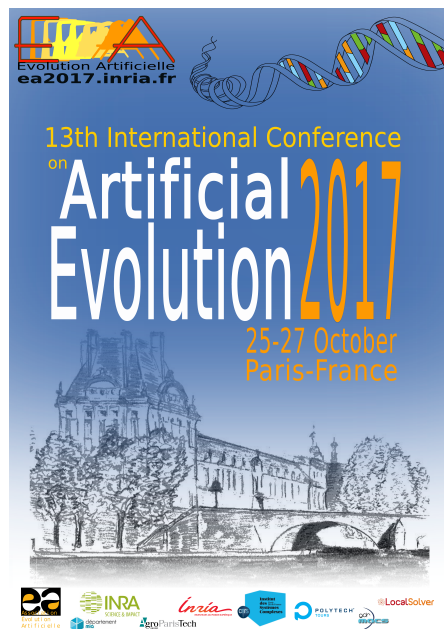# Artificial Evolution 2017

13th Biennal International Conference on Artificial Evolution

Proceedings

25-27 October 2017, Paris, France

Evelyne Lutton, Pierrick Legrand, Pierre Parrend
Nicolas Monmarché, Marc Schoenauer

Chair

Evelyne Lutton - *INRA Versailles-Grignon, France*

Steering Committee

Nicolas Monmarché - *Univ. of Tours, France*
Marc Schoenauer -*INRIA Saclay, France*

Organizing Committee

Marc Barnabé - *INRA Versailles-Grignon, France*
Nadia Boukhelifa - *INRA Versailles-Grignon, France*
Emmanuel Cayla - *Galerie Louchard, Paris, France*
Thomas Chabin - *INRA Versailles-Grignon, France*
Pierre Collet - *Univ. of Strasbourg, France*
Aurélien Dumez - *INRIA Grenoble, France*
Benoît Génot - *INRA Versailles-Grignon, France*
Lhassane Idhoumgar - *Univ. of Mulhouse, France*
Anne Jeannin-Girardon - *Univ. of Strasbourg, France*
Laetitia Jourdan - *Univ. of Lille 1, France*
Pierrick Legrand - *Univ. of Bordeaux, France*
Pierre Parrend - *Univ. of Strasbourg, France*
Alberto Tonda - *INRA Versailles-Grignon, France*

Program Committee

Hernan Aguirre - *Univ. of Shinshu, Japan*
Anne Auger - *INRIA Saclay, France*
Sebastien Aupetit - *Univ. of Tours, France*
Stefan Balev - *Univ. of Le Havre, France*
Sana Ben Hamida - *Univ. of Paris Ouest, France*
Christian Blum - *IIIA-CSIC, Spain*
Stéphane Bonnevay - *Univ. of Lyon 1, France*
Nadia Boukhelifa - *INRA Versailles-Grignon, France*
Amine Boumaza - *Univ. of Lorraine, France*
Nicolas Bredeche - *Univ. Pierre & Marie Curie, France*
Stefano Cagnoni - *Univ. of Parma, Italy*
Maurice Clerc - *Independent Scholar, France*
Manuel Clergue - *U.A.G., France*
Pierre Collet - *Univ. of Strasbourg, France*
Fabio Daolio - *Univ. of Shinshu, Japan*
Fatima Debbat - *Univ. of Mascara, Algeria*
Laurent Deroussi - *Univ. of Clermont-Ferrand, France*
Carola Doerr - *Sorbonne Univ., France*
Nicolas Durand - *ENAC, Toulouse, France*
Marc Ebner - *Univ. of Greifswald, Germany*
Francisco Fernández de Vega - *Extremadura Univ., Spain*
Cyril Fonlupt - *Univ. of Littoral, France*
Edgar Galvan - *Trinity College, Dublin, Ireland*
Mario Giacobini - *Univ. of Turin, Italy*
Jin-Kao Hao - *Univ. of Angers, France*
Lhassane Idoumghar - *Univ. of Mulhouse, France*
Thomas Jansen - *Univ. of Aberystwyth, UK*
Anne Jeannin-Girardon - *Univ. of Strasbourg, France*
Laetitia Jourdan - *Univ. of Lille 1, France*
Bill Langdon - *Univ. College London, UK*
Pierrick Legrand - *Univ. of Bordeaux, France*
Julien Lepannot - *Univ. of Haute-Alsace, France*
Arnaud Liefooghe - *Univ. of Lille 1, France*
Manuel López-Ibáñez - *Free Univ. of Brussels, Belgium*

Jean Louchet - *INRIA Saclay, France*
Evelyne Lutton - *INRA Versailles-Grignon, France*
Virginie Marion-Poty - *Univ. of Littoral, France*
Eric Medvet - *Univ. of Trieste, Italy*
Juan Julián Merelo Guervós - *Univ. of Granada, Spain*
Nicolas Monmarché - *Univ. of Tours, France*
Amir Nakib - *Univ. of Paris, France*
Gabriela Ochoa - *Univ. of Stirling, Scotland, UK*
Damien Olivier - *LITIS, Univ. of Le Havre, France*
Luis Paquete - *Univ. of Coimbra, Portugal*
Andrew Parkes - *Univ. of Nottingham, UK*
Pierre Parrend - *Univ. of Strasbourg, France*
Francisco Pereira - *Univ. of Coimbra, Portugal*
Nathalie Perrot - *INRA Versailles-Grignon, France*
Alain Petrovsky - *Telecom Sud Paris, France*
Denis Robilliard - *Univ. of Littoral, France*
Eduardo Rodriguez-Tello - *CINVESTAV, Mexico*
Frédéric Saubion - *Univ. of Angers, France*
Marc Schoenauer - *INRIA Saclay, France*
Ines Sghir - *Univ. of Manouba, Tunisia*
Patrick Siarry - *Univ. of Paris-Est Creteil, France*
Sara Silva - *Univ. of Coimbra, Portugal*
Dan Simon - *Univ. of Cleveland State, USA*
Christine Solnon - *INSA Lyon, France*
Giovanni Squillero - *Royal Turin Polytechnic, Italy*
Thomas Stützle - *IRIDIA, Brussels, Belgium*
El-Ghazali Talbi - *INRIA Lille, France*
Fabien Teytaud - *Univ. of Littoral, France*
Alberto Tonda - *INRA Versailles-Grignon, France*
Leonardo Trujillo - *Tijuana Inst. of Tech., Mexico*
Paulo Urbano - *Univ. of Lisboa, Portugal*
Sébastien Verel - *Univ. of Littoral, France*
Nicolas Zufferey - *Univ. of Geneva, Switzerland*

Contents

Foreword

These proceedings collect all papers presented at the 13th Biennial International Conference on Artificial Evolution, EA[1] 2017, held in Paris (France). This conference proceeds a long series of previous issues, that took place in Lyon (2015), Bordeaux (2013), Angers (2011), Strasbourg (2009), Tours (2007), Lille (2005), Marseille (2003), Le Creusot (2001), Dunkerque (1999), Nimes (1997), Brest (1995), and Toulouse (1994).

We sought original contributions relevant to Artificial Evolution, including, but not limited to: evolutionary computation, evolutionary optimisation, co-evolution, artificial life, population dynamics, theory, algorithmics and modelling, implementations, application of evolutionary paradigms to the real world (industry, biosciences, ...), other biologically-inspired paradigms (swarm, artificial ants, artificial immune systems, cultural algorithms...), memetic algorithms, multi-objective optimisation, constraint handling, parallel algorithms, dynamic optimisation, machine learning and hybridisation with other soft computing techniques.

Each submitted paper was reviewed by four members of the International Program Committee. Among the 33 submissions received, 17 papers were selected for oral presentation and 7 other papers for poster presentation. As for the previous editions (see LNCS volumes 1063, 1363, 1829, 2310, 2936, 3871, 4926, 5975 and 7401), a selection of best papers presented at the conference will be published as a post-conference volume of Springer's LNCS series, after further revisions.

As usual, the EA2017 success is indebted to dedicated team work, for which I would like to express my gratitude:

- Gabriela Ochoa and Jean-Daniel Fekete, who accepted to be our keynote speakers;
- The Program Committee for their careful work: the high quality of the selected papers is a proof of their strong commitment;
- The Organising Committee for their efficient work and kind availability, in particular the local team, Nadia Boukhelifa, Alberto Tonda and our student volunteers, Marc Barnabé, Thomas Chabin and Benoît Génot ;
- The ISC-PIF who hosted this event: David Chavalarias, his director, and the local ISC organisation team, Margaux Calon and Franck Leclerc, for their kind, efficient and everyday help;
- The members of the Steering Committee for their valuable assistance;
- Aurélien Dumez and Pierrick Legrand for the administration of the conference website;
- Marc Schoenauer and Anne Jeannin-Girardon for their support and management of the MyReview system;
- Laetitia Jourdan for publicity;
- Pierrick Legrand and Pierre Parrend for the edition of the proceedings;
- Lhassane Idoumghar for registrations;
- Emmanuel Cayla and Nicolas Monmarché for the organisation of the Twin Event: "Art and Artificial Evolution" at Galerie Louchard.

I take this opportunity to thank the different partners whose financial and material support were precious: the MIA department of INRA, the INRIA Saclay research unit, AgroParisTech, ISC-PIF, Polytech-Tours, the Local Solver company, RO and MACS GDR of CNRS.

We are as always deeply grateful to all authors who submitted their research work to the conference, to all artists who contribute to the art exposition, and to all attendees who make the conference so lively. The scientific quality as well as the warm and friendly atmosphere of this series of conferences is the result of a rare alchemy that is still maintained. Thank you for all these years of fidelity, thank you for EA 2017.

Evelyne Lutton

EA 2017 Chair
INRA Versailles-Grignon, France

---

[1] As for previous editions of the conference, the EA acronym is based on the original French name "Évolution Artificielle".

Invited conferences

Gabriela Ochoa

The Cartography of Computational Search Spaces

Gabriela Ochoa[2] has worked on Evolutionary Computation since the late 90s. Her very first article used genetic algorithms to evolve plant-like structures encoded as Lindenmayer systems. During her PhD at the University of Sussex, UK she worked with the notion of error thresholds in genetic algorithms. She held academic positions at the University Simon Bolivar in Caracas, Venezuela and the University of Nottingham, UK before joining the University of Stirling, Scotland in 2012. Her current research includes autonomous search and hyper-heuristics, fitness landscape analysis and applications to combinatorial optimisation, software engineering and healthcare. She is an associate editor of the Journal of Evolutionary Computation, MIT Press and the IEEE Transactions of Evolutionary Computation. She has served as an organiser for EvoCOP, FOGA and PPSN, is a member of the board of ACM SIGEVO, and served as the Editor-in-Chief for GECCO 2017.

Abstract

This talk will present our recent findings and visual (static and animated) maps characterising combinatorial and program search spaces. We seek to lay the foundations for a new perspective to understand problem structure and improve heuristic search algorithms: search space cartography.

A multitude of heuristic and bio-inspired search algorithms has been proposed, each trying to be more powerful and innovative. However, little attention has been devoted to understanding the problems' structure and what makes them hard to solve for a given algorithm. Formal theoretical results are difficult to obtain, and they may only apply to problem classes and algorithms chosen more for their amenability to analysis than for their relevance and difficulty.

Heuristic methods operate by searching a large space of candidate solutions. The search space can be regarded as a spatial structure where each point (candidate solution) has a height (objective or fitness function value) forming a fitness landscape surface. The performance of optimisation algorithms crucially depends on the fitness landscape structure, and the study of landscapes offers an alternative to problem understanding where realistic formulations and algorithms can be analysed.

Most fitness landscapes analysis techniques study the local structure of search spaces. There is currently a lack of tools to study instead their global structure, which is known to impact algorithms' performance. Our recently proposed model, local optima networks, fills this gap by bringing tools from complex networks to study optimisation. This model provides fundamental new insight into the structural organisation and the connectivity pattern of a search space with given move operators. Most importantly, it allows us to visualise realistic search spaces in ways not previously possible and brings a whole new set of quantitative network metrics for characterising them.

---

[2]http://www.cs.stir.ac.uk/∼goc

Jean-Daniel Fekete

Progressive Data Analysis: a new computation paradigm for scalability in exploratory data analysis

Jean-Daniel Fekete[3] is Senior Research Scientist (DR1) at INRIA, Scientific Leader of the INRIA Project Team AVIZ that he founded in 2007.

He received his PhD in Computer Science in 1996 from Université Paris-Sud. From 1997 to 2001, he joined the Graphic Design group at the Ecole des Mines de Nantes that he led from 2000 to 2001. He was then invited to join the Human-Computer Interaction Laboratory at the University of Maryland in the USA in 2001-2002. He was recruited by INRIA in 2002 as a confirmed researcher and became Senior Research Scientist in 2006. In 2015, he was on Sabbatical at the Visualization and Computer Graphics group at NYU-Poly, and at the Visual Computing Group at Harvard.

His main Research areas are Visual Analytics, Information Visualization and Human Computer Interaction. He published more than 150 articles in multiple conferences and journals, including the most prestigious in visualization (TVCG, InfoVis, EuroVis, PacificVis) and Human-Computer Interaction (CHI, UIST).

He is the chair of the IEEE Information Visualization Conference Steering Committee, member of the IEEE VIS Executive Committee, member of the Eurographics EuroVis Steering Committee, and member of the Eurographics publication board. He is also an ACM Distinguished Speaker.

Jean-Daniel Fekete was the General Chair of the IEEE VIS Conference in 2014, the first time it was held outside of the USA in Paris, Associate Editor of the IEEE Transactions on Visualization and Computer Graphics (TVCG) 2011-2015, the President of the French-Speaking HCI Association (AFIHM) 2009-2013, he was Conference Chair of the IEEE InfoVis Conference in 2011, Paper Co-Chair of the IEEE Pacific Visualization conference in 2011.

Jean-Daniel Fekete is a member of the Association for Computer Machinery (ACM) and Senior Member of the IEEE.

Abstract

Exploring data requires a short feedback loop, with a latency of at most 10 seconds because of human cognitive capabilities and limitations. When data becomes large or analyses become complex, sequential computations can no longer be completed in a few seconds and interactive exploration is severely hampered. This talk will describe a novel computation paradigm called Progressive Data Analysis that brings at the programming language level the low-latency guarantee by performing computations in a progressive fashion. Moving this progressive computation at the language level relieves the programmer of exploratory data analysis systems from implementing the whole analytics pipeline in a progressive way from scratch, streamlining the implementation of scalable exploratory analytics systems. I will describe the new paradigm, report on novel experiments showing that human can cope effectively with progressive systems, and show demos using a prototype implementation called ProgressiVis, explain the requirements it implies through exemplar applications, and present opportunities and challenges ahead, in the domains of visualization and machine-learning.

---

[3]http://www.aviz.fr/∼fekete

LONG PAPERS - ORAL PRESENTATIONS

SESSION 1 - THEORY

On the Design of a Master-Worker Adaptive Algorithm Selection Framework

Christopher Jankee[1], Sébastien Verel[1] Bilel Derbel[2], and Cyril Fonlupt[1]

[1] Université du Littoral Côte d'Opale, LISIC
[2] Université Lille 1, LIFL – CNRS – INRIA Lille

Abstract. We investigate the design of a master-worker schemes for adaptive algorithm selection with the following two-fold goal: (i) choose accurately from a given portfolio a set of operators to be executed in parallel, and consequently (ii) take full advantage of the compute power offered by the underlying distributed environment. In fact, it is still an open issue to design online distributed strategies that are able to optimally assign operators to parallel compute resources when distributively solving a given optimization problem. In our proposed framework, we adopt a reward-based perspective and investigate at what extent the average or maximum rewards collected at the master from the workers are appropriate. Moreover, we investigate the design of both homogeneous and heterogeneous scheme. Our comprehensive experimental study, conducted through a simulation-based methodology and using a recently proposed benchmark family for adaptive algorithm selection, reveals the accuracy of the proposed framework while providing new insights on the performance of distributed adaptive optimization algorithms.

1 Introduction

The selection of an accurate algorithm from a given portfolio, as well as the effective choice of the relevant algorithmic components of a general-purpose search heuristic, are among the major issues that one has to face in practice when tackling an optimization problem; in particular, in a black-box optimization scenario when no problem-specific properties can be known beforehand [3]. In fact, from a theoretical point of view, several parallel compute ressources, possibly distributed over a large scale environment, are provided, it is even more challenging to design an efficient distributed cooperative strategy, since the algorithmic design space gets huge and we still lack knowledge on the optimal mapping of the implied search computational flows to the available resources. The motivation of this paper is precisely to investigate these issues by proposing a master-worker algorithm selection framework and precisely analyzing the impact of its different possible design components. On the one hand, algorithm selection (or the related topic of parameter setting), although being one of the oldest research topic in evolutionary computation [14], is attracting more and more attention [17] due to its crucial importance and the difficult, and yet unsolved, challenges it implies

in practice. In this work, we are interested in *adaptive algorithm selection*. Indeed, there are two main and tightly related methodologies that are commonly adopted to select an algorithm [9]. In the offline setting, usually called *tuning*, an algorithm is first selected, and only then it is executed from scratch on the target and unseen problem instance. In the online setting, called *control*, an algorithm is selected all along the optimization process (see for example [11, 3]). An online selection scheme is typically and continuously getting feedback from the optimization algorithm being executed, and deciding accordingly on the next choice. Hence, online algorithm selection can be viewed as an adaptive optimization algorithm which follows the multi-armed bandit framework where the arms are the algorithms of the portfolio [5]. The adaptive algorithm selection is then performed as follows. A reward is computed according to the performance observed when previously executing an algorithm. Then, in every iteration, a reinforcement machine learning is applied in order to select from the portfolio the next algorithm to execute, typically according to some exploration-exploitation rules.

On the other hand, numerous real wold optimization problems, such as engineering design which are often based on numerical simulation, are computationally expensive, e.g., one fitness function evaluation can take several minutes [19]. Besides, the advent of new compute facilities and the establishment of robust and large scale massively parallel platforms, such as grids and pay-as-you-go clouds, open tremendous research opportunities for pushing forward the development and uptake of parallel and distributed evolutionary optimization algorithms. In this context, a number of evolutionary optimization models have been investigated [20], e.g., from centralized to fully decentralized, from fined-grained (cellular model) to coarse grained (island model). In this work, we adopt the centralized Master-Worker (M/W) architecture, where each worker process is basically responsible of executing locally in parallel the actions scheduled for him by the master (e.g., evaluate a candidate solution), whereas the master process is responsible of collecting the local results from the workers (e.g., the fitness values) and deciding on the next actions to send them (e.g., next candidate solutions to evaluate). It is worth-noticing that this framework is often adopted in practice, not only due to the simplicity of deploying it over a real test-bed, but also due to its high accuracy when dealing with computationally expensive optimization problems [6].

In this context, we argue that a master-worker approach to adaptive algorithm selection requires specific coordination mechanisms in order to achieve optimal performances. In a sequential setting, the observed rewards are in fact updated according to the performance of the algorithm executed previously in the last round by one single process. In a M/W approach, one can benefit from the set of performances observed by several parallel processes, i.e., the workers. However, switching to such a scenario requires to carefully define the aggregated reward with respect to a selected algorithm given a set of observed performance values instead of just a single one. Additionally, one can adopt either a homogenous strategy in which all workers execute the same algorithm at each iteration (e.g., the best rewarded one so-far) or instead a heterogeneous strategy where

the workers can execute different algorithms. Several existing machine learning technics have previously been used and studied in the sequential setting [11], as well as in the decentralized island model [7, 15]. However, to our best knowledge, the design and analysis of online selection strategies have not been investigated within a M/W framework. We argue that the M/W scheme make it more convenient, as a first step, to reason about the optimal distributed decisions to make since the master has the ability of acquiring a global view of the whole distributed system before selecting the most accurate algorithms to execute in parallel by the workers. This allows us to focus on the critically important selection strategy at the master level. To summarize, we propose a M/W algorithm selection framework contributing to the solving of the following questions:

- How to define a reward function on the master based on the performance of the algorithm(s) executed by the workers?
- How the master can decide on the set of algorithms to be executed next by the workers based on the reward function?
- What is the relative quality that can be achieved by different algorithm selection strategies ?

Our M/W framework is evaluated using a tunable benchmark family and a simulation-based experimental procedure in order to abstract away the technical implementation issues, and instead provide a fundamental and comprehensive analysis of the expected empirical parallel performance of the underlying adaptive algorithm selection. The rest of the paper is organized as follows. In Section 2, we review some related works. In Section 3, the design components of our M/W adaptive framework is described in details. In Section 4, we report our main experimental findings. In Section 5, we conclude the paper and discuss future research directions.

2   Related works

In the following, we provide an overview of related studies on the algorithm selection problem in the sequential and distributed setting, as well as a brief summary of exiting optimization benchmark problems designed at the aim of evaluating their dynamics and behavior.

2.1   Sequential Adaptive Algorithm Selection

In the sequential setting, a number of reinforcement machine learning technics have been proposed for the online and adaptive selection of algorithms from a given portfolio. Back to the early works of Grefenstette [14], one standard technique consists in predicting the performance of a set of operators using a simple linear regression and the current average fitness of the population, which then allows to select the best operator to be chosen according to the prediction given by the regression. However, recent works embeds this selection

problem into a multi-armed bandit framework dealing more explicitly with the tradeoff between the exploitation of the best so far identified algorithm, and the exploration of the remaining potentially under-estimated algorithms.

A simple strategy is the so-called ϵ-greedy (ϵ-G) strategy which consists in selecting the algorithm with the best estimated performance at rate $(1 - \epsilon)$, and a random one at rate ϵ. In that case, the performance of an operator i is estimated with the empirical mean $\hat{\mu}_i$ of rewards on a sliding window where only the W previous reward observations are considered. The Upper Confidence Bound (UCB) strategy [2] is a state-of-the-art framework in machine-learning which consists in estimating the upper confidence bound of the expected reward of each arm by $\hat{\mu}_i + C \cdot e_i$; where $\hat{\mu}_i$ is the estimated (empirical) mean reward, and e_i is the standard error of the prediction. It then selects the algorithms with the higher bound (for maximization problem). The parameter C allows to tune the exploitation/exploration trade-off. In the context of algorithm selection [11] where the arms could be neither independent nor stationary, the estimation of the expected reward is refined using a sliding window of size W. The Adaptive Pursuit (AP) strategy [22] is another technique using an exponential recency weighted average to estimate the expected reward with a parameter α to tune the adaptation rate of the estimation. This is used to define the probability p_i of selecting every algorithm from the portfolio. At each iteration, these probability values are updated according to a learning rate β, which basically allows to increase the selection probability for the best algorithm, and to decrease it for the other ones.

One key aspect to design a successful adaptive selection strategy is the estimation of the quality of an algorithm based on the observed rewards. Some authors showed that the maximum reward over a sliding window improves the performance compared to the mean on some combinatorial problems [11, 4]; but no fundamental analysis of this result was given. In genetic algorithms, the reward can be computed not only based on the quality but also on the diversity of the population [18]. In the context of parallel adaptive algorithm selection, the estimation of quality of each available algorithm is also a difficult question since not only one but many algorithms instances could be executed in each iteration.

2.2 Parallel Adaptive Algorithm Selection

The Master-Worker (M/W) architecture has been extensively studied in evolutionary computation (e.g., see [8]). It is in fact simple to implement, and does not require sophisticated parallel operations. Two communication modes are usually considered. In the synchronous mode, the distributed entities operate in rounds, where in each round the master communicates actions to the workers and then waits until receiving a response from every worker before starting a new round, and so on. In the asynchronous mode, the master does not need to wait for all workers; but instead can initiate a new communication with a worker, typically when that worker has terminated executing the previous action and is idle. When the evaluation time of the fitness function can vary substantially during the course of execution, the asynchronous mode is generally preferred [24] since

it can substantially improve parallel efficiency. However, the synchronous mode can allow to have a more global view of the distributed system which can be crucially important to better coordinate the workers [23].

Adaptive selection approaches designed to operate in a distributed setting are not new. The island model, which is considered as inherently distributed, has been investigated in the past. To cite a few, in [21, 12], it is also demonstrated that a randomly setting the parameters at each iteration in a heterogeneous manner can outperforms static homogeneous parameter settings. Nonetheless, embedding a reinforcement machine learning technique instead of random selection can improve the performance of the adaptive distributed system. In [4], a dynamic island model is proposed to select online the relevant algorithm. Each island is associated to one algorithm, and the migration rates of solutions between islands are controlled by the operators performance of each island. As commented by the authors, this technique is not designed to fit directly in a scalable distributed system and requires some further adaptations. In [7, 15], a distributed adaptive metaheuristic selection framework is proposed which can be viewed as a natural extension of the island model that was specifically designed to fit the distributed nature of the target compute platforms. The adaptive selection is performed locally by selecting the best rewarded metaheuristic from the neighboring nodes (islands) or a random one with small probability like in ϵ-greedy strategy. Notice however that we are not aware of any in-depth analysis addressing the design principles underlying a M/W adaptive algorithm selection approach. In this work, we propose and empirically analyze the behavior of such an approach in an attempt to fill the gap between the existing sequential algorithm selection methods and the possibility to deploy them in a parallel compute environment using a simple, yet effective, parallel scheme like the M/W one.

2.3 Benchmarks: The Fitness Cloud Model

The understanding of the dynamics of a selection strategy according to the problem at hand is a difficult issue. A number of artificial combinatorial problems have been designed and used in the literature. We can distinguish between two main benchmark classes. In the first one, a well-known combinatorial problem in evolutionary algorithm is used, such as oneMax or long-path problems, with basic operators, such as bit-flip, embedded in a $(1 + \lambda)$-EA [5]. This however can only highlight the search behavior according to few and problem-specific properties. In the second class of benchmarks, the problem and the stochastic operators are abstracted. The performance of each available operator is then defined according to the state of the search [22, 11, 13, 16]. This allows to study important black-box (problem independent) features such as the number of operators, the frequency of change of the best operators, the quality difference between operators, etc.

In this work, we use a tunable benchmark, called the Fitness Cloud Model (FCM), introduced recently in [16]. The FCM is a benchmark from the second class where the state of the search is given by the fitness of the solution. The fitness of a solution after applying a search operator is modeled by a random variable for which the probability distribution depends on the fitness of the

current solution. A normal distribution with tunable parameters is typically used. More specifically, given the fitness $z = f(x)$ of the current solution x, the probability distribution of the fitness $f(y)$ of one solution obtained by a specific operator is defined by: $\Pr(f(y) = z' \mid f(x) = z) \sim \mathcal{N}(\mu(z), \sigma^2(z))$ where $\mu(z)$ and $\sigma^2(z)$ are respectively the mean and the variance of the normal distribution. In [16], a simple scenario with two operators is studied. The mean and variance of the conditional normal distribution are defined as follows: $\mu_i(z) = z + K_{\mu_i}$ and $\sigma_i^2(z) = K_{\sigma_i}$ for each operator $i \in \{1, 2\}$. Parameters K_{μ_i} and K_{σ_i} are different constant numbers. An adaptive algorithm is assumed to start with a search state where the fitness value is 0, and stops when a fitness value of 1 is reached. Notice that in the FCM, on the contrary of benchmark of the first class (oneMax, etc.), one can control the average quality and the variance of each operator as well the relative difference between the considered operators which are two of the main features to analyze from the perspective of adaptive selection of operators. Please refer to [16] for more details on the design and motivation of the FMC benchmark.

3 M/W Framework Description

First, a portfolio of k (local search) operators is assumed to be given, and no a priori knowledge is assumed on the behavior of the operators with respect to the black-box problem under consideration. Naturally, k is an integer value greater or equal than 2. The global architecture of the proposed adaptive M/W framework is summarized in Algorithm 1 depicting the high level code executed by the master and in Algorithm 2 depicting the high level code executed in parallel by each worker. The overall algorithm operates in different parallel rounds. At each round, the master sends the best solution $x^\star$ and the operator identifier θ^i assigned to each worker node i. Based on $x^\star$ and θ^i, the role of each worker is to compute a new candidate solution to be send back to master. Although one could consider and study different alternatives, in this work, a standard $(1+1)$-EA is simply executed by each worker. In addition, the worker computes a local reward in order to render the quality of its assigned operator θ^i. Different kinds of local rewards can be considered at this stage [10]. In our work, and since an elitist selection is applied locally by each worker, the local reward of an operator is the positive improvement observed when applying the $(1+1)$-EA. The master waits for all local solutions computed in parallel by the workers, and updates the global best solution $x^\star$ to be considered in the next round, and so on. More importantly, the local rewards collected by the master are used in order to select a new set of operators to be assigned to the workers in the subsequent rounds, which actually constitutes the adaptive and core part of our framework. Two tightly coupled issues are to be handled by the master in order to set up an effective adaptive mechanism: (i) how to aggregate the local rewards sent by the workers and (ii) how to select the new set of operators accordingly. This is described next.

Algorithm 1 Adaptive M/W algorithm for the master node

1: $(\theta^1, \theta^2, ..., \theta^n) \leftarrow$ Selection_Strategy_Initialization()
2: $x^\star \leftarrow$ Solution_Initialization() ; $f^\star \leftarrow f(x^\star)$
3: **repeat**
4:     **for** each worker i **do**
5:         Send $\mathsf{Msg}(\theta^i, x^\star, f^\star)$ to worker i
6:     **end for**
7:     Wait until all messages are received from all workers
8:     **for** each worker i **do**
9:         $(r^i, x^i, f^i) \leftarrow$ Receive $\mathsf{Msg}()$ from worker i
10:    **end for**
11:    $x^\star \leftarrow x^i; f^\star \leftarrow f^i$ s.t. $f^i = \max\{f^\star, f^1, f^2, \ldots, f^n\}$
12:    $(R_1, R_2, ..., R_k) \leftarrow$ Reward_Aggregation$((\theta^1, r^1), ..., (\theta^n, r^n))$
13:    $(\theta^1, \theta^2, ..., \theta^n) \leftarrow$ Decision_Strategy$(R_1, R_2, ..., R_k)$
14: **until** stopping criterion is true

Algorithm 2 Adaptive M/W algorithm for each worker node

1: $(\theta, x^\star, f^\star) \leftarrow$ Receive Msg() from master
2: $x' \leftarrow$ Apply operator θ on $x^\star$; $f' \leftarrow$ Evaluate fitness of x'
3: $\delta_b \leftarrow \max(0, f' - f^\star)$
4: **if** $f(x^\star) < f(x')$ **then**
5:    $x^\star \leftarrow x'$; $f^\star \leftarrow f'$
6: **end if**
7: Send $\mathrm{Msg}(\delta_b, x^\star, f^\star)$ to master

3.1 Aggregation of local reward values

On one hand, all adaptive operator selection strategies such as ϵ-greedy, Adaptive Pursuit, Upper Confidence Bound, etc. (see Sec. 2.1) need to get one single reward value as a feedback when one operator is executed. On the other hand, in our framework, a set of local rewards are computed by the workers and provides us with a feedback on the quality of an operator when executed in parallel by several workers. Unlike sequential algorithms, the set of local rewards observed in parallel cannot be viewed simply as a sequence of independent rewards that would be given iteratively to a sequential strategy. Hence, one specific design component of an adaptive M/W algorithm is the way to aggregate the local reward values into one global reward value. Consequently, we distinguish two main aggregation strategies: (i) the mean or the (ii) maximum of the local rewards. In other words, at each round, the (global) reward computed by the master, with respect to one operator executed by at least one worker, is either the average or the maximum of the local values sent by the corresponding workers.

Despite their simplicity, the two previous local reward aggregation strategies are fundamentally different. In fact, assuming that the fitness improvement after applying a stochastic operator is given by a probability distribution, the mean of the reward values computed by the n workers allows to estimate the expectation of this distribution with a high accuracy, whereas the maximum gives information

on its extremes [10]. Additionally, we consider a sliding window of size W to estimate the expected reward $\hat{\mu}_i$ in ϵ-greedy, and UCB as considered in previous works.

3.2 Homogeneous *vs.* Heterogeneous Adaptive Selection

As mentioned previously, the master needs to select one operator for each worker. We consider both (i) a *Homogeneous* (Ho) adaptive strategy, in which the same operator is selected by the master and assigned to all worker, and (ii) a *Heterogeneous* (He) adaptive strategy, in which the master selects, possibly different, operators to be assigned to the workers. The rationale behind a homogeneous strategy is that in each round there exists one relevant operator providing an optimal performance, and hence should be executed simultaneously in parallel by all workers. This a rather exploitation-guided strategy which aims at avoiding to loose function evaluations, and to post-pone the exploration component to act in-between two consecutive rounds. In contrast, the rationale behind a heterogeneous strategy is that a set containing a mixture of different operators is expected to perform better than a set containing the same operator, in the sense that: (i) the probability of obtaining a better solution when executing different operators in each round is larger, and/or (ii) a relatively small number of evaluations spent exploring non-necessarily optimal operator(s) at each round allows to better predict the best operator(s) to select next.

In the homogeneous setting, we consider the three standard selection strategies (cf. Sect. 2.1), namely, ϵ-greedy, AP, and UCB. The same operator computed by any of these strategies is assigned by the master to the workers. Notice that the difference with a sequential selection is the way the reward is computed by the workers and maintained by the master, which is crucially important for those methods to operate accurately. In the heterogeneous setting, we consider to execute either the ϵ-greedy strategy or the AP strategy iteratively for each worker. Notice in fact that these two strategies are randomized, i.e., for ϵ-greedy, the best operator is selected with rate $1 - \epsilon$ and the other ones with rate ϵ, and for AP, each operator is selected proportionally to a rate p_i. Hence, by running iteratively those strategies, the selected operators is likely to be different in each execution and the master is then able to assign different operators to the workers. In contrast, running iteratively an UCB selection does not give an heterogenous strategy due to its deterministic nature (same operator is given at each selection step). Designing an heterogeneous UCB-based strategy is actually a challenging open question which is left for future investigations.

4 Experimental analysis

We consider the Fitness Cloud Model as an abstract benchmark. We have three competing adaptative selection mechanisms (ϵ-G, UCB, and AP) which combined accordingly with the two considered reward aggregation strategies (mean

Table 1. Parameters setting of the selection strategies

Selection strategy	Parameters value			Selection strategy	Parameters value	
ϵ-G He. Max.	$\epsilon = 0.5$	$W = 400$		UCB Max.	$C = 0.005$	$W = 700$
ϵ-G He. Mean	$\epsilon = 0.5$	$W = 400$		UCB Mean	$C = 0.05$	$W = 5000$
ϵ-G Ho. Max	$\epsilon = 0.05$	$W = 4500$		AP	$\alpha = 0.2$	$\beta = 0.2$
ϵ-G Ho. Mean	$\epsilon = 0.05$	$W = 4500$				

and max), and the two homogeneity scenarios (Ho and He), provide us 10 variants. Moreover, we consider two baseline random strategies, which consist in selecting the next operator randomly, both in a homogeneous or in a heterogeneous setting. In the following, we first start discussing the overall relative performance, then provide a more focused analysis to better understand the behavior and the dynamics of the different variants.

4.1 Overall Relative Performance

We adopt a simulation-based approach where we count the number of rounds performed by the master until reaching the optimal fitness value. This allows us to abstract away the communication issues and to evaluate the accuracy of the considered algorithms in adapting the search process to operate optimally. We consider a portfolio with $k = 2$ operators. Following the Fitness Cloud Model, each operator impacts differently the fitness of solution: the first one follows the normal distribution $\mathcal{N}(-10^{-4}, 10^{-4})$, and the second one $\mathcal{N}(-10^{-3}, 5 \times 10^{-4})$. These distributions are fixed and do not change in the course of the optimization which is a simple, yet challenging, scenario in order to elicit the behavior of adaptive algorithms in a black-box scenario. For the sake of presentation, the choice of the benchmark parameters will be discussed later. The parameter set of the different selection strategies is given in Table 1. This setting can be shown to be robust and is in accordance with previous studies [16, 15]. Each variant is executed 100 times and an overview of the performance in terms of number of distributed rounds is given in Fig. 1. Three main observations can be made.

Firstly, using the mean reward aggregation function is clearly outperformed by the maximum reward function. Secondly, the difference between a homogeneous and heterogeneous setting is mitigated and depends on the selection it-self. Thirdly, according to a Mann-Whitney statistical test at confidence level of 5%, and when comparing the best setting of given selection variant, the UCB strategy appears to be the best one, followed by ϵ-greedy strategy and are better than the AP strategy. These first results can be explained by the ability of the UCB machine-learning inspired strategy to efficiently learn the best operator to apply in a given round when using the maximum reward. The other strategies, although being competitive, spend some rounds to explore non-relevant operators. More importantly, all adaptive strategies are found to share a relatively good performance when the other design components, that is the choice of the reward, and the heterogeneity, are well tuned. To better understand such a behavior, we provide next a more throughout analysis.
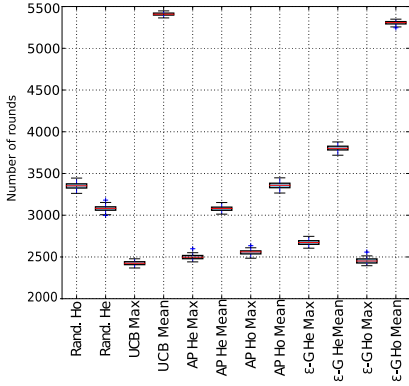
Fig. 1. Number of rounds to the optimal fitness using operator $1 \sim \mathcal{N}(-10^{-4}, 10^{-4})$, operator $2 \sim \mathcal{N}(-10^{-3}, 5 \times 10^{-4})$ and $n = 256$ workers.
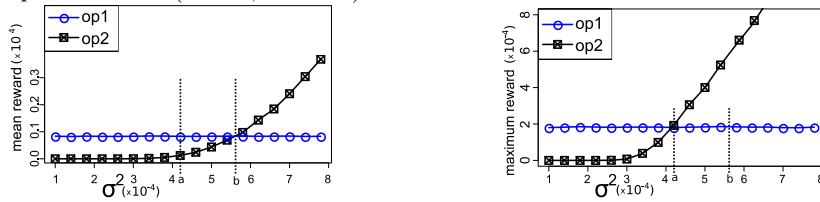
Fig. 2. Mean (top) and maximum (bottom) reward values with operator $1 \sim \mathcal{N}(-10^{-4}, 10^{-4})$, operator $2 \sim \mathcal{N}(-10^{-3}, \sigma^2)$ and $n = 256$ workers as a function of the variance σ^2.

4.2 Analysis of the Reward Aggregation Functions

To understand the fundamental difference between using the maximum or the mean as a reward function, as well as its crucial importance when designing an adaptive strategy, we consider to study the property of the considered fitness cloud benchmark in an extended setting. More precisely, let us fix the parameters of the normal distribution corresponding to the first operator in the portfolio to $\mu_1 = -10^{-4}$ and $\sigma_1^2 = 10^{-4}$. Let us also fix the mean of the normal distribution of the second operator to $\mu_1 = -10^{-3}$. Since both means are negative, the fitness value is decreased *in expectation* by both operators. For the fixed number of workers, and since the parameters of the normal law does not change in the course of optimization, operator 1 would always provide the same expected improvement, i.e., 8.33×10^{-2} [15], which corresponds to the local reward. Let us now study how the relative reward value would be for operator 2 if its variance was set to take different values than in our initial setting. This is summarized in Fig. 2 showing the expected rewards of both operators when using a mean aggregation function (top) and a maximum aggregation function (bottom), for $n = 256$ workers as a function of a range of variance values σ^2 for operator 2.

For both reward aggregation functions, the reward value of operator 2 increases with the variance σ^2. Below the value of $a = 4.17 \times 10^{-4}$, the reward of
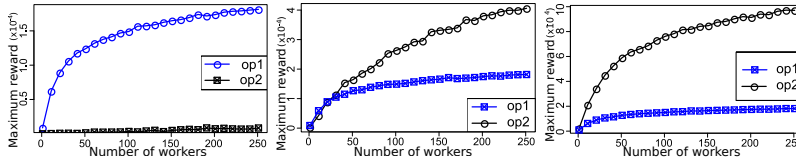
Fig. 3. Maximum reward value as function of the number of workers n for operator $1 \sim \mathcal{N}(-10^{-4}, 10^{-4})$ and different variance values of operator $2 \sim \mathcal{N}(-10^{-3}, \sigma^2)$: $\sigma^2 = 10^{-4}$ (left), $\sigma^2 = 5 \times 10^{-4}$ (middle), and $\sigma^2 = 7 \times 10^{-4}$ (right).

operator 1 is higher than the reward of operator 2 for both mean and maximum functions. Hence, an (elitist) operator selection strategy which selects the operator according to the highest reward value would select the same operator 1, no matter which aggregation function is used, i.e., there is no difference between the two aggregation function in the case the difference between the fitness variances of both operators is relatively large, given that their mean fitness is same. Similarly, when the variance σ^2 of operator is much larger than the variance of operator 1 ($\sigma^2 > b = 5.6 \times 10^{-4}$), the reward value for operator 2 is larger compared to operator 1 no matter the reward aggregation used. Hence, a selection strategy based on one or the other reward function would likely take the same decision, i.e., select operator 2. However, the challenging situation is when the variance σ^2 is in the interval $[a, b]$; since, according to the mean reward function, operator 1 (resp. 2) is better (resp. worst), but according to the maximum reward function, operator 1 (resp. 2) is worst (resp. better). In this case, it is not clear that two selection strategies following the mean and the maximum reward function would select the same operator.

Additionally, the mean reward value does not depend on the number of workers. In fact, increasing the number of local rewards computed by the workers simply reduces the confidence interval around the global mean reward value. In contrast, the maximum reward value increases logarithmically with the number of workers. This is shown in Fig. 3 where three representative examples are considered ($\sigma^2 = 3 \times 10^{-4} < a$, $\sigma^2 = 5 \times 10^{-4} \in [a, b]$, and $\sigma^2 = 7 \times 10^{-4} > b$). When the variance is small or large, the number of workers does not change the rank of each operator with respect to the maximum reward value. However, when the variance σ^2 is between a and b, the best operator according to the maximum reward changes: for low number of workers, operator 1 has a highest maximum reward, whereas operator 2 is preferred when $n \geq 30$.

These empirical observations explain why the maximum reward was found to clearly outperform the mean reward (Sec. 4.1, Fig. 1), since the variance of the second operator was set to a the value $5 \times 10^{-4} \in [a, b]$ which corresponds to a challenging scenario for adaptive selection. In fact, the mean reward can only measure the expected quality of an operator when executed locally and independently by each individual worker, whereas the maximum reward measures the expected quality of the next solution that would be obtained more globally by the cooperative master-worker system in one round. In this sense, an accurate *distributed* selection strategy has to acquire information about the quality of an operator when executed cooperatively by all the entities of the system, and not

only on the quality of one operator taken independently of the distributed and cooperative environment where it can be executed. Hence, the maximum reward aggregation has to be preferred when the goal of the adaptive master-worker algorithm is to increase as much as possible the fitness value in each round of computation which is typically the case of a $(1+\lambda)$-EA. More generally, it should be possible to extend this kind of results for others adaptive M/W algorithms which are less explorative, i.e., the global reward should then take explicitly into account an additional diversity measure.

4.3 Analysis of the Heterogeneity Scenarios

The impact of selecting and assigning workers different operators can also be studied as a function of the relative variance of the portfolio operators. In the following, we only consider the maximum reward strategy since it was proved to perform better. For the sake of analysis, let us consider the fitness cloud benchmark where operator 1 follows $\mathcal{N}(-10^{-4}, 10^{-4})$, and operator 2 follows $\mathcal{N}(-10^{-3}, \sigma^2)$. By varying σ, we compute the maximum global reward, *i.e.* the expected improvement of one round of the M/W algorithm when using $n = 256$ workers, in a heterogeneous setting that would split the workers into those that execute operator 1 and those that execute operator 2. By varying the proportion of heterogeneous workers we are then able to compute the optimal number n_1 of workers which should executes operator 1 (the $n - n_1$ executing operator 2) as a function of operator 2 variance σ^2. More precisely, for each value $n_1 \in [0, n]$, the average of the maximum reward on 1000 independent rounds is computed, and the value n_1 with the highest maximum reward is selected and reported in Fig. 4. Clearly, for a wide range of σ values, the optimal value n_1 is either 256, or $n_1 = 0$ for large variance. This indicates that a homogenous setting (with only operator 1 or 2) is optimal except for a small range of variance (between 3.4×10^{-4} and 4.4×10^{-4}). Moreover when an heterogenous strategy is optimal, the gain of maximum reward with an homogeneous strategy is very small (cf. Fig. 4 right). Given these observations we can know understand better the relative performance observed for the different strategies in Fig. 1 for which $\sigma = 5 \times 10^{-4}$.

For the baseline random strategy, the heterogeneous setting is clearly better; since because of the elitism of a $(1 + \lambda)$-EA, it is better to select the wrong operator for half of the workers than one over two rounds. Notice that the baseline heterogeneous random strategy is never better than any others adaptive strategies when using the maximum reward. The homogeneous version of the ϵ-greedy strategy based on the maximum reward significantly outperforms the heterogenous version according to the Mann-Whitney test at level 5%. In contrast, the heterogenous AP outperforms the homogeneous one. Nevertheless, the best strategy is UCB which is homogeneous. According to the exploration power of the strategy, the heterogeneity could help to select the relevant operator; but, when the selection strategy is able to detect the best operator, and when the relative expected gain in fitness improvement is small, a homogeneous setting is to be preferred.

21
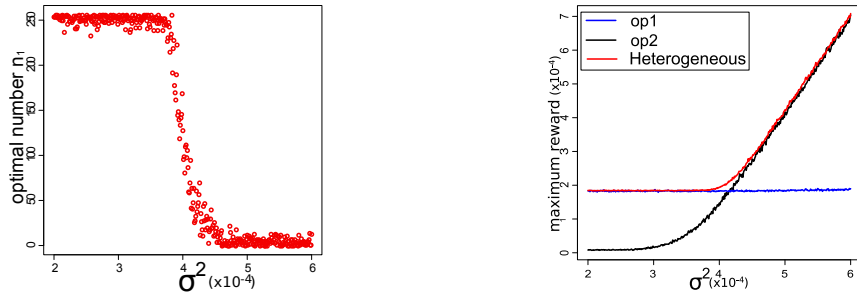
Fig. 4. Optimal number of workers n_1 with operator 1 which maximizes the maximum reward value for an heterogeneous strategy as a function of variance parameter σ^2. The operator 1 and 2 follow respectively the normal law $\mathcal{N}(-10^{-4}, 10^{-4})$, and $\mathcal{N}(-10^{-3}, \sigma^2)$ for $n = 256$ workers. Left: optimal number n_1. Right: Maximum reward values for homogeneous strategies with operator 1 and operator 2, and for the optimal heterogeneous strategy.

5   Conclusions

We conducted an in-depth analysis of the design components of a synchronous M/W adaptive algorithm selection framework. Our main findings can be summarized as follows. The reward associated to each algorithm, which gives the feedback measure for the adaptive selection method, must take into account the performance of the global system, and not only the local performance of each worker. Except when all algorithms have very close performance, an optimal set of algorithms is homogeneous. However, with respect to a particular adaptive strategy, a heterogeneous set could be helpful to continuously enhance its corresponding exploration level. At last, adaptive algorithm selection strategies can be highly effective when their design components in a master-worker architecture are well tuned.

Besides, this first work shall allow us to extend our results for expensive real-world problems, where the evaluation of the fitness function is typically based on computing intensive simulations, e.g., [1]. Another interesting question is the design of reward functions for the asynchronous M/W communication mode. Since a global snapshot of the distributed system is difficult to acquire by the master in such a setting, the reward function is expected to be critically important depending on the different communication to computation trade-offs faced by the master. It is our hope that the new insights provided by our fundamental analysis in the synchronous setting will help addressing such challenging issues.

References

1. R. Armas, H. Aguirre, S. Zapotecas-Martínez, and K. Tanaka. Traffic signal optimization: Minimizing travel time and fuel consumption. In *EA 2015*, pages 29–43, 2016.
2. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

3. P. Baudiš and P. Pošík. Online black-box algorithm portfolios for continuous optimization. In *PPSN XIII*, pages 40–49. Springer, 2014.
4. C. Candan, A. Goëffon, F. Lardeux, and F. Saubion. Non stationary operator selection with island models. In *GECCO*, pages 1509–1516, 2013.
5. L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *GECCO*, page 913. ACM Press, 2008.
6. D. Dasgupta and Z. Michalewicz. *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013.
7. B. Derbel and S. Verel. DAMS: distributed adaptive metaheuristic selection. In *GECCO*, pages 1955–1962. ACM Press, 2011.
8. M. Dubreuil, C. Gagne, and M. Parizeau. Analysis of a master-slave architecture for distributed evolutionary computations. *IEEE T. on Systems, Man, and Cybernetics: Part B*, 36:229–235, 2006.
9. A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 19–46. Springer, 2007.
10. A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In *LION'09*, volume 5851, pages 176–190. Springer, 2009.
11. A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *AMAI*, 60:25–64, 2010.
12. M. García-Valdez, L. Trujillo, J. J. Merelo-Guérvos, and F. Fernández-de Vega. Randomized parameter settings for heterogeneous workers in a pool-based evolutionary algorithm. In *PPSN XIII*, pages 702–710. Springer, 2014.
13. A. Goëffon, F. Lardeux, and F. Saubion. Simulating non-stationary operators in search algorithms. *Appl. Soft Comput.*, 38:257–268, 2016.
14. J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122–128, 1986.
15. C. Jankee, S. Verel, B. Derbel, and C. Fonlupt. Distributed Adaptive Metaheuristic Selection: Comparisons of Selection Strategies. In *EA 2015*, pages 83–96, 2015.
16. C. Jankee, S. Verel, B. Derbel, and C. Fonlupt. A fitness cloud model for adaptive metaheuristic selection methods. In *PPSN 2016*, pages 80–90. Springer, 2016.
17. L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, pages 48–60, 2012-10-30.
18. J. Maturana, Á. Fialho, F. Saubion, M. Schoenauer, and M. Sebag. Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In *CEC 2009*, pages 365–372. IEEE, 2009.
19. M. Muniglia, J.-M. Do, L. P. Jean-Charles, H. Grard, S. Verel, and S. David. A Multi-Physics PWR Model for the Load Following. In *ICAPP*, Apr. 2016.
20. D. Sudholt. Parallel evolutionary algorithms. In *Springer Handbook of Computational Intelligence*, pages 929–959, Berlin, Heidelberg, 2015. Springer.
21. R. Tanabe and A. Fukunaga. Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms. In *CEC*, pages 1263–1270, 2013.
22. D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *GECCO'05*, pages 1539–1546, 2005.
23. S. Wessing, G. Rudolph, and D. A. Menges. Comparing asynchronous and synchronous parallelization of the sms-emoa. In *PPSN XIV*, pages 558–567, 2016.
24. M. Yagoubi and M. Schoenauer. Asynchronous master/slave moeas and heterogeneous evaluation costs. In *GECCO*, pages 1007–1014, 2012.

Comparison of Acceptance Criteria in Randomized Local Searches

Alberto Franzin and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium
`alberto.franzin@ulb.ac.be`, `stuetzle@ulb.ac.be`

Abstract. One key component of stochastic local search algorithms is the acceptance criterion that determines whether a solution is accepted as the new current solution or it is discarded. One of the most studied local search algorithms is simulated annealing. It often uses the Metropolis condition as acceptance criterion, which always accepts equal or better quality solutions and worse ones with a probability that depends on the amount of worsening and a parameter called temperature. After the introduction of simulated annealing several other acceptance criteria have been introduced to replace the Metropolis condition, some being claimed to be simpler and better performing. In this article, we evaluate various such acceptance criteria from an experimental perspective. We first tune the numerical parameters of the algorithms using automatic algorithm configuration techniques for two test problems, the quadratic assignment problem and a permutation flowshop problem. Our experimental results show that, while results may differ depending on the specific problem, the Metropolis condition and the late acceptance hill climbing rule are among the choices that obtain the best results.

1 Introduction

Stochastic local search (SLS) methods are generic procedures commonly used to tackle hard optimization problems [9]. They are composed of a set of general rules of how to design effective heuristics for specific optimization problems; hence, an alternative name for these methods is *meta*-heuristics. Often, the sometimes rather problem-specific heuristic algorithms derived from these rules are very effective in finding high quality solutions in short computation time, and for many problems such algorithms define the state of the art.

To achieve good solutions, SLS methods balance the intensification of the search in narrow regions, often needed to find the best solutions in promising search space areas, with the exploration of different areas of the search space. One mechanism that many trajectory-based SLS methods use to promote diversification is the acceptance of solutions that are worse than the current incumbent solution. In this article, we call *acceptance criterion* the function devoted to determining whether a newly proposed candidate solution should replace the current one. A first metaheuristic that proposed a probabilistic acceptance criterion for acceptinf a worse candidate solution is simulated annealing (SA) [10,23].

It uses the so called Metropolis condition from statistical mechanics, which relies on a parameter called temperature, as acceptance criterion [14]. The name of the parameter mimics the temperature of a physical system, which was used in a Monte-Carlo simulation of physical systems proposed by Metropolis *et al.* [14]. According to the Metropolis condition, an improving or equal quality candidate solution is always accepted, while a worsening candidate solution is accepted with a probability that depends both on the quality difference between the current solution and the newly proposed one and the temperature parameter. To create a transition from search diversification to intensification of the search, in a typical SA algorithm the temperature is initially set to some high value (corresponding to a rather likely acceptance of worsening candidate solutions) and then subsequently lowered to make the acceptance of worsening candidate solutions less likely.

Over the years, various new ideas have been conceived with the motivation to improve over this usual acceptance criterion of SA algorithms. These new acceptance criteria have been compared in individual papers often directly to basic SA algorithms and in various such papers potential improvements have been reported. These new ideas include refinements of the Metropolis condition, such as generalized SA [2] and the bounded Metropolis condition [6]; a criterion where the acceptance probability of worsening solutions decreases geometrically [17]; and deterministic criteria such as threshold acceptance [8,15], the great deluge and record-to-record travel algorithms [7], and the late acceptance hill climbing [5]. The latter four methods all accept a solution with probability one when it meets the specific, deterministic acceptance conditions. In our experiments, we also include a basic hill climbing acceptance criterion [1], which accepts a solution if and only if it improves over the incumbent, as a baseline the other criteria need to outperform.

The original articles proposing these acceptance criteria often report experimental results on few problem instances or on very small instance sizes. One reason is that many of these acceptance criteria were introduced when experimental conditions available were quite different from today. Hence, there is limited indication in the original works on how to apply the various methods to different problems. To just cite one example, in the original paper on threshold acceptance, the authors present a sequence of values for the "threshold" parameter, stating that *"We have the feeling (really only the feeling, not, for instance, the impression) that the sequence above is somewhat better* [than another sequence mentioned]" [8]. The comparisons in these papers are also usually performed against the Metropolis condition and a limited set of the other criteria.

In this work, we compare well-known acceptance criteria on common benchmark sets, derived from two classical, NP-hard problems, namely the quadratic assignment problem (QAP) and the permutation flow-shop problem with the total completion time objective (PFSP-TCT). To obtain unbiased results we tune the numerical parameters of the algorithms, using the automatic algorithm configuration tool irace [11]. We evaluate the impact of the nine different criteria we study in terms of the quality of the final solutions and the robustness

of the criterion. Our experiments show that the results may change according to different problems, instance classes, or experimental condition. Overall, the Metropolis condition, its generalized version and, in particular, the rather recent late acceptance hill climbing are the criteria that gave the best results.

2   Literature review

We first introduce the notation used in the remainder of this work. We consider NP-hard combinatorial optimization problems, in which for a given problem instance π a globally optimal solution $s^* \in S$, where S is the search space of candidate solutions, is to be found. The quality of solutions is evaluated according to an objective function $f : S \mapsto \mathbb{R}$ and $f(s)$ is the objective function value for a generic solution s. Without loss of generality, we consider minimization problems, that is, for a globally optimal solution it holds that $f(s^*) \leq f(s), \forall s \in S$. Each algorithm we consider uses an iteration counter of the search process, which is denoted by i, and s_i is the new candidate solution evaluated in that iteration. The difference in terms of objective function value between two solutions s_i and s_j is denoted with $\Delta(i, j)$, or simply Δ when no confusion may arise. With $\hat{s}$ we indicate the incumbent solution. The neighbourhood of $\hat{s}$ is denoted by $\mathcal{N}(\hat{s})$ and comprises all candidate solutions that can be reached from s by one application of the neighborhood operator.

Algorithm 1: Outline of a generic randomized search algorithm.

   Input: problem instance Π, $\mathcal{N}$, initial solution s_0, control parameters
   Output: best solution s^*
1 best solution $s^* =$ incumbent solution $\hat{s} = s_0$;
2 parameter initialization, $i := 0$;
3 **while** *stopping criterion is not met* **do**
4   **while** *parameters settings fixed* **do**
5 $i := i + 1$;
6   generate a random solution $s_i \in \mathcal{N}(\hat{s})$;
7 $\hat{s} :=$ accept $(\hat{s}, s_i)$;
8 $s^* :=$ best $(s^*, \hat{s})$;
9   **end**
10   update parameters;
11 **end**
12 return s^*;

All SLS methods that we consider can be interpreted as instantiations of the generic algorithm outlined in Algorithm 1. It starts from a given initial solution as incumbent (line 1), and iteratively generates one new candidate solution in the neighbourhood of the incumbent uniformly at random (line 6); at iteration i the new candidate solution s_i can be chosen to replace the current incumbent $\hat{s}$

if it meets some criteria (e.g. it is an improving solution, line 7), otherwise it is discarded. Periodically, the parameter(s) that control the search may get updated (line 10). All the algorithms we examine here fit in this generic template. They only differ in the acceptance criterion. However, some of these algorithms may not use all the components of the algorithm; for example, the late acceptance hill climbing relies only on one parameter that is held constant during the run of the algorithm and, hence, does not need to be updated in the outer loop (lines 3 to 11). In the following, the counter k refers to the number of times the outer loop has been invoked. Conversely, SA and others evaluate solutions using the same parameter values in the inner loop (controlled by the temperature length, lines 4 to 9), and update the parameters in the outer loop.

SA, proposed independently in [10] and [23], is inspired by work in statistical physics [14]. In the usual, basic variants, SA iteratively generates and evaluates one random solution $s \in \mathcal{N}(\hat{s})$; if the new solution is better or equal to the incumbent in terms of objective function value, it replaces the incumbent one; otherwise it gets accepted with a probability that depends on the relative difference in terms of objective function values, $\Delta(s, \hat{s})$, and on the temperature parameter, denoted as T. The acceptance criterion of SA can be written as

$$p = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq 0 \\ \exp\left(-\Delta(s, \hat{s})/T\right) & \text{otherwise.} \end{cases} \tag{1}$$

This probabilistic criterion is known as Metropolis acceptance criterion or Metropolis condition, and it is the distinctive feature of SA. We refer to this criterion simply as SA in the rest of this paper.

More recently, in [6] the authors argue that solutions that are worse with respect to the incumbent by a quantity that exceeds a certain threshold ϕ_{BM} are not worth considering at all. This *bounded* Metropolis criterion (BSA) accepts a solution s with a probability

$$p = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq 0 \\ \exp\left(-\Delta(s, \hat{s})/T\right) & \text{if} \quad 0 < \Delta(s, \hat{s}) \leq \phi_{BM} \\ 0 & \text{if} \quad \Delta(s, \hat{s}) > \phi_{BM}, \end{cases} \tag{2}$$

where ϕ_{BM} is a parameter.

Soon after the introduction of SA, the Metropolis acceptance criterion has been generalized in [2], where a variant of SA called generalized simulated annealing (GSA) was introduced. The GSA acceptance criterion is defined as

$$p = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq 0 \\ \exp\left(-\beta f(\hat{s})^{\gamma} \Delta(s, \hat{s})\right) & \text{otherwise,} \end{cases} \tag{3}$$

where β and γ are control parameters. Even if the temperature parameter is not explicitly considered in GSA, it is possible to recreate the original Metropolis condition by defining $\beta = 1/T$.

In [17], the authors propose a criterion in what is the first occurrence of a SA variant that does not consider the temperature value in the acceptance of solutions. They propose to accept a solution with probability

$$p^k = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq 0 \\ p_0 \times \rho^{k-1} & \text{otherwise.} \end{cases} \tag{4}$$

where p_0 is the initial acceptance probability, $0 < \rho < 1$ is a reducing factor, and k is the number of times the probability has been updated. In this *geometric* acceptance criterion, the temperature value is (possibly) related only to the initial acceptance probability; during the search, the updating process of the probability matters, rather than the actual value of a temperature.

The actual need of stochasticity in the Metropolis acceptance criterion is questioned independently in [8] and [15]. In both works, the authors propose a criterion that accepts any move that is either improving or worsening by at most a given threshold $\phi^k > 0$:

$$p = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq \phi^k \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where ϕ^k is the value at step k of the threshold, which gets updated periodically. In [8], the authors consider a sequence of thresholds, without giving any indication on how to set its initial value or how to update it. Our implementation follows [15], maintaining the SA terminology: the initial value of ϕ is the initial temperature of SA, and the updating process of the threshold is called cooling. This threshold acceptance (TA) is a deterministic version of SA. At the time of its introduction, it was argued that using TA is faster than evaluating the Metropolis condition as it does not require the generation of a random number and the computation of an exponential. This advantage may be important when the computation of the objective function value of a neighboring candidate solution is very fast. However, for problems that benefit little from incremental update schemes or where the computation of the objective function value of neighboring candidate solutions is expensive (as is the case in the problems we study here), the advantage of a faster computation of the acceptance test diminishes.

Two acceptance criteria have been derived from TA and proposed in [7] as new algorithms. The first algorithm and criterion proposed in [7] is called record-to-record travel (RTR), and accepts solutions that do not deviate from the best solution found so far plus a given threshold ϕ:

$$p_{RTR} = \begin{cases} 1 & \text{if} \quad f(s) \leq f(s^*) + \phi \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

RTR is therefore a stricter version of TA, which compares the newly proposed candidate solution with the current incumbent; moreover, in the RTR algorithm ϕ does not get updated.

The second algorithm proposed in [7] is called great deluge algorithm (GDA) and is a radical change in terms of solution evaluation, as it moves away from the idea of comparing solutions. The acceptance criterion of GDA accepts every move whose objective function value is lower than a certain threshold that gets progressively lowered during the search

$$p^k = \begin{cases} 1 & \text{if} \quad f(s) \le \phi^k \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

with $\bar{\phi}^{k+1} = \phi^k - \lambda$, λ being a fixed parameter. The consequence of a lowering bound is that GDA becomes increasingly strict for accepting solutions.

A more recent work proposes another simple deterministic acceptance criterion, called late acceptance hill climbing (LAHC) [4,5]. This algorithm makes no use of a temperature-like parameter, but maintains limited knowledge about the history of the search. It accepts every solution s that is improving either with respect to the current incumbent $\hat{s}$ or with respect to a solution visited κ iterations before, for a fixed κ:

$$p = \begin{cases} 1 & \text{if} \quad f(s) \le \min\{f(\hat{s}), f(s_{i-\kappa})\} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Finally, we consider as baseline for the comparison a simple hill climbing (HC) algorithm [1] that accepts a solution if and only if it improves over the incumbent. Obviously, we expect the other criteria to obtain better results with respect to HC. While in practice one would implement HC using a systematic enumeration of the neighbourhood, we implemented it inside the framework of Algorithm 1 for convenience.

3   Experimental Setup

The nine acceptance criteria presented in Section 2 are evaluated as candidate acceptance criteria for a generic algorithm outlined in Algorithm 1. The common components of the nine implementations are: (i) a random exploration of the neighbourhood, (ii) no parameter restarting rule (e.g. temperature restart in SA), and (iii) a termination condition based on runtime. The runtime differs for each problem, so the actual value is given below.

For the criteria that need initial values for their parameters (such as the temperature for the SA family of algorithms, or the threshold ϕ in TA), we use a value proportional by a coefficient ϵ to the maximum gap between consecutive solutions observed during an initial random walk of length 10000 in the solution space. The parameters that need to be modified during the algorithm run time (e.g. temperature in SA or threshold in TA) are updated using a geometric decreasing; e.g., the temperature T in SA is updated according to the formula $T_{k+1} = \alpha \times T_k$, where α is a parameter. The inner loop of Algorithm 1 evaluates a number of solutions that is given by $\tau \cdot |\mathcal{N}(s)|$, where τ is a parameter and $|\mathcal{N}(s)|$ is the size of the neighbourhood of a solution s.

Table 1. Parameter values for the algorithms.

	Metro	BMetro	GSA	Geom	TA	GDA	RTR	LAHC
ϵ	$[0, 10]$	$[0, 10]$	$[0, 10]$	$[0, 10]$	$[0, 10]$	$[0, 10]$	–	–
α	$[0, 1]$	$[0, 1]$	$[0, 1]$	$[0, 1]$	$[0, 1]$	$[1, 100]$	–	–
τ	$[1, 100]$	$[1, 100]$	$[1, 100]$	$[1, 100]$	$[1, 100]$	$[1, 100]$	–	–
ϕ, ϕ_{BM}	–	$[0, 1]$	–	–	–	–	$[0, 1]$	–
β	–	–	$[10^{-4}, 10]$	–	–	–	–	–
γ	–	–	$[0, 10]$	–	–	–	–	–
ρ	–	–	–	$[0, 1]$	–	–	–	–
κ	–	–	–	–	–	–	–	$[1, 10^4]$

We choose to not use parameter restart schemes to better observe the impact of the main algorithm component that is studied, the acceptance criterion. The periodic reset or increase of parameters such as the temperature or the threshold is often beneficial to obtain better results, as it facilitates search space exploration, but it also has the side effect of smoothening the difference in terms of impact of the other components.

The parameter values, and their presence for each algorithm, are given in Table 1. Parameters equivalent in scope and values are grouped together. The only algorithm that does not use the components described above is GDA. During the experimental phase, we have observed very poor results when using the GDA acceptance criterion with the choices above, indicating a lack of flexibility of the method. We thus consider the GDA algorithm in its original settings, which are anyway valid components that fit in the template of Algorithm 1. The initial threshold value ϕ is computed proportional to the objective function value of the initial solution, using a coefficient $\epsilon \in [0, 10]$; ϕ is updated according to the formula $\phi_{k+1} = \phi_k - \alpha$, where α is an integer in the interval $[1, 100]$; we bound this decrease to 0. The other components are as described above.

Our setup considers as test problems the quadratic assignment problem (QAP) [3] and the permutation flow-shop problem with the total completion time objective (PFSP-TCT) [19,18]. The QAP models the location of a set of facilities, with the goal of minimizing the overall distance between facilities taking into account also the flow between them. PFSP instead is a scheduling problem where a set of jobs have to be ordered to be executed on a set of machines.

For the QAP we use a randomly generated initial candidate solution and the exchange neighbourhood, which is defined as

$$\mathcal{N}(s) = \{s' \mid s'(j) = s(h) \wedge s'(h) = s(j) \wedge \forall : l \notin \{j, h\} \, s'(l) = s(l)\}, \quad (9)$$

where $s(j)$ is the solution vector at position j. The neighbourhood size is $n(n - 1)/2$, where n is the instance size. The running time considered for termination is 10 seconds. We consider two different instance sets of size 100, one where all QAP instance data are generated uniformly at random, and one randomly generated in analogy to structured real-world like QAP instances. Each instance set is divided into a training set of 25 instances and a test set of 25 instances.

From here onwards, we refer to these two scenarios as random instances and structured instances, respectively. The two scenarios are not mixed, that is, the configurations obtained for the random instances are evaluated on the random instances and not on the structured ones, and viceversa.

For the PFSP-TCT we use the `NEH` heuristic [16] for the initial solution generation. For an instance of size $n \times m$, where n is the number of jobs and m the number of machines, the neighbourhood is the insert neighbourhood that randomly picks one element $s(j)$ in position j of the permutation $s = [s(1), \ldots, s(n)]$ and inserts it in position $k \neq j$, obtaining

$$s' = [s(1), \ldots, s(j-1), s(j+1), \ldots, s(k), s(j), s(k+1), \ldots, s(n)] \qquad (10)$$

if $j < k$ and

$$s' = [s(1), \ldots, s(k-1), s(j), s(k), s(k+1), \ldots, s(j-1), s(j+1), \ldots, s(n)] \quad (11)$$

if $j > k$. The neighbourhood size is $n(n-1)$. In this case, we use an instance-based maximum runtime of $n \times m \times 0.015$ seconds. The training set consists of 40 randomly generated instances of size ranging from 50 jobs and 20 machines to 250 jobs and 50 machines [13] and the test set is composed by the instances `Tai31-110` of the Taillard benchmark [21]. We will, however, discuss separately the instances whose size is smaller than those covered by the training set (those with $n = 20$), covered by the training set (`Tai31-110`), and larger ($n = 500$).

We tune the numerical parameters using irace [11] with a budget of 2000 experiments per tuning on an Intel Xeon E5-2680 v3 CPU, with a speed of 2.5GHz, 16MB cache and 2.4GB of RAM available for each job. For each algorithm we run nine tunings, evaluate the best configuration obtained from each tuning on the test set, and average the final solution quality obtained on each instance by the nine configurations. The real valued parameters have a precision of 4 decimal digits.

4   Experiments on the Quadratic Assignment Problem

In Figure 1 we show the results obtained by the nine algorithms after the tuning on the random instances and on the structured instances respectively. Each boxplot reports the results obtained on the test instances in terms of the average relative percentage deviation (ARPD) from the best known solutions. In Table 2, we report the results of the Friedman rank sum test, obtained for the nine algorithms on the two QAP instance classes. The algorithms are ordered according to the sum of their ranks, and the difference in terms of rank sum with the best ranked algorithm is computed along with a statistical significance threshold. Algorithms whose rank sum differs from the best ranked one by a value larger than the significance threshold are statistically significantly worse than the best one.

On the random instances, RTR obtains the best results, with a mean ARPD slightly lower than 1%. The criteria based on the Metropolis condition (SA,
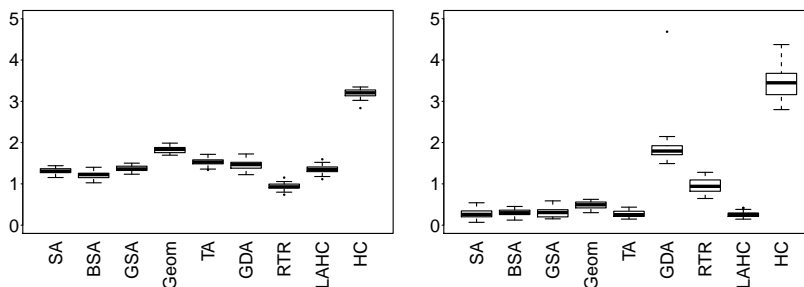
Fig. 1. Average Relative Percentage Deviation (ARPD) from the best known solutions obtained on random (left plot) and structured instances (right plot).

BSA, GSA) and the LAHC algorithm obtain similar results, with mean ARPDs around 1.2 to 1.3%. Though the results are similar, BSA is consistently slightly better than the other ones. TA, GDA and the geometric criteria are worse, but still within the 2% average deviation, while HC stands around 3%. On the structured instances it is instead TA, LAHC and the family of the Metropolis criteria that obtain the best results, with average ARPDs all around 0.3%. The ARPDs among these five criteria are not statistically significantly different. The geometric criterion also obtains reasonably good results when considering the ARPD values, though from the rank-based analysis it is already clearly worse than the top-ranking group of acceptance criteria. RTR and GDA obtain solutions around 1% and 2% worse than the best known ones and, thus perform clearly worse than the other acceptance criteria. HC, as expected, is overall the worst, with ARPDs around 3 to 4%.

The difference of the results on the two scenarios can be explained by the different landscape of the instances [22,20]. The random instances present a relatively flat landscape, where it is easy to discover local optima and move through them, but difficult to converge to very good solutions. On the other hand, the landscape of the structured instances is less flat, with "deeper" local optima than in the random instances. The criteria that strengthen the intensification along the search process are the ones that apparently benefit from this landscape. RTR compares candidate solutions to the global best, making it therefore more difficult to accept a worsening solution; additionally, using a same parameter settings across all instances may make it less robust.

5   Experiments on the Permutation Flow-Shop Problem

In Figure 2 we report the results obtained on the PFSP-TCT on the 80 instances of the Taillard benchmark with 50 to 200 jobs. The results of the Friedman rank sum test for the nine algorithms are reported in Table 3, separated for the

Table 2. Results of the Friedman rank sum test for the nine algorithms on the QAP instances. Algorithms are ranked according to their results. Δ_R is the minimum rank-sum difference that indicates significant difference from the best one. Algorithms in boldface are significantly better than the following ones.

Instance class	Δ_R	Acceptance criteria ranking
Random	13.15	**RTR** (0), BSA (33), SA (70), LAHC (80), GSA (88),
		GDA (115), TA (140), Geom (174), HC (200)
Structured	16.08	**TA** (0), **LAHC** (0), **SA** (11), **BSA** (16), GSA (21),
		Geom (81), RTR (109), GDA (135), HC (158)
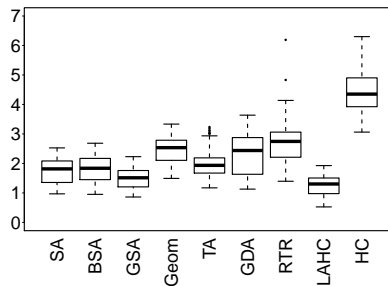
Fig. 2. Average Relative Percentage Deviation (ARPD) from the best known solutions obtained on the instances `Ta031-110` of the Taillard benchmark.

three sets of instance subclasses considered (smaller than in the training set, size covered by the training set, and larger).

The results in Figure 2 for the PFSP-TCT exhibit higher variance than on the QAP, because they report results obtained on 8 subclasses of instances, with a different number of jobs and machines. Inside each instance subclass, the variance is much lower, indicating consistent results for each instance size.

Late acceptance hill climbing is the criterion that obtains clearly the best results, with an average ARPD of 1.2%. It is also more robust than the others: its worst results are below 2% of ARPD. GSA comes second best, with an average deviation of 1.5%, followed by SA and BSA (respectively 1.7% and 1.8% on average; a Wilcoxon test shows no statistically significant difference between them). TA obtains results comparable to BSA. The other criteria obtain results between 2% and 3% of ARDP, still significantly better than HC.

The different instance sizes in both the training and test sets favour the more robust solutions. GSA appears to be more robust than the original SA, thanks to the increased flexibility given by the additional parameters. Looking at the different instance subclasses, however, LAHC consistently outperforms all other acceptance criteria.

Table 3. Results of the Friedman rank sum test for the nine algorithms on the Taillard Benchmark. Algorithms are ranked according to their results. Δ_R is the minimum rank-sum difference that indicates significant difference from the best one. Algorithms in boldface are significantly better than the following ones.

Instance class	Δ_R	Acceptance criteria ranking
Ta001-030	13.57	**LAHC** (0), GSA (30), TA (74), SA (82), BSA (87), Geom (142), RTR (193), GDA (206), HC (212)
Ta031-110	27.58	**LAHC** (0), GSA (94), SA (229), TA (267), BSA (279), GDA (412), Geom (455), RTR (490), HC (636)
Ta111-120	2.93	**LAHC** (0), GSA (11), RTR (19), GDA (31), HC (39), Geom (50), TA (63), SA (67), BSA (80)
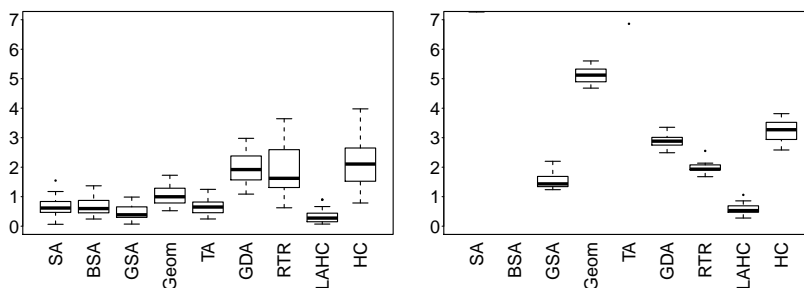
Fig. 3. Average Relative Percentage Deviation (ARPD) from the best known solutions obtained on the `Ta001-030` (left plot) and on the `Tai111-120` instances of the Taillard benchmark set (right plot, SA, BSA, and in part TA results are not shown as they were very poor).

We focus now on the instance subclasses not covered by the training set, either because they are too small (`Ta001-030`) or because they are too big `Tai111-120`. We can observe in Figure 3 and in Table 3 that LAHC is consistently the best performing one, followed by GSA. Overall, on the small instances all algorithms obtain results that are according to the ARPD values at least as good as on medium size instances of `Ta031-110`, with GDA and RTR being the only ones for which this is not true.

On the large instances, LAHC and GSA remain the top-performing algorithms with an average ARPD of 0.59% and 1.57%, respectively. SA and BSA instead obtain good results on the small instances, but perform very poorly on the larger ones, with ARPDs ranging around $9-10\%$, much worse than even HC. This effect is due to the parameters selected by the tuning phase, which are calibrated for instance sizes occurring in the training set and the given running time. The convergence behaviour of SA and BSA does not scale to large instance sizes for which the evaluation per solution is much more costly (the evaluation
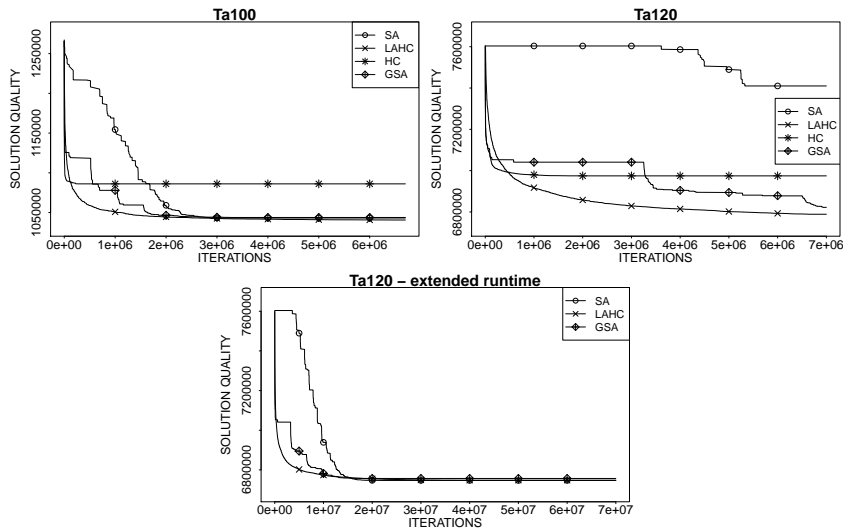
Fig. 4. Convergence behaviour of the SA, GSA, LAHC and HC algorithms for the `Ta100` (top left plot) and `Ta120` (top right plot) instances; in the bottom plot, the results for SA, GSA and LAHC on `Ta120` with 10× the original running time.

scales quadratically with instance size while the computation time only increases linearly). This is illustrated in Figure 4, where we compare the development of the solution quality over the number of iterations for SA, GSA, LAHC and HC on two instances: `Ta100`, whose size is 200×10 and is covered by the training set, and `Ta120`, whose size is 500×20. On `Ta100`, the four algorithms quickly discover good solutions; still, the convergence of SA is slower with respect to the other three. On `Ta120`, SA is clearly unable to converge within the originally allocated computation time. In the right plot of Figure 4 we observe the convergence of SA, GSA and LAHC on `Ta120` with a runtime ten times higher (1500s instead of 150s on that instance – HC not included in the plot): the convergence is more similar to the one observed for `Ta100`, with also SA discovering high quality solutions. In particular, GSA finds a solution very close to the best known one (6756860 vs 6755722), while SA and LAHC both find a solution of better quality than the currently best known one (6746818 and 6748131, respectively). It is interesting to note that SA has now found the best solution, while LAHC has continued improving until more than half the time available.

6   Conclusions

We have observed how a careful tuning of the numerical parameters is crucial to obtain good results, in terms of both solution quality and convergence. Across our

two benchmark problems, the algorithm that obtained the best results is LAHC. It exhibits a good convergence behaviour, quickly discovering good solutions in the beginning of the search, and continuously improving afterwards. It is also very robust, as it is the best performing algorithm across the whole Taillard benchmark for the PFSP-TCT, and it scales well also to instances of different sizes, unseen in the training set. Despite its simplicity (only one parameter to be tuned), LAHC makes a good use of the history of the search, as any solution it accepts is never worse than at least another one it has accepted in the past.

SA obtains overall good results, but it requires a proper tuning, as we have observed, in particular, for the large PFSP instances. It is able to obtain good results, but it might do so slowly; it is therefore advisable to tune SA for anytime behaviour [12] to obtain good results in a shorter time. BSA performs similarly to the standard SA. GSA, instead, has been shown to be flexible, outperforming SA also in terms of scalability and anytime behaviour. The geometric acceptance criterion is overall inferior to tthose derived from the original SA.

TA has obtained results overall not very different from SA. RTR has obtained good results on the random QAP instances, but was among the worse performers in the other scenarios, probably because of the fixed value of its threshold. GDA also showed a lack of flexibility, requiring a different setup and thus making its use within other algorithms more problematic.

As future work, we plan to extend the analysis to different conditions that might improve the performance of the various criteria. For example, a temperature restart, which is a common option in various SA algorithms, may change the conclusions of especially those criteria that rely on the temperature parameter. In addition, we plan to extend the set of acceptance criteria that are considered in this work and also extend the set of test problems to increase the experimental basis on which our conclusions rely. Finally, we intend to test the various acceptance criteria considering other aspects such as anytime behavior or robustness to other scenarios that differ in instance size and termination condition.

References

1. Appleby, J., Blake, D., Newman, E.: Techniques for producing school timetables on a computer and their application to other scheduling problems. The Computer Journal 3(4), 237–245 (1961)
2. Bohachevsky, I.O., Johnson, M.E., Stein, M.L.: Generalized simulated annealing for function optimization. Technometrics 28(3), 209–217 (1986)
3. Burkard, R.E., Çela, E., Pardalos, P.M., Pitsoulis, L.S.: The quadratic assignment problem. In: Handbook of Combinatorial Optimization, vol. 2, pp. 241–338. Kluwer Academic Publishers (1998)

4. Burke, E.K., Bykov, Y.: The late acceptance hill-climbing heuristic. Tech. Rep. CSM-192, University of Stirling (2012)
5. Burke, E.K., Bykov, Y.: The late acceptance hill-climbing heuristic. European Journal of Operational Research 258(1), 70–78 (2017)
6. Chen, R.M., Hsieh, F.R.: An exchange local search heuristic based scheme for permutation flow shop problems. Applied Mathematics & Information Sciences 8(1), 209–215 (2014)
7. Dueck, G.: New optimization heuristics: the great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104(1), 86–92 (1993)
8. Dueck, G., Scheuer, T.: Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 90(1), 161–175 (1990)
9. Hoos, H.H., Stützle, T.: Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco, CA (2005)
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
11. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58 (2016)
12. López-Ibáñez, M., Stützle, T.: Automatically improving the anytime behaviour of optimisation algorithms. European Journal of Operational Research 235(3), 569–582 (2014)
13. Mascia, F., López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T.: From grammars to parameters: Automatic iterated greedy design for the permutation flow-shop problem with weighted tardiness. In: Proc. LION 7, LNCS, vol. 7997, pp. 321–334. Springer, Heidelberg, Germany (2013)
14. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21, 1087–1092 (1953)
15. Moscato, P., Fontanari, J.F.: Stochastic versus deterministic update in simulated annealing. Physics Letters A 146(4), 204–208 (1990)
16. Nawaz, M., Enscore, Jr, E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11(1), 91–95 (1983)
17. Ogbu, F.A., Smith, D.K.: The application of the simulated annealing algorithm to the solution of the n/m/C max flowshop problem. Computers & Operations Research 17(3), 243–253 (1990)
18. Pan, Q.K., Ruiz, R.: Local search methods for the flowshop scheduling problem with flowtime minimization. European Journal of Operational Research 222(1), 31–43 (2012)
19. Pan, Q.K., Ruiz, R.: A comprehensive review and evaluation of permutation flow-shop heuristics to minimize flowtime. Computers & Operations Research 40(1), 117–128 (2013)
20. Stützle, T.: Iterated local search for the quadratic assignment problem. European Journal of Operational Research 174(3), 1519–1539 (2006)
21. Taillard, É.D.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64(2), 278–285 (1993)
22. Taillard, É.D.: Comparison of iterative searches for the quadratic assignment problem. Location Science 3(2), 87–105 (1995)
23. Černý, V.: A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications 45(1), 41–51 (1985)

SESSION 2 - FITNESS LANDSCAPES

A fitness landscape view on the tuning of an asynchronous master-worker EA for nuclear reactor design

Mathieu Muniglia[1], Sébastien Verel[2],
Jean-Charles Le Pallec[1], and Jean-Michel Do[1]

[1] CEA (french Commissariat à l'Energie Atomique), France
[2] Université du Littoral Côte d'Opale, LISIC, France

Abstract. In the context of the introduction of intermittent renewable energies, we propose to optimize the main variables of the control rods of a nuclear power plant to improve its capability to load-follow. The design problem is a black-box combinatorial optimization problem with expensive evaluation based on a multi-physics simulator. Therefore, we use a parallel asynchronous master-worker Evolutionary Algorithm scaling up to thousand computing units. One main issue is the tuning of the algorithm parameters. A fitness landscape analysis is conducted on this expensive real-world problem to show that it would be possible to tune the mutation parameters according to the low-cost estimation of the fitness landscape features.

1 Introduction

In the actual context of energetic transition, the increase of the intermittent renewable energies contribution (as wind farms or solar energy) is a major issue. On the one hand, the French government aims at increasing their part up to 30% [6] by 2030, against 6% today. On the other hand, their intermittent production may lead to an important imbalance between production and consumption. Consequently, the other ways of production must adapt to those variations, especially nuclear energy which is the most important in France. The power variations occur at different time scales (hour, day, or even week) and in order to counterbalance their effects on the electric grid, the nuclear power plants (NPP) are able to adjust their production. NPPs which take part in the response of the power variations operate in the so-called load-following mode. In this operating mode, the power plant is mainly controlled using control rods (neutron absorber) that may introduce unacceptable spatial perturbations in the core, especially in case of huge power variations. The purpose of this work is to optimize the manageability of the power plants to cope with a large introduction of intermittent renewable energies. Its final goal is to tune the control parameters (called variables) in order to be able to make the load following at a shorter time scale and larger power amplitude scale, meeting the safety constraints.

Such a real-world optimization problem is a challenge, considering the size of the search domain, the computation cost and the unknown properties of the

39

fitness function. Due to the design of the nuclear power plants, and in a goal to propose only simple modifications of the current management, 11 integer variables are used to describe the control rods such as speed, overlaps between rods, etc. (details are given in Sect. 3). Therefore, the optimization is a large size combinatorial problem where no full enumeration is possible. Moreover, a multi-physic simulator is used which is able to compute several criteria such as the evolution of the axial power offset, the rejected volume of effluent, etc. according to the variables of the problem. So, the computation of the fitness function is computationally expensive, and one evaluation typically takes on average about 40 minutes. This optimization problem is considered for the first time, and no property on the search space is *a priori* known. Hence, in this work, the Nuclear Reactor Operation Optimization problem (NROO problem) is an original combinatorial black-box problem with expensive fitness function evaluation for which only few candidate solutions and their corresponding fitness value can be computed.

The ability of Evolutionary Algorithm (EA) to find high quality solutions is likely to depend strongly on its parameters settings. In this work, we propose a parallel master-worker EA for large scale computing environment to solve the NROO problem. Despite the expensive cost, an analysis of the mutation parameters is then proposed. Such a study is not always possible for expensive optimization problems. Hence, we achieve a fitness landscape analysis of the NROO problem using low-cost features to argue that it helps to select the relevant parameters of the mutation operator.

The main goals of the paper are then : (i) perform for the first time an offline optimization of the control rods using an evolutionary algorithm (ii) analyze the fitness landscape structure using a random walk (details are given in Sect. 5.3) to tune the algorithm parameters, especially the ones of the mutation, in order to (iii) propose an efficient mono-objective master-worker that will be used in the next step of the work, consisting in a multi-objective optimization using a decomposition approach.

The rest of this paper is organized as follows. The next section introduces previous works on nuclear energy problems, and main definitions used in this work. The NROO problem and the proposed algorithm are described respectively in Sect. 3 and 4. The experimental analysis of the algorithm and of the fitness landscape is conducted in Sect. 5. At last, the paper concludes on the main results, and future works.

2   Preliminaries

2.1   Evolutionary optimization for nuclear energy problems

The use of Evolutionary Algorithms (EA) in order to optimize some variables of a nuclear power plant as regards performance or safety is not new. Offline optimizations can already be found, and studies such as [2] or [4] deal with the In-Core Fuel Management Optimization (ICFMO) and loading pattern optimization

which is a well-known problem of Nuclear Engineering and aims for instance at maximizing the use of the fuel (increase the cycle length for example) while keeping the core safe (minimize the power peak). Pereira and Lapa consider in [16] an optimization problem that consists in adjusting several reactor cell variables, such as dimensions, enrichment and materials, in order to minimize the average peak-factor in a reactor core, considering some safety restrictions. This is extended in [18] to stochastic optimization algorithms conceptually similar to Simulated Annealing. Sacco *et al.* even perform in [19] an optimization of the surveillance tests policy on a part of the secondary system of a Nuclear Power Plant, using a metaheuristic algorithm, which goal is to maximize the system average availability for a given period of time.

To our best knowledge, the only optimizations of the plant operation are made online, like in [15], where Na *et al.* develop a fuzzy model predictive control (MPC) method to design an automatic controller for thermal power control in pressurized water reactors. The objectives are to minimize both the difference between the predicted reactor power and the desired one, and the variation of the control rod positions. A genetic algorithm is then used to optimize the fuzzy MPC. Kim *et al.* propose in [10] another MPC by applying a genetic algorithm, to optimize this time the discrete control rod speeds. This paper proposes a new approach to do so, by optimizing offline the main characteristics of the control mechanisms, using an EA.

2.2 Parallel evolutionary algorithms

With the increasing number of computing units (cores, etc.), parallel EA become more and more popular to solve complex optimization problems. Usually, two main classes of types of parallel EA [1] can be distinguished : the coarse-grained model (island model) in which several EA share solutions within the migration process, and the fine-grained model (cellular model) where the population is spread into a grid and evolutionary operators are locally executed. Besides, a Master-Worker (M/W) architecture with the fitness evaluation on workers have been extensively used and studied [5]. It is simple to implement, and does not require sophisticated parallel techniques. Two communication modes are usually considered. In the synchronous mode, the parallel algorithm is organized by round. The master sends candidate solutions on each worker for evaluation, and waits until receiving a response from all workers before the next round. In the asynchronous mode, the master does not need to wait, and communicates with each worker individually on-the-fly. The asynchronous mode could improve the parallel efficiency when the evaluation time of the fitness function vary substantially [24]. We also propose an asynchronous parallel EA in this work.

2.3 Landscape aware parameter tuning

The performance of EA strongly depends on the value of their parameters (mutation rate, population size, etc.). Parameters setting is then one of the

major issues in practice for EA, and two methodologies are commonly used [7]. In the online setting, called *control*, the parameters are selected all along the optimization process. In the offline setting, called *tuning*, the parameter values are set before the execution of the algorithm. In offline setting, most of the methods, such as the irace framework [13], are based on a smart trial and error technic of parameter values on a set of problem instances. Those methods may require a large number of tests/executions on representative problem instances which can be difficult to afford in a black-box scenario with expensive costs on large scale computing environment. Alternatively, following Rice's framework [17], one can use a fitness landscape aware methodology to first extract features from the given problem instance, then select the relevant parameters according to those fitness landscape features.

Fitness landscapes are a powerful metaphor to describe the structure of the search space for a local search algorithm, and peaks, valley or plateaus for instance are used to depict the shape of the search space in this picture. Formally, a fitness landscape [21] is defined by a triplet $(\mathcal{X}, \mathcal{N}, f)$ where $\mathcal{X}$ is the set of candidate solutions, $\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}}$ is the neighborhood relation between solutions, and $f : \mathcal{X} \to \mathbb{R}$ is the fitness function (here assumed to be minimized) which associates to each candidate solution the scalar value to minimize. The neighborhood relation can be defined by a distance between solutions or by a local search operator.

Two main geometries are commonly used in fitness landscape. A multimodal fitness landscape is a search space with a lot of local optima (solution with no improving solution in the neighborhood). This geometry is also associated with the *ruggedness* which is the local regularity of the landscape. The more rugged the more multimodal the landscape is. The ruggedness can be measured by the autocorrelation of fitness [23] during a random walk over the landscape. A random walk is a sequence $(x_1, \ldots, x_\ell)$ of solutions such that for all $t \in \{2, \ell\}$, x_t is a neighboring solution selected uniformly at random from $\mathcal{N}(x_{t-1})$, or according to the local search operator. The autocorrelation function $\hat{\rho}$ is defined by the correlation of fitness between solutions of the walk : $\hat{\rho}(k) = \frac{\sum_{t=1}^{\ell-k}(f(x_t)-\bar{f})\cdot(f(x_{t+k})-\bar{f})}{\sum_{t=1}^{\ell}(f(x_t)-\bar{f})^2}$ with $\bar{f}$ the average value of $f(x_t)$. The main feature of ruggedness is then the autocorrelation length [9] which is the length τ such that there is no more significant fitness correlation at level ϵ between solutions of the walk : $\tau = \min\{k \; : \; |\hat{\rho}(k)| < \epsilon\}$. Usually, a smooth fitness landscape with long autocorrelation length is supposed to be easier to solve.

A neutral fitness landscape is another main geometry where the search space is dominated by large flat plateaus with many equivalent solutions. The dynamics of EA on such landscape is characterized by punctuated equilibrium dynamics where long neutral moves on plateaus are interrupted by rapid improving moves toward better solutions. One of the main features of this landscape is the neutral rate which is the proportion of neighboring solutions with the same fitness value [22] : $\mathrm{E}_{\mathcal{X}}[\sharp\{y \; : \; f(y) = f(x) \text{ and } y \in \mathcal{N}(x)\}/\sharp\mathcal{N}(x)]$. To avoid the computation of large neighborhood, the neutral rate can be estimated with a random walk [11] by : $nr = \sharp\{(x_t, x_{t+1}) \; : \; f(x_t) = f(x_{t+1}), t \in \{1, \ell-1\}\}/(\ell-1)$.

According to the local search operator, which could be the mutation operator for EA, the features of fitness landscape can characterize the shape of the landscape. First fundamental works have demonstrated the relevance of fitness landscape analysis for the parameters tuning [3]. However, to our best knowledge, no work has used such methodology for a real-world problem with expensive fitness function.

3   Problem definition

The optimization process is based on the current load-following transient [12] and this analysis focuses on a single Pressurized Water Reactor (PWR) type (1300 MW) of the French nuclear fleet. When an electrical power variation occurs (demand of the grid) a chain of feedback is setting up in the whole reactor, leading to a new steady state. It is usual to take advantage of this self-regulation in the case of small variations, but the regulated variables such as the temperature or the pressure in the primary or secondary circuits may reach unacceptable values in case of load-following, possibly leading to damages of the whole system. The control rods are then used in order to cope with this variation, and maintain the primary coolant temperature close to the target. However, those control rods have to be handled carefully as they could cause axial or radial heterogeneity in the core, inducing high power peaks or Xenon oscillations.

3.1   Description of the system

The reactor core is a grid of square assemblies (21cm length) in a cylindrical vessel. There are 193 assemblies, split into two kinds : 120 assemblies made of Uranium oxide (UOX) and 73 ones made of Uranium plus Gadolinium oxides (UGd). Each control rod is made of pins of a neutron absorber that are inserted together from the top of the core in some assemblies. The positions of the assemblies where they are inserted and the materials of which they are made correspond to the French "G" mode [12]. The rods are organized in two families: (i) the power shimming rods (PS) and (ii) the regulation rods (TR). The first ones are used to shim the power effects during the power transient, and are split in four groups (4 rods G1, 8 rods G2, 8 rods N1 and 8 rods N2). All the rods of a same group move together, and the groups are inserted successively in this order : G1, G2, N1, N2, as it is shown in Fig.1. An overlap is also defined between all the groups, so that they follow an insertion program as illustrated from frames (a) to (d). The position of those rods is linked to the electrical power by a calibration function. The second family enable a control of the average coolant temperature of the core (the targeted temperature, called reference temperature, is a linear function of the thermal power) and is made of 9 rods gathered in a single group. This group moves independently and automatically, following a speed program depending on the difference between the reference temperature (T_{ref}) and the mean temperature (T_m) as shown in Fig.2. One can see a dead band of ±0.8°C in which the rods do not move, avoiding continuous displacement and corresponding to the self-regulation of the core. Finally, as they are very efficient and for safety
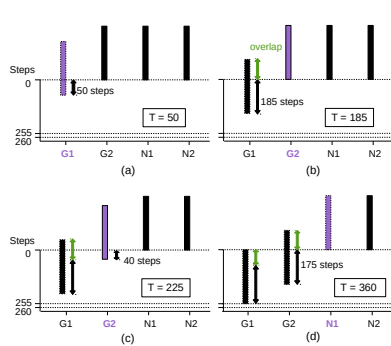
Fig. 1: Insertion sequence of the Power Shimming rods (PS). The totalizer value (T) is given on each frame, and the last moving group is in purple.
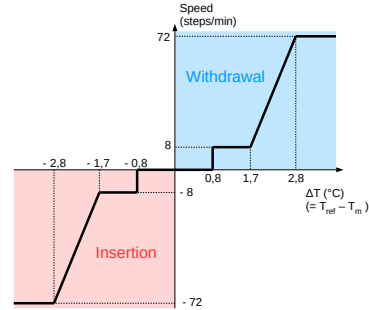
Fig. 2: Speed program of the Temperature Regulation rods (TR). The dead band corresponds to the null speed and the maximal and minimal speeds (± 72 steps/min) are for an absolute temperature difference larger than 2.8℃.

reasons, they are shut into a maneuvering band of about 50 centimeters in the upper part of the core. For more details, please refer to [8].

The variables to be tuned for the optimization are then the 4 nominal speeds and the 3 overlaps for the PS rods, the maximal and minimal speeds, the dead band width and the maneuvering band height for the TR rods. 11 variables are then considered, and they are coded as integer values corresponding to a discrete number of steps or of temperature (the dead band is discretized by steps of 0.1℃). Table 1 summarizes the variables, their initial values (current management) and ranges. The values take into account some technological and logical constraints. For example, the overlaps cannot be greater than the total height of the rods, the velocity ranges are bounded by the mechanisms, etc. A number of other variables could have been studied, like swaps between groups, or splitting groups, but the study is confined to the variables listed for two reasons: simplify the problem for a first optimization, and be able to propose a solution without major technological breakthroughs and similar to the current one. Nevertheless, the search domain is huge (at least 3×10^{20} possible configurations).
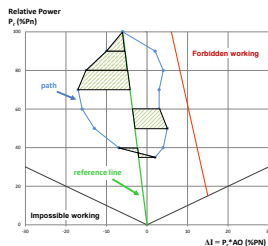
Fig. 3: Control diagram and criteria calculation principle.

Table 1: Integer variables of the design: lower bound (l.), upper bound (u.), and value of the current reference (r.). The dead band (db) variable is expressed in tenth of degree, and all the other variables are expressed in steps.

	PSR Overlaps			PSR Velocities				TRR V.			
	o_1	o_2	o_3	v_1	v_2	v_3	v_4	V	v	mb	db
l.	0	0	0	10	10	10	10	3	3	7	8
u.	255	255	255	110	110	110	110	13	13	117	16
r.	185	175	160	60	60	60	60	72	8	27	8

44

3.2 Criterion of interest

This seek of simplification is even more understandable when it is known that the black-box evaluation function is very costly. Each unitarian calculation corresponds to a given management configuration running on a complete typical load-following transient, corresponding to about 11 hours. The value of interest is then determined thanks to a model of the whole reactor described in [14], and developed within the APOLLO3® [20] calculation code. The optimization aims at minimizing this value of interest, which represents a global operating criterion, based on the control diagram. This control diagram is used by the operator to manage the power plant and represents the evolution of the relative thermal power (P_r) as a function of the power axial imbalance given by: $\Delta I = P_r \times AO$ where AO is the axial offset defined as $AO = \frac{P_T - P_B}{P_T + P_B}$ and standing for the unbalance between the lower and upper half parts of the core as regards the power. P_T (resp. P_B) is the power in the upper (resp. lower) part of the core.

An example of such a diagram is to be found on Fig.3, which draws the path of the state of the core during a power variation (blue line) and the bounds for this path. On the right side, the forbidden region (red line) is based on many studies and ensures the safety of the core in case of accidental situations. The impossible working region just comes from the definition ($AO \in [-1, 1]$). Finally, the green line starting at the same point as the path corresponds to a constant axial offset, and is called reference line in the following.

The criterion derived from the control diagram to be minimized is defined by:

$$\frac{1}{4} \sum |P_{r,i+1}^2 - P_{r,i}^2| \cdot \Big(D(\Delta I_{i+1}) + D(\Delta I_i) \Big) \tag{1}$$

where $D(\Delta I_i) = |\Delta \dot{I}_i - \Delta I_i^{ref}|$. The pair ($P_{r,i}$, ΔI_i) represents the state of the core at the time step i, and ΔI_i^{ref} the power axial imbalance given by the reference line at the power $P_{r,i}$. The criterion corresponds to the sum of all the areas as illustrated on Fig. 3, weighted by the relative power to take into account the fact that an important axial offset at high power is worse than at low power. Minimizing this criterion enables to reduce the area of the path and avoids being close to the forbidden region while staying close to the reference line.

4 Asynchronous parallel EA

The design of the EA is guided both by the expensive cost of fitness evaluation of the problem computed by a numerical simulation, and by the computing environment available to solve this problem.

4.1 Algorithm definition

On the one hand, the fitness evaluation duration is about 40 minutes on average with a large variance. On the other hand, a large number of computing units ($w = 3072$) are available to run the optimization algorithm, but they are only free for few hours (around 15 hours per experiment). Hence, we propose a master-worker (M/W) framework for the EA. On average one fitness evaluation is

completed every 0.78 second, meaning that the master node is not to be overflowed by the request of the workers, and with respect to the fitness evaluation time, an idle working time of few seconds will not reduce the performance. In addition, some simulations crash before the end of the calculation, increasing even more the discrepancies in calculation times. All considered, the model of the M/W has been made asynchronous: the workers are updated on the fly without a synchronization barrier, and each worker only computes the fitness value using the multi-physic simulator.

A lot of efficient EA can be considered in an asynchronous M/W framework with fitness evaluation on workers. The number of evaluations per worker is small, on average 23 fitness function evaluations is possible on each worker within 15 hours of computation. As a consequence, the EA should converge quickly. We propose then an asynchronous $(1 + \lambda)$-EA where λ is the number of computation units minus one. The Algo. 1 show the details of the algorithm.

Algorithm 1: Asynchronous M/W $(1 + \lambda)$-EA on master

1 **for** i *in Workers* **do**
2      $x^i \leftarrow$ **Initialization** using quasi-random numbers
3   **Send (non-blocking)** $\mathrm{Msg}(x^i)$ to worker i
4 **end**
5 $f^\star \leftarrow$ maximal value
6 **while** *pending message* **and** *time is not over* **do**
7   **Receive** Msg from worker i
8      $f^i \leftarrow \mathrm{Msg}[0]$
9   **if** $f^i \leqslant f^*$ **then**
10          $x^\star \leftarrow x^i$; $f^\star \leftarrow f^i$
11   **end**
12      $x^i \leftarrow$ **Mutate**$(x^\star)$
13   **Send (non-blocking)** $\mathrm{Msg}(x^i)$ to worker i
14 **end**
15 **return** $x^\star$

First, the algorithm on master node produces $\lambda = w - 1$ quasi-random solutions (integer vectors of dimension $n = 11$) using a Design of Experiments (DoE) based on Sobol of quasi-random numbers. This initialization is used to improve the spreading of the initial solutions in the search space. Every initial solution is then sent asynchronously to a worker who receives the solution from the master, computes the fitness value by running the multi-physic simulator, and send back the result to the master node. In the meantime, the main loop of the Algo. 1 is executed on the master node : wait for a message from a worker i, and when the fitness value is received, the best so far solution is updated if necessary. Notice that the best solution is replaced by the new solution evaluated by the worker even when the fitness values are equals. In that way, the algorithm is able to drift on plateaus of the search space. A new candidate solution is then computed by

the mutation (detailed in the next section) of the best-known solution and sent in non-blocking mode to the same worker i. The master is then able to manage the requests of the other worker nodes by the asynchronous communication mode. The algorithm stops after an arbitrary time limit is reached.

4.2 Mutation operator

The mutation operator is based on the classical mutation for vectors of numbers. The mutation rate p defines the parameter of the Bernouilli distribution to modify each number of the vector. Therefore, the number of modified variables follows a binomial distribution of parameters n and p, and the expectation of the number of modified variables is np. When an integer variable is modified according to the mutation rate, a random integer number is drawn using a uniform distribution centered on the current value. Let x_j be the current value of the variable j, and δ_j the gap defined by $\lfloor r.(ub_j - lb_j) \rfloor$ where lb_j and ub_j are respectively the lower bound and the upper bound of the variable j defined in the Tab. 1, and $r \in [0, 1]$ is a mutation parameter. The new value of variable j after mutation is selected uniformly in the interval $[x_j - \delta_j, x_j + \delta_j] \cap [lb_j, ub_j] \setminus \{x_j\}$. The parameter r tunes the range width for the new value of variable after the mutation, and is expressed relatively to the total range width of the variables ($r \leq 0.5$).

In addition, to avoid multiple costly evaluations of the same candidate solution, a hash-map is used on the master node to save all evaluated solutions. The mutation is applied on the solution until a new candidate solution which is not in the hash-map is produced by the mutation random process.

5   Experimental analysis

First, the performance of the algorithm with a baseline parameters setting is studied with 3072 computing units during 24 hours (approx. $73, 728$ hours of CPU time). Then, the mutation parameters are analyzed with the algorithm launched on 3072 computing units during 5 hours (approx. $15, 360$ hours of CPU time per run). At last, a fitness landscape analysis is conducted.

5.1   Baseline parameters setting

Following the value of the mutation rate parameter of $1/n$ commonly used in EA, the mutation rate has been set roughly to the inverse of the number of variables ($p_0 = 0.1$), so that the mutation operator modify on average one variable. The width of the random variation range has been arbitrarily set to about $r_0 = 0.05$ (5% of the total variation range of the variable). Those parameters have been chosen for the first optimization process and are called in the following the *baseline* settings. The use of an asynchronous algorithm to avoid idle time is justified by the discrepancies of the computation costs from a candidate solution to another one. The mean computation time is 2426 seconds, and the faster computation is done in 1629 seconds whereas the longer is performed in 6169 seconds. Fig.4 shows the dynamic of the run. The normalized best fitness is drawn
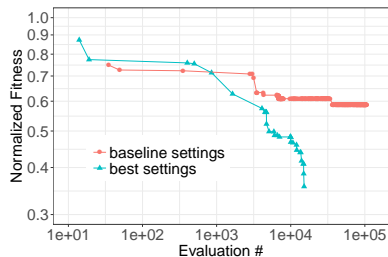
Fig. 4: Dynamic of the asynchronous M/W algorithm for the *baseline* and optimal mutation parameters settings
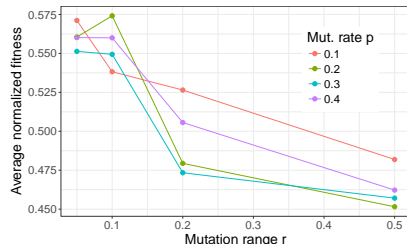
Fig. 5: Average normalized best fitness as a function of the mutation range width r and mutation rate p.

as a function of the number of evaluations received by the master node. A point is plotted when the best solution so far is updated (included for equal fitness values). The fitness values are normalized by the fitness value of the current management (see Tab. 1). The solutions for which the number of evaluations is lower than 3072 are from the initial quasi-random population. Even if the best solution obtained with *baseline* settings enable to reduce the fitness of about 40% compared to the current management, it can be seen that the number of strictly improving solutions is low (about 10 improving steps). The dynamic is a punctuated equilibrium dynamic with a lot of neutral moves on plateaus, and few improving solutions. For instance, the process is stuck on a plateau at the end of the run : almost $50,000$ fitness evaluations are necessary to find a strictly better solution. Subsequently, one can say that the neutrality is really important in the NROO problem. This first experiment shows the relevance of the algorithm to found better solutions than the current management, but it suggests that the setting of mutation parameters could also be improved.

5.2 Impact of the mutation parameters

This section deeply analyzes the influence of the mutation parameters on the performance of the M/W algorithm. Four values of mutation rates p and mutation ranges r are investigated: $p \in \{0.1, 0.2, 0.3, 0.4\}$ and $r \in \{0.05, 0.1, 0.2, 0.5\}$. All the combinations are considered, given 16 possible mutation settings of the mutation operator. To reduce the intrinsic random effect of the algorithm, each couple of mutation parameters values (p, r) have been launched five times with different initial populations generated by the Sobol sequence of quasi-random numbers. However, the 5 initial populations are the same for each couple of parameters settings. The total computation cost is more than $1, 2 \times 10^6$ hours of computation times, and we were not able to execute more than five runs.

The Fig. 5 shows the average normalized best fitness found for each parameter setting. The standard deviation of the best fitness found is also computed to measure the robustness of the parameters settings (not shown here to save space). In addition, for each initial population, the rank of each parameters setting is computed, and the average of the ranks gives another performance measure of

the settings. However, statistical tests will not give exploitable results because of the very low number of runs, and are then not considered. The variation of average fitness is larger according to the mutation range width parameter r than according to the mutation rate parameter p. The average fitness decreases with the parameter r whereas there is no clear trend as a function of the mutation rate p. The best sets as regards this criterion are then the ones for which the mutation range r is maximal. Inversely, the worse are the one for which the mutation range is minimal. Given the huge discrepancies of the average fitness as a function of the mutation range, the impact of the mutation rate cannot clearly be seen in this figure, and it is then difficult to choose the best mutation rate.

The performance according to the rank instead of best fitness value share the same result. Indeed, the Spearman correlation between the average fitness and the mean rank appears to be really high ($\rho = 0.91$), meaning that the best parameters as regards the first one is likely to be good also as regards the second. For example, the first five parameters settings with respect to the best average fitness are $(0.5, 0.2)$, $(0.5, 0.3)$, $(0.5, 0.4)$, $(0.2, 0.3)$, $(0.2, 0.2)$, and they are respectively third, first, sixth, second and fourth with respect to the average rank. However, the correlation between the average and the standard deviation of best fitness is low ($\rho = 0.48$) and thus, the five previous parameters settings are now in first, ninth, second, fifth and thirteenth position with respect to the standard deviation. It was decided to prefer the mutation parameters leading to low fitness value and rank rather than to low standard deviation. Future works will investigate ways to improve the robustness of the algorithm with respect to the initial population and thus to reduce the standard deviation.

The selected parameters setting is then $r = 0.5$, and $p = 0.3$ which is the first (resp. second) with respect to the rank (resp. best average fitness) because $(r, p) = (0.5, 0.2)$ is the first one as regards the best average fitness is only third as regards the rank, and also because the very best fitness so far is obtained with $(0.5, 0.3)$. In the framework of the greedy $(1 + \lambda)$-EA with large λ value and low numbers of iterations, very large mutation parameters with large exploration seem to be suggested. The dynamic of the optimal parameters setting is shown on Fig. 4. On the contrary of the common value of mutation parameters, the search is not stuck on plateaus, and the number of improving steps is high. Besides, those parameters setting found an optimal solution which reduces almost 65% of the reference fitness of current management, with only the quarter of the computation cost of the baseline settings.

5.3   Fitness landscape analysis

In this section, we investigate the fitness landscape of the NROO problem. For each parameters setting of the mutation operator, a random walk of length $\ell = 1024$ starting from a random candidate solution is computed. The cost of the walk is about 5% of the computation cost of the EA, and the length is smaller than the initial population size. Notice that by construction, all the solutions of the walk are strictly different. From the random walks, the autocorrelation length and the neutral rate are both estimated (see Sect. 2.3). The significant level ϵ used to estimate the autocorrelation length is set to $4/\sqrt{\ell}$.
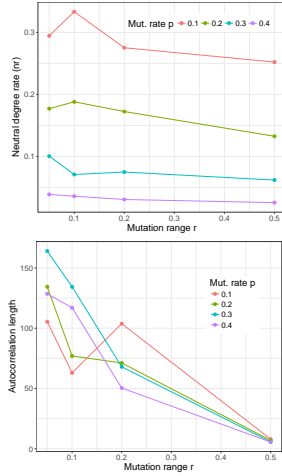
Fig. 6: Features of the fitness landscape as a function of mutation parameters r and p. neutral rate (top) and autocorrelation length (bottom).
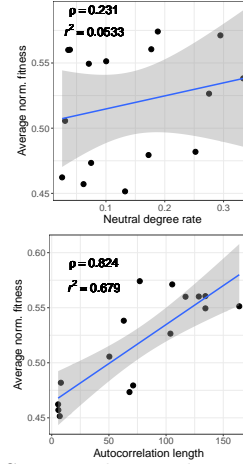
Fig. 7: Scatter plots and regression linear models between the average best normalized fitness and the features of fitness landscapes. Neutral rate (top) and autocorrelation length (bottom).

The Fig. 6 shows the features of the fitness landscape according to the mutation parameters. The mutation range width r does not impact the neutral rate. On the contrary, the neutral rate decreases with the mutation rate p : from 25% for the *baseline* setting with $p = 0.1$ to 3% for a high mutation rate value $p = 0.4$. The neutrality of NROO fitness landscape is high, and is dominated by large plateaus for common value of the mutation rate p. The neutral geometry explains the punctured equilibrium dynamics of the EA. As expected, a stronger mutation implies a more rugged fitness landscape. However, the ruggedness of the landscape is more impacted by the mutation range width r than by the mutation rate p. The autocorrelation length decreases with the mutation range width r from approximately 120 for $r = 0.05$ to 6 for the largest value $r = 0.5$ which picks a random new value. However, the landscape can be considered as a smooth landscape. For instance, when the mutation range is $r = 0.2$, more than 50 steps are required to reach a correlation of fitness between solutions smaller than $\epsilon = 0.125$. This feature should explain the good performances of the EA.

The Fig. 7 shows the correlation between the performance of the EA in terms of average normalized best fitness found and the feature values of the fitness landscape. Each point corresponds to a mutation parameters setting and the regression line of the linear model is also drawn. Surprisingly, although the neutral rate could be high, it is not linearly correlated to the performance of the EA. Only $r^2 = 5.3\%$ of the performance variance is explained by the linear regression model, and the Pearson correlation coefficient is below 0.23. On the contrary, the autocorrelation length is highly correlated with EA performance. The Pearson correlation coefficient is 0.82, and $r^2 = 67.9\%$ of variance is explained by the

simple linear regression. The result of the real-world NROO problem with costly fitness function is in accordance with fundamental works in EA such as on the well-known NK-landscapes : the problem difficulty and the performances are correlated to the ruggedness of the fitness landscapes. In contrast to the classical result obtained on the previous fundamental works however, the more rugged the landscape, the better the performance of the parallel EA. Our first result shows that a fitness landscape approach could be used to tune the parameters, but for highly selective parallel $(1 + \lambda)$-EA with a large number of computing units, rugged landscapes should be preferred.

6 Conclusions

In this paper, a real-world black-box combinatorial optimization problem with an expensive fitness function has been studied, and to solve it, an asynchronous master-worker $(1 + \lambda)$-EA running on a massively parallel architecture was used. The tough point of this exercise was the design of the algorithm, and mainly the mutation parameters. To do so, a parametric study was launched, giving satisfactory results, but requiring a lot of resources. In a second time, a fitness landscape analysis on this expensive problem showed that it is possible to tune the mutation parameters, and surprisingly, in the case of a large scale computing environment, with a limited user computation time, the mutation parameters associated to the most rugged landscape are relevant. It has then been possible to improve the considered criterion of almost 65%, meaning that on a given load-following transient, the operation of the core keep the axial power offset almost constant. This is encouraging for the following as some margins have been generated so that more heckled transients can now be considered.

The next step of this work is the minimization of the rejected effluent by the nuclear power plant. While there are many steps to be taken, our methodology opens the opportunity to tune the evolutionary algorithm from fitness landscape features, and pushes to design an efficient bi-objective algorithm for combinatorial black-box problems with expensive fitness functions.

References

1. E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.
2. G. Arnaud, J.-M. Do, J.-J. Lautard, A.-M. Baudron, and S. Douce. Selection combinatory algorithm for loading pattern design of light water reactor with two levels of heterogeneity. *Proceedings of ICAPP*, 2011.
3. F. Daolio, A. Liefooghe, S. Verel, H. Aguirre, and K. Tanaka. Problem features vs. algorithm performance on rugged multi-objective combinatorial fitness landscapes. *Evolutionary Computation*, 2016.
4. A. A. de Moura Meneses, L. M. Gambardella, and R. Schirru. A new approach for heuristics-guided search in the in-core fuel management optimization. *Progress in Nuclear Energy, Vol. 52*, 2010.

5. M. Dubreuil, C. Gagne, and M. Parizeau. Analysis of a master-slave architecture for distributed evolutionary computations. *IEEE T. on Systems, Man, and Cybernetics: Part B*, 36:229–235, 2006.

6. O. Dumont. Ademe energie 2030 : Production d'énergies renouvelables. Technical report, Ademe, 2012.

7. A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 19–46. Springer, 2007.

8. H. Grard. *Physique, fonctionnement et sûreté des REP*. EDP Sciences, 2014.

9. W. Hordijk. A measure of landscapes. *Evo. comp.*, 4(4):335–360, 1996.

10. J. H. Kim, S. H. Park, and M. G. Na. Design of a model predictive load-following controller by discrete optimization of control rod speed for PWRs. *Annals of Nuclear Energy, Vol. 71*, 2014.

11. A. Liefooghe, B. Derbel, S. Verel, H. Aguirre, and K. Tanaka. Towards landscape-aware automatic algorithm configuration: preliminary experiments on neutral and rugged landscapes. In *EvoCOP*, Amsterdam, 2017.

12. A. Lokhov. Technical and economic aspect of load following with nuclear power plants. *Nuclear Energy Agency, OECD*, June 2011.

13. M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The Rpackageirace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, 2011.

14. M. Muniglia, J.-M. Do, J.-C. Le Pallec, H. Grard, S. V. Verel, and S. David. A multi-physics PWR model for the load following. *ICAPP*, 2016.

15. M. G. Na, I. J. Hwang, and L. Y. Joon. Design of a fuzzy model predictive power controller for pressurized water reactors. *IEEE Transactions on Nuclear Science, Vol. 53 (3)*, JUNE 2006.

16. C. M. Pereira and C. M. Lapa. Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem. *Annals of Nuclear Energy, Vol. 30*, 2003.

17. J. R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

18. W. F. Sacco, C. R. De Oliveira, and C. M. Pereira. Two stochastic optimization algorithms applied to nuclear reactor core design. *Progress in Nuclear Energy, Vol. 48*, 2006.

19. W. F. Sacco, C. M. Lapa, C. M. Pereira, and H. A. Filho. A metropolis algorithm applied to a nuclear power plant auxiliary feedwater system surveillance tests policy optimization. *Progress in Nuclear Energy, Vol. 50*, 2008.

20. D. Schneider, F. Dolci, F. Gabriel, and J.-M. Palau. Apollo3® : Cea/den deterministic multi-purpose code for reactor physics analysis. *PHYSOR*, 2016.

21. P. F. Stadler. Fitness landscapes. In *Biological Evolution and Statistical Physics*, volume 585 of *Lecture Notes Physics*, pages 187–207, 2002.

22. L. Vanneschi, M. Tomassini, P. Collard, S. Vérel, Y. Pirola, and G. Mauri. A comprehensive view of fitness landscapes with neutrality and fitness clouds. In *European Conference on Genetic Programming*, pages 241–250, 2007.

23. E. D. Weinberger. Local properties of Kauffman's NK model, a tuneably rugged energy landscape. *Phys. Rev. A*, 44:6399–6413, 1991.

24. S. Wessing, G. Rudolph, and D. A. Menges. Comparing asynchronous and synchronous parallelization of the SMS-EMOA. *PPSN XIV*, 2016.

Sampled Walk and Binary Fitness Landscapes Exploration

Sara Tari, Matthieu Basseur, Adrien Goëffon

Laboratoire d'Etude et de Recherche en Informatique d'Angers
UFR sciences, 2 boulevard Lavoisier 49045 Angers cedex 01
{sara.tari, matthieu.basseur, adrien.goeffon}@univ-angers.fr

Abstract. In this paper we present and investigate partial neighborhood local searches, which only explore a sample of the neighborhood at each step of the search. We particularly focus on establishing links between the structure of optimization problems and the efficiency of such local search algorithms. In our experiments we compare partial neighborhood local searches to state-of-the-art tabu search and iterated local search and perform a parameter sensitivity analysis by observing the efficiency of partial neighborhood local searches with different size of neighborhood sample. In order to facilitate the extraction of links between instances structure and search algorithm behavior we restrain the scope to binary fitness landscapes, such as NK landscapes and landscapes derived from UBQP.

1 Introduction

Fitness landscapes are nowadays used in various fields to better apprehend the behavior of complex systems. In particular, in evolutionary computation, the study of combinatorial and continuous search spaces through fitness landscapes analysis helps to understand and predict the behavior of evolutionary algorithms. The concept of fitness landscape was first introduced by Wright [16] in the field of theoretical biology. Originally, landscapes represent an abstract space of genotypes where each individual is surrounded by all individuals differing by a mutation on a single gene. Once an adaptation value (fitness) is assigned to each genotype, such a model illustrates the repartition of peaks, valleys, and plateaus which are helpful to highlight the effect of mutations on genotypes. In evolutionary computation, such a model can help to observe difficulties induced by a given problem when tackled with an optimization method. Some studies using the concept of fitness landscapes focus on basic methods in order to better isolate and study some mechanisms used among search algorithms. In particular many studies have investigated hill-climbing algorithms [13, 15, 2] which are basic methods often incorporated within more sophisticated metaheuristics.

The aim of this work is to obtain insights to conceive local search algorithms. We focus on establishing links between optimization problem structure and efficiency of local searches; the purpose here is not to tackle and optimize specifically

a particular problem. More precisely, we present and focus on *partial neighborhood local search* algorithms, simple solution-based local searches which explore a sample of the neighborhood at each step of the search. In our experiments we perform a parameter sensivity analysis on partial neighborhood local searches and compare them to state-of-the-art local searches (iterated local search and tabu search) on two binary fitness landscapes: NK landscapes and the Unconstrained Binary Quadratic Programming problem (UBQP). Focusing on such landscapes facilitates the extraction of links between landscapes properties and search algorithms behavior. Here our experimental analysis highlights some links between ruggedness and both overall efficiency of considered methods as well as parameter sensivity of partial neighborhood local searches.

The paper is organized as follows. Section 2 is dedicated to the concept of fitness landscapes and related features. In section 3 we introduce the *sampled walk* local search algorithm as well as a similar partial neighborhood search algorithm called ID walk, previously introduced by Neveu *et. al* [11]. In section 4, experiments are presented and analyzed. In the concluding section, we provide possible ways forward.

2   Fitness landscapes

A fitness landscape is a triplet $(\mathcal{X}, \mathcal{N}, f)$ where $\mathcal{X}$ denotes the search space, $\mathcal{N}$ the neighborhood relation which assigns a set of neighbors to each solution, and f the fitness value which assigns a score to each solution. The search space and fitness function are directly derived from the instance of the considered problem whereas the method used to tackle the instance often induces a particular neighborhood function. One of the main interests of fitness landscapes in evolutionary computation is to study the behavior of neighborhood-based optimization methods in function of landscapes properties (typically their size, neutrality and ruggedness). These properties and associated indicators are discussed in [9]. Yet, main landscape characterization features cannot be calculated exactly since it induces an exhaustive enumeration of the search space on landscapes that are usually derived from large-scale NP-hard problems. They are generally estimated through indicators which sample the search spaces.

The neutrality rate of a fitness landscape corresponds to the proportion of neighboring solutions which have the same fitness value. While some landscapes contain no neutrality, the presence of such a feature can have a non-negligible effect on the number and distribution of local optima. In fact, landscapes with high neutrality are in general harder to solve and induce questions about the acceptance of neutral moves within local searches [10, 1].

The ruggedness of a landscape is a major property that determines the difficulty to optimize the underlying problem using the considered neighborhood relation. It mainly refers to the number of local optima, their distribution through the search space, and the size of their basins of attraction [12].

The autocorrelation function [14] is generally used to estimate the ruggedness of a fitness landscape. Such a measure requires the execution of several random

walks through the considered landscape. It calculates the correlation between fitness and distances of solutions encountered during the random walk. The result is a plot of autocorrelation where correlations usually decrease from 1 to 0 with respect to increasing distances between solutions.

The definition of ruggedness is not clearly established and ruggedness can also refer to the epistasis phenomenon, related to the degree of variable interdependency between genes [4]. When the interdependence between genes is high, knowing if the presence of a given gene positively affects the individual is difficult, if not impossible. Such landscapes have high epistasis since the effect of a mutation depends on the presence of other mutations. The sign-epistasis phenomenon between two genes A and B is depicted in fig. 1 (in lower case when the gene is not present). Considering two solutions and a given mutation (or neighborhood operator application), there exists a sign epistasis when the sign of the fitness variation resulting from the application of the mutation on both solutions differs. The 1-ruggedness of a landscape corresponds to the rate of sign epistasis between neighboring solutions, whereas the k-ruggedness of a landscape corresponds to the rate of sign epistasis between k-distant solutions [2].
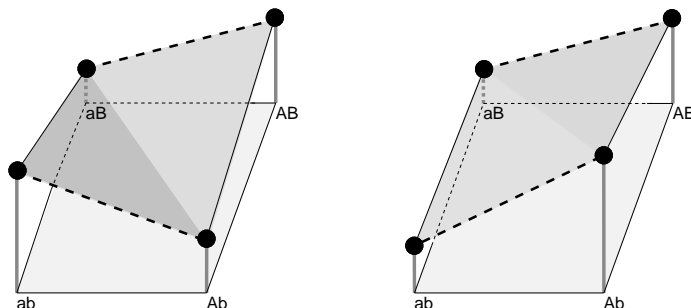
Fig. 1. Epistasis occurs when the presence of a mutation $a \rightarrow A$ affects the effect of another mutation ($ab \rightarrow aB$, $Ab \rightarrow AB$). We observe sign epistasis once a deteriorating mutation becomes beneficial when occuring after another mutation (left-hand side). Ride-hand side: no sign epistasis.

In the following, we mainly focus on the relation between ruggedness and behavior of partial neighborhood local searches.

3   Partial neighborhood local searches

This work follows previous studies related to the links between neutrality and ruggedness of combinatorial landscapes and the efficiency of hill-climbing algorithms. In particular, we investigated the ability of different neutral move policies within climbers to find good local optima.

First, it is obvious that accepting neutral solutions can potentially drive toward better local optima since it often helps not to be stuck in low-quality local optima. During the climbing process, most intensification mechanisms focus as a priority on improving the current solution rather than on considering neutral moves. In such cases, the selection of a neutral solution is only considered once a strict local optimum is reached. Yet a stochastic hill-climbing which indifferently selects the first improving or neutral neighbor encoutered clearly outperforms climbers which select improving neighbors as a priority [1].

Since accepting both improving and neutral neighbors during the search process helps to reach higher pikes, the effect of adding artificial neutrality in landscapes (by discretizing the fitness function) in order to reduce the ruggedness was studied in [3]. With an appropriate neutrality rate, stochastic climbers can efficiently tackle harder landscapes. Artificial neutrality-based climbers tend to favor better solutions without exclusively focusing on the improvement or deterioration of the real fitness values.

Here, we aim to simplify as far as possible the idea of favoring better neighbors. The key concept is to ignore if a move improves or not the current fitness value while maintaining a selection pressure. We then propose the *sampled walk* algorithm (SW), a local search which is based on randomly sampled neighborhoods (see alg. 1). At each step of the search, SW evaluates λ_{SW} random neighbors of the current solution and selects the one with the highest fitness value. Except λ_{SW}, the only choice to make concerns the stopping criterion. Yet, the stopping criterion is not necessarily fully considered as a parameter since in practice to be compared runs have to stop for any algorithm. Moreover such a parameter is irrelevant in an any-time optimization context.

$\lambda_{SW} = 1$ corresponds to a random walk whereas $\lambda_{SW} = N$ (with N is the neighborhood size) corresponds to a tabu search mechanism with an empty tabu list.

Due to the extreme simplicity of SW, its implementation is easy and does not require heavy design choices which depend upon the considered neighborhood function. Moreover, the SW simplicity greatly facilitates its analysis and allows many specific advanced variants. Note that SW, which is defined in a local search context, can also be viewed as an (1,λ) evolution strategy (with $\lambda = \lambda_{SW}$).

ID Walk (Intensification/Diversification Walk) [11] is based upon a similar concept. Like SW, ID Walk can be considered as a *partial neighborhood search* since it consists of evaluating (at most) λ_{ID} solutions at each step of the search. However, ID Walk selects the first encountered improving neighbor and therefore considers the fitness of the current solution to select the move to apply. When no improving solution is found among the λ_{ID} neighbors, the selected one depends upon the considered variant. ID_{best} selects the best one among the λ_{ID} deteriorating neighbors, whereas ID_{any} randomly selects one of them.

It is obvious that these partial neighborhood local seaches (SW, ID_{best}, ID_{any}), which use randomly generated subneigborhoods, leads to similar behaviors. As stated by Neveu *et al.* [11], ID walk was proposed with the aim to combine intensification and diversification during the search process. Although SW

follows the same principle, it emphasizes that the diversification aspect (brought by the partial neighborhood) is not explicitly determined by the sign of the fitness variation. The next section experiments show that such approach is efficient even if its selection strategy does not consider the fitness of the current solution.

Algorithm 1 Sampled Walk algorithm (maximization)

1: Choose $x_0 \in \mathcal{X}$ (initialization)
2: $x \leftarrow x_0$
3: $x^* \leftarrow x$
4: **while** stop criterion not reached **do**
5:     P $\leftarrow \lambda_{SW}$ random neighbors in $\mathcal{N}(x)$
6:     $x \leftarrow \text{argmax}_{x' \in P}(f(x'))$
7:     **if** $f(x) > f(x^*)$ **then**
8:         $x^* \leftarrow x$
9:     **end if**
10: **end while**
11: **return** x^*

---

4   Analysis on binary fitness landscapes

4.1   Experimental protocol

In order to properly assess the capacity of partial neighborhood local searches to lead toward good quality solutions, we compared the three variants SW, ID_{best} and ID_{any} to two widely-used local searches: tabu search (TS) [6] and iterated local search (ILS) [8]. Like SW, the classic tabu search does not use the current fitness for the selection process, but the whole neighborhood is considered. The tabu list prevents cycles which naturally occur by selecting iteratively the best neighbor among the complete neighborhood. ILS separates intensification and diversification phases. We choose here to use a first-improvement strategy during hill-climbing (intensification) phases, as first-improvement regularly reaches better local optima than best-improvement on landscapes difficult to climb [1]. Moreover, this leads to use for comparison two metaheuristics (ILS and TS) sufficiently different.

ILS and TS can be implemented with some variants which affect their behavior. Here we designed them as classical as possible. ILS performs $\mathcal{M}$ random moves when a local optimum is reached. At each step of the algorithm, TS selects the best move using a tabu list of forbidden bit-flips of size $\mathcal{L}$, that ensures a minimal distance between following solutions.

ILS, TS, ID_{best}, ID_{any} and SW require to set two parameters: a stopping criterion and the aforementioned structuring parameter. In this study, the stopping criterion is a maximum number of evaluations to permit a fair comparison between methods. The maximum number of evaluations is fixed to 10^8 for all

runs regardless to the landscape size. Such a value allows a sufficient convergence which ensures methods to almost never improve the best encountered solution after a significant number of evaluations. For each method we perform runs using several parameter values in order to establish appropriate settings.

For each triplet (landscape, method, parameter value) 100 runs are performed from the same initial set of 100 randomly generated solutions in order to reduce the stochastic bias. For each triplet, we retain the average of the 100 best encountered solutions (one per run). Since several values are tested, for each couple (landscape, method) only the best average is reported, i.e. the average obtained with the best considered parameter value. We also indicate if the method having the best average statistically dominates the other ones with respect to a binomial test (with a confidence level of 99%) for each considered couple.

In our experiments, we consider two types of fitness landscapes: NK landscapes and UBQP landscapes (ie. landscapes derivated from UBQP instances), the neighborhood operator under consideration being the one-flip operator.

NK landscapes are a model of binary fitness landscapes introduced by Kaufmann [7]. They are widely used when it comes to study the link between ruggedness and methods since their specificity is to have a tunable ruggedness. Such landscapes are defined by means of two parameters N and K. N specifies the number of variables and then the search space size (2^N). K determines the degree of variable interdependency (the fitness contribution of each variable being affected by K other variables) and greatly influences the ruggedness rate. Setting K to zero leads to a completely smooth landscape with no variable interdependency whereas setting K to $N-1$ leads to an extremely rugged (random) landscape. We used landscapes of various sizes $N \in \{128, 256, 512, 1024\}$ and ruggedness parameter $K \in \{1, 2, 4, 6, 8, 10, 12\}$.

The Unconstrained Binary Quadratic Programming problem (UBQP) is a NP-hard problem [5] which can reformulate a large scope of real-life problems in various fields. An instance of UBQP is composed of a matrix Q of size $n \times n$ of constants q_{ij} which can be positive or negative. A solution is a binary vector x of size n where $x_i \in \{0, 1\}$ corresponds to the i-th element of x. The UBQP objective function $f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j$ has to be maximized.

We used an instance generator (proposed and provided by Gintaras Palubeckis on www.personalas.ktu.lt/~ginpalu/ubqop_its.html) to generate some instances of different sizes and density. The density d affects the rate of values equal to zero in the matrix Q, $d = 0$ leads to a matrix full of zero except on the diagonal whereas a $d = 100$ leads to matrix with no zero.

4.2 Results

Results (see table 1) show in most cases that on the considered NK landscapes the sampled walk SW leads toward best solutions in average. SW efficiency does not seem to be affected by the ruggedness, which is mostly tuned by means of the parameter K. On smooth and small landscapes ($K \leq 4$ and $N = 128$) almost all methods lead toward the same solution which seems to be the global optimum. The explanation behind these results is that regardless the size and ruggedness

Land.	Average fitness					Best parameter value				
N K	SW	ID_{best}	ID_{any}	ILS	TS	λ_{SW}	λ_{IDb}	λ_{IDa}	$\mathcal{M}$	$\mathcal{L}$
128 1	**.7245**	.7245	.7245	.7245	.7165	8, 12	8, 12	$16 \to 128$	$5 \to 20$	20
128 2	**.7424**	.7424	.7420	.7423	.7369	12, 16, 20	16	40	10	20
128 4	**.7959**	.7959	.7959	.7958	.7952	16, 20	16, 20	$40 \to 128$	5	20
128 6	**.8004**	.8003	.8000	.7994	.7976	16	16	56	5	15
128 8	**.8021**	.8015	.7980	.7949	.7923	20	20	72	5	15
128 10	**.7937**	.7930	.7893	.7847	.7828	24	32	120	5	10
128 12	**.7819**	.7817	.7785	.7724	.7729	28	36	96	5	10
256 1	**.7220**	.7220	.7199	.7200	.7118	16	16	96	15	20
256 2	**.7444**	.7444	.7426	.7424	.7249	24	24	96	5, 10, 20	20
256 4	**.7934**	.7933	.7921	.7916	.7823	20	20	192	5	20
256 6	**.8048**	.8045	.8017	.8007	.8020	24	24	184	5	20
256 8	**.7964**	.7960	.7915	.7892	.7894	32	32	112	5	15
256 10	**.7869**	.7860	.7822	.7782	.7779	36	40	184	5	15
256 12	.7756	**.7756**	.7718	.7663	.7657	44	52	184	5	15
512 1	**.7079**	.7077	.7038	.7040	.7007	16	16	256	20	50
512 2	.7509	**.7509**	.7451	.7453	.7316	16	24	128	5	50
512 4	**.7860**	.7857	.7802	.7806	.7845	24	24	$128 \to 512$	5	50
512 6	**.7989**	.7984	.7944	.7940	.7965	24	32	256	5	30
512 8	**.7939**	.7935	.7894	.7886	.7849	40	40	256	5	30
512 10	**.7829**	.7825	.7790	.7781	.7760	56	48	256	5	20
512 12	**.7720**	.7719	.7682	.7671	.7618	64	64	256	5	15
1024 1	**.7163**	.7160	.7083	.7087	.7051	16	16, 24	256	15	50
1024 2	**.7522**	.7521	.7427	.7428	.7274	24	24	256	5, 10, 20	50
1024 4	**.7878**	.7872	.7800	.7797	.7654	24	24	256	5	50
1024 6	**.7949**	.7943	.7893	.7890	.7899	32	32	256	5	50
1024 8	**.7901**	.7888	.7859	.7850	.7850	40	48	256	5	40
1024 10	**.7793**	.7786	.7758	.7753	.7740	56	64	256	10	30
1024 12	**.7694**	.7689	.7664	.7656	.7653	72	80	256	5	20

Table 1. Results on NK landscapes. Left-hand side: average fitness obtained with the best parameter value for each couple (landscape, method). For each landscape, the best average fitness obtained appears in bold, whereas non statistically dominated methods appear in shaded. Right-hand side: the best parameter value(s).

UBQP	SW	ID_{best}	ID_{any}	ILS	TS
2048 10	$_{100}$4035.71	$_{100}$4052.02	$_{100}$3773.58	$_{100}$**4293.54**	$_{100}$4254.14
2048 25	$_{164}$0792.90	$_{164}$0823.45	$_{164}$0432.32	$_{164}$1183.63	$_{164}$**1192.63**
2048 50	$_{239}$7652.20	$_{239}$7695.08	$_{239}$7215.93	$_{239}$8106.97	$_{239}$**8443.35**
2048 100	$_{309}$7976.50	$_{309}$8266.91	$_{309}$7122.46	$_{309}$8566.65	$_{309}$**9318.75**
4096 10	$_{280}$7921.35	$_{280}$7955.71	$_{280}$6968.13	$_{280}$7632.68	$_{280}$**8263.77**
4096 25	$_{459}$4746.56	$_{459}$5136.29	$_{459}$3264.07	$_{459}$3665.15	$_{459}$**5741.73**
4096 50	$_{652}$6291.28	$_{652}$6692.46	$_{652}$4326.94	$_{652}$5133.66	$_{652}$**7995.10**
4096 100	$_{90}$90355.70	$_{90}$90761.04	$_{90}$86936.83	$_{90}$87492.74	$_{90}$**93039.30**

Table 2. Results on UBQP landscapes. Average fitness obtained with the best parameter value for each couple (landscape, method). For each landscape, the best average fitness obtained appears in bold, whereas non statistically dominated methods appears in shaded.

of landscapes, runs are always performed with a credit of 100 million of evaluations. Since smaller and smoother landscapes tend to be easier to tackle, they require less computational effort to attain good solutions. On every considered landscapes, results obtained by ID walk are very close to those obtained by SW, but SW often statistically dominates on large landscapes.

Ruggedness does not seem to affect the overall comparative efficiency of the considered methods. Yet, best parameter values (among the considered ones) for each method evolve in function of the value of K. SW and ID_{best} require very similar parameter values in order to reach good solutions. One can observe that the most appropriate parameter values increase when K increases. ID_{any} parameters requirement does not evolve the same way as ID_{best} and SW. For ILS and TS, the number of perturbations and the length of the tabu list also evolve in function of K. On very smooth landscapes, ILS requires more perturbations and TS a longer tabu list than on more rugged landscapes.

Such results, in addition to those observed on SW, seem indicate that the search process needs more diversification on smooth landscapes than on rugged ones. Actually smooth landscapes contain few local optima and then have large basins of attraction. On such a configuration, a more important diversification helps to get out of some basins and to attain different local optima.

All methods were also tested on several landscapes derived from UBQP instances and results differ from those obtained on NK landscapes. In table 2, we only report large landscapes results ($N = 2048$ and $N = 4096$) since on smaller ones the considered methods with various parameters value almost always lead toward the same solution (which is expected to be the global optimum). Such a fact indicates that for a given N, N-dimensional UBQP landscapes are *easier* to tackle than N-dimensional NK landscapes.

On large landscapes, the tabu search almost always leads toward the best average. ILS is rarely dominated when $N = 2048$, but always outperformed by TS when $N = 4096$.

4.3 Landscapes ruggedness and partial neighborhood LS efficiency

Experiments show that the sampled walk is particularly efficient on NK landscapes, regardless of the ruggedness level. Yet SW and other partial neighborhood local searches (ID walk) is less efficient on UBQP landscapes, which seem to be easier to tackle than NK landscapes. In order to determine if there is a link between those results and the structure of landscapes, we analyzed the ruggedness of landscapes by means of two indicators: autocorrelation and the k-ruggedness.

The plot of autocorrelation (fig. 2) on NK landscapes shows that its evolution is affected by K. Indeed, the more rugged a landscape is, the faster the correlation fitness-distance decreases. The plot of autocorrelation (fig. 3) on UBQP landscapes shows that its evolution is globally identical on all considered landscapes and evolves similarly as very smooth NK landscapes ($K = 1$).

The evolution of k-ruggedness (fig. 4) on NK landscapes follows the same scheme as the evolution of autocorrelation. Yet, the k-ruggedness indicator evolves quite differently on UBQP landscapes (fig. 5) than on smooth NK landscapes, especially on the first steps with a faster evolution. Such observation evokes locally-rugged landscapes.
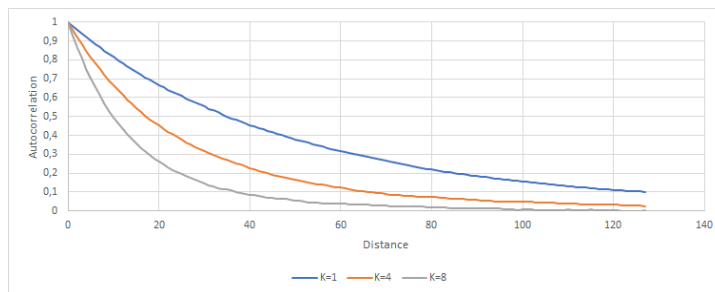
Fig. 2. Autocorrelation evolution on NK landscapes of size 128 (similar outputs can be observed for higher size of landscapes).

Considering NK landscapes, analogies between the evolution of autocorrelation and k-ruggedness seem to indicate that such landscapes have a uniform ruggedness repartition. On the contrary, it appears that UBQP landscapes have a less uniform ruggedness repartition, which we can describe as a local ruggedness and a global smoothness. This could also explain why smaller UBQP instances are easy to solve by local search as long as some diversification is applied. An hypothesis which could possibly explain the lower efficiency of partial neighborhood local searches on such landscapes is that such methods tends to explore solutions scattered through the entire landscape, whereas tabu search naturally intensifies around promising areas. In this type of landscapes, the correlation between
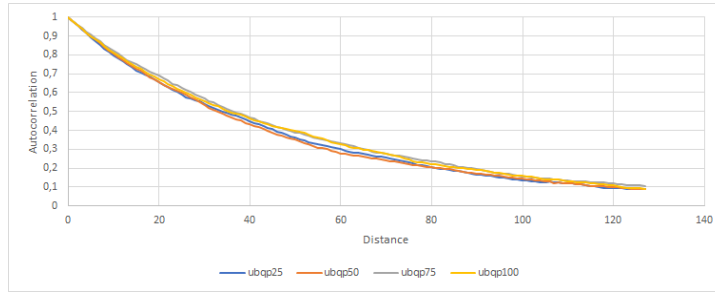
Fig. 3. Autocorrelation evolution on landscapes derived from UBQP of size 128 (similar outputs can be observed for higher size of landscapes).

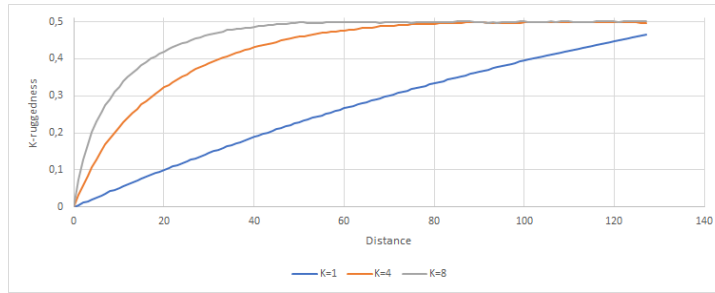

Fig. 4. K-ruggedness evolution on NK landscapes of size 128 (similar outputs can be observed for higher size of landscapes).
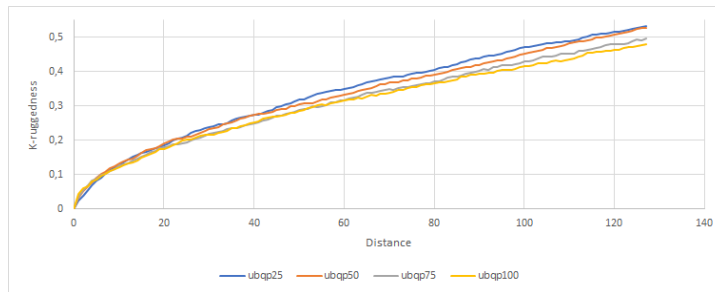

Fig. 5. K-ruggedness evolution on landscapes derived from UBQP of size 128 (similar outputs can be observed for higher size of landscapes).

fitness and distance of solutions decreases progressively. When the decorrelation fitness-distance is fast, the use of a sampled walk seems more appropriate to efficiently explore the search space.

5   Conclusion

In this paper we investigate partial neighborhood local searches and, more particularly, the sampled walk algorithm which can be viewed as a local search transposition of an $(1,\lambda)$-ES. We show that the sampled walk is efficient to tackle common binary landscapes. Conducted experiments on NK landscapes highlighted the fact that the sampled walk behavioral parameter can be principally set according to the landscape ruggedness. Experiments also show that such a method is globally competitive in comparison to metaheuristic searches like tabu search and iterated local search. Even if the sampled walk is outperformed by a tabu search on UBQP, we are able to establish links between respective efficiency of methods and ruggedness repartition thanks to the k-ruggedness indicator.

Future works include the consideration of permutation-based landscapes. The use of other solution representation brings some difficulties such as the criterion on which the tabu list is based, as well as the way to evaluate the k-ruggedness since this indicator is related to the concept of sign epistasis. It would provide useful information to analyze the behavior of the considered methods all along the search process (any-time optimization). Finally, this family of local searches like ID walk, based on a random sampling of the neighborhood, constitutes very simple search algorithms and have not been deeply investigated in the metaheuristics literature. There are thus many ways to differently use the sampled walk principle and to improve its efficiency, for instance by adapting its parameter during the search according to landscapes features.

References

1. Matthieu Basseur and Adrien Goëffon. Hill-climbing strategies on various landscapes: an empirical comparison. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 479–486. ACM, 2013.
2. Matthieu Basseur and Adrien Goëffon. Climbing combinatorial fitness landscapes. *Applied Soft Computing*, 30:688–704, 2015.
3. Matthieu Basseur, Adrien Goëffon, and Hugo Traverson. Exploring non-neutral landscapes with neutrality-based local search. In *International Conference on Learning and Intelligent Optimization*, pages 165–169. Springer, 2015.
4. William Bateson, ER Waunders, and Reginald Crundall Punnett. Experimental studies in the physiology of heredity. *Molecular and General Genetics MGG*, 2(1):17–19, 1909.
5. Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of NP-completeness, 1979.
6. Fred Glover and Manuel Laguna. *Tabu Search*. Springer, 2013.

7. Stuart A. Kauffman and Edward D. Weinberger. The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of Theoretical Biology*, 141(2):211 – 245, 1989.

8. Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

9. Katherine M Malan and Andries P Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013.

10. Marie-Éléonore Marmion, Laetitia Jourdan, and Clarisse Dhaenens. Fitness landscape analysis and metaheuristics efficiency. *Journal of Mathematical Modelling and Algorithms*, pages 1–24, 2013.

11. Bertrand Neveu, Gilles Trombettoni, and Fred Glover. ID walk: A candidate list strategy with a simple diversification device. In *International conference on principles and practice of constraint programming*, pages 423–437. Springer, 2004.

12. Gabriela Ochoa, Marco Tomassini, Sebástien Vérel, and Christian Darabos. A study of NK landscapes' basins and local optima networks. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 555–562. ACM, 2008.

13. Gabriela Ochoa, Sébastien Verel, and Marco Tomassini. First-improvement vs. best-improvement local optima networks of NK landscapes. *Parallel Problem Solving from Nature, PPSN XI*, pages 104–113, 2010.

14. Edward Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics*, 63(5):325–336, 1990.

15. Darrell Whitley, Adele E Howe, and Doug Hains. Greedy or not? best improving versus first improving stochastic local search for MAXSAT. In *AAAI*, 2013.

16. Sewall Wright. *The roles of mutation, inbreeding, crossbreeding, and selection in evolution*, volume 1. 1932.

Session 3 - Genetic Programming

Semantics-based Crossover for Program Synthesis in Genetic Programming

Stefan Forstenlechner, David Fagan, Miguel Nicolau, and Michael O'Neill

Natural Computing Research & Applications Group, School of Business,
University College Dublin, Ireland
`stefan.forstenlechner@ucdconnect.ie`,
{`david.fagan,miguel.nicolau,m.oneill`}`@ucd.ie`

Abstract. Semantic information has been used to create operators that improve performance in genetic programming. As different problem domains have different semantics, extracting semantics and calculating semantic similarity is of tantamount importance to use semantic operators for each domain. To date researchers have struggled to effectively do this beyond the boolean and regression problem domain. In this paper, a semantic similarity-based crossover is tested in the problem domain of program synthesis. For this purpose, a similarity measure based on the execution trace of a program is introduced. Subtree crossover as well as semantic similarity-based crossover are analysed on performance and semantic aspects. The goal is to introduce the Semantic Similarity-based Crossover in the program synthesis domain and to study the effects of using semantic locality. The results show that semantic crossover produces more semantically different children as well as more children that are better than their parents compared to subtree crossover.

Keywords: Genetic Programming; Program Synthesis; Crossover

1 Introduction

Semantic information has helped improve operators to achieve better performance in Genetic Programming (GP) compared to syntactical operators [13] [15]. The properties semantic diversity (keeping a semantically diverse population) and semantic locality (small change in the genotype results in a small change in the phenotype) are of major importance to this process. One such operator that makes use of semantic information, especially of semantic locality, is the Semantic Similarity-based Crossover (SSC) introduced by Nguyen et al. [12], which was able to achieve performance improvements over several other crossover operators.

Although semantic information can be a key aspect to improve performance in GP, it is also problem dependent. Therefore, operators have to be adapted to the problem domain. In this paper, SSC is introduced to the domain of program synthesis to be able to benefit from semantic information in this area. For this purpose, a semantic similarity measure for code snippets of programs is proposed.

The paper proceeds as follows. More information on SSC as well as semantics itself is given in Section 2. A semantic crossover for program synthesis as well as a semantic similarity measure are described in Section 3. The semantic crossover is tested on a set of benchmark problems. The experimental setup is explained in Section 4 and the results are analysed in Section 5. The conclusion and future work of the study are discussed in Section 6.

2   Related Work

This section explains certain terms that are related to this study and Semantic Similarity-based Crossover is described, which the proposed crossover operator in Section 3 is based on.

2.1   Semantics

A standard genetic programming system relies upon syntactical genetic operators. The benefits of these operators are that they are problem independent, but they can be outperformed by more specialised operators. Semantics, which defines "the behavior of a program, once it is executed on a set of data" [15], can be used to gain additional information about a program to improve performance. Performance improvements are often gained by making use of semantic diversity and semantic locality [13] [15]. These properties are important as a high semantic diversity helps exploring the search space and semantic locality improves the efficiency of the search algorithm [15]. A direct approach to using semantics instead of relying on diversity and locality that should be mentioned is Geometric Semantic GP [10], which uses tailored genetic operators that produce solutions which cannot be worse than the solutions they are derived from. But this approach is limited to certain problem domains and increases the size of the possible solutions rapidly.

As semantic information is dependent on input and output of a program, it is also problem dependent. A measure for semantic similarity or difference can only be applied in a certain problem domain, therefore semantic operators cannot be applied to all problem domains without adaptation. In this paper, a semantic crossover operator is introduced in the domain of program synthesis. Although this study focuses on crossover, semantic information has also been used for mutation operators [1] and selection [4] [5]. A more detailed overview of semantics and how it has been used so far can be found in this survey [15].

2.2   Semantic Crossover

Using semantics in crossover operators has been mainly studied in the domain of boolean problems [2] [8] and regression problems [11] [12] [13]. The main idea is to promote semantic locality by exchanging semantically similar subtrees to make an overall small change to the output of the whole GP tree.

The proposed semantic crossover in this study is mainly inspired by Nguyen semantic similarity-based crossover [11], which was improved multiple times [12] [13]. The crossover selects one subtree of each parent and compares their semantic output with each other. The semantic output is produced by using random values as input for the subtrees. Then, the semantic similarity is measured as the sum of the absolute differences of the outputs. Initially, an upper bound was used to decide if the subtrees were similar enough for crossover [11]. This was improved by using an upper bound and a lower bound to avoid equivalent and too similar subtrees [12]. In both cases, multiple tries can be used to find similar subtrees. After that, the crossover was adapted to calculating the semantic similarity of multiple subtrees at once and using the most similar ones for crossover, which are not equivalent according to a lower bound [13].

3   Semantics in Program Synthesis

In contrast to other problems tackled with GP, in program synthesis, nodes are usually typed to be able to produce syntactically correct programs, like using grammars or abstract syntax trees as representations. It should be noted that experiments carried out in this paper, see Section 4, are executed with a grammar-based GP system, but the proposed approach is generally applicable. Therefore, nodes are implicitly typed, because crossover and mutation are only allowed to create individuals which apply to the specifications of the grammar. Section 3.1 describes the semantic similarity measure proposed in this study and Section 3.2 explains the details of the adapted semantic crossover for program synthesis.

3.1   Semantic Similarity Measure with Traces

The semantic crossover proposed by Nguyen et al. [13], which the crossover proposed in this paper is based upon, uses a semantic distance measure. As mentioned, semantics are problem specific and the existing semantic distance measures were designed for regression problems. To apply semantic crossovers in the program synthesis domain a distance measure for code snippets of a program are required, which is the main contribution of this paper.

Semantics is defined as the output or the behaviour of a program. For a regression problem, the output is a vector of real values. In the case of program synthesis, the output can be multiple vectors of different data types. A semantic similarity can be calculated on the difference of the variables of two programs. Similar to semantics in regression, it is not only possible to get the semantics of the final output, but also of intermediate steps. In the case of program synthesis, intermediate steps can be one or more executable statements. After every statement the change of variables can be checked. Therefore, the semantic of every statement can be saved and used in a genetic operator. To measure the semantics of a subtree which only represents part of a program statement (e.g. a binary

comparison), the first parent node representing an actual program statement is used instead.

The process of logging variable changes in a program is called *tracing* and produces a *trace*. These traces are used to check the semantics of every statement in the program and to measure semantic similarity. An example of a trace of a short program is shown in Figure 1. A short code snippet and a possible corresponding tree representation are shown, as well as the trace of the variable settings. Every statement that can be executed on its own has a corresponding variable setting before and after the execution of the code. The variable setting of `1*` is the initial input setting for three different inputs. `2*` displays the variable settings after executing the first statement and `3*` after the second statement. Variable changes are marked in red and bold.
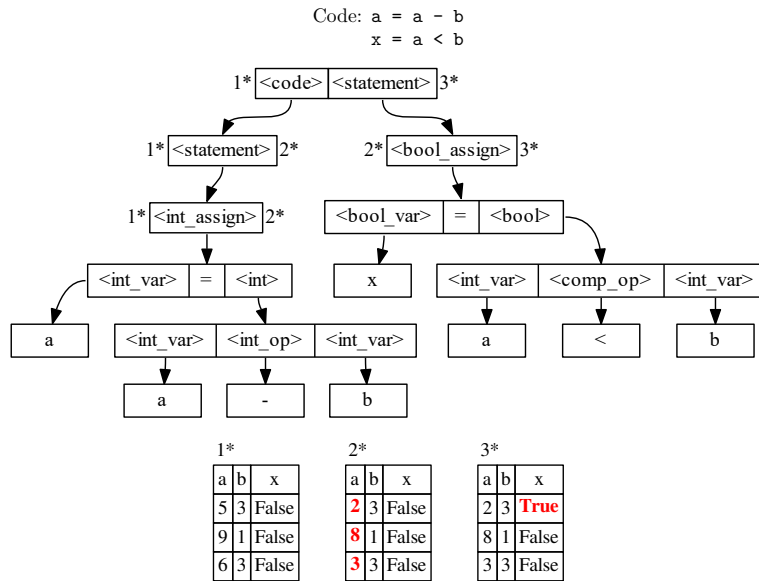
Fig. 1. Example trace of a tree which has an initial variable setting `1*` for three different inputs. The state of the variables after executing the tree/code is shown in the variable setting `3*`. `2*` is an intermediate variable setting produced by executing the first statement. The numbers `1*-3*` denote the variable settings or semantics before and after executing that node and are the trace of this tree.

The trace of a variable produces a vector of that type, as the trace is produced with multiple different inputs. Therefore, the trace of a boolean variable produces a vector of boolean values. A variable of the type list of integers produces a vector of lists of integers. The similarity measures used are listed in Table 1. These measures are only suggestions as this is an initial study on using semantics in

program synthesis. The similarity measures should be self-explanatory, except the "List of any type". As list elements cannot only be changed, but also removed or additional ones can be inserted, Levenshtein distance gives an approximation of how much of the list has changed, similar to a string.

Table 1. Similarity measures per variable

Variable type	Similarity measure
Boolean	Hamming distance
Integer	Sum of absolute differences
Float	Sum of absolute differences
String	Sum of Levenshtein distances
List of any type	Sum of Levenshtein distances

These similarity measures only calculate the difference between two variables of the same type. Two programs might have more variables of the same type. The similarity of two variables that mainly contain large values might be by far bigger than the similarity of two variables that contain only small values. Therefore, the influence on the overall similarity of two programs of variables containing small values will always be proportionally smaller. To counteract that problem, all similarities are normalized to be between 0 and 1. Nevertheless, due to the different similarity measures used and to check which data types show a difference, only one data type is used per crossover for measuring semantic similarity. The tree in Figure 1 shows a difference in the integer variable a and boolean variable x. One of the data types, integer or boolean, will then be chosen for the similarity measure. If integer is chosen, then a and b will be used and x will be ignored, otherwise x will be used and the other two will be ignored.

3.2   Semantic Crossover for Program Synthesis

The semantic crossover for program synthesis proposed in this study is based on the Most Semantic Similarity-based Crossover (MSSC) by Nguyen et al. [13], which is explained in Section 2.2. The pseudocode in Algorithm 1 describes the proposed crossover algorithm. As with subtree crossover, a crossover point from the first parent is selected. Then, instead of selecting one random subtree from the second parent, which is of the same type as the selected node from the first parent, up to a maximum value of subtrees (*Max_Tries*) are selected at random without repetition. *Max_Tries* is a parameter that can be set. If the second parent does not contain a subtree of the same node type as the selected one from parent one, no crossover is executed.

In the next step, the semantic differences between the subtree of the first parent and all the selected subtrees from the second parent are calculated. This is done in the following way. During the evaluation, the semantic information of every individual is saved in form of an execution trace as explained in Section

---
Algorithm 1 Semantic similarity-based crossover for program synthesis
---
select crossover point from first parent
select *Max_Tries* possible subtrees from second parent
if no subtrees of same type as crossover point available **then**
    return do nothing
end if
get semantics of every selected subtree from second parent
calculate semantic differences for every selected subtree per type
if differences **then**
    select random type
    select most semantically similar subtree based on selected type
else
    select random subtree for crossover from second parent
end if
crossover with selected subtree

Algorithm 2 Semantic similarity calculation for two subtrees
---
input1, output1 ← semantics of subtree from first parent
set variables to input1
output2 ← execute one subtree from second parent
calculate semantic distance between output1 and output2

3.1. The variables setting before and after the execution of each statement and therefore the corresponding subtree is saved as well. The variable setting before the execution of a subtree can be viewed as the *input* and the setting afterwards as the *output* of that code snippet. For each selected subtree from the second parent, the variables are set to *input* of the subtree of the first parent, followed by executing the subtree of the second parent and comparing the variable outputs to the *output* variable setting of the subtree from the first parent. A more concise description of this process as pseudocode is given in Algorithm 2. It should be noted that this is not a fitness evaluation, but a necessary process to find the semantic differences. If there is no difference for any subtree, one subtree is chosen randomly. If there is a semantic difference, a random data type is chosen, which shows a semantic difference. For all variables of that data type, the sum of semantic similarities is calculated with the corresponding similarity measure, shown in Table 1. The most semantic similar subtree that is not equal, is chosen for crossover.

The reason why only one data type is chosen, is because different data types use different distance measures and mixing them might result in unwanted behaviour. Investigating combining these measures and choosing different measures is left for future work.

4 Experimental Setup

A tree-based grammar guided genetic programming system is used for the experiments. The used grammars produce executable Python programs and have been automatically generated. Multiple small Python grammars exist, where each grammar only defines rules for one specific data type. The automatic generation process combines these grammars according to the data types required by a specific problem. The design of the grammars has been taken from [3]. Five benchmark problems are used and are explained in more detail in the next Section. The parameter settings are displayed in Table 2, which are the same as in [3], except the number of generations has been reduced. Three variables of each data type required by a problem are available in the grammars. An execution timeout has been set to one second to avoid non-halting programs. An execution on the problem should usually only take a few milliseconds. Lexicase selection [6] was used as it has shown better performance on program synthesis problems than tournament selection [7]. All experiments are run with the semantic similarity-based crossover for program synthesis proposed in the previous section and subtree crossover. Subtree crossover was modified to operate only on nodes with the same type, similar to Strongly Typed Genetic Programming [9], as grammars have implicit typing due to grammar rules. The maximum number of selected subtrees from the second parent (*Max_Tries*) has been set to 10, which has been established in preliminary experiments.

Table 2. Experimental parameter settings

Parameter	Setting
Runs	50
Generations	100
Population size	500
Selection	Lexicase
Crossover probability	0.9
Mutation probability	0.05
Elite size	1
Node limit	250
Variables per type	3
Max execution time	1 second
Max_Tries	10

The experiments have been executed with HeuristicLab [16].

4.1 Benchmark Problems

The problems chosen are of various difficulties, namely Collatz Number, Compare String Lengths, Grade, Number IO and Super Anagrams. These problems

have been introduced with others as a general program synthesis benchmark suite by Helmuth et al. [7] and are introductory computer science programming problems. The benchmark suite has been tackled before with PushGP [14] and the system used in this paper. Collatz Number has neither been solved by the system used, nor by the PushGP. Compare String Lengths, Grade and Super Anagrams have been solved before, but the success ratio was rather small, from 3 times out of 100 runs to 28 times out of 100 runs. Number IO is a rather easy problem, solved nearly every time and is only used as a sanity check to see if everything works correctly.

5 Results

In this section, the results of the experiments described in Section 4 are analysed. Subtree crossover is referred to as "Default", while the semantic similarity-based crossover is called "Semantic". As mentioned, the overall goal of the experiments is to analyse and draw conclusions from the behaviour of a semantic crossover. Due to the additional computation that is required for the semantic crossover for program synthesis, an increased runtime is expected, but it has not been analysed, because computational cost was added to collect the measurements which will be discussed in this Section. Additionally the current implementation of semantic crossover has in no form be optimised.

5.1 Successful Runs and Fitness

Table 3 shows the number of times a run was able to find a correct solution to a problem and Table 4 shows the average test fitness of the best training solution found over 50 runs. When comparing subtree crossover to semantic crossover (SC), Table 3 shows that the correct solutions found on 50 runs is quite similar, but Table 4 shows that on all problems except Super Anagrams the semantic approach improved the average best fitness on the test dataset.

Table 3. Number of times correct solutions were found within 50 runs.

	Default	Semantic	Diff
Collatz Numbers	0	0	0
Compare String Lengths	5	2	-3
Grade	1	4	3
Number IO	48	48	0
Super Anagrams	9	11	2

Additionally, Figure 2 depicts the average best training fitness over 50 runs. The plots show that on average with SC better solutions are found in earlier generations. Due to the simplicity of Number IO and to make better use of the

Table 4. Average test fitness of the best training individual found over 50 runs

	Default	Semantic	Diff
Collatz Numbers	79852.54	79404.52	0.56%
Compare String Lengths	158.10	135.78	14.12%
Grade	1434.42	1012.14	29.44%
Number IO	14.59	0.06	99.61%
Super Anagrams	26.48	28.16	-6.34%

space available, plots of Number IO have been omitted. Statistical tests with Wilcoxon rank sum test on the best test fitness of the last generation does not show a significant difference on any of the problems.
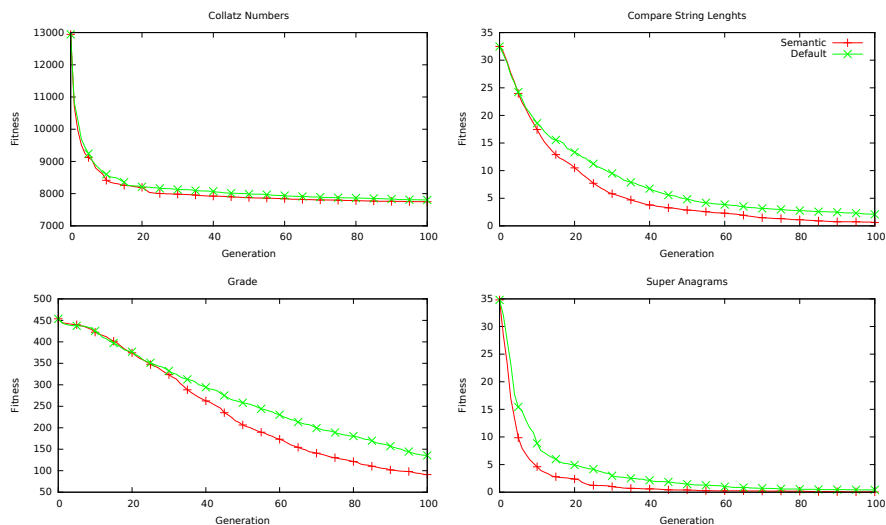
Fig. 2. Average best training fitness over 50 runs

5.2 Parent Comparison

The goal of SC is to promote semantic locality and exchange similar subtrees, but not equivalent ones. Therefore, this change should be visible in the child by having a different semantics than its parent. McPhee et al. noticed in the boolean domain that more than 50% of subtree crossover operations were not able to change the semantics [8] and Nguyen et al. reported that even though subtree crossover was able to change semantics in the regression domain in 60%-80% of the operation, semantic crossover was often 20% higher [13]. Figure 3 shows the

percentage of individuals that are different from their rooted parent. The rooted parent is the parent which removes a subtree to add the subtree from the second parent, therefore the child and the parent have the same root node. The plots in Figure 3 confirm that SC is more suited to create children whose semantics differ from their rooted parent. Additionally, Wilcoxon rank sum tests were conducted on each problem to confirm that the difference of the average percentage of semantically different children produced by SC is statistically significant to subtree crossover.
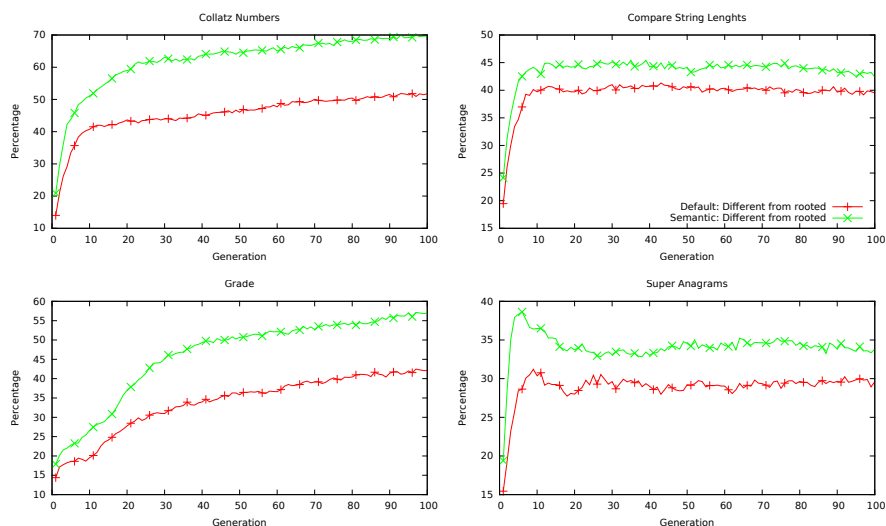
Fig. 3. Percentage of children semantically different from their rooted parent

More interesting than just if a child is different than a parent, is if a child is better than its parents. Figure 4 shows the percentage of children that have a better fitness than their rooted parent and a better fitness than both parents. For Collatz Numbers, SC is able to continually produce more children which are better than either the rooted parent or both parents compared to subtree crossover, although no correct solution was found for Collatz Numbers. In the case of Grade, SC is also able to produce more children which are better than their parents over all generations. SC and subtree crossover achieve similar percentages on Compare String Lengths and Super Anagram. Although SC initially does better than subtree crossover, it only achieves similar or slightly worse percentages later on. Again Wilcoxon rank sum tests was used for the statistical test, which shows that for Collatz Numbers, Grade and Super Anagrams the average percentage of children that are better than their rooted parent and both parents is significantly higher with SC than with subtree crossover. For Compare String Lengths the statistical test did not show a statistical significant difference.
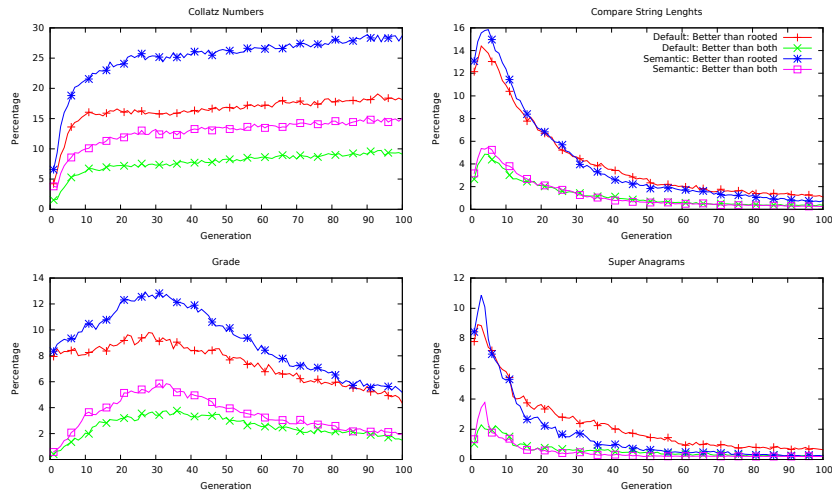
Fig. 4. Percentage of children that are better than rooted or better than both parents

## 5.3 Types Selected for Similarity Measurement

As described in Section 3.2, one data type of all available data types that shows a semantic difference is randomly chosen to measure semantic similarity. Figure 5 shows the percentages of the likelihood of selecting a certain type for the semantic similarity measure or if random crossover or no crossover was executed. As mentioned before, no crossover can happen, if the second parent does not contain a subtree of the same type as the selected node from the first parent or no subtree applies the node limits set. Random crossover happens if no semantic difference can be found with any data type on the specified number of subtrees selected.

As can be seen in all cases, the data type that has been most often chosen for calculating the semantic similarity is the data type that is used as return value and therefore has the most influence on the fitness. Although even for Grade the output data type is selected more often, the percentage does not increase as drastically as in the other cases, which might occur because GP is not able to improve the population as fast as on the other problems, as shown in Figure 2.

Another observation that can be made with the plots in Figure 5 is that the amount of random crossover is high on all problems, which happens because no semantic difference can be found with any data type. Super Anagrams is the problem with the highest amount of random crossover, which keeps random crossover around 50% over all generations, which might be due to the fast convergence on that problem. The high percentage of random crossover indicates two things. First, that many crossover operations are not able to produce individuals that are different from their parents, which correlates with the plots displayed in Figure 3. Second, that such a detailed semantic similarity measure as has been

used in this study might not be required. A relatively high number of subtrees that have been checked during the semantic crossover for program synthesis seem not to be able to create a semantically different individual. Adapting the crossover to using the first subtree that is semantically different instead of using the most semantically similar one might be sufficient and improve run time, which will be more similar to the original semantic similarity-based crossover proposed in [12]. This has been noted for future work, see Section 6. Obviously increasing the number of *Max_Tries* could also increase the number of times the semantic crossover finding a semantic difference, but that would increase run time.

Another interesting observation that can be made when looking at Figure 2 and 5 is that around the same generation as semantic crossover is using a specific type instead of falling back to random crossover, is around the same generation as fitness improves more with the semantic operator compared to subtree crossover.
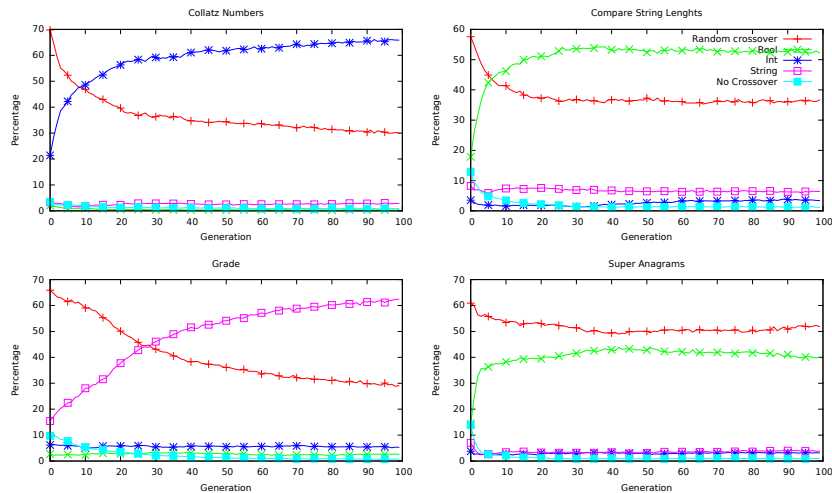
Fig. 5. Percentage of crossover of a specific type with Semantic similarity-based crossover

6   Conclusion & Future Work

In this study, a semantic similarity-based crossover was adapted to be able to use in the program synthesis domain. To this end, methods for semantic distance measure were proposed, which use the execution trace of a program, and the semantic crossover was applied to a suite of benchmark problems.

Semantic similarity crossover for program synthesis was able to produce more children that are semantically different from their parents as well as more children that are better than their rooted parent and both parents. Nevertheless,

that did not lead to better overall performance. A reason might be that that a high percentage of times the semantic crossover was still falling back to random crossover, if it does not find any semantic difference on any selected subtree.

As mentioned before, a simpler check for semantic similarity, like checking for any semantic difference, might be sufficient to improve performance and might reduce run time over the semantic similarity measure proposed in this study, which is part of future work. Additionally, adapting the crossover to consider multiple different crossover points in the first parent instead of a single one might also lead to finding semantic differences more often.

7 Acknowledgments

References

1. Beadle, L., Johnson, C.: Semantically driven mutation in genetic programming. In: Evolutionary Computation, 2009. CEC '09. IEEE Congress on. pp. 1336–1342 (May 2009)
2. Beadle, L., Johnson, C.: Semantically driven crossover in genetic programming. In: Wang, J. (ed.) Proceedings of the IEEE World Congress on Computational Intelligence. pp. 111–116. IEEE Computational Intelligence Society, IEEE Press, Hong Kong (1-6 Jun 2008), http://results.ref.ac.uk/Submissions/Output/1423275
3. Forstenlechner, S., Fagan, D., Nicolau, M., O'Neill, M.: A Grammar Design Pattern for Arbitrary Program Synthesis Problems in Genetic Programming, pp. 262–277. Springer International Publishing, Cham (2017), http://dx.doi.org/10.1007/978-3-319-55696-3_17
4. Forstenlechner, S., Nicolau, M., Fagan, D., O'Neill, M.: Introducing semantic-clustering selection in grammatical evolution. In: Johnson, C., Krawiec, K., Moraglio, A., O'Neill, M. (eds.) GECCO 2015 Semantic Methods in Genetic Programming (SMGP'15) Workshop. pp. 1277–1284. ACM, Madrid, Spain (11-15 Jul 2015), http://doi.acm.org/10.1145/2739482.2768502
5. Galván-López, E., Cody-Kenny, B., Trujillo, L., Kattan, A.: Using semantics in the selection mechanism in genetic programming: A simple method for promoting semantic diversity. In: Evolutionary Computation (CEC), 2013 IEEE Congress on. pp. 2972–2979 (June 2013)
6. Helmuth, T., Spector, L., Matheson, J.: Solving uncompromising problems with lexicase selection. IEEE Transactions on Evolutionary Computation 19(5), 630–643 (Oct 2015)
7. Helmuth, T., Spector, L.: General program synthesis benchmark suite. In: GECCO '15: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference. pp. 1039–1046. ACM, Madrid, Spain (11-15 Jul 2015), http://doi.acm.org/10.1145/2739480.2754769
8. McPhee, N.F., Ohs, B., Hutchison, T.: Semantic Building Blocks in Genetic Programming, pp. 134–145. Springer Berlin Heidelberg, Berlin, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-78671-9_12

9. Montana, D.J.: Strongly typed genetic programming. Evol. Comput. 3(2), 199–230 (Jun 1995), http://dx.doi.org/10.1162/evco.1995.3.2.199

10. Moraglio, A., Krawiec, K., Johnson, C.: Geometric semantic genetic programming. In: Coello, C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) Parallel Problem Solving from Nature - PPSN XII, Lecture Notes in Computer Science, vol. 7491, pp. 21–31. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-32937-1_3

11. Nguyen, Q.U., Nguyen, X.H., O'Neill, M.: Semantic aware crossover for genetic programming: The case for real-valued function regression. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009. LNCS, vol. 5481, pp. 292–302. Springer, Tuebingen (Apr 15-17 2009)

12. Nguyen, Q.U., Nguyen, X.H., O'Neill, M., McKay, R.I., Galvan-Lopez, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines 12(2), 91–119 (Jun 2011)

13. Nguyen, Q.U., Nguyen, X.H., O'Neill, M., McKay, R.I., Phong, D.N.: On the roles of semantic locality of crossover in genetic programming. Information Sciences 235, 195–213 (20 Jun 2013), http://www.sciencedirect.com/science/article/pii/S0020025513001175

14. Spector, L., Robinson, A.: Genetic programming and autoconstructive evolution with the push programming language. Genetic Programming and Evolvable Machines 3(1), 7–40 (2002), http://dx.doi.org/10.1023/A:1014538503543

15. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines 15(2), 195–214 (2014), http://dx.doi.org/10.1007/s10710-013-9210-0

16. Wagner, S., Kronberger, G., Beham, A., Kommenda, M., Scheibenpflug, A., Pitzer, E., Vonolfen, S., Kofler, M., Winkler, S., Dorfer, V., Affenzeller, M.: Advanced Methods and Applications in Computational Intelligence, Topics in Intelligent Engineering and Informatics, vol. 6, chap. Architecture and Design of the HeuristicLab Optimization Environment, pp. 197–261. Springer (2014), http://dx.doi.org/10.1007/978-3-319-01436-4_10

On the Use of Dynamic GP Fitness Cases in Static and Dynamic Optimisation Problems

Edgar Galván-López[1⋆], Lucia Vázquez-Mendoza[2],
Marc Schoenauer[3] and Leonardo Trujillo[4]

[1] Department of Computer Science, National University of Ireland Maynooth, Ireland
`edgar.galvan@mu.ie`
[2] School of Social Sciences and Philosophy, Trinity College Dublin, Ireland
`lucyvaz@gmail.com`
[3] TAU, INRIA and LRI, CNRS & U. Paris-Sud, Université Paris-Saclay, France
`marc.schoenauer@inria.fr`
[4] Posgrado en Ciencias de la Ingeniería, Instituto Tecnológico de Tijuana, México
`leonardo.trujillo@tectijuana.edu.mx`

Abstract. In Genetic Programming (GP), the fitness of individuals is normally computed by using a set of fitness cases (FCs). Research on the use of FCs in GP has primarily focused on how to reduce the size of these sets. However, often, only a small set of FCs is available and there is no need to reduce it. In this work, we are interested in using the whole FCs set, but rather than adopting the commonly used GP approach of presenting the entire set of FCs to the system from the beginning of the search, referred as static FCs, we allow the GP system to build it by aggregation over time, named as dynamic FCs, with the hope to make the search more amenable. Moreover, there is no study on the use of FCs in Dynamic Optimisation Problems (DOPs). To this end, we also use the Kendall Tau Distance (KTD) approach, which quantifies pairwise dissimilarities among two lists of fitness values. KTD aims to capture the degree of a change in DOPs and we use this to promote structural diversity. Results on eight symbolic regression functions indicate that both approaches are highly beneficial in GP.

1 Introduction

Normally, the fitness of Genetic Programming (GP) [9] programs is obtained by using a set of fitness cases: a fitness case is an input/output pair and the fitness of an individual is measured on how well it matches the output(s) from input(s).

Research on the use of fitness cases has primarily focused on how to reduce the number of these cases when running a GP system given that this is a major element that affects speed [6, 11, 19, 14].

⋆ Research conducted during Galván's stay at TAU, INRIA and LRI, CNRS & U. Paris-Sud, Université Paris-Saclay, France.

There are, however, some problems where only a few fitness cases are available for the GP system to work with. For instance, when dealing with highly binary unbalanced data for a classification task, the positive (minority) class has only a few cases and the use of all the available fitness cases is necessary [2, 4]. Other times, it may be the case that the dataset is highly contaminated by outliers and sampling is required to detect true examples of the system [12].

In this work, rather than using only a subset of fitness cases from the entire set [6, 11, 19], we are interested in using them all in a way to make the search more robust. To do so, we propose an approach called dynamic fitness cases, wherein cases are built by aggregation over generations instead of using the commonly adopted approach of using them all from the beginning of the search.

Moreover, there is no study that has focused its attention on the study of fitness cases on dynamic optimisation problems (DOPs). These are problems that are solved online by an optimisation algorithm as time progresses [16]. This work uses both static problems and DOPs to test the proposed approach.

Multiple elements have been reported to be beneficial in DOPs (see [16] for a detailed analysis on the area). One key element is diversity. This is a key element of the biological theory of natural selection and it is used in EAs to describe, for instance, structural variety, and it is expected that an EA with a mechanism to promote diversity will greatly improve its performance [16].

Diverse approaches have been proposed to promote diversity in EAs. One commonly adopted approach is the replacement of individuals in a population by newly generated genetic material. However, a common element observed when doing so is that frequently researchers use an arbitrary approach to decide the number of individuals that need to be replaced [13, 17, 21]. However, this process is purely intuitive and often expensive due to its trial-and-error nature.

To address this issue, we also use a mechanism to make a more informed decision to determine the proportion of individuals that need to be replaced in DOPs. To this end, we use the fitness values of individuals as indicators to determine how big/small a change is, and consequently, use this information to determine, for instance, the number of individuals in a population that need to be replaced by new individuals. Any population-based EA can adopt our proposed approach and in this work, as stated previously, we use a GP system.

Thus, the main contributions of this work are: (a) the use of dynamic fitness cases, wherein cases are built gradually over time to make the GP search more amenable, (b) to study for the first time the impact that fitness cases have in DOPs, and to this end, we also use the use of pair-wise fitness disagreements, based on the Kendall Tau distance, as a metric to promote diversity, which has constantly been reported beneficial in DOPs [16].

2 Related Work

2.1 Fitness Cases in Genetic Programming

As discussed previously, the fitness of a GP individual is normally computed by using a set of fitness cases and the way it is used is highly important in GP. The

size of the fitness cases vary e.g., in [15] the authors reported problems that use a small size of fitness cases up to dozens of thousands of fitness cases.

To decide on the number of fitness cases that a GP system may need to use, the rational allocation of trials algorithm can be a good alternative [19]: before a new generation takes place, individuals are evaluated using only a fraction of all the fitness cases available to the system. GP programs are further evaluated on new fitness cases when e.g., there is a possibility of winning some selection mechanism (e.g., tournament selection) that they are losing.

Another approach to determine the necessary number of fitness cases to solve a given problem in GP is that based on well-known statistical and information-theoretic considerations e.g., Central Limit Theorem and entropy of random variables [6]. The authors tested their theoretical framework on discrete fitness-valued cases and showed that their estimations agree with experimental results. Specifically, they showed that when the GP system uses at least the estimated number of fitness cases yield by their approach, the system achieves reliable results and the opposite is true when a lower number of fitness cases is used.

When having a large number of fitness cases, it may be necessary to adopt a mechanism to determine how many and which cases to use. Multiple works have been proposed. For instance, a topology-based mechanism [11] promotes the use of certain fitness cases based on how well or bad these are solved by individuals; historical subset selection [5] uses part of the set of all fitness cases based on how well the elitist individual is able to solve them; active data selection [22] uses small training case sizes and during search, these subsets are recombined and enlarged by a few fitness cases taken from the entire set.

Other sampling methods have been proposed that use a single fitness case in some generations and the entire training set is used in others. Such is the case for Interleaved Sampling and Random Interleaved Sampling [7]. More recently, the Lexicase selection algorithm has been proposed [18], wherein fitness cases are randomly shuffled at each parent selection event, and the best performing individual on the first fitness cases is kept. The method was extended to real-valued problems [10], with performance improving in almost all cases. An evaluation of some of these techniques, as well as others, is reported in [14].

In this work, we take a different approach: rather than determining how many FC the GP system should use, we use them all. The rationale for doing so it is because often there are a few fitness cases available to the system. The novelty of our approach is that instead of presenting all the cases to the system as traditionally done in GP, we build by aggregation these fitness cases over time with the hope to make the search more amenable. Moreover, as indicated before, there are no studies on the impact of fitness cases in DOPs and this works also considers this scenario. It is well-known that diversity plays an important role in evolutionary search, in general, and in DOPs in particular [16]. We present some works on this area next.

2.2 Promoting and Maintaining Diversity

Multiple works have been proposed to promote and maintain diversity in EAs. In this section, we focus only on DOPs tackled by GP (see [16] for a more

general discussion on the subject). Among those approaches proposed to promote diversity in the face of DOPs using GP are: (a) adaptable genetic operators, (b) behavioural diversity, and (c) injection of new genetic structural material. In this work, we only focus on the latter and briefly discuss some approaches that have been proposed to promote diversity via the injection of new individuals.

One of the easiest forms of promoting diversity is adopting the injection of new genetic material into the GP population. The generation of GP individuals is done by using common techniques, like the adoption of the ramped half-and-half method [9]. This can take place when, for instance, detecting a change [13] or when bloat (dramatic increase of tree sizes as evolution proceeds) reaches a limit and there is a need to substitute individuals contained in the population by new GP programs [21]. Injecting new GP individuals into the population has also been promoted via culling [17]. That is, removing the worst individuals and replacing them by randomly generated programs. Variable population size [20] also promotes diversity by adding new GP individuals into the population.

A common element in all these works that promote structural diversity is that the number of individuals to be replaced by the same number of newly created individuals is chosen rather arbitrarily. Next, we present an approach that aims to overcome this limitation.

3 Proposed Approaches

As discussed previously, we are interested in making the GP search more amenable and to do so we propose a dynamic fitness cases approach, wherein cases are built by aggregation over time. We test this approach in both static and DOPs, and for the latter, we also use the adoption of the Kendall Tau Distance (KTD) that quantifies pairwise dissimilarities among two lists of fitness values with the hope to make a better informed decision in terms of the number of individuals that need to be replaced in a population by new individuals to promote diversity.

3.1 Dynamic Fitness Cases

To make the GP search more amenable, we build the fitness cases over time. More specifically, at the beginning of an evolutionary run or just after a change has occurred (for the dynamic setting), we use in order a subset of fitness cases, $C_{g=0}$ which is chosen from all the fitness cases C^N of size N,

$$C_{g=0} \subset C^N, |C_{g=0}| = k \qquad (1)$$

where k is a constant and $k < N$. After a few i generations another k fitness cases of the C^N fitness cases are added to $C_{g=0}$,

$$C_{g=0} \cup C_{g=i}, C_{g=0} \cap C_{g=i} = \{\} \qquad (2)$$

We continue this process until all the fitness cases have been used. Thus, the complete sequence of fitness cases is build as follows,

$$C_{g=0} \cup C_{g=i} \cup \cdots \cup C_{g=M} = C^N \qquad (3)$$

where M is a constant and $M < K$, where K is either the maximum number of generations or the number of generations that are necessary for a change to take place (for the dynamic scenario). By defining the latter, we guarantee that the GP system accounts for all the fitness cases before a change takes place and it has all the necessary elements to, potentially, find the solution. The values of the variables are defined in Table 2 and discussed in Section 5.

3.2 Kendall Tau Distance

As indicated before, there is no study that has focused its attention on the study of fitness cases in a dynamic setting and this work also considers such scenario.

As seen in Section 2, we know that there is strong evidence indicating that the adoption and/or encouragement of diversity in GP search on DOPs is highly beneficial. Normally, when adopting this type of diversity, researchers have focused their attention on setting arbitrarily a number of individuals to be generated and then used them to e.g., replace the worst GP individuals in a population. The major drawback with this approach is that often this process is based on trial and error and can be computationally expensive.

We believe that it is possible to adopt a more informed way of determining the number of individuals that should be replaced from a population by using fitness values. The use of these values as indicators to perform a specific task (e.g., prediction of problem hardness) is common in EAs. The most well-known example of this is the fitness-distance correlation [8], where these values are used in conjunction with a metric that informs us how distant two individuals are in the search space to determine problem difficulty.

In this work, we use a distance, studied in the first author's works [1, 3], that accounts for pairwise disagreements between two lists of ranked fitness values. We hope that these disagreements can inform us on whether an evolved population is useful in the face of a change. Our proposed approach works in three phases:

1. Firstly, it is necessary to account for a method that can indicate when a change is about to take place. We do this in a non-expensive manner: before a new generation is about to take place, we use one individual (the elitist individual), whose fitness (f_e^g) is assessed again in the next generation ($g+1$).
2. Secondly, if f_e^g and f_e^{g+1} are different, then we regard this as a change in the environment and we then proceed to compute the KTD (defined in Eq. 4) between the ranking of the fitness values of all individuals at generation g and the next generation ($g+1$). This distance counts the number of pairwise disagreements between two ranked lists and it is normalised by the maximum number of possible disagreements. This distance gives a discrete value $k = [0, 1]$ and this is used to generate a percentage of T new individuals with respect to the population size.
3. Thirdly, the worst (less fit) individuals at $g+1$ are replaced by the newly generated individuals (using ramped half-and-half initialisation method, details are discussed in Section 4) keeping the size of the population constant.

The KTD between two ranked lists is defined as,

Table 1. Symbolic regression benchmarks problems used in our work.

Function	Objective function
f_1	$x^3 + x^2 + \alpha x$
f_2	$x^4 + x^3 + x^2 + \alpha x$
f_3	$x^5 + x^4 + x^3 + x^2 + \alpha x$
f_4	$x^6 + x^5 + x^4 + x^3 + x^2 + \alpha x$
f_5	$\sin(x^2) \cos(\alpha)$ - 1
f_6	$\sin(\alpha x) + \sin(x + x^2)$
f_7	$\log(\alpha x + 1) + \log(x^2 + 1)$
f_8	$\mathrm{sqrt}(\alpha x)$

Table 2. Summary of Parameters.

Parameter	Value
Population Size	800
Generations	200
Type of Crossover	Any node
Crossover Rate	0.80
Type of Mutation	Subtree
Mutation Rate	0.20
Selection	Tournament (size = 7)
Initialisation Method	Ramped half-and-half
Initialisation Depths:	
Initial Depth	2
Final Depth	5
Maximum Length	1200 nodes
Maximum Final Depth	8
Independent Runs	50
Changes	Every 50 generations
Dynamic Fitness Cases	$k = 1$, $i = 2$, $M = 39$

$$k(\tau_1, \tau_2) = \sum_{(i,j) \in P} \bar{k}_{i,j}(\tau_1, \tau_2) \tag{4}$$

where, P is the set of pairs of elements in τ_1 and τ_2, $\bar{k}_{i,j}(\tau_1, \tau_2) = 0$ if i and j are in the same order in both τ_1 and τ_2; and 1 if i and j are in opposite order.

It is worth mentioning that when the change to the objective function is monotonically increasing (order preserving), the computed KTD will be 0. This is a good property because in this case the evolved individuals are expected to behave well in the changed objective function, so there is no need to replace individuals. A mirror image is seen in the presence of a monotonically decreasing change of the objective function, which will yield the maximal normalised distance of 1, meaning that the order of both fitness lists is completely different. The latter will indicate that our approach based on the KTD will replace the entire population by newly generated genetic material.

4 Experimental Setup

To test our approach, we use eight symbolic regression functions of various difficulties, shown in Table 1. The fitness function is computed as the sum of absolute errors of the Euclidean distance to the output vector of the target uni-variate function queried on 20 inputs in the equally drawn range [1, 1]. A solution is regarded as correct when its fitness is less than a threshold set at 0.01. The function set is F = {+,-,*,/}, where / is protected division.

To test separately and in conjunction our two approaches, we use a static and a dynamic setting. We define three different type of changes for the latter: we use α as a variable (see Table 1) that can be tuned to achieve this along with a constant L, set at 50, that denotes when α changes to simulate a change (in this work, the maximum number of generations is set at 200, hence only three values for α are required for a dynamic setting, as defined next). For the static

Table 3. Success rate (%) and avg. of best fitness using either static (SFC) or dynamic fitness cases (DFC). No changes take place during evolution. All the results on the avg. of best fitness are statistical significant (Wilcoxon Test at 95% level of significance). Higher is better.

Function	Success Rate		Avg. Best Fitness	
	SFC	DFC	SFC	DFC
f_1	92.0%	100.0%	0.9371	1.0000
f_2	54.0%	88.0%	0.6656	0.9969
f_3	18.0%	70.0%	0.4501	0.9915
f_4	4.0%	72.0%	0.3280	0.9895
f_5	0.0%	60.0%	0.4580	0.9896
f_6	0.0%	64.0%	0.3438	0.9893
f_7	0.0%	36.0%	0.4988	0.9739
f_8	0.0%	16.0%	0.3068	0.9665

scenario, $\alpha = 1$. For the dynamic setting, we define a smooth, an 'abrupt' and a random change, where $\alpha = \{0.9, 0.8, 0.7\}$, $\alpha = \{0.1, 0.9, 0.1\}$, and finally, α is set with a random value between 0 and 1 every L generations, respectively.

For comparative purposes, we use a static fitness case-scenario and our proposed dynamic fitness case-approach, where all the cases are presented to the system at the beginning of the search as commonly adopted in the GP community and where the cases are built over time, respectively.

Moreover, for the DOPs defined in this work, we use an arbitrary approach, wherein the number of individuals to be replaced in a population is generated randomly and compared it against the results yield by our proposed Kendall Tau distance. We generate the individuals in these two approaches using the ramped half-and-half method, where the initial and final depths used are the same as when generating the population (see Table 2).

The experiments were conducted using a generational approach. The parameters used are shown in Table 2. To obtain meaningful results, we performed an extensive empirical experimentation (50 * 2 * 3 * 8 runs, plus 50 runs for each fitness case scenario: static and dynamic; 2,500 independent runs in total)[5].

5 Results and Discussion

5.1 Performance on a Static Setting

Let us analyse the results when using our proposed dynamic fitness cases (DFC), wherein cases are built over time, where one fitness case is added every two generations until all of them have been used, as defined at the bottom of Table 2, (see Section 3 for details on how this works) and compared the results obtained by DFC against the widely adopted mechanism of using all the fitness cases at the beginning of the search, denominated in this work as static fitness cases (SFC).

Table 3 shows the success rate (shown in the 2nd and 3rd column, from left to right), defined as the number of times that the GP system was able to find

---

[5] 50 independent runs, 2 types of replacement of individuals (arbitrary, Kendall tau distance-based), 3 types of changes, 8 problems.

the solution and the average of the best fitness at the end of each independent run (shown in the last two columns).

It is clear to see that the proposed DFC achieves good results in terms of finding the solution, as indicated in Table 3. The traditional SFC has a good performance only on the relatively easy f_1 and its performance decreases significantly with the rest of the functions used in this work, where SFC is not able to find a single solution for functions f_5, f_6, f_7 and f_8 in any of the independent runs. Our proposed DFC, on the other hand, achieves better results e.g., 60%, 64%, 36% and 16% for functions f_5, f_6, f_7 and f_8, respectively.

The results shown in the last two columns of Table 3, which are the average of the best individuals' fitness values at the end of each run, are aligned to the performance achieved by SFC and FDC. These results are all statistically significant, Wilcoxon Test set at 95% level of significance.

5.2   Performance on a Dynamic Setting

Let us first focus our attention on the performance achieved by the static and the dynamic function case-based approach, when these two are now used in conjunction with our proposed Kendall Tau Distance (KTD) approach to promote diversity in three type of changes: smooth, random and 'abrupt' change, as defined in Section 4.

These results, shown in Table 4, are similar to those discussed above: the SFC approach has a poor performance: less than 3.0%, for functions f_5 - f_8, defined in Table 1, regardless of the type of change used. These results are significantly better when using the proposed DFC in conjunction with the KTD approach. For example, the proposed approach achieves more than 48%, 67%, 59% and 52% for the same referred functions, respectively, regardless of the change used.

The average of best fitness values just before a change takes place, defined at every 50 generations, is shown in Table 5. These results (all statistically significant, Wilcoxon Test set at 95% level of significance) are aligned to the performance discussed above: the average fitness values is poor when using SFC compared to those results achieved by DFC. For example, for f_3 the average fitness values achieved by our proposed DFC is around 0.93 (almost three times better compared to the results yield by SFC), regardless of the change used.

Now, let us discuss the results when using the commonly adopted approach of replacing a random number of individuals from a population to promote diversity, referred in this work as the arbitrary approach. These results are shown in Tables 6 and 7 for the percentage of success rate and the average of the best fitness values just before a change occurs, respectively. In these tables, we can observe a similar scenario compared to what we discussed when using the KTD for an informed way to replace a number of individuals. That is, the SFC approach yields significantly worse results compared to DFC.

If we now compare, for instance, the performance achieved by the KTD and the arbitrary approach focusing on either using static fitness cases or using dynamic fitness cases, we do not see much difference. For example, the performance for the function f_5 is 49.0% (Table 4) and 47.0% (Table 6), using the KTD approach and the arbitrary approach in the presence of a smooth change,

Table 4. Percentage of success rate using both static and dynamic fitness cases on eight different regression functions in the presence of three different types of changes. Replacement type used: <u>Kendall</u> approach.

Function	Smooth Change		Random Change		Abrupt Change	
	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases
f_1	21.5%	25.0%	24.5%	33.5%	21.5%	29.0%
f_2	10.0%	92.5%	11.0%	90.0%	10.5%	91.5%
f_3	2.5%	79.0%	4.5%	89.0%	2.5%	82.0%
f_4	0.5%	87.0%	1.5%	86.0%	0.5%	90.5%
f_5	0.0%	49.0%	0.0%	60.5%	0.0%	57.0%
f_6	0.0%	68.0%	0.5%	.80.5%	0.0%	77.5%
f_7	0.0%	76.0%	0.0%	.80.0%	2.5%	60.0%
f_8	0.0%	53.0%	0.5%	64.0%	1.0%	61.0%

Table 5. Avg. of best fitness values at every 50th generation (just before a change takes place) using both static and dynamic fitness cases on eight symbolic regression functions in the presence of three different types of changes. Replacement type used: <u>Kendall</u> approach. All the results are statistical significant (Wilcoxon Test at 95% level of significance). Higher is better.

Function	Smooth Change		Random Change		Abrupt Change	
	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases
f_1	0.482	0.7594	0.5444	0.8115	0.5574	0.7583
f_2	0.4245	0.9572	0.4913	0.9566	0.5088	0.9648
f_3	0.3372	0.9457	0.3886	0.9547	0.4420	0.9508
f_4	0.3332	0.9435	0.3862	0.9512	0.4218	0.9589
f_5	0.3326	0.8901	0.4153	0.9129	0.4122	0.9156
f_6	0.3099	0.9218	0.3939	0.9397	0.4470	0.9401
f_7	0.4639	0.9249	0.5006	0.9312	0.5092	0.9042
f_8	0.3288	0.8831	0.4113	0.9010	0.4453	0.8968

Table 6. Percentage of success rate using both static and dynamic fitness cases on eight different regression functions in the presence of three different types of changes. Replacement type used: <u>Arbitrary</u> approach.

Function	Smooth Change		Random Change		Abrupt Change	
	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases
f_1	21.5%	25.0%	24.5%	35.0%	21.5%	33.5%
f_2	10.0%	90.0%	11.0%	92.5%	10.5%	95.0%
f_3	2.5%	84.0%	3.5%	85.0%	2.5%	85.0%
f_4	0.5%	89.5%	1.5%	87.5%	0.5%	90.0%
f_5	0.0%	47.0%	0.5%	61.0%	0.5%	56.0%
f_6	0.0%	73.5%	0.5%	80.5%	0.5%	85.5%
f_7	0.0%	75.0%	0.5%	74.5%	6.5%	71.0%
f_8	0.0%	55.5%	0.0%	67.5%	1.0%	66.5%

Table 7. Avg. of best fitness values at every 50th generation (just before a change takes place) using both static and dynamic fitness cases on eight symbolic regression functions in the presence of three different types of changes. Replacement type used: <u>Arbitrary</u> approach. All the results are statistical significant (Wilcoxon Test at 95% level of significance). Higher is better.

Function	Smooth Change		Random Change		Abrupt Change	
	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases
f_1	0.4793	0.7714	0.567	0.8099	0.5889	0.8337
f_2	0.4288	0.9561	0.5089	0.9638	0.5160	0.9701
f_3	0.3337	0.9477	0.4059	0.9521	0.4705	0.9563
f_4	0.3439	0.9514	0.3887	0.9538	0.4394	0.9628
f_5	0.3444	0.8927	0.4228	0.9098	0.4257	0.9202
f_6	0.3478	0.9342	0.4108	0.9424	0.4747	0.9514
f_7	0.4675	0.9285	0.5137	0.9296	0.5397	0.9214
f_8	0.3282	0.8863	0.4149	0.9121	0.4622	0.9072

respectively. The same trend is observed for the rest of the functions regardless of the type of change used. However, the benefit of using the KTD in DOPs instead of using an arbitrary approach (random number of individuals replaced in a population), as normally adopted in EAs DOPs when promoting diversity via the replacement of individuals, can be observed when analysing the number of individuals created by either approach. We discuss this next.
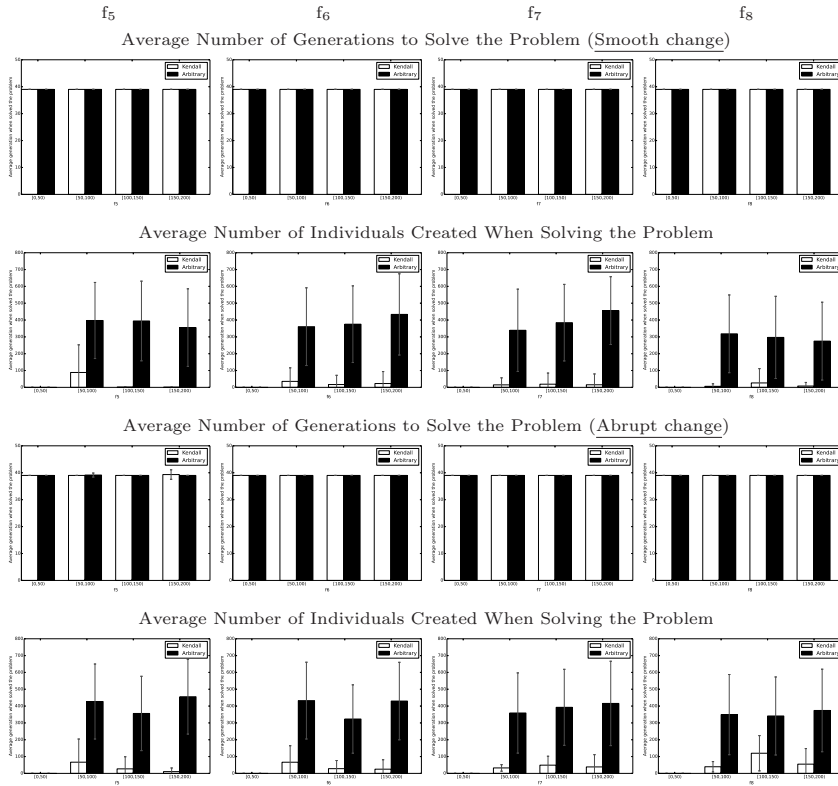
Fig. 1. Average number of generations to solve a problem (odd rows) and average number of created individuals (even rows) along with standard deviation using either the arbitrary approach (black-filled rectangle) or our proposed Kendall approach (white-filled rectangle) for functions f_5 - f_8 when using dynamic fitness cases. Notice that for the avg. number of individuals created, the first generations [0,50), show nothing given that a change occurs after this.

5.3 Analysis of the Number of Created Individuals

To see the benefit of using the proposed KTD approach in DOPs compared to the arbitrary approach to promote diversity via the replacement of individuals in the population by new genetic material, it is necessary to see the number of individuals created by each of these two approaches. This is shown in the second and fourth rows, from top to bottom, of Figure 1, for functions f_5 - f_8, where the vertical line denotes the standard deviation. Due to space constraints, we only show the results when using the DFC approach on these functions and in the presence of a smooth and an 'abrupt' change that yield better results compared to the SFC approach. However, a similar trend was observed for the rest of the functions and type of change.

Let us discuss a particular example: when a smooth change takes place for functions f_5 - f_8, shown in the second row of Figure 1. It is clear to see that the number of individuals created by the KTD, shown by a white-filled bar is significantly lower compared the number of GP programs created by the arbitrary approach, shown by a black-filled bar. This is to be expected since a smooth change took place and our proposed approach was able to capture this by creating a few individuals in the presence of this type of change. The same trend is observed for the rest of the functions (not shown due to space constraints).

Moreover, we can see that the KTD approach is able to capture the level of a change. For instance, see the number of created individuals in the presence of an smooth change *vs.* an 'abrupt' change, as denoted in the second and fourth row of Figure 1: the KTD approach creates less number of individuals in the presence of a smooth change compared to the 'abrupt' change. Although the difference of created individuals in the presence of either these two changes is small. This is due to the type of changes proposed in this work (see Section 4) rather than the KTD approach failing at capturing a change. To illustrate this, we adopted a more radical change where the last sign in f_1 changes every 50 generations (from '+' to '-' and *vice versa*) to simulate a DOP and compare this result against those yield by the other three types of changes. This is shown in Figure 2, where we can clearly see that the KTD yields values accordingly: the number of created individuals is increased as the change is more severe.

In addition to this, we also analyse the average number of generations required to solve a problem. This is shown in the first and third row in Figure 1. Interestingly, we can see that the majority of problems, regardless of the change used, are solved once all the fitness cases are presented to the system: observe how they finish before generation 40 (recall that all fitness cases are presented to the GP system at generation $M = 39$ as indicated in Table 2), with a few runs finding the solution just after generation 40 as denoted by the small standard deviation (see, for example f_5, in the presence of an abrupt change, third row first column of Figure 1).

5.4 Size of GP Programs

We have learnt that DFC behaves better than the widely-adopted SFC in GP. We believe that the reason is due to the fact that GP system gradually solves
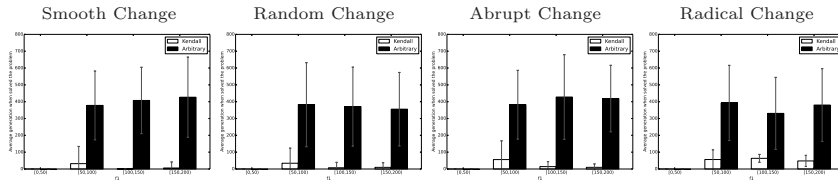
Fig. 2. Average number of created individuals along with standard deviation using either the arbitrary approach (black-filled rectangle) or our proposed Kendall approach (white-filled rectangle) for function f_1 when using static fitness cases. Notice that for the average of created individuals, the first generations [0,50) show nothing, given that a change occurs after this.

the problem in accordance to the proposed DFC, wherein fitness cases are built by aggregation over time (see Section 3). This in consequence could mean that the GP program gradually starts growing as more cases are presented to the system. Indeed, this is what can be observed in Fig. 3, where we report the average length of individuals, along with the standard deviation, on f_3 and f_4 using the arbitrary replacement approach and an abrupt change (the same trend is observed for the other two type of changes and replacement mechanisms not shown due to space constraints). It is evident that the size of programs created by the DFC approach, denoted by grey lines, is significantly lower compared to the traditional SFC approach, indicated by black lines.
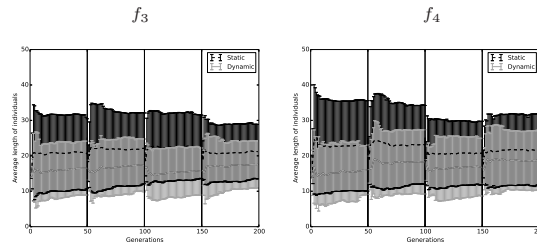
Fig. 3. Average (along with a standard deviation) length of programs using both static and dynamic fitness cases, shown in black and grey lines, respectively, on f_3 and f_4, using the arbitrary replacement approach. Vertical lines at every 50th generations indicate an (abrupt) change.

6   Conclusions

Traditionally, the fitness value of a GP program is computed by using a set of fitness cases. It is common that all the fitness cases are presented to GP from the beginning of the search, an approach we call static fitness cases. In this

work, we propose a dynamic fitness cases approach, wherein the cases are built by aggregation over time, making it an incremental search. We showed that the proposed approach achieves better better performance, in some problems achieving a 60% success rate compared to 0% achieved by the standard approach.

Furthermore, we tested these two approaches in the presence of dynamic changes, where the results achieved by the DFC are consistently better compared to the SFC. Moreover, we also showed how the DFC approach encourages a smooth increase of GP trees compared to SFC where the size of trees are bigger.

Finally, we also studied the impact/use of fitness cases in DOPs, where the adoption of diversity has consistently been reported as beneficial. To this end, we proposed an approach based on the Kendall Tau Distance that aims to capture the degree of a change in a dynamic setting and we use this consequently to determine the proportion of individuals that need to be replaced to promote structural diversity. We compared this against the commonly adopted arbitrary approach where the number of individuals is set randomly. We showed that the performance of both replacement mechanisms is similar, with the added benefit that the proposed KTD approach creates only the necessary individuals with regards to the amount of change.

References

1. E. Galván-López and O. Ait ElHara. Using fitness comparison disagreements as a metric for promoting diversity in dynamic optimisation problems. In *IEEE Symposium Series on Computational Intelligence*. Springer, 2016.
2. E. Galván-López, E. Mezura-Montes, O. Ait ElHara, and M. Schoenauer. On the use of semantics in multi-objective genetic programming. In J. Handl et al., editors, *Parallel Problem Solving from Nature – PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*, pages 353–363. Springer, 2016.
3. E. Galván-López, L. Vázquez-Mendoza, M. Schoenauer, and L. Trujillo. Dynamic GP fitness cases in static and dynamic optimisation problems. In P. A. N. Bosman, editor, *Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings*, pages 227–228. ACM, 2017.
4. E. Galván-López, L. Vázquez-Mendoza, and L. Trujillo. Stochastic semantic-based multi-objective genetic programming optimisation for classification of imbalanced data. In O. Pichardo-Lagunas and S. Miranda-Jiménez, editors, *Advances in Soft Computing*, chapter 22, pages 261–272. Springer, 2016.
5. C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In Y. Davidor et al., editors, *Parallel Problem Solving from Nature III*, volume 866 of *LNCS*, pages 312–321, Jerusalem, 9-14 Oct. 1994. Springer-Verlag.
6. M. Giacobini, M. Tomassini, and L. Vanneschi. Limiting the number of fitness cases in genetic programming using statistics. In *Parallel Problem Solving from Nature VII*, pages 371–380, London, UK, 2002. Springer-Verlag.

7. I. Gonçalves and S. Silva. Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In K. Krawiec et al., editors, *Genetic Programming*, volume 7831 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin Heidelberg, 2013.

8. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

9. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

10. W. La Cava, L. Spector, and K. Danai. Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 741–748, New York, NY, USA, 2016. ACM.

11. C. W. G. Lasarczyk, P. W. G. Dittrich, and W. W. G. Banzhaf. Dynamic subset selection based on a fitness case topology. *Evol. Comput.*, 12(2):223–242, June 2004.

12. U. López, L. Trujillo, Y. Martinez, P. Legrand, E. Naredo, and S. Silva. Ransac-gp: Dealing with outliers in symbolic regression with genetic programming. In J. McDermott et al., editors, *Genetic Programming: 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*, pages 114–130, Cham, 2017. Springer International Publishing.

13. J. Macedo, E. Costa, and L. Marques. *Genetic Programming Algorithms for Dynamic Environments*, pages 280–295. Springer, Cham, 2016.

14. Y. Martnez, E. Naredo, L. Trujillo, P. Legrand, and U. Lpez. A comparison of fitness-case sampling methods for genetic programming. *Journal of Experimental & Theoretical Artificial Intelligence*, pages 1–22, 2017.

15. J. McDermott et al. Genetic programming needs better benchmarks. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 791–798, New York, NY, USA, 2012. ACM.

16. T. T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm Evol. Comput.*, 6:1 – 24, 2012.

17. M. Riekert, K. M. Malan, and A. P. Engelbrect. Adaptive genetic programming for dynamic classification problems. In *Proceedings of the Eleventh Conference on Congress on Evolutionary Computation*, CEC'09, pages 674–681, Piscataway, NJ, USA, 2009. IEEE Press.

18. L. Spector. Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, GECCO Companion '12, pages 401–408. ACM, 2012.

19. A. Teller and D. Andre. Automatically choosing the number of fitness cases: The rational allocation of trials. In J. R. Koza et al., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 321–328, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

20. L. Vanneschi and G. Cuccu. A study of genetic programming variable population size for dynamic optimization problems. In *IJCCI*, pages 119–126, 2009.

21. N. Wagner, Z. Michalewicz, M. Khouja, and R. R. McGregor. Time series forecasting for dynamic environments: The dyfor genetic program model. *IEEE Transactions on Evolutionary Computation*, 11(4):433–452, Aug 2007.

22. B.-T. Zhang and D.-Y. Cho. Genetic programming with active data selection. In *Simulated Evolution and Learning on Simulated Evolution and Learning*, SEAL'98, pages 146–153, London, UK, 1999. Springer-Verlag.

Session 4 - Cooperation-Coevolution

MEMSA: a robust *Parisian EA* for multidimensional multiple sequence alignment

Julie D. Thompson, Renaud Vanhoutrève and Pierre Collet

[thompson,collet]@unistra.fr, vanhoutreve.renaud@gmail.com

ICube laboratory, UMR CNRS 7357, Strasbourg University, France
Fédération de Médecine Translationnelle de Strasbourg,
CS-DC UNESCO UniTwin http://cs-dc.org

Abstract. This paper describes a new approach for the multiple alignment of biological sequences (DNA or proteins) using a Parisian Evolution approach called MEMSA, for *Multidimensional Evolutionary Multiple Sequence Alignment*, coded using the EASEA platform. This approach evolves individual sub-alignments called "patches" that are used to create a new kind of Multiple Sequence Alignment where alternative solutions are computed simultaneously using different fitness functions. Solutions are generated by combining coherent sets of high-scoring individuals that are used to reconstruct multi-dimensional multiple sequence alignments. The alignments of this prototype version show a quality comparable to ClustalW (one of the most widely used existing methods) on the 218 samples of the BAliBASE benchmark in reasonable time.

1 Introduction

The incredible increase in the output of Next generation Genome Sequencing (NGS) technologies in the recent years is making sequence data analysis a major bottleneck for the biologist. New bioinformatics solutions are needed to allow end-users to fully exploit the progress of these technologies in various applications, including genome annotation, analysis of genetic mutations, evolutionary studies, or the characterization of gene products (*e.g.* proteins). Proteins are large macromolecules, consisting of one or more long chains of amino acid residues. They perform a wide variety of biological functions in organisms, from catalysis of biochemical reactions, transport of nutrients or recognition and transmission of signals to structural and mechanical roles within the cell. As a consequence, one of the most important applications of bioinformatics has been the study of the relationships between the sequence of a protein and its 3D structure, cellular function and evolution.

In this context, protein multiple sequence alignments (MSA) play a central role in comparative analyses of the data produced by NGS. Recently, new methods for the construction of MSA (such as MAFFT[10], MUSCLE[5], KALIGN[14], PROBCONS[4]) have been developed that use heuristic approaches that are fast enough to handle this "big data" and that allow comparison of sequences from hundreds of diverse organisms. However, the current flood of data also poses

other challenges, in addition to the obvious scalability issues. For example, large protein families are often complex, with multidomain architectures, long unstructured (natively disordered) regions, splicing variants, *etc.* In addition, the new sequences are mostly predicted by automatic methods and contain many sequence errors. A recent comparative study of MSA algorithms [19] showed that the current methods can identify most of the shared sequence features that determine the broad molecular functions of a protein family, such as the 3D structure or catalytic sites, that have been conserved throughout evolution. However, the locally conserved regions, that reflect functional specificities or that modulate a protein's function in a given cellular context, are less well aligned. The complexity of the problem means that new MSA representations are now crucial. This motivated us to develop MEMSA, a Multidimensional Evolutionary Multiple Sequence Alignment tool, a new MSA approach that exploits an alternative genetic algorithm called Parisian Evolution [2], in order to produce multi-dimensional multiple alignments depending on "patches" of interest for the biologist.

1.1 Multiple Sequence Alignment (MSA)

In the most general terms, a protein multiple sequence alignment represents a set of sequences using a single-letter code for each amino acid. Each horizontal row in the alignment represents a single sequence and structurally, functionally or evolutionarily equivalent amino acids are aligned vertically. During evolution, mutation events occur. They include *point mutations* (single amino acid changes) that appear as differing characters in an alignment column and *indels* (INsertions or DELetions) or *gaps*, generally represented by a "–" character in one or more of the sequences. Most MSA methods represent these events *via* two sets of parameters: an "amino acid substitution matrix" (*e.g.* BLOSUM62 [8]) that assigns scores to the alignment of each possible pair of amino acids and a "gap penalty" for the introduction of gaps in a sequence. However, one of the main challenges for MSA algorithms is that there is no such thing as a *single optimal alignment*. Indeed, many distance matrices have been proposed, that offer different metrics to evaluate the quality of an alignment [1].

1.2 Evolutionary Algorithms for MSA

One of the first genetic algorithms (GA) for multiple sequence alignment was SAGA (Sequence Alignment by Genetic Algorithm) [17]. SAGA tries to find an optimal MSA by creating a population of MSAs and allowing them to evolve based on a natural selection process that mimics biological evolution (with crossover and mutation operators). In this case, an individual in the population represents one complete solution to the problem, *i.e.* an individual is a multiple alignment of all sequences with gaps at given positions. Unfortunately, the results are not very good on today's complex problems, probably because it tries to tackle an unsolvable problem as a whole.

Since then, other multiple sequence alignment strategies based on GAs have been introduced that use better mutation operators to improve the efficiency

and the accuracy of the algorithms, *e.g.* [22] or [12]. Other attempts such as MSAGMOGA [11] have used multi-objective algorithms to take into account the multi-dimensionality of the problem. These methods show promising results in some specific cases, but they are generally too slow for large-scale alignment applications. An alternative approach involves the use of GAs to improve an initial population of alignments constructed with a heuristic algorithm, such as PHGA [16], or MOMSA [6].

In this paper, we propose a new evolutionary algorithm for MSA that does not attempt to provide the user with a single MSA option, but rather finds as many good "patches" as possible in the studied sequences, possibly using different metrics, which will be subsequently used to propose alternative alignments compatible with a chosen patch. The result could evoke paintings by Piet Mondrian, with patches of different colours that could correspond to patches computed with different metrics, cf. Fig. 1.
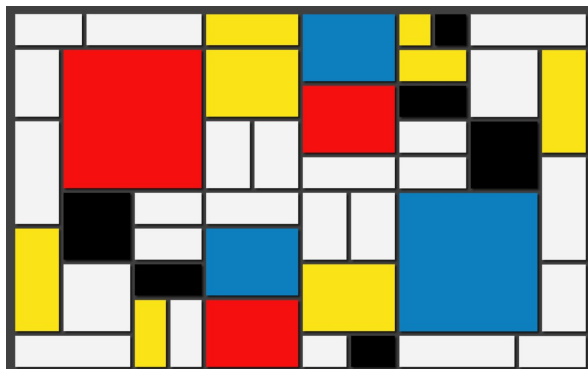

Fig. 1: A painting by Piet Mondrian (1872 - 1944)

1.3 Parisian Evolution Approach

The problem with all existing MSA approaches (evolutionary or not) is that they try to evolve individuals that represent a *complete MSA*, which is a conceptually impossible task, because is is ackowledged that there are several valid ways to align sequences. Therefore, using a global approach implies making a choice on the fitness function. Similarly to the Michigan Approach of Learning Classifier Systems [7] [9] [21], the Parisian approach [2] has been designed to: (i) decompose a large problem into smaller sub-problems that would be several orders of magnitude simpler to solve than the global one and (ii) reassemble the sub-problems into a global solution.

In the Parisian approach, an individual represents only a part of the solution and in the final stage of the algorithm, the global solution is reconstructed from

the individual parts. This has two advantages: (i) it reduces the computational requirements of the MSA algorithm, since an alignment is divided into smaller sub-problems, meaning that smaller individuals can be constructed and evaluated quickly. (ii) It also makes it possible to address the fact that there is no unique global fitness function for MSA. Indeed, different fitness functions exist, designed specifically for different protein regions.

Therefore, an evolutionary algorithm based on the Parisian Approach can evolve many good partial MSAs using different fitness functions that can be re-assembled differently, depending on what the biologist wants to study. MEMSA evolves individuals that represent good "patches", rather than complete solutions, that can be reassembled differently for different global evaluations.

Fig. 2: Individual representation in MEMSA

2 Genetic Algorithm with Parisian Approach for MSA

In this section we explain the evolutionary operators implemented in MEMSA.

2.1 Individuals / Patches

In MEMSA, individuals represent local alignments called "patches", containing a small number of "segments" of identical length (at least 2 amino acids) from different sequences, as depicted in Fig. 2.

2.2 Initialisation

As is usual in artificial evolution, initialisation is done using random values (within constraints) so as to avoid biases. Each individual in the initial population consists of short segments of 2 amino acids taken from 2–5 sequences.

2.3 Crossover

When running the algorithm, we observed that good individuals having more than 2 segments are very rare. Therefore, the crossover operator is used to favour the creation of larger individuals containing 3 or more segments. To this effect, the crossover associates segments coming from two parents that use the same metrics to generate a child that has more than two segments.

If both parents have an identical sequence in common then the child takes the segment of one of the parents for this sequence.

As seen in section 2.5 below, evaluation is very fast, as it is made up of very simple additions and multiplications. This means that evaluation is not the most time-consuming part of the algorithm. Therefore, it is a great advantage that this crossover is very simple, because it makes it also very quick, resulting in the possibility of testing more possibilities than if an "intelligent" crossover were used.

2.4 Mutator

Mutating an individual (a patch) may:

- add a segment to the individual (if < max number of segments),
- remove a segment from the individual (if the number of segments is > 3)
- shift the whole patch 0 to 10 amino acids to the right or to the left,
- shift one segment 0 to 10 amino-acid to right or to the left

An alignment is a succession of conserved and unconserved groups of columns. Shifting a good patch in any direction has a good chance of generating another good patch.

2.5 Evaluation

The evaluator must evaluate the quality of each individual (the quality of each patch). Because it is the evaluation function that drives the algorithm, it is the most important function of the algorithm. First of all, a fitness F is computed as the norMD [20] (mean distance score representing the similarity of the sequence segments) of the patch defined as follows:

$$S = \frac{1}{NA} * \frac{2}{(NS-1)*NS} * \sum_{a=0}^{NA} \sum_{i=0}^{NS} \sum_{j=i+1}^{NS} d_M(AA_{a,i}, AA_{a,j}) \qquad (1)$$

where NS is the number of segments in the individual, NA the number of amino acids in each segment, $AA_{x,y}$ is the x-th amino acid in y-th segment, and d is the distance between two amino acids in Euclidean space using a particular aminoacid substitution matrix M that can be different for different individuals.

In addition to the similarity of the segments, other information about the fitness of a patch is calculated. In order to determine the size of the patch, we

define two variables: NS is the number of segments in the individual (height) and NA is the number of amino acids in each segment (length). Then, because we are interested in patches that have a height > 5, we favour such individuals by computing:

$$H = \min\left(1, \frac{NS + 5}{10}\right) \tag{2}$$

and because we are interested in patches that have 10 or more amino acids, we favour such individuals by computing:

$$L = \min\left(1, \frac{NA + 10}{20}\right) \tag{3}$$

The number of conserved columns is also an important factor. We calculate:

$$I = \frac{caa + 1}{NA + 1} \tag{4}$$

where caa is the number of conserved amino-acid columns. 1 has been added to the numerator so that $I > 0$ and to the denominator in the case where $caa = NA$.

Another factor to also take into account is cm, the maximum number of consecutive columns that are not conserved, which is normalised through:

$$C = \frac{NA - cm + 1}{NA + 1} \tag{5}$$

where cm is the maximum number of consecutive columns that are not conserved (*i.e.* columns that contain more than one type of amino acid).

Finally, the fitness calculation is the following :

$$F = S * C * I * (1 + H + L * H + L) + (2 * H + L * H + 2 * L) \tag{6}$$

The first part of the equation favours well formed individuals where the second part favours larger individuals.

2.6 Diversity Preservation

MEMSA uses an operator to preserve diversity in the population. Loss of diversity appears because MSAs are mostly constructed by an alternation of conserved and non-conserved blocks. Without a diversity preservation operator, individuals end up being concentrated on well-conserved blocks (high fitness value) and other interesting blocks are less explored. Therefore, in MEMSA, all unique individuals obtain an arbitrarily fixed bonus (larger than the best possible fitness) so as to preserve them during the reduction step that selects the individuals to create the next generation of parents.

2.7 Selection of Individuals for the new generation

After many tests, a 4-tournament is used to select the best individuals to create the new generation.

2.8 Patchwork to Create an MSA

After each generation, a MSA is created out of several individuals. This is done thanks to the following algorithm:

- During the evolution, an archive is created that contains the best individuals of each generation, sorted depending on their fitness.
- Then, the individuals of the last generation are sorted and added after the archive population.
- For each of the individuals of (archive+last generation), if the individual is "compatible" with the current patchwork, add the individual to the patchwork.

Testing whether an individual is compatible with a patchwork under construction is a complicated task because the individual must not only be compatible with all the other individuals of the patchwork, but also with all combinations of individuals created from the patchwork, taking into account the different fitness functions.

Several refined algorithms have been tested to perform this task, but the brute force one is currently the most efficient. It involves adding the individual to the patchwork and attempting to align all the patches to create a complete alignment, within a limited number of iterations. If a stable alignment is not found in the predetermined number of iterations, the individual is discarded.

2.9 Run parameters and behaviour of the algorithm

The EASEA [3] [15] platform has been used for the implementation. Its parameters are the following:

```
Number of generations : 200
Population size : 800000
Offspring size : 800000
Mutation probability : 1 // Probability to call the mutation function
Crossover probability : .3 // Cloning parent 1 if no crossover
Evaluator goal : maximise
Parents selection operator : Tournament 2
Next generation selection operator : Tournament 4
Elitism : Weak
Elite : 1
```

Evaluation is very fast, making it possible to use a very large population. This is a huge advantage as it means that no complex diversity preservation scheme needs to be used. A huge population means that the algorithm can be both very exploratory and at the same time, tuned to converge fast on good individuals (0.3 crossover probability). This is comparable to fast converging Genetic Programming algorithms that evolve huge populations for a reduced number of generations. During the very first generations, we observe that individuals in the population have relatively few segments (mostly two) and their segment size

is mostly close to three. After a few generations, the size of the segments of the individuals increases considerably (the best individuals can have more than 15-20 amino acids). Then after tens of generations, more complex individuals appear, which have more than two segments. Individuals that have more than two segments are very interesting because they make it possible to create links between sequences more easily.
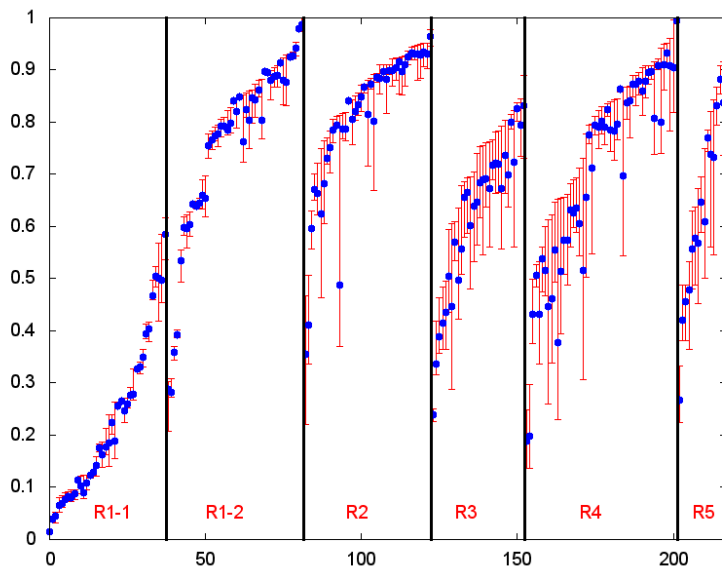
Fig. 3: Normalised SP scores for MSA constructed by MEMSA, using the 218 BAl-iBASE alignments in reference sets R1-1 to R5.

3   Experiments and Validation

In order to objectively evaluate the quality of the multiple sequence alignments constructed by MEMSA, we use a large scale benchmark specifically designed for MSA algorithms, called BAliBASE [18]. BAliBASE contains 218 reference multiple alignments based on 3D structural superpositions that are manually refined to ensure the correct alignment of conserved residues. The alignments are organised into reference sets that are designed to represent real multiple alignment problems. Reference 1 contains alignments of equidistant sequences and is divided into 2 subsets: R1-1 (10-30% amino acid identity) and R1-2 (30-50% amino acid identity). R2 contains families aligned with one or more highly

divergent "orphan" sequences. R3 contains divergent subfamilies, R4 contains sequences with large N/C-terminal extensions and R5 contains sequences with large internal insertions.

We aligned each of the test cases in BAliBASE with MEMSA and compared the resulting MSAs with the hand-made reference alignments in BAliBASE. We calculated the SP score, defined as the number of amino acid pairs aligned correctly by MEMSA, using the baliscore program provided with the BAliBASE benchmark. Fig. 3 shows the average SP score for 10 repeated applications of MEMSA, as well as the minimum and maximum SP scores for each test case. The results are quite stable over the 10 runs.
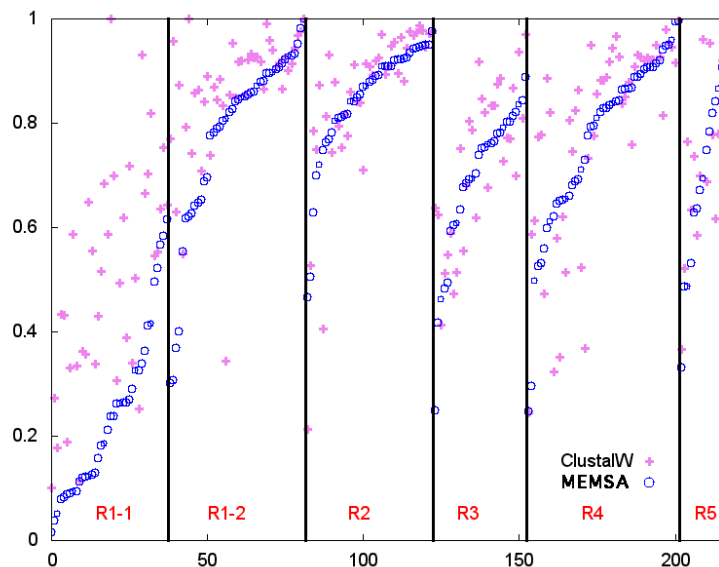
Fig. 4: Normalised SP scores for the best MSA constructed by MEMSA and ClustalW, using the 218 BAliBASE alignments in reference sets R1-1 to R5.

We also compared the accuracy of the MEMSA alignments with one of the most widely-used heuristic MSA methods: ClustalW [13]. We used ClustalW because it remains a reference among MSA methods due to its versatility, and in this work which focuses on exploiting different aligning methods, versatility is what is valued. Fig. 4 shows the best SP scores obtained by MEMSA compared to ClustalW for each of the reference sets in BAliBASE. We observe lower quality alignments in R1-1, probably due to the small number of sequences in this reference set. However, R1-1 is no longer relevant in the context of NGS "big

data". In all the other reference sets, MEMSA achieves comparable accuracy to ClustalW with higher homogeneity (less dispersion).

4 Discussion and Conclusion

To our knowledge, this is the first time a genetic algorithm is capable of aligning *all* the test cases of the large-scale BAliBASE benchmark within a reasonable time. Nevertheless, MEMSA still requires significant computational resources, compared to heuristic methods like MAFFT, MUSCLE, K-ALIGN or ProbCons but it must be noted that the Parisian approach proposed in this paper aims not at providing the user with a single MSA knowing that no single optimal MSA exists.

However, the objective of MEMSA is different from what usual MSA algorithms are doing: it is working as a complex system, that can be defined as a large number of autonomous entities in interaction. We propose a paradigm shift by acknowledging the multi-dimensionality of the problem and offering patch-oriented MSAs centered on interesting patches.

In MEMSA, the entities are patches, that are in interaction through common fitness measurements. Complete MSAs can be built by assembling matching patches taken from the population of individuals. The spatial visualisation interface (called Mondrian) allowing the user to explore the different MSAs according to different fitness functions (that will depend on the specific interests of the user) is currently under development.

5 Acknowledgements

Bibliography

[1] Blackburne, B.P., Whelan, S.: Measuring the distance between multiple sequence alignments. Bioinformatics 28 (4), 495–502 (Feb 2012)

[2] Collet, P., Lutton, E., Raynal, F., Schoenauer, M.: Polar ifs+parisian genetic programming=efficient ifs inverse problem solving. Genetic Programming and Evolvable Machines 1(4), 339–361, http://dx.doi.org/10.1023/A:1010065123132

[3] Collet, P., Lutton, E., Schoenauer, M., Louchet, J.: Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings, chap. Take It EASEA, pp. 891–901. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)

[4] Do, C.B., Mahabhashyam, M.S., Brudno, M., Batzoglou, S.: Probcons: Probabilistic consistency-based multiple sequence alignment. Genome Research (Feb 2005)

[5] Edgar, R.C.: Muscle: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research (2004)

[6] H. Zhu, Z.H., Jia, Y.: A novel approach to multiple sequence alignment using multiobjective evolutionary algorithm based on decomposition. IEEE J Biomed Health Inform 20, 717–727 (2016)

[7] Hayes-Roth, F.: Review of "adaptation in natural and artificial systems by john h. holland", the u. of michigan press, 1975. SIGART Bull. (53), 15–15 (Aug 1975), http://doi.acm.org/10.1145/1216504.1216510

[8] Henikoff, S., Henikoff, J.G.: Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences 89(22), 10915–10919 (1992), http://www.pnas.org/content/89/22/10915.abstract

[9] Holland, J.H.: Computation & intelligence. chap. Escaping Brittleness: The Possibilities of General-purpose Learning Algorithms Applied to Parallel Rule-based Systems, pp. 275–304. American Association for Artificial Intelligence, Menlo Park, CA, USA (1995), http://dl.acm.org/citation.cfm?id=216000.216016

[10] Katoh, K., Standley, D.M.: Multiple Sequence Alignment Methods, chap. MAFFT: Iterative Refinement and Additional Methods, pp. 131–146. Humana Press, Totowa, NJ (2014)

[11] Kaya, M., Sarhan, A., Alhajj, R.: Multiple sequence alignment with affine gap by using multi-objective genetic algorithm. Comput Methods Programs Biomed 114, 38–49 (Apr 2014)

[12] L. Cai, D.J., Liaknovitch, E.: Evolutionary computation techniques for multiple sequence alignment. Proceedings of the IEEE Congress on Evolutionary Computation 2000 (2000)

[13] Larkin, M., Blackshields, G., Brown, N., Chenna, R., McGettigan, P., McWilliam, H., Valentin, F., Wallace, I., Wilm, A., Lopez, R., Thompson, J., Gibson, T., Higgins, D.: Clustal w and clustal x version 2.0. Bioinformatics 23, 2947–2948 (2007)

[14] Lassmann, T., Sonnhammer, E.L.: Kalign - an accurate and fast multiple sequence alignment algorithm. BMC Bioinformatics (2005)

[15] Maitre, O., Krüger, F., Querry, S., Lachiche, N., Collet, P.: Easea: specification and execution of evolutionary algorithms on gpgpu. Soft Computing 16(2), 261–279 (2011)

[16] Nguyen, H.D., Yoshihara, I., Yamamori, K., Yasunaga, M.: Aligning multiple protein sequences by parallel hybrid genetic algorithm. Genome Informatics (2002)

[17] Notredame, C., Higgins, D.G.: Saga: Sequence alignment by genetic algorithm. Nucleic Acids Research (Apr 1996)

[18] Thompson, J.D., Koehl, P., Ripp, R., Poch, O.: Balibase 3.0: latest developments of the multiple sequence alignment benchmark. Proteins: Structure, Function and BioInformatics (Oct 2005)

[19] Thompson, J.D., Linard, B., Lecompte, O., Poch, O.: A comprehensive benchmark study of multiple sequence alignment methods: Current challenges and future perspectives. PLoS One (2011)

[20] Thompson, J.D., Plewniak, F., Ripp, R., Thierry, J.C., Poch, O.: Towards a reliable objective function for multiple sequence alignments1. Journal of Molecular Biology 314(4), 937 – 951 (2001), http://www.sciencedirect.com/science/article/pii/S0022283601951873

[21] Wilson, S.W., Goldberg, D.E.: A critical review of classifier systems. In: Proceedings of the 3rd International Conference on Genetic Algorithms. pp. 244–255. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989), http://dl.acm.org/citation.cfm?id=645512.657260

[22] Zhang, C., Wong, A.: A genetic algorithm for multiple molecular sequence alignment. Comput Appl Biosci. 13, 565–581 (1997)

Basic, Dual, Adaptive, and Directed Mutation Operators in the Fly Algorithm

Zainab Ali Abbood and Franck P. Vidal

School of Computer Science, Bangor University, Dean Street, Bangor, LL57 1UT, United Kingdom, f.vidal@bangor.ac.uk

Abstract. Our work is based on a Cooperative Co-evolution Algorithm – the Fly algorithm – in which individuals correspond to 3-D points. The Fly algorithm uses two levels of fitness function: i) a local fitness computed to evaluate a given individual (usually during the selection process) and ii) a global fitness to assess the performance of the population as a whole. This global fitness is the metrics that is minimised (or maximised depending on the problem) by the optimiser. Here the solution of the optimisation problem corresponds to a set of individuals instead of a single individual (the best individual) as in classical evolutionary algorithms (EAs). The Fly algorithm heavily relies on mutation operators and a new blood operator to insure diversity in the population. To lead to accurate results, a large mutation variance is often initially used to avoid local minima (or maxima). It is then progressively reduced to refine the results. Another approach is the use of adaptive operators. However, very little research on adaptive operators in Fly algorithm has been conducted. We address this deficiency and propose 4 different fully adaptive mutation operators in the Fly algorithm: Basic Mutation, Adaptive Mutation Variance, Dual Mutation, and Directed Mutation. Due to the complex nature of the search space, (kN-dimensions, with k the number of genes per individuals and N the number of individuals in the population), we favour operators with a low maintenance cost in terms of computations. Their impact on the algorithm efficiency is analysed and validated on positron emission tomography (PET) reconstruction.

Keywords: evolutionary algorithms, Parisian approach, reconstruction algorithms, positron emission tomography, mutation operator

1 Introduction

This paper focuses on the application of a particular Cooperative Co-evolution Algorithm (CCEA), the *Fly algorithm* [12], to reconstruct 3-D tomographic data in nuclear medicine. The general public is more familiar with computed tomography (CT) when considering 3-D medical imaging. It offers a high spatial resolution and signal-to-noise ratio (SNR), which is suitable for anatomic examinations. Tomography in nuclear medicine is called emission tomography (ET). It has a much lower resolution and SNR (see Fig. 1). There are two types of 3-D imaging modalities in nuclear medicine: Single-photon emission computed
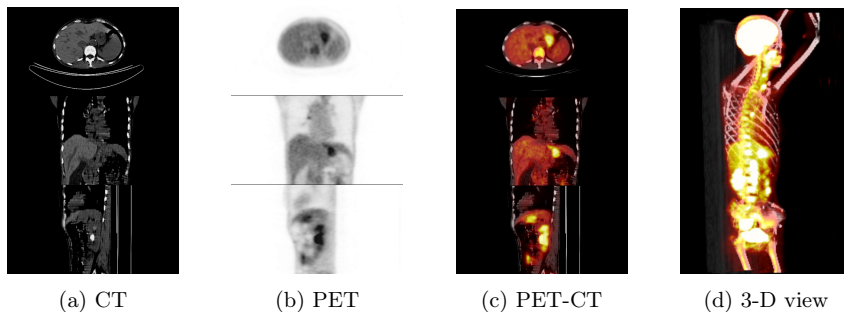
(a) CT   (b) PET   (c) PET-CT   (d) 3-D view

Fig. 1: PET-CT of a cancer patient (data available on The Cancer Imaging Archive (TCIA) [6] at `https://public.cancerimagingarchive.net/`).

tomography (SPECT) and positron emission tomography (PET). They both use radioactive substances for the labelling of a physiological process (e.g. bone fracture, growth of cancer cells, or areas of low blood pressure). The radioactive concentration in the body is proportional to the physiological process of interest. The radioactive emission is detected by the imaging system. Tomography reconstruction "translates" it into a stack of 2-D cross-sections, which corresponds to an estimation of the 3-D radioactive concentration. Evolutionary computing has been successfully used in tomography reconstruction in both SPECT and PET [4,19]. This method heavily relies on the selection process, mutation operators and a diversity mechanism. We focus here on PET as it is now the main 3-D imaging modality in nuclear imaging. The focus of this article is on the choice and combination of mutation operators.

The problem definition and the motivations for this research are given in Section 2. The general principles of the evolutionary reconstruction for PET reconstruction are given in Section 3. It is followed by the definition of the mutation operators, which are used in our implementation. The results of these operators are analysed in Section 5. The article ends with a conclusion in Section 6.

2   Problem definition and motivations

Tomography reconstruction is an inverse problem: Projection data (Y) acquired by a medical scanner (Y is the known data) has to be inverted by a computer program to generate an estimate ($\hat{f}$) of an unknown image (f). The nature of f and Y is problem-dependant (see Fig. 2 for an example in ET) and it is often modelled as:

$$Y = P[f] \tag{1}$$

with P a projection operator that is also problem-dependant. The reconstruction corresponds to solving:

$$\hat{f} = P^{-1}[Y] \tag{2}$$

(a) Ground-truth (f).
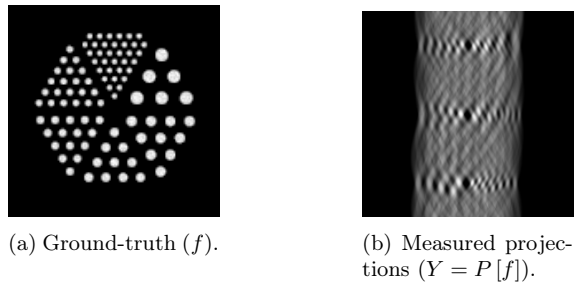

(b) Measured projections ($Y = P[f]$).

Fig. 2: Test case using the Jaszczak phantom with hot rods.

The problem is also ill-posed due to missing data and photonic noise (Poisson noise) in Y. Noise is actually a major concern in ET. The reconstruction can be considered as an optimisation problem:

$$\hat{f} = \arg\min_{f \in \mathbb{R}^2} \left\| Y - P\left[\hat{f}\right] \right\|_1 \tag{3}$$

Maximum-Likelihood Expectation-Maximization (MLEM) and its derivative Ordered Subset Expectation-Maximization (OSEM) are the main algorithms used in nuclear medicine [18,11].

An alternative approach would be the use of artificial evolution. Most people are familiar with black-box optimisation using simple genetic algorithms (binary encoding of individuals) and real-valued genetic algorithms. Using this traditional approach is not suitable for tomography reconstruction. Consider an optimisation problem that consists in finding the best possible 3-D position of N points. The search space has $3N$-dimensions. When a black-box evolutionary algorithm (EA) is considered, including CMA-ES, there will be k individuals in the population, with $3N$ genes per individual. When N is large, this approach is likely to fail due to its computing time. A more recent class of algorithms is the Parisian approach. Using this framework, it is possible to only require N individuals with 3 genes per individual. Each individual is a partial solution. The individuals collaborate to build the overall solution.

In [4,19], the Fly algorithm is deployed to minimise the error between the simulated image $\hat{Y}$ and an input image Y (global evaluation) by optimising the individual position of radioactive emission points (local evaluation of the points). As the goal includes both a local and a global evaluation, the optimisatin problem is perfectly suited for the Parisian approach [7], which includes the Fly algorithm. It heavily relies on the selection process, mutation operators and a diversity mechanism. Ideally, the amount of random change needs first to be set to a large value to better explore the search space. However, a constant large mutation variance will lead to blurred reconstructed volumes. As a consequence, the mutation variance has to be gradually reduced. The usefulness of adaptive mutations in EAs is a well established [2,5,9,14]. Such techniques

have been proven effective in various cases, depending on the fitness function and the genetic engine used. However, complex schemes for the adaptivity of the mutation operator have a computational cost that may not be negligible. More simplistic schemes can actually perform better due to lower computational needs [8]. Our main motivation is to investigate the use of such operators in the Fly algorithm. The aim is to determine which sets of operators are the best in terms of accuracy of the results, and amongst them which one is the best in terms of computational cost. Destroying a bad fly and creating new and better ones has to be a fast process because it is performed at a much higher rate in the Parisian approach than in classical EAs. This is because the solution to the optimisation problem in our case is the whole population [1] rather than the best individual as in classical EAs. Using the best set of mutation operators to create new flies is therefore important.

3   Overview of the Fly Algorithm for PET Reconstruction

The Fly algorithm was initially developed as a fast EA in stereovision for robotic applications such as obstacle detection [12]. The Fly algorithm is based on the Parisian approach. In conventional artificial evolution (AE), the solution is represented by the best individual. In the Parisian approach, the solution corresponds to the whole population (or a subgroup of the population).

The individuals, called Flies, are 3-D points. In its original implementation, a fly is projected onto the image planes corresponding to the pair of stereo images. Its fitness is proportional to the difference of pixel values between the neighbourhoods of the projected point on both images. The flies are evolved using the typical steps of EAs. The flies eventually gather on the surfaces of obstacles (e.g. walls). The final population can be used by autonomous robots to avoid collision when moving.

Following its success in robotics, the Fly algorithm has been adapted to ET in nuclear medicine, first in SPECT [4], then PET [19]. The population of individuals is randomly generated within the search space contained in the scanner. Each individual corresponds to a 3-D point that simulates radioactive emissions. The emitted photons are projected. Each fly keeps track of its own simulated photons. The estimated projections $(\hat{Y})$ is the amalgamation of the projected photons of all the flies of the population. After optimisation, $\hat{Y}$ matches the projection data Y measured by the scanner, and the population $\hat{f}$ is an estimate of the unknown f.

The *global fitness* is used to evaluate the performance of the population as a whole toward an optimal global solution. It is a specific feature of the Parisian approach. In our case, it is an error metrics corresponding to the $\mathcal{L}^1$-norm (also known as sum of absolute errors (SAE)) between Y and $\hat{Y}$:

$$SAE(Y, \hat{Y}) = \left\| Y - \hat{Y} \right\|_1 = \sum_i \sum_j \left| Y(i,j) - \hat{Y}(i,j) \right|$$

SAE is measured using all the individuals and it minimised by the algorithm. Note that Y and $\hat{Y}$ have to be normalised between 0 and 1 before computing

the $\mathcal{L}^1$-norm as the lowest and highest pixel values in Y and $\hat{Y}$ may be significantly different. To evaluate the performance of a single individual (i), we use the *marginal fitness* ($F_m(i)$). It is based on the leave-one-out cross-validation principle:

$$F_m(i) = SAE(Y, \hat{Y} \setminus \{i\}) - SAE(Y, \hat{Y})$$

with $\hat{Y} \setminus \{i\}$ the estimated projections without the photons simulated by fly i. If F_m is positive, the error is smaller when the fly is included: The fly has a positive impact on the population's performance. It is a good fly, i.e. a good candidate for reproduction. If F_m is negative, the error is larger when the fly is included: The fly has a negative impact on the population's performance. It is a bad fly, i.e. a good candidate for death. F_m is therefore a measure maximised by the algorithm.

Repeated applications of the genetic operators are used to optimise the position of all the flies to get to the state where the difference between the projections ($\hat{Y}$) simulated by the population and the actual data (Y) is as small as possible. Our implementation heavily relies on different mutation operators and on new blood (also called immigration).

We use a steady state evolutionary strategy where, at each iteration, a bad fly is selected for death and replaced using a genetic operator (mutation or new blood). We demonstrated in [19] the usefulness of the *Threshold Selection* over tournament selection. To select a bad fly, a random fly is repeatedly picked up until one is found with a marginal fitness below or equal to a given threshold (e.g. 0); to select a good fly, a random fly is repeatedly picked up until one is found with a marginal fitness above the threshold.

When the number of flies with negative fitness decreases too much, the threshold selection fails to select flies to kill in an acceptable time. It indicates that the population has converged. A mitosis operator can be activated to increase the population size, i.e. the population size is doubled: Each fly is split into two new flies (one of the two is then mutated). The benefit of this strategy in terms of computing time has been demonstrated in [19].

If there are enough flies in the population, the solution is extracted to create volume data using voxels. Two voxelisation methods can be exploited [1]. In the simpler one, flies are binned into voxels. The flies are considered as Dirac functions and the voxel intensity corresponds to the number of flies within it. However it may generate noisy images. In the most advanced method, implicit modelling is used to produce smoother images. The principle is to treat the fitness of a fly as a level of confidence in the fly's position. Each fly corresponds to a 3-D Gaussian kernel whose variance depends on the fly's fitness. A fly is spread over several voxels. The contribution to final volume dataset is the same for each fly.

4 Varying Mutation Operators in the Fly Algorithm

Our implementation relies on mutation to create better flies. The aim of the mutation operators is to create new flies in the neighbourhood of good flies.

Note that new blood is also used to preserve a minimum level of diversity in the population. The following steps are necessary to use a mutation operator:

1. A bad fly is selected using the Threshold Selection.
2. Its projections are removed from $\hat{Y}$.
3. A good fly is selected using the Threshold Selection.
4. The bad fly is replaced by the good fly.
5. The position of the newly created fly is altered by random changes.
6. The projections of the mutated fly are computed.
7. These projections are added to $\hat{Y}$.

The only step which is different, depending on the mutation operator used, is 5. Only the Dual mutation was used in our initial implementation [19]. We added three other adaptive mutation operators that are automatically tuned without any human intervention. An individual has 9 genes: x, y, and z for the fly's position; $P_{basicMut}$, the probability of the basic mutation operator, $P_{adaptiveMut}$, the probability of the adaptive mutation operator, $P_{dualMut}$, the probability of the dual mutation operator, $P_{directedMut}$, the probability of the directed mutation operator, $P_{newBlood}$, the probability of the immigration/new blood operator; and σ, the mutation rate associated with the fly (it is used by the basic and directed mutation operators). Alg. 1 shows how random changes are applied for all our mutation operators, which are described below.

4.1   Basic Mutation

The mutation variance can be subject to an adaptive pressure itself and be self-adapted [2]. In our implementation the probability of all the operators is encoded in the genome of each individual. The mutation variance is too. The probabilities and the variance are then subject to random mutations as well. The major advantage of this scheme is to provide a fully automatic method to adapt the mutation variance, whilst keeping the administration cost null.

4.2   Adaptive Mutation Variance

The mutation variance can be directly adapted to local measurements, such as fitness [15] and local regularity [13]. For example, when the evolutionary algorithm is used to minimise an error function, the variance can be bigger when fitness is high and smaller when fitness is low [20]. The idea is to favour large exploration around the weakest individuals, whilst performing fine tuning in the vicinity of good individuals. In our case, we want to maximise the marginal fitness of flies: The higher the marginal fitness (F_m), the lower the variance, and *vice versa*. We define the mutation variance here as a piecewise-defined function of F_m:

$$\sigma\left(F_m\right) = \begin{cases} \sigma_{max}, & F_m < fit_{min} \\ \sigma_{min}, & F_m > fit_{max} \\ \sigma_{min} + \left(\sigma_{max} - \sigma_{min}\right) \times \frac{\cos\left(\pi \times \left(\frac{F_m - fit_{min}}{fit_{max} - fit_{min}}\right)\right) + 1.0}{2.0}, & \text{otherwise} \end{cases}$$

Algorithm 1 Procedure mutate

Input: μ^+ # The good fly on which λ will be based
Input: σ # The mutation rate to use for small random alterations
Input: use_dir_mut # A boolean flag
Output: λ # The fly create by mutation of μ^+

  $\lambda.parentF_m = \mu^+.F_m$ # Record the parent's fitness
  $\lambda.created\_by\_mutation \leftarrow TRUE$

  # Mutate each gene
  for each gene i **do**
     $\Delta \leftarrow \sigma \times \mathrm{rand}(0,1) \times \frac{range[i]}{2}$ # Amount of random variation

     # Get the direction of change of gene i
     if NOT use_dir_mut OR NOT $\mu^+.created\_by\_mutation$ **then**
        # Not using directed mutation
        $\lambda.dir[i] \leftarrow \mathrm{sign}(\mathrm{rand}(-1,1))$ # Random direction
     else if $\mu^+.F_m > \mu^+.parentF_m$ **then**
        # Parent better than grand-parent
        $\lambda.dir[i] \leftarrow \mu^+.dir[i]$ # Go in the same direction as parent
     else   # Grand-parent better than parent
        $\lambda.dir[i] \leftarrow -\mu^+.dir[i]$ # Go in the opposite direction as parent
     end if

     # Apply the random change in the corresponding direction
     $\lambda.gene[i] \leftarrow \mu^+.gene[i] + \lambda.dir[i] \times \Delta$
     $\mathrm{check}(\lambda.gene[i])$ # Apply constraints on value of gene if necessary
  end for

---

with F_m the fitness of the individual who will undergo a mutation. Using the cosine function, $\sigma(F_m)$ smoothly varies between the smaller (fit_{min}) and the larger (fit_{max}) fitness thresholds respectively. If F_m is smaller than fit_{min}, σ is equal to σ_{max}; if the individual's fitness is greater than fit_{max}, σ is equal to σ_{min} (with σ_{min} and σ_{max} two constant values set by the user). The major advantage of this scheme is similar to the previous one: It provides a fully automatic method to adapt the mutation rate, whilst keeping its administration cost negligible.

4.3   Dual Mutation

Another approach, called Rechenberg's rule, is to modulate the mutation variance based on the success/failure rate of the current mutation variance [16,3]. It relies on the notion of "evolution window": Increase the mutation variance to speed-up the search-space exploration, or decrease it to refine the results. For this purpose, the algorithm must keep track of the success rate, which has an obvious computational cost. This category includes the well-known $1/5^{\mathrm{th}}$ rule proposed by Schewefel [17,3]. A single σ value is used. It is updated at regular

intervals. It records the number of successful and unsuccessful mutations over a given number of mutations (M). If the rate of successful mutation is greater than $1/5$, then increase σ; if it is lower, decrease σ.

The *Dual mutation* operator in the Fly algorithm is based on the concurrent testing of two alternative variance values (σ_{low} and σ_{high}, with $\sigma_{high} = k\sigma_{low}$) [19]. The update rule is multiplicative as for the $1/5^{\text{th}}$ rule. If mutations with σ_{high} provide the best results during the previous M iterations, then both mutation variances are multiplied by a predefined factor (pf, with $pf > 1$). If mutations with σ_{low} provide the best results during the previous M iterations, then both mutation variances are divided by pf. For every Dual mutation, we check the global fitness before and after the mutation. Note that these numbers are always pre-computed during the selection of individuals. Therefore, we cannot affect their computation to the administration cost of this mutation operator. Using two accumulators, we can assess which variance amongst σ_{low} and σ_{high} is the best. This scheme also requires a very limited number of user inputs. The administration cost of the algorithm is sightly higher than the previous two schemes but still relatively light. Also, the dual mutation does not need to make any assumption on the ideal success rate of the mutation as in the $1/5^{\text{th}}$ rule.

4.4   Directed Mutation

We introduce here a new operator, the *Directed Mutation*, which is related to the evolution path in CMA-ES [10]. Its objective is to lead new individuals toward areas of the search space that have been previously defined as "interesting" by older flies. This principle follows well the fundamentals of CCEAs as new individuals have to cooperate with older ones to benefit from their knowledge to locate areas of interest.

To illustrate how our implementation works (see Alg. 1 with $use\_dir\_mut = TRUE$), let us consider the case as follows: A fly (Fly_2) has been created by mutation of another fly (Fly_1). We are now going to create a new fly (Fly_3) by mutation of Fly_2. The position of the new fly will be biased toward the position of the best fly among Fly_1 and Fly_2. If Fly_1 is better than Fly_2, we will look for a new Fly_3 from the location of Fly_2 in the direction toward Fly_1 (see Fig. 3a); if Fly_2 is better than Fly_1, we will look for Fly_3 from the location of Fly_2 in the direction away from Fly_1 (see Fig. 3b). For any fly created by any kind of
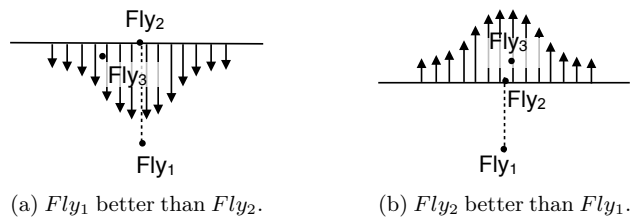

(a) Fly_1 better than Fly_2.   (b) Fly_2 better than Fly_1.

Fig. 3: Directed Mutation Principle.

mutation, we record its parent's fitness and in which direction the new fly has been moved with respect to its parent. This is the main administration cost.

5   Results

We consider the Jaszczak phantom with hot rods as a test case (see Fig. 2). We assess the algorithm with all the possible combinations of mutation operators. There are 2^4 possible configurations (see Tab. 1). Due to the stochastic

Table 1: The combinations of mutation operators.

Type	Operators	Type	Operators
0000	no mutation	1000	basic
0001	directed	1001	basic + directed
0010	adaptive	1010	basic + adaptive
0011	adaptive + directed	1011	basic + adaptive + directed
0100	dual	1100	basic + dual
0101	dual + directed	1101	basic + dual + directed
0110	dual + adaptive	1110	basic + dual + adaptive
0111	dual + adaptive + directed	1111	basic + dual + adaptive + directed

nature of artificial evolution, 15 reconstructions per configuration are performed to gather statistically meaningful results. For each reconstruction, we record i) the normalised cross-correlation (NCC) between the ground-truth (f) and the reconstructed volume ($\hat{f}$), and ii) the reconstruction time:

$$NCC(f,\hat{f}) = \frac{1}{w \times h} \sum_{i=0}^{i<w} \sum_{j=0}^{j<h} \left(\frac{\left(f(i,j) - \overline{f}\right)\left(\hat{f}(i,j) - \overline{\hat{f}}\right)}{\sigma_f \sigma_{\hat{f}}} \right)$$

with w and h the number of pixel along the horizontal and vertical axis respectively, $\overline{f}$ and $\overline{\hat{f}}$ the average pixel value in f and $\hat{f}$ respectively, and σ_f and $\sigma_{\hat{f}}$ the standard deviation in f and $\hat{f}$ respectively. The NCC provides a measure of similarity between two images. It is 100% if they are perfectly correlated. It is 0% is they are totally uncorrelated. It is -100% if there is a negative correlation (also called anticorrelation or inverse correlation) between them.

Figure 4 shows the median results in term of performance for duration and NCC for each mutation operator combination. The dual mutation combined with the directed mutation (see Configuration 0101 in Table 1) looks effective. The dual mutation only (0100) as in [19] is not as good. The combinations whose NCC is less than 1% smaller than the best combination are highlighted in green and blue-violet in the figure. We can see that all of the combinations using the directed mutation are performing relatively well. Only combinations that are shown in both green and blue-violet should be considered (i.e. Configuration

Type/mm #rank	Dirac #rank	Gaussian #rank	Type/mm #rank	Dirac #rank	Gaussian #rank
0000/14.53 #1	(68.77%) #16	(84.38%) #16	1000/15.40 #3	(85.29%) #9	(92.14%) #12
0001/15.67 #5	(86.25%) #2	(92.56%) #10	1001/15.67 #5	(86.06%) #3	(92.17%) #11
0010/14.53 #1	(74.28%) #15	(88.67%) #15	1010/16.00 #8	(85.59%) #7	(92.81%) #3
0011/18.93 #16	(84.56%) #12	(92.65%) #7	1011/17.80 #14	(86.01%) #4	(92.99%) #1
0100/15.67 #5	(77.42%) #13	(90.05%) #13	1100/16.73 #11	(85.76%) #6	(92.60%) #8
0101/16.40 #10	(86.28%) #1	(92.96%) #2	1101/16.73 #11	(85.77%) #5	(92.73%) #5
0110/15.40 #3	(74.41%) #14	(88.71%) #14	1110/16.13 #9	(84.68%) #11	(92.57%) #9
0111/16.87 #13	(84.79%) #10	(92.68%) #6	1111/17.87 #15	(85.42%) #8	(92.80%) #4

Fig. 4: Performance comparison of the different combinations of mutation operators. The combinations whose NCC is less than 1% smaller than the best combination are highlighted in green and blue-violet.

Table 2: Performance comparison in terms of NCC (Gaussian voxelisation only) and duration of all the combinations of mutation operators. $\mathcal{W}$ is the Wilcoxon signed-rank test between each entry with Combination 0101, and S is the total rank sum. Possible good candidates against 0101 are highlighted in green.

type	NCC (in %)	$\mathcal{W}_{NCC}/S$	Duration (in minute)	$\mathcal{W}_{duration}/S$
0000	84.38 ±1.07	1.00	14.53 ±1.88	0.64
0001	92.556 ±0.65	0.47	15.67 ±2.02	0.20
0010	88.67 ±0.64	1.00	14.53 ±1.19	0.54
0011	92.65 ±0.55	0.37	18.93 ±6.18	-0.28
0100	90.05 ±0.89	0.98	15.67 ±1.35	0.29
0101	92.96 ±0.67	N/A	16.4 ±2.29	N/A
0110	88.71 ±0.36	1.00	15.4 ±1.64	0.42
0111	92.68 ±0.7	0.40	16.87 ±3.14	-0.08
1000	92.14±0.54	0.78	15.4 ±1.76	0.33
1001	92.17 ±0.61	0.80	15.67 ±2.66	0.26
1010	92.81 ±0.32	0.27	16 ±2.42	0.22
1011	92.99 ±0.58	-0.03	17.8 ±2.91	-0.43
1100	92.6 ±0.5	0.43	16.73 ±3.03	-0.11
1101	92.73 ±0.53	0.35	16.73 ±1.79	-0.09
1110	92.5 ±0.71	0.40	16.13 ±2.64	0.08
1111	92.8 ±0.6	0.35	17.87 ±3.38	-0.33

0001, 0101, 1000, 1001, 1010, 1011, 1100, 1101 and 1111). Ideally, the ones with a short run-time should be favoured.

Quantitative results for each configuration are presented using boxplots in Figure 5. The dark turquoise and mangenta configurations of Figure 5a shows the NCC between the ground-truth and the reconstructions using all flies as a finite point (or Dirac) and all flies using Gaussian kernel, respectively [1]. The experiments with the dual and directed mutation operator (0101) (green circle) seem to provide best results (86.28 ±0.71) and (92.96 ±0.67) in terms of NCC in both configurations. It is much better than with the Dual mutation only (0100) (red circle) when we use all flies as finite points (77.42% ±2.41) as in [19] and all flies as Gaussian kernels (90.05 ±0.89). However it is still hard to assess which configuration is the best in term of reconstruction speed (see Figure 5b).

Our hypothesis is that 0101 is the best combination of operators. To validate it, we apply a non-parametric statistical hypothesis test (Wilcoxon signed-rank test, noted $\mathcal{W}$). The size of the samples is 15. The aim is to identify all other combinations that are statistically relatively similar in terms of NCC as 0101 (see Tab. 2). We only consider the voxelisation using Gaussian kernels as we already know it provides the most accurate reconstructions [1]. We also apply the Wilcoxon signed-rank test on duration (see Tab. 2). The idea is to identify which possible good combination provides accurate results the quickest. $\mathcal{W}$ is divided by the total rank sum S to account for the effect size. $\mathcal{W}/S$ is within the
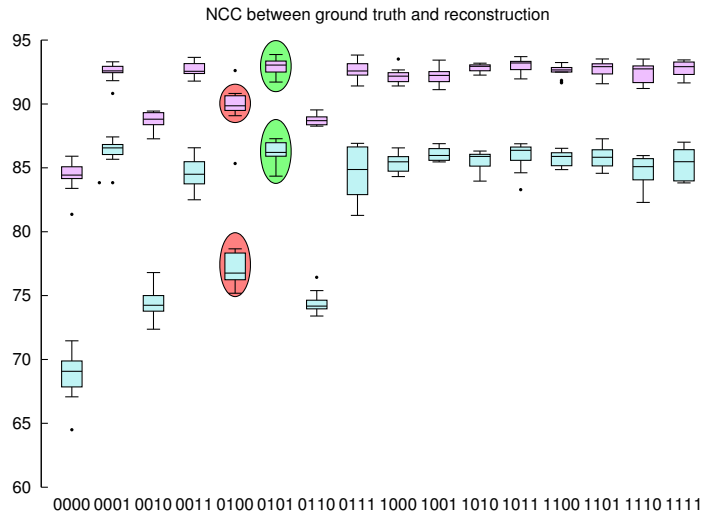
(a) NCC for all flies as a finite point (in dark turquoise) and all flies as Gaussian kernel (in magenta) [1].
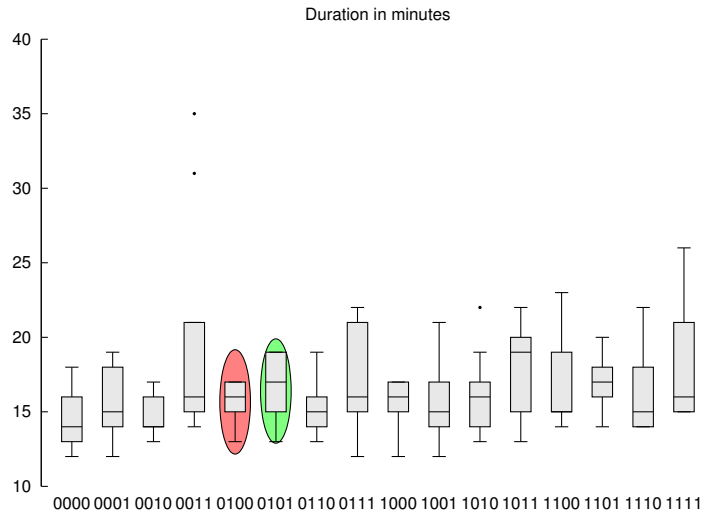
(b) Time required.

Fig. 5: Performance of mutation operators. The performance of our initial implementation with dual mutation only as in [19] is highlighted in red. The performance of the combination of the dual and directed mutation operators is highlighted in green.

range between -1 and +1. For the NCC, any combination whose corresponding value is close to -1 performs much better than 0101; close to 0 performs similarly well; and close to +1 performs much worse. In practice, any configuration with $\mathcal{W}_{NCC}/S < 0.5$ should be considered as a possible good candidate. It includes 0001, 0011, 0111, 1010, 1011, 1100, 1101, 1110 and 1111. Almost all of them use a combination of at least two mutation operators. For this selection of combinations, the directed mutation is used 6 times, the adaptive mutation 6 times, the dual mutation 5 times, and the basic mutation 6 times. It shows the benefit of our new operators and the usefullness of combining several types of mutation.

When the duration is considered, any combination whose corresponding value is close to +1 performs much better than 0101; close to 0 performs similarly well; and close to -1 performs much worse. It is because the NCC should be as high as possible whereas the duration should be as small as possible. Any configuration with $\mathcal{W}_{duration}/S > -0.5$ could be considered as a possible candidate. It is therefore impossible to objectively distinguish the possible good candidates in terms of shortest time required.

6   Conclusion

We have presented a fully adaptive implementation of a CCEA based on the Fly algorithm. The purpose of this algorithm is to optimise the location of 3-D points. The final set of points corresponds to the solution of the optimisation problem. The Fly algorithm heavily relies on the mutation operator to find the best positions. In our initial implementation, we proposed the Dual Mutation operator to self-tune the mutation variance. In this paper we complete our implementation with three other adaptive mutation operators and assessed their behaviours in tomographic reconstruction in PET. The probability of the genetic operators are now part of the individuals' genome. It includes a basic mutation operator whose mutation variance is also encoded in the genome. There is also a mutation operator whose variance is a function of the fitness of the individual to mutate. Finally we introduced a new operator, the Directed Mutation, that looks at the history of the individual that is going to be mutated to guide its mutations in a direction that is likely to be worth exploring based on the experience of the individual's ancestors. We demonstrate using the Jaszczak phantom with hot rods that this approach and this new operator lead to better results in terms of accuracy (improvement of NCC by ∼10%) without sacrificing the reconstruction speed. The problem considered here is, however, relatively specific to claim general results. Further research is needed to evaluate the effectiveness of our operators i) against state-of-the-art operators, ii) in alternative EAs, and iii) with other reconstruction data.

Acknoledgements

References

1. Ali Abbood, Z., Lavauzelle, J., Lutton, E., Rocchisani, J.M., Louchet, J., Vidal, F.P.: Voxelisation in the 3-D Fly algorithm for PET. Swarm Evol Comput (2017), in press
2. Bäck, T.: Self-adaptation in genetic algorithms. In: Proc 1st European Conf Artif Life. pp. 263–271. MIT Press (1992)
3. Beyer, H.G., Schwefel, H.P.: Evolution strategies - a comprehensive introduction. Nat Comput 1(1), 3–52 (2002)
4. Bousquet, A., Louchet, J., Rocchisani, J.M.: Fully three-dimensional tomographic evolutionary reconstruction in nuclear medicine. Lect Notes Comput Sc 4926, 231–242 (2007)
5. Chellapilla, K.: Combining mutation operators in evolutionary programming. IEEE T Evolut Comput 2(3), 91–96 (1998)
6. Clark, K., et al.: The cancer imaging archive (TCIA): Maintaining and operating a public information repository. Journal of Digital Imaging 26(6), 1045–1057 (2013)
7. Collet, P., Louchet, J.: Applications in the Processing of Signals and Images, chap. Chapter 2. Artificial Evolution and the Parisian Approach, pp. 15–44. Wiley (2010)
8. Collet, P., Lutton, E., Louchet, J.: Issues on the optimisation of evolutionary algorithm code. In: IEEE C Evol Computat (2002)
9. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE T Evolut Comput 3(2), 124–141 (1999)
10. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 11(1), 1–18 (2003)
11. Hudson, H.M., Larkin, R.S.: Accelerated image reconstruction using ordered subsets of projection data. IEEE Trans Med Imaging 13(4), 601–609 (1994)
12. Louchet, J.: Stereo analysis using individual evolution strategy. In: Proc Int C Patt Recog. vol. 1, pp. 908–911 (2000)
13. Lutton, E., Lévy Véhel, J.: Pointwise regularity of fitness landscapes and the performance of a simple ES. In: IEEE C Evol Computat. pp. 16–21 (2006)
14. Ochoa, G.: Setting the mutation rate: Scope and limitations of the 1/L heuristic. In: Proc GECCO'02. pp. 495–502 (2002)
15. Orlowska-Kowalska, T., Lis, J.: Application of evolutionary algorithms with adaptive mutation to the identification of induction motor parameters at standstill. COMPEL 28(6), 1647–1661 (2009)
16. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag (1973)
17. Schwefel, H.P.: Numerical optimization of computer models. Wiley (1981)
18. Shepp, L.A., Vardi, Y.: Maximum likelihood reconstruction for emission tomography. IEEE Trans Med Imaging 1(2), 113–122 (1982)
19. Vidal, F.P., Lutton, E., Louchet, J., Rocchisani, J.: Threshold selection, mitosis and dual mutation in cooperative coevolution: application to medical 3D tomography. Lect Notes Comput Sc 6238, 414–423 (2010)
20. Vidal, F.P., Villard, P., Lutton, E.: Tuning of patient specific deformable models using an adaptive evolutionary optimization strategy. IEEE T Bio-Med Eng 59(10), 2942–2949 (2012)

Session 5 - Metaheuristics

A New High-Level Relay Hybrid Metaheuristic for Black-Box Optimization Problems

Julien Lepagnot[1], Lhassane Idoumghar[1], Mathieu Brévilliers[1], and Maha Idrissi-Aouad[2]

[1] Université de Haute-Alsace, LMIA (E.A. 3993)
12 rue des Frères Lumière, 68093 Mulhouse, France
{julien.lepagnot,lhassane.idoumghar,mathieu.brevilliers}@uha.fr
[2] Institut supérieur du Génie Appliqué
41 avenue 2 Mars, 20250 Casablanca, Maroc

Abstract. In this paper, a high-level relay hybridization of three metaheuristics with different properties is proposed. Our objective is to investigate the use of this kind of hybridization to tackle black-box optimization problems. Indeed, without any knowledge about the nature of the problem to optimize, combining the strengths of different algorithms, belonging to different classes of metaheuristics, may increase the probability of success of the optimization process. The proposed hybrid algorithm combines the multiple local search algorithm for dynamic optimization, the success-history based adaptive differential evolution, and the standard particle swarm optimization 2011 algorithm. An experimental analysis using two well-known benchmarks is presented, i.e. the Black-Box Optimization Benchmarking (BBOB) 2015 and the Black Box optimization Competition (BBComp). The proposed algorithm obtains promising results on both benchmarks. The ones obtained at BBComp show the relevance of the proposed hybridization.

Keywords: High-level Relay, Hybrid Metaheuristic, Black-Box Optimization, Local Search, Differential Evolution, Particle Swarm Optimization

1 Introduction

In High-level Relay Hybrid (HRH) metaheuristics, self-contained metaheuristics are executed in sequence [8]. For instance, a local search can be applied after an evolutionary algorithm (EA) in order to fine-tune the solution found by the EA. The HRH hybridization may also use a greedy heuristic to generate a good initial population for the EA. In this paper, we investigate the use of a HRH metaheuristic for black-box optimization problems. The idea is to combine the strengths of different kind of metaheuristics, in order to successfully tackle a wider range of problems. The first metaheuristic executed in the proposed HRH should especially focus on exploration, while the last one should focus on exploitation.

The *Multiple Local Search algorithm for Dynamic Optimization* (MLSDO) [6] is a dynamic optimization algorithm that shows interesting properties for a HRH. Indeed, it has been designed to widely explore the search space, in order to quickly detect and keep track of the local optima in a problem that changes over time. For a static optimization problem, it can be used to perform a first exploration of the search space in order to quickly locate a promizing area, and eventually different local optima of a multimodal objective function. These local optima can then be used to create a good initial population of an EA. For this last one, we have chosen the Success-History based Adaptive Differential Evolution (SHADE) algorithm, which is one of the state-of-the-art differential evolution (DE) algorithms [9]. As a last component of the proposed HRH, we investigate using an algorithm based on particle swarm optimization (PSO). Compared to EAs, that make use of a population of individuals, PSO uses a swarm of particles that adjust their flying trajectories in the search space according to their own flying experiences and the ones of all neighboring particles [5]. PSO often suffers from premature convergence [1], but it can be used as a powerful exploitation procedure. In the proposed HRH, we have chosen Standard PSO 2011 (SPSO2011) [2].

In order to validate the proposed hybrid metaheuristic, we have selected two well-known benchmarks: the Black-Box Optimization Benchmarking (BBOB) 2015 [3], and the Black Box optimization Competition (BBComp) [7].

The rest of this paper is organized as follows: Section 2 presents an overview of MLSDO, SHADE and SPSO2011. Then, the proposed HRH algorithm, called MSS, is described in detail in Section 3. Experimental protocol and parameter setting are presented in Section 4. Experimental results are discussed in Section 5. Finally, a conclusion is given in Section 6.

2 Presentation of the hybridized components

2.1 Overview of MLSDO algorithm

In dynamic optimization, it is important for a metaheuristic to continuously and widely explore the search space, in order to quickly locate a promising area, and to quickly detect and react to a change in the objective function. To do so, MLSDO uses several local searches, each one performed in parallel with the others, to explore the search space, and to track the found optima over the changes in the objective function. These local searches consist of moving step-by-step in the search space, from a current solution to its best neighbor one, until a stopping criterion is satisfied, reaching thus a local optimum. Each local search is performed by an agent, and all the agents are coordinated by a dedicated module (the coordinator). Two types of agents exist in MLSDO: the exploring agents (to explore the search space in order to discover the local optima), and the tracking agents (to track the found local optima over the changes in the objective function). The local searches performed by the exploring agents have a greater initial step size than the one of the tracking agents, because the exploring agents have to widely explore the search space. The strategies used to coordinate

these local search agents enable the fast convergence to well diversified optima, in order to quickly react to a change and find the global optimum. Especially, each agent performs its local search in an exclusive area of the search space: an exclusion radius is attributed to each agent. This way, if several agents converge to a same local optimum, then only one of them can continue to converge to this local optimum: all the other conflicting agents are reinitialized elsewhere in the search space. Another important strategy is the use of two levels of precision in the stopping criterion of the local searches of the agents. In this way, we prevent the fine-tuning of low quality solutions, which could lead to a waste of fitness function evaluations; only the best solution found by MLSDO is fine-tuned. Furthermore, the local optima found during the optimization process are archived, to accelerate the detection of the global optimum after a change in the objective function. More details about this algorithm are in [6].

In static optimization, there is no need for the tracking agents of MLSDO. However, the exploring agents and their coordination can be useful to perform an initial wide exploration of the search space. MLSDO makes indeed use of fast converging local searches initialized in order to cover at best the search space. The local optima found by these local searches can then be used to create the initial population of a population-based metaheuristic.

2.2 Overview of SHADE algorithm

SHADE maintains two historical archives of H entries for the CR and F control parameters of DE, denoted by M_{CR} and M_F, respectively. It also maintains an archive of C_A inferior individuals, denoted by A. Its overall implementation is shown in Figure 1, where the population, denoted by P, is made of N individuals. For each generation, in order to generate the control parameters CR_i and F_i of each individual x_i, it is necessary to randomly select an index r_i. It is the index of an entry of M_{CR} denoted by M_{CR,r_i}, and of an entry of M_F denoted by M_{F,r_i}. The functions $randn$ and $randc$ generate random values from normal and Cauchy distributions, respectively. They take two parameters: the mean followed by the variance of the distribution. The control parameters CR_i and F_i used by successful individuals are stored in two archives, denoted by S_{CR} and S_F, respectively. In line 11, a mutant vector u_i is generated by applying the current-to-$p$best/1/bin mutation strategy: $u_i = x_i + F_i(x_{pbest} - x_i) + F_i(x_{r1} - x_{r2})$. Individual x_{pbest} is randomly selected from the best $N \times p$ ($p \in [0, 1]$) members of the current generation. The individuals x_{r1} and x_{r2} are randomly selected from P and $P \cup A$ such that they differ from each other as well as x_i. More details about this algorithm are in [9].

2.3 Overview of SPSO2011 algorithm

In SPSO2011, each particle has a *position* and a *velocity* in the search space. For the ith particle, they are denoted by X_i and V_i, respectively. The best-known position of the ith particle (known as *personal best*) is denoted by P_i,

SHADE
---

1 $A \leftarrow \varnothing$
2 Randomly initialize $P = \{x_1, x_2, \ldots, x_N\}$
3 Set all values in M_{CR} and M_F to 0.5
4 **while** the stopping criteria are not met **do**
5 $\quad$ $S_{CR} \leftarrow \varnothing$
6 $\quad$ $S_F \leftarrow \varnothing$
7 $\quad$ **for** $i = 1$ **to** N **do**
8 $\quad\quad$ Randomly initialize $r_i \in \{1, 2, \ldots, H\}$
9 $\quad\quad$ $CR_i \leftarrow randn(M_{CR,r_i}, 0.1)$
10 $\quad\quad$ $F_i \leftarrow randc(M_{F,r_i}, 0.1)$
11 $\quad\quad$ Generate trial vector u_i
12 $\quad$ **end**
13 $\quad$ **for** $i = 1$ **to** N **do**
14 $\quad\quad$ **if** $f(u_i)$ is better than $f(x_i)$ **then**
15 $\quad\quad\quad$ $A \leftarrow A \cup \{x_i\}$
16 $\quad\quad\quad$ $S_{CR} \leftarrow S_{CR} \cup \{CR_i\}$
17 $\quad\quad\quad$ $S_F \leftarrow S_F \cup \{F_i\}$
18 $\quad\quad\quad$ $x_i \leftarrow u_i$
19 $\quad\quad$ **end**
20 $\quad$ **end**
21 $\quad$ If necessary, delete randomly selected individuals from A so that $|A| \le C_A$
22 $\quad$ **if** $S_{CR} \ne \varnothing$ and $S_F \ne \varnothing$ **then**
23 $\quad\quad$ Update M_{CR} and M_F based on S_{CR} and S_F
24 $\quad$ **end**
25 **end**

Fig. 1. SHADE algorithm

whereas the best-known position of its neighboring particles (known as *local best*) is denoted by G_i. The position of the ith particle is updated according to (1), where $i = 1, 2, \ldots, N$ and N is the size of the swarm.

$$X_i \leftarrow X_i + V_i \tag{1}$$

SPSO2011 exploits the idea of *rotational invariance*. It starts by defining a center of gravity (Gr_i) around three points: the current position (X_i), a point a little "beyond" the best previous position (p_i), and a point a little "beyond" the best previous position in the neighborhood (l_i), as follows:

$$p_i = X_i + c_1(P_i - X_i) \tag{2}$$

$$l_i = X_i + c_2(G_i - X_i) \tag{3}$$

$$Gr_i = \frac{1}{3}(X_i + p_i + l_i) \tag{4}$$

where c_1 and c_2 are two parameters of the algorithm. Then, a random point is generated in the hypersphere $\mathcal{H}(Gr_i, \|Gr_i - X_i\|)$, and the velocity is updated as follows:

$$V_i \leftarrow \omega V_i + r\mathcal{H}(Gr_i, \|Gr_i - X_i\|) - X_i \qquad (5)$$

where ω is a parameter of the algorithm and r is a random number in $[0, 1]$. The position of the particle is updated according to (1). A parameter K is used to generate the particles neighborhood. More details are in [2].

3 The proposed hybrid algorithm

The proposed algorithm, called MSS, makes use of a HRH of the MLSDO, SHADE and SPSO2011 algorithms. At first, MLSDO is used to explore the search space using fast local searches starting from distant initial solutions. Since the objective function is not dynamic for the problems at hand, only one exploring agent and no tracking agent is used. A stagnation criterion is used to stop MLSDO if it is not able to improve the fitness value of the best solution found for a given number, denoted by $stop_{MLSDO}$, of successive objective function evaluations. Furthermore, a maximum number of evaluations is defined for MLSDO, denoted by max_{MLSDO}. Hence, if the stagnation criterion is satisfied or if the number of evaluations performed by MLSDO reaches max_{MLSDO}, then MLSDO stops its execution and SHADE (the next algorithm in the proposed HRH) starts its execution. The population of SHADE is initialized with the best local optima found by the local searches performed by MLSDO. The number of local optima used to create the initial population of SHADE is denoted by top_{MLSDO}. The other individuals of the population are randomly initialized uniformly in the search space. As for MLSDO, a stagnation criterion is also defined for SHADE. The maximum number of successive non improving evaluations that SHADE can perform is denoted by $stop_{SHADE}$. If this stagnation criterion is satisfied, then SPSO2011 is executed (the last algorithm in the proposed HRH) for the remaining evaluations that can be performed by MSS. In the proposed hybrid algorithm, SPSO2011 is especially used for exploitation. Its initial population is the same as the one of the last generation of SHADE.

4 Experimental protocol and parameter setting

4.1 The BBOB 2015 benchmark

The BBOB 2015 benchmark is made of 24 noise-free real-parameter single-objective test functions categorized into five groups (see Table 1) [3]. These functions have been proposed to reflect, at least to a certain extend and with a few exceptions, a difficult portion of the problem distribution that will be seen in practice. The search interval for each dimension of all functions is $[-5, 5]$. Each function is randomly shifted to produce 15 instances with different positions

Separable functions	Multi-modal functions with adequate global structure
f_1 : Sphere	f_{15} : Rastrigin
f_2 : Ellipsoidal	f_{16} : Weierstrass
f_3 : Rastrigin	f_{17} : Schaffers F7
f_4 : Büche-Rastrigin	f_{18} : Schaffers F7, moderately ill-conditioned
f_5 : Linear Slope	f_{19} : Composite Griewank-Rosenbrock F8F2

Functions with low or moderate conditioning	Multi-modal functions with weak global structure
f_6 : Attractive Sector	f_{20} : Schwefel
f_7 : Step Ellipsoidal	f_{21} : Gallagher's Gaussian 101-me Peaks
f_8 : Rosenbrock, original	f_{22} : Gallagher's Gaussian 21-hi Peaks
f_9 : Rosenbrock, rotated	f_{23} : Katsuura
	f_{24} : Lunacek bi-Rastrigin

Functions with high conditioning and unimodal

f_{10} : Ellipsoidal	f_{13} : Sharp Ridge
f_{11} : Discus	f_{14} : Different Powers
f_{12} : Bent Cigar	

Table 1. BBOB 2015 test functions

and values of the global optimum. A performance measure, called the expected running time (ERT), is typically used to quantify and compare performance of numerical optimization algorithms on this benchmark. It depends on a given target function value, $f_{\mathrm{t}} = f_{\mathrm{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_{t}, summed over all trials and divided by the number of trials that actually reached f_{t} [4]. Statistical significance is tested with the rank-sum test for a given target Δf_{t} using, for each trial, either the number of needed function evaluations to reach Δf_{t} (inverted and multiplied by −1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

In our empirical analysis, two experimental protocols are used to evaluate the performance of MSS on the BBOB 2015 benchmark. In the first one, called PROTOCOL1, we fixed the maximum number of evaluations allowed to solve a function at the same value as in BBComp, i.e. $100D^2$, where D is the number of dimensions. We chose to solve 5 times each instance of each test function (5 independent trials), which makes a total of 75 runs of an algorithm per test function. The goal of this first protocol is to be closer to the one of BBComp. In the second protocol, called PROTOCOL2, we run the algorithms with a budget of $10^5 D$ evaluations on each instance of each test function. We follow the typical methodology of BBOB 2015, i.e. each instance of each test function is solved only one time, and ERT is used to present the results.

4.2 The Black Box Optimization Competition

BBComp is the first competition in continuous black-box optimization where test problems are truly black boxes for participants. It is also the first web/online optimization competition in the direct search domain. The nature of the test functions used for the competition is unknown to the participants. Furthermore, each participant can use only one algorithm to solve each test function only once. His algorithm have to solve test functions in 2, 4, 5, 8, 10, 16, 20, 32, 40 and 64 dimensions. For each of these numbers of dimensions, 100 test functions have to be solved, which makes a total of 1000 problems. The search bound for each dimension of all functions is $[0, 1]$. The maximum number of evaluations allowed to solve a function is $100D^2$, where D is the number of dimensions.

After the competition, all participants are ranked for each problem based on their performance. Let k be the rank of a participant for a problem. Then, a score can be computed for this problem according to k. The sum of these scores, for all problems, gives the overall rank of a participant. This overall rank is used to sort the participants and to determine the winner. A simple way to compute the score of a participant for a problem is to set it to k. It leads to an overall ranking system called "sum of ranks". Another way to compute the score of a participant for a problem is to set it to $\max\{0, \log((n+1)/2) - \log(k)\}$, where n is the number of participants. In effect, these scores amplify differences of good (low) ranks and hide differences of bad (high) ranks k. It puts an emphasis on the top ranks, and it leads to the official overall ranking system of BBComp. More details about this competition can be found in [7].

4.3 Parameter setting

In MSS, since MLSDO is especially used for exploration, it does not need to precisely converge to a local optimum. Hence, the parameters that control the precision of the convergence in MLSDO are set consequently, i.e. the parameters denoted by δ_{ph} and δ_{pl}, called the highest and the lowest precision parameters of the stagnation criterion of the local searches, respectively, can be left to high values. All the parameters of MSS, empirically fitted, are presented in Table 2, where D is the number of dimensions of the problem and B is the budget allowed for its optimization, i.e. the maximum number of allowed evaluations. These parameters are used for all functions of BBOB 2015 and BBComp. They are also fitted for the unmodified SHADE and SPSO2011 algorithms, i.e. not as components of MSS. For the unmodified MLSDO algorithm, changing the parameters in Table 2 can significantly improve its performance for 10 functions of BBOB 2015 (using $\delta_{ph} = 1.0E{-}11$ and $\delta_{pl} = 1.0E{-}7$). However, it also significantly worsen its performance for 11 functions. Hence, the parameters in Table 2 can be considered as fitted for MSS and for the unmodified MLSDO, SHADE and SPSO2011 algorithms.

5 Experimental results and discussion

5.1 Results for the BBOB 2015 benchmark

The results obtained by MSS, MLSDO, SHADE and SPSO2011 on the BBOB 2015 benchmark, following PROTOCOL1, are presented in Table 3, where the columns "D" and "Pb" give the number of dimensions and the test function used, respectively. For each test function, the best value of the objective function found by an algorithm, averaged over 75 runs, is presented. The standard deviation, denoted by SD, is also given. The Kruskal-Wallis statistical test has been used, at 95% confidence level, to determine if a significant difference exists between the results obtained by the algorithms for each test case. If this test indicates that there is a significant difference between the performance of the algorithms, then the Tukey-Kramer post hoc test is used to determine which algorithms perform differently from MSS. If an algorithm performs significantly better than MSS, then the letter B is written in the "C" column. If it performs significantly worse, then the letter W is written. The best results obtained for each test case, along with the results that are not significantly different from the best ones according to the Tukey-Kramer post hoc test, are written in bold.

As we can see, in 32 and 64 dimensions, MSS performs significantly better than MLSDO for 18 functions, similarly for 3 functions, and worse for 3 functions. Compared to SPSO2011, MSS performs significantly better for 20 functions in 32 dimensions and for 21 functions in 64 dimensions. It performs similarly for 3 functions in 32 dimensions and for 2 functions in 64 dimensions. Finally, it performs worse than SPSO2011 for only 1 function both in 32 and 64 dimensions. Hence, compared to MLSDO and to SPSO2011, the performance of MSS is significantly better for most test functions.

Compared to SHADE, MSS performs better for 6 functions in 32 dimensions and for 7 functions in 64 dimensions. It performs similarly for 14 functions in 32 dimensions and for 13 functions in 64 dimensions. Finally, it performs worse than SHADE for 4 functions both in 32 and 64 dimensions. Among the functions for which SHADE performs better than MSS, only two are the same in 32 and in 64 dimensions, i.e. f_{11} (Discus function) which is unimodal but with a very high conditioning, and f_{15} (Rastrigin function). This last one is actually a modified version of the original well-know Rastrigin function, in order to make it a non-separable less regular counterpart of f_3. The functions for which MSS performs significantly better than SHADE are the same in 32 and in 64 dimensions. Most of these functions belong to the category of "multi-modal functions with weak global structure".

Table 4 presents the contribution of each component of MSS in the convergence of the algorithm. In columns "Av." and "SD", the average number of evaluations and its standard deviation, given for each test case, are expressed in percentage of the total number of evaluations allowed for this test case. The average fitness improvement (column "AFI"), given for SHADE (respectively SPSO2011), is the percentage by which the fitness value of the best solution

	Parameter	Value
MLSDO	Exclusion radius of the agents	0.07
	Initial step size of tracking agents	0.005
	δ_{ph}	0.001
	δ_{pl}	0.1
SHADE	N	$\lceil 0.85D+9\rceil$
	C_A	$2N$
	H	N
	p	0.1

	Parameter	Value
SPSO2011	ω	0.721
	c_1, c_2	1.193
	K	3
Hybridization	max_{MLSDO}	$0.3B$
	$stop_{MLSDO}$	$0.112B$
	top_{MLSDO}	$0.5N$
	$stop_{SHADE}$	$0.337B$

Table 2. MSS parameters

		MSS		MLSDO			SHADE			SPSO2011		
D	Pb	Average	SD	Average	SD	C	Average	SD	C	Average	SD	C
32	f_1	**0.00E+00**	**0.00E+00**	1.11E−03	2.20E−04	W	**0.00E+00**	**0.00E+00**		**0.00E+00**	**0.00E+00**	
	f_2	**0.00E+00**	**0.00E+00**	3.63E−04	3.75E−04	W	**0.00E+00**	**0.00E+00**		1.52E+03	5.95E+02	W
	f_3	**2.65E−02**	**1.61E−01**	3.14E+01	4.45E+00	W	**3.18E−01**	**9.13E−01**		8.57E+01	2.17E+01	W
	f_4	**4.17E−01**	**8.49E−01**	3.61E+01	5.64E+00	W	2.47E+00	2.98E+00	W	1.32E+02	3.34E+01	W
	f_5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		**0.00E+00**	**0.00E+00**		2.11E+01	8.10E+00	W
	f_6	4.57E−05	1.91E−04	1.94E+00	2.21E+00	W	2.44E−05	1.37E−04	B	**1.78E−07**	**1.47E−06**	B
	f_7	**9.69E+00**	**4.71E+00**	4.11E+01	1.21E+01	W	**1.03E+01**	**4.78E+00**		2.41E+01	1.01E+01	W
	f_8	**6.38E−01**	**1.47E+00**	1.72E+01	7.42E+00	W	**7.44E−01**	**1.56E+00**		3.48E+01	2.49E+01	W
	f_9	**8.91E+00**	**1.83E+00**	2.25E+01	4.06E+00	W	**7.27E+00**	**1.94E+00**		2.64E+01	1.18E+01	W
	f_{10}	**2.78E+02**	**1.64E+02**	2.06E+03	8.95E+02	W	**1.71E+02**	**1.03E+02**		1.46E+03	7.15E+02	W
	f_{11}	1.26E+01	4.03E+01	3.10E+02	4.80E+01	W	**6.60E+00**	**3.18E+01**	B	4.16E+01	1.24E+01	W
	f_{12}	**1.22E+00**	**2.36E+00**	**2.65E−01**	**5.64E−01**		**1.16E+00**	**3.71E+00**		4.49E+00	8.21E+00	W
	f_{13}	**4.72E−01**	**2.04E+00**	5.87E−01	6.65E−01	W	4.01E+00	6.12E+00	W	5.55E+00	8.01E+00	W
	f_{14}	**3.55E−05**	**1.14E−05**	1.41E−02	2.91E−03	W	**3.01E−05**	**7.99E−06**		2.47E−04	2.15E−05	W
	f_{15}	6.79E+01	2.22E+01	2.10E+02	3.42E+01	W	**4.03E+01**	**1.06E+01**	B	7.28E+01	2.43E+01	
	f_{16}	9.78E+00	2.28E+00	**7.62E+00**	**1.51E+00**	B	1.13E+01	1.76E+00	W	1.61E+01	3.02E+00	W
	f_{17}	**1.69E−01**	**1.13E−01**	1.03E+01	2.82E+00	W	**1.66E−01**	**1.46E−01**		8.22E−01	3.63E−01	W
	f_{18}	**8.56E−01**	**5.01E−01**	4.02E+01	9.96E+00	W	**8.20E−01**	**4.77E−01**		3.01E+00	1.18E+00	W
	f_{19}	3.19E+00	9.57E−01	5.13E+00	1.00E+00	W	**2.76E+00**	**4.65E−01**	B	4.23E+00	4.33E−01	W
	f_{20}	**6.84E−01**	**1.12E−01**	8.85E−01	9.82E−02	W	8.10E−01	1.28E−01	W	2.06E+00	2.05E−01	W
	f_{21}	2.69E+00	3.33E+00	**5.98E−01**	**7.26E−01**	B	4.65E+00	5.10E+00		4.37E+00	6.40E+00	
	f_{22}	**3.57E+00**	**4.15E+00**	**1.09E+00**	**1.16E+00**		8.78E+00	8.19E+00	W	7.41E+00	6.62E+00	W
	f_{23}	8.04E−01	2.79E−01	**5.46E−01**	**1.78E−01**	B	1.03E+00	2.22E−01	W	2.22E+00	3.49E−01	W
	f_{24}	**8.75E+01**	**1.73E+01**	6.15E+02	2.77E+02	W	**8.34E+01**	**1.15E+01**		1.65E+02	1.74E+01	W
64	f_1	**0.00E+00**	**0.00E+00**	2.12E−03	3.05E−04	W	**0.00E+00**	**0.00E+00**		**0.00E+00**	**0.00E+00**	
	f_2	**0.00E+00**	**0.00E+00**	2.52E−04	2.23E−04	W	**0.00E+00**	**0.00E+00**		2.66E+03	6.21E+02	W
	f_3	**0.00E+00**	**0.00E+00**	6.54E+01	7.20E+00	W	**5.70E−01**	**2.35E+00**		2.23E+02	4.30E+01	W
	f_4	**1.59E−01**	**4.02E−01**	7.85E+01	7.85E+00	W	5.80E+00	8.39E+00	W	3.50E+02	5.67E+01	W
	f_5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		**0.00E+00**	**0.00E+00**		7.15E+01	1.79E+01	W
	f_6	1.90E−04	9.21E−04	4.71E+00	3.61E+00	W	6.05E−05	3.75E−04		**4.19E−06**	**1.61E−05**	B
	f_7	**2.99E+01**	**9.00E+00**	1.41E+02	2.56E+01	W	**3.19E+01**	**8.42E+00**		8.36E+01	2.33E+01	W
	f_8	**4.25E−01**	**1.24E+00**	4.92E+01	6.60E+00	W	**4.78E−01**	**1.30E+00**		8.80E+01	3.46E+01	W
	f_9	**3.51E+01**	**6.52E+00**	5.34E+01	3.11E+00	W	**3.46E+01**	**8.77E+00**		6.32E+01	2.06E+01	W
	f_{10}	4.54E+02	1.87E+02	2.25E+03	1.10E+03	W	**2.95E+02**	**1.43E+02**	B	2.80E+03	9.58E+02	W
	f_{11}	5.21E−02	4.66E−02	6.32E+02	7.80E+01	W	**5.74E−03**	**6.37E−03**	B	2.63E+01	6.08E+00	W
	f_{12}	**1.61E+00**	**2.73E+00**	**3.62E−01**	**3.43E−01**		2.38E+00	3.69E+00		3.83E+00	6.78E+00	
	f_{13}	**2.82E−01**	**3.86E−01**	6.09E−01	6.08E−01	W	2.49E+00	3.56E+00	W	4.23E+00	6.60E+00	W
	f_{14}	3.09E−05	5.86E−06	2.04E−02	3.38E−03	W	**2.61E−05**	**4.46E−06**	B	3.09E−04	1.51E−05	W
	f_{15}	1.19E+02	2.95E+01	6.25E+02	7.83E+01	W	**8.93E+01**	**1.84E+01**	B	2.07E+02	5.03E+01	W
	f_{16}	1.51E+01	2.70E+00	**1.24E+01**	**2.09E+00**	B	1.83E+01	1.73E+00	W	2.29E+01	2.26E+00	W
	f_{17}	**2.70E−01**	**1.26E−01**	1.33E+01	2.87E+00	W	**2.24E−01**	**1.06E−01**		1.90E+00	4.60E−01	W
	f_{18}	**1.39E+00**	**5.46E−01**	5.07E+01	1.15E+01	W	**1.34E+00**	**4.96E−01**		6.17E+00	1.50E+00	W
	f_{19}	**3.96E+00**	**5.19E−01**	7.53E+00	1.14E+00	W	**3.83E+00**	**3.38E−01**		4.93E+00	3.96E−01	W
	f_{20}	**6.55E−01**	**6.97E−02**	9.33E−01	7.08E−02	W	9.46E−01	2.22E−01	W	2.13E+00	1.12E−01	W
	f_{21}	1.63E+00	1.10E+00	**2.91E−01**	**5.43E−01**	B	6.06E+00	5.54E+00	W	5.57E+00	5.34E+00	W
	f_{22}	**1.98E+00**	**2.22E+00**	**1.38E+00**	**1.51E+00**		1.20E+01	1.03E+01	W	1.20E+01	1.06E+01	W
	f_{23}	9.75E−01	2.64E−01	**7.46E−01**	**1.82E−01**	B	1.48E+00	2.27E−01	W	2.69E+00	3.94E−01	W
	f_{24}	**1.74E+02**	**2.12E+01**	1.97E+03	2.70E+02	W	**1.67E+02**	**1.89E+01**		3.43E+02	4.21E+01	W

Table 3. Results of the compared algorithms on the BBOB 2015 benchmark

Table 4 — 32-D

Pb	MLSDO Av.	SD	SHADE AFI	Av.	SD	SPSO2011 AFI	Av.	SD
1	12.53	0.06	100.00	58.85	6.45	0.00	28.62	6.46
2	13.67	0.09	100.00	63.49	7.51	0.00	22.84	7.52
3	20.15	5.78	99.93	79.85	5.78	0.00	0.00	0.00
4	18.69	5.91	99.11	81.31	5.91	0.00	0.00	0.00
5	11.56	0.02	0.00	33.82	0.00	0.00	54.62	0.02
6	23.24	4.91	100.00	76.76	4.91	0.00	0.00	0.00
7	18.80	6.28	83.78	52.80	14.07	0.18	28.40	15.03
8	24.32	5.50	97.79	75.68	5.50	0.00	0.00	0.00
9	21.49	5.93	65.23	78.51	5.93	0.00	0.00	0.00
10	30.00	0.00	95.11	70.00	0.00	0.00	0.00	0.00
11	23.99	6.56	96.55	76.01	6.56	0.00	0.00	0.00
12	17.65	4.40	58.06	82.35	4.40	0.00	0.00	0.00
13	18.93	4.85	78.06	81.04	4.84	0.00	0.03	0.22
14	14.27	2.22	99.79	85.73	2.22	0.00	0.00	0.00
15	19.83	6.01	74.67	80.17	6.01	0.00	0.00	0.00
16	20.26	5.45	14.75	52.80	22.56	0.02	26.94	22.29
17	23.05	6.51	98.86	76.84	6.42	0.00	0.12	0.96
18	24.36	6.36	98.99	75.47	6.21	0.00	0.18	1.13
19	20.79	5.94	55.38	73.15	17.19	0.23	6.06	14.92
20	18.11	5.56	35.01	81.89	5.56	0.00	0.00	0.00
21	19.59	6.13	0.07	61.08	7.54	0.00	19.33	10.00
22	18.05	5.05	0.05	60.91	7.35	0.00	21.04	9.00
23	17.03	4.87	1.83	35.87	10.15	0.36	47.10	10.81
24	19.75	5.97	89.75	80.25	5.97	0.00	0.00	0.00

32-D

Table 4 — 64-D

Pb	MLSDO Av.	SD	SHADE AFI	Av.	SD	SPSO2011 AFI	Av.	SD
1	11.92	0.01	100.00	56.02	11.48	0.00	32.06	11.48
2	12.51	0.04	100.00	62.48	13.28	0.00	25.00	13.28
3	18.73	5.89	100.00	81.27	5.89	0.00	0.00	0.00
4	18.26	6.17	99.83	81.74	6.17	0.00	0.00	0.00
5	11.40	0.01	0.00	33.77	0.00	0.00	54.82	0.01
6	20.47	3.62	100.00	79.53	3.62	0.00	0.00	0.00
7	19.77	6.08	83.39	61.60	19.03	0.08	18.63	18.97
8	23.75	6.25	99.33	76.25	6.25	0.00	0.00	0.00
9	19.19	5.38	38.56	80.81	5.38	0.00	0.00	0.00
10	30.00	0.00	93.55	70.00	0.00	0.00	0.00	0.00
11	21.60	6.56	99.99	78.40	6.56	0.00	0.00	0.00
12	16.42	4.40	22.91	83.58	4.40	0.00	0.00	0.00
13	16.04	3.99	81.54	81.10	4.40	0.00	2.86	3.81
14	13.07	1.55	99.86	86.93	1.55	0.00	0.00	0.00
15	21.10	6.22	84.49	78.90	6.22	0.00	0.00	0.00
16	19.88	6.00	2.81	47.25	20.56	0.004	32.86	20.11
17	21.92	5.92	98.44	74.24	7.65	0.00	3.83	7.23
18	23.70	5.89	98.02	71.44	6.60	0.00	4.86	6.77
19	18.72	5.28	57.17	81.28	5.28	0.00	0.00	0.00
20	18.48	5.65	37.52	81.52	5.65	0.00	0.00	0.00
21	20.40	6.38	0.12	59.90	10.15	0.00	19.70	11.46
22	17.96	5.11	0.07	57.79	11.80	0.00	24.25	13.60
23	15.35	3.99	1.36	39.10	14.07	0.05	45.55	13.83
24	15.89	3.79	91.74	84.11	3.79	0.00	0.00	0.00

64-D

Table 4. Contribution of MLSDO, SHADE and SPSO2011 in the convergence of MSS

found by MLSDO (respectively SHADE) is improved by SHADE (respectively SPSO2011).

In this table, we can see that the test functions for which MLSDO contributes the most in improving the solution found by MSS (the functions for which the AFI of SHADE and SPSO2011 are close to 0) are f_5, f_{21}, f_{22} and f_{23} in 32 dimensions, and f_5, f_{16}, f_{21}, f_{22} and f_{23} in 64 dimensions. For the other functions, SHADE provides an important contribution in the convergence of MSS. Besides, we can see that all AFI values of SPSO2011 are close to 0, which means that SHADE is able to precisely converge to a local or global optimum. Yet, SPSO2011 contributes to the exploitation process of MSS for the functions f_7, f_{16}, f_{19} and f_{23} in 32 dimensions, and f_7, f_{16} and f_{23} in 64 dimensions.

To further study the effect of the hybridization of SHADE with MLSDO and SPSO2011, a comparison of the performance of MSS and SHADE following PROTOCOL2 is presented in Table 5. Considering the lowest target Δf-value for which at least one of the two compared algorithms has an ERT different from ∞, MSS obtains a significantly better ERT than SHADE for 5 functions. On the other hand, SHADE gets also a significantly better ERT than MSS for 5 functions. Hence, the results are mitigated in terms of ERT. However, as we can see in the column "#succ", MSS has a better success rate in reaching the final target than SHADE for 4 functions, i.e. f_3, f_4, f_8 and f_{21}.

5.2 Results at the Black Box Optimization Competition

BBComp occurs every year since 2015. The BBComp editions where we participated are called "BBComp2015CEC" and "BBComp2016-1OBJ" tracks. In the following subsections, we present the results obtained at BBComp using MSS.

Δf_{opt}	10	0.1	1e-3	1e-5	1e-7	#succ
f_1	83	83	83	83	83	30/30
1: MSS	**7.8**^{*3}	**14**^{*3}	750	806	854	15/15
2: SHA	22	54	**85**^{*3}	**116**^{*3}	**150**^{*3}	15/15
f_2	796	799	800	802	804	15/15
1: MSS	**2.5**^{*3}	**3.1**^{*3}	**3.8**^{*3}	96	102	15/15
2: SHA	11	15	18	**23**^{*3}	**27**^{*3}	15/15
f_3	15526	15612	15646	15651	15656	15/15
1: MSS	7.1	8.2	9.0	10	11	15/15
2: SHA	**2.1**^{*3}	12	13	13	14	11/15
f_4	15536	15659	15703	15733	2.8e5	9/15
1: MSS	7.2	11	12	13	0.75	14/15
2: SHA	**2.7**^{*3}	106	106	106	6.0	3/15
f_5	98	120	121	121	121	15/15
1: MSS	**3.6**^{*3}	**3.0**^{*3}	**3.0**^{*3}	**3.0**^{*3}	**3.0**^{*3}	15/15
2: SHA	33	59	91	123	155	15/15
f_6	3507	7168	11538	15007	19222	15/15
1: MSS	7.3	14	12	12	11	15/15
2: SHA	4.0	**7.1**^{*3}	**8.0**^{*3}	**9.0**^{*3}	10	15/15
f_7	10698	41037	66294	66294	68145	15/15
1: MSS	83	∞	∞	∞	∞	0/15
2: SHA	524	∞	∞	∞	∞	0/15
f_8	7080	11012	11430	11701	11969	15/15
1: MSS	22	19	19	19	20	14/15
2: SHA	**8.6**^{*3}	21	21	21	22	11/15
f_9	6122	13300	13651	13909	14142	15/15
1: MSS	38	34	∞	∞	∞	0/15
2: SHA	**32**^{*3}	29	145	∞	∞	0/15
f_{10}	25890	36796	56007	65128	70824	15/15
1: MSS	∞	∞	∞	∞	∞	0/15
2: SHA	∞	∞	∞	∞	∞	0/15
f_{11}	2368	11681	29749	38949	48211	15/15
1: MSS	61	22	23	∞	∞	0/15
2: SHA	**24**^{*3}	**16**^{*3}	**10***	**51***	∞	0/15
f_{12}	4169	9174	13146	22758	25192	15/15
1: MSS	2.2	45	225	∞	∞	0/15
2: SHA	10	30	144	∞	∞	0/15

Δf_{opt}	10	0.1	1e-3	1e-5	1e-7	#succ
f_{13}	2029	8734	71936	98467	1.2e5	15/15
1: MSS	**2.1***	**26***	∞	∞	∞	0/15
2: SHA	34	299	∞	∞	∞	0/15
f_{14}	304	777	2207	4825	57711	15/15
1: MSS	**1.9**^{*3}	**2.4**^{*3}	31	161	∞	0/15
2: SHA	4.9	6.7	**8.7**^{*3}	125	∞	0/15
f_{15}	1.9e5	1.0e6	1.1e6	1.1e6	1.1e6	15/15
1: MSS	∞	∞	∞	∞	∞	0/15
2: SHA	∞	∞	∞	∞	∞	0/15
f_{16}	5244	3.2e5	1.4e6	2.0e6	2.0e6	15/15
1: MSS	**120***	∞	∞	∞	∞	0/15
2: SHA	1090	∞	∞	∞	∞	0/15
f_{17}	399	14158	51958	1.3e5	2.7e5	14/15
1: MSS	168	190	∞	∞	∞	0/15
2: SHA	**1.4**^{*3}	43	∞	∞	∞	0/15
f_{18}	1442	47068	1.9e5	6.7e5	9.5e5	6/15
1: MSS	65	∞	∞	∞	∞	0/15
2: SHA	**2.7**^{*3}	∞	∞	∞	∞	0/15
f_{19}	1	1.4e6	2.6e7	4.5e7	4.5e7	8/15
1: MSS	7350	∞	∞	∞	∞	0/15
2: SHA	**842**^{*2}	∞	∞	∞	∞	0/15
f_{20}	222	1.6e8	∞	∞	∞	0
1: MSS	**3.3**^{*3}	∞	∞	∞	∞	0/15
2: SHA	8.0	∞	∞	∞	∞	0/15
f_{21}	1044	1.0e5	1.0e5	1.0e5	1.0e5	26/30
1: MSS	6.3	**3.9**^{*2}	**3.9**^{*2}	**4.2**^{*2}	**4.5**^{*2}	8/15
2: SHA	98	∞	∞	∞	∞	0/15
f_{22}	3090	6.5e5	6.5e5	6.5e5	6.5e5	8/30
1: MSS	1.6	∞	∞	∞	∞	0/15
2: SHA	48	∞	∞	∞	∞	0/15
f_{23}	7.1	75453	1.3e6	3.2e6	3.4e6	15/15
1: MSS	1.3	∞	∞	∞	∞	0/15
2: SHA	1.7	∞	∞	∞	∞	0/15
f_{24}	5.8e6	3.0e8	3.0e8	3.0e8	3.0e8	1/15
1: MSS	∞	∞	∞	∞	∞	0/15
2: SHA	∞	∞	∞	∞	∞	0/15

Table 5. Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in 40 dimensions. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\mathrm{opt}} + 10^{-8}$. Bold entries are statistically significantly better compared to the other algorithm, with $p = 0.05$ or $p = 10^{-k}$ where $k \in \{2, 3, 4, \dots\}$ is the number following the $\star$ symbol, with Bonferroni correction of 48

Results at the BBComp2015CEC track This edition of the competition occured in 2015, and the results obtained by the participants were presented at the IEEE Congress on Evolutionary Computation (CEC'2015). There were 25 participants to this competition, and MSS is among the winners at the 3rd place using the official ranking system. Using the "sum of ranks" ranking system, MSS is ranked at the 2nd place. MSS obtained bad results on low dimensional problems, i.e. in 2, 4, 5 and 8 dimensions. For these number of dimensions, MSS is ranked at the 16th, 13th, 9th and 5th place, respectively. However, using a higher number of dimensions, MSS shows a good performance compared to the other algorithms. For these higher dimensions, i.e. in 10, 16, 20, 32, 40 and 64 dimensions, the convergence curves and/or the ranking of all competing

algorithms are presented in figures 2 and 3 [7]. As we can see, MSS is the 1st ranked algorithm in 10 dimensions, the 2nd ranked algorithm in 32, 40 and 64 dimensions, and the 3nd ranked algorithm in 20 dimensions.

Results at the BBComp2016-1OBJ track This edition of the competition occured in 2016, and the results obtained by the participants were presented at the Genetic and Evolutionary Computation Conference (GECCO'2016). There were 14 participants to this competition, and MSS is ranked at the 6th place using the official ranking system. Using the "sum of ranks" ranking system, MSS is ranked at the 4th place. The results obtained at this edition of BBComp confirmed that MSS can be easily outperformed on low dimensional problems. Indeed, MSS is ranked at the 8th place and below for problems with a number
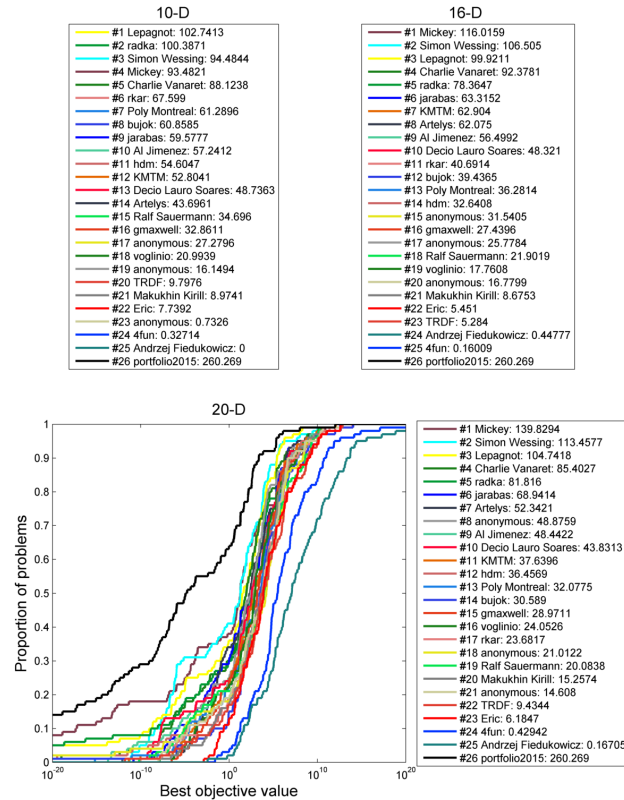
Fig. 2. Ranking of algorithms competing on the BBComp2015CEC track in 10, 16 and 20 dimensions. Due to space limitations, their convergence curves are displayed only in 20 dimensions (taken from [7])

132
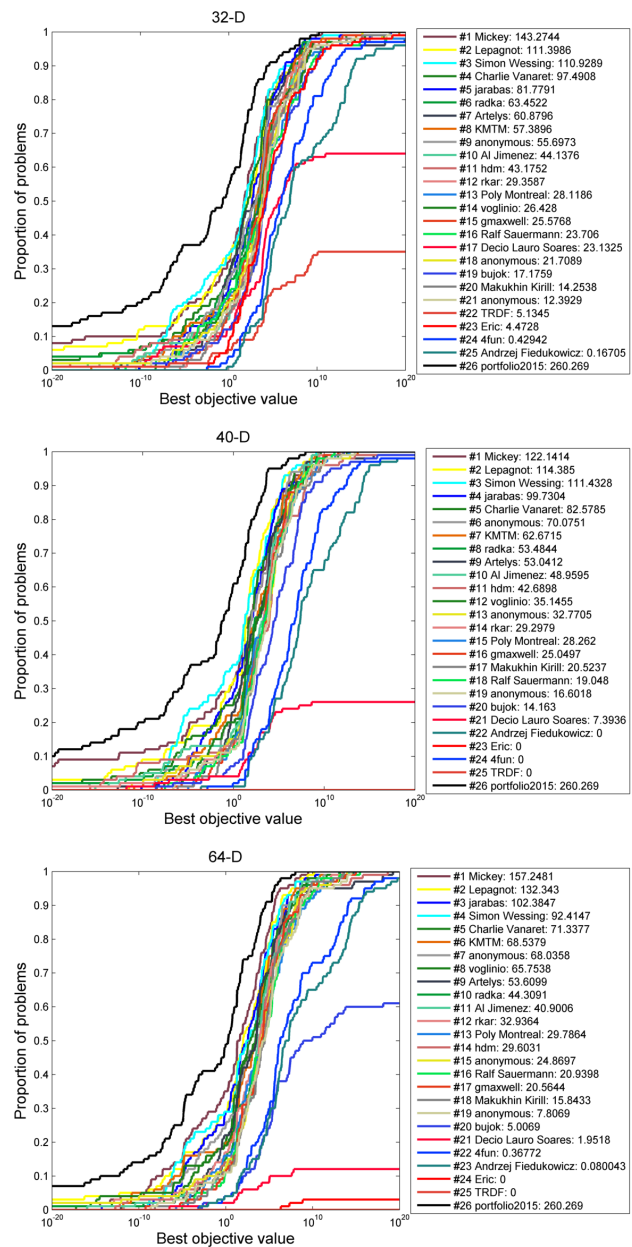
Fig. 3. Convergence curves and ranking of algorithms competing on the BB-Comp2015CEC track in 32, 40 and 64 dimensions (taken from [7])

of dimensions lower or equal to 5, but it is ranked at the 5th place and above for problems with a number of dimensions higher or equal to 8.

6   Conclusion

In this paper, a high-level relay hybrid metaheuristic, called MSS, is proposed. It combines three algorithms, i.e. MLSDO, SHADE and SPSO2011, that belong to different classes of metaheuristics. This combination of different metaheuristics leads to promising results on the BBOB 2015 benchmark, and at the BBComp competition. At the 2015 edition of BBComp, MSS has notably got the 3rd prize of the competition, among 25 competing algorithms. The experimental analysis suggests that MSS still needs to be improved for low dimensional problems.

As a perspective, we can investigate the replacement of the local searches of MLSDO by more adapted and efficient ones. SHADE and SPSO2011 can also be replaced by improved variants. Finally, the integration of machine learning techniques in the proposed hybrid metaheuristic can also be studied.

References

1. Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Natural Computing 7(1), 109–124 (2008)
2. Clerc, M.: Standard Particle Swarm Optimisation (2012), https://hal.archives-ouvertes.fr/hal-00764996, 15 pages
3. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noiseless Functions (2015), http://coco.lri.fr/downloads/download15.03/bbobdocfunctions.pdf, 126 pages
4. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-Parameter Black-Box Optimization Benchmarking: Experimental Setup. Tech. rep., INRIA (2014), http://coco.lri.fr/downloads/download15.02/bbobdocexperiment.pdf, 20 pages
5. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks IV. pp. 1942–1948. Perth, Australia (November 1995)
6. Lepagnot, J., Nakib, A., Oulhadj, H., Siarry, P.: A multiple local search algorithm for continuous dynamic optimization. Journal of Heuristics 19(1), 35–76 (2013)
7. Loshchilov, I., Glasmachers, T.: Black-Box Optimization Competition (2017), https://bbcomp.ini.rub.de
8. Talbi, E.G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics 8(5), 541–564 (2002)
9. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for differential evolution. In: IEEE Congress on Evolutionary Computation. pp. 71–78. Cancun, Mexico (June 2013)

Improved Hybrid Iterative Tabu Search for QAP using Distance Cooperation

Omar Abdelkafi[1], Lhassane Idoumghar[2], and Julien Lepagnot[2]

[1] Université Lille 1, LIFL/UMR CNRS 8022 - INRIA Lille Nord Europe, 59655 -
Villeneuve d'Ascq cedex, France
{omar.abdelkafi}@uha.fr
[2] Université de Haute-Alsace, LMIA, EA 3993, F-68093 Mulhouse, France
{lhassane.idoumghar, julien.lepagnot}@uha.fr

Abstract. The quadratic assignment problem can be considered as one of the hardest and most studied combinatorial problems. In this paper, we propose and analyze three distributed algorithms based on hybrid iterative tabu search. These algorithms follow the design of the parallel algorithmic level. A new mechanism to exchange information between processes is introduced. Through 34 well-known instances from QAPLIB benchmark, our algorithms produce competitive results. This experimentation shows that our best propositions can exceed or equal several leading algorithms from the literature in almost all the hardest benchmark instances.

Keywords: metaheuristics, Iterative tabu search, Quadratic assignment problem, cooperative and distributed algorithms.

1 Introduction

The Quadratic Assignment Problem (QAP) is an NP-hard problem. It is an important challenge for different areas. This problem is well known for its multiple applications in various fields such as: chemistry, transport, industry and many others. Works on some significant applications of QAP can be found in [6, 18–20]. The QAP was first introduced by Koopmans and Beckmann [10] to model a facility location problem. The objective is to find a minimum cost assignment of facilities to locations considering the flow of materials between facilities and the distance between locations. The problem can be formulated as follows:

$$\min_{p \in P} z(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)} \tag{1}$$

where f and d are the flow and distance matrices respectively, $p \in P$ represents a solution where p_i is the location assigned to facility i and P is the set of all n vector permutations. The objective is to minimize $z(p)$, which is the total cost assignment for the permutation p.

In this work, we propose and analyze three distributed Hybrid Iterative Tabu Search (HITS) algorithms using cooperation strategies or running independently in parallel machines. Our main objective is to compare the efficiency of information exchange and the efficiency of some well-known diversification methods.

2   Background

One of the first efficient approaches to solve the QAP is the Robust Tabu search (Ro-Ts) proposed by Taillard in 1991 [16]. Many other works are based on Ro-Ts to solve the QAP like [1, 8, 9]. P. Merz et al [12] propose a memetic algorithm to solve QAP (MA-QAP). This approach is a genetic algorithm using the Uniform Crossover (UX) combined with a 2-opt local search. In 2004, an Improved Hybrid Genetic Algorithm (IHGA) was implemented in [13]. Then, A. Misevicius et al [14] propose an Iterative Tabu Search (ITS) which is a set of successive Tabu Search (TS) using a random perturbation after each TS to give the algorithm a new starting solution. A complete survey until 2007 can be found in [11].

Later, Z. Drezner [4] experiments different hybrid genetic algorithms to solve the QAP. The best variant of this work is the MRT60 which is a genetic algorithm hybridized with a modified Ro-Ts [16]. Then, T. James et al [9] propose five TS variants extended from Ro-Ts and separated in two categories: the continuous diversification TS process and the discontinuous diversification TS process. Later, the work of U. Benlic et al [1] proposes an Iterative Local Search with a breakout strategy based on the history of the search (BLS). The local search uses also the delta matrix. According to the evolution of the search, the BLS chooses one from different degrees of perturbation to escape the local optima. Recently, U. Benlic et al [2] presented a memetic algorithm (BMA) for the well-known QAP. They combine the previous BLS [1] with the standard UX. This approach performs particularly well on unstructured instances from the QAP.

The parallel and distributed design for metaheuristics approaches has the capacity to improve the solution quality and to reduce the execution time. The computational cost of the QAP and its difficult search space make this problem suitable for parallelization.

E. G. Talbi [17] classifies the parallel and distributed design of metaheuristics in 3 levels:

- Algorithmic level: This level allows the run of many algorithms in parallel. The algorithms can run independently with different starting solutions and/or different parameters and choose the best results of the run. In this case, the result will be the same as if we execute all these algorithms sequentially, which means that we reduce the execution time. The algorithms can also cooperate with each other which means that the behavior of the metaheuristics will change and, in this case, the parallelization can improve the quality of the solutions.
- Iteration level: This level allows a parallelization in each iteration. This is a parallel evaluation and/or generation of neighborhood. Different parts of the neighborhood are processed in parallel. The behavior of the metaheuristic is not altered. The main objective is to speed up the algorithm by reducing the search time.
- Solution level: This level allows a parallelization of a single solution such as the evaluation of the objective functions or constraints for a generated solution. The behavior of the metaheuristic is not altered. The objective is mainly the speed up of the search.

3 Distributed and Cooperative Algorithms

In this work, the design used for all the versions is the algorithmic level. The aim of these versions is exclusively the improvement of results thanks to the distributed environment. The acceleration factor is untreated in this paper. We propose three versions: Distributed Hybrid Iterative Tabu Search called *D-HITS*, DIStance COoperation Hybrid Iterative Tabu Search using the Glover Diversification called *DISCO-HITS-GD* and DIStance COoperation Hybrid Iterative Tabu Search using the Uniform Crossover called *DISCO-HITS-UX*.

3.1 Distributed Hybrid Iterative Tabu Search

The Distributed Hybrid Iterative Tabu Search (D-HITS) approach is the first variant of our work. Different HITS are executed in a set of parallel machines from different starting solutions. In this version, there is no exchange of information between the distributed processes. Each HITS runs independently from the others with a different starting solution and different parameters such as the threshold values (*L1, L2* in Algorithm 1).

The TS represents the intensification mechanism of our work. It produces the best possible solution after a set of movements in the search space. After each TS, an adaptive diversification is applied to the global best solution. The aim is to discover a new promising region for the exploration of the next TS. The use of the global best solution structure allows the algorithm to reach a promising region of the search space. all these mechanisms constitute the HITS of this variant.

In each machine the search history is used to apply a preventive measure to escape from stagnancy. A counter w is initialized with 0 and after each TS without improvement, the counter w is incremented. If there is an improvement, w is reset to 0. The solution is perturbed after a set of TS executions without improvement. This way, if the algorithm is trapped inside a region of the space, then, additionally of the usual diversification, a part of the solution is perturbed to unlock the search. If w continues to grow, a complete re-localization is needed to explore other regions of the search space. Algorithm 1 is the pseudo-code of the D-HITS. Algorithm 2 [16] contains the TS method used in this work.

As shown in Algorithm 1 line 20, in every global iteration of the HITS (iteration between two consecutive TS), there is a diversification applied to the global best solution. For a set of successive global iterations, the global best solution can stay the same. Hence, if the diversification used is constant, then it will provide the same new starting solution over and over again. For this reason, the diversification needs to change from a global iteration to another. It has to be efficient and to use the structure of the global best solution. All these conditions are satisfied for the diversification proposed by [7]. The diversification procedure takes a solution (in our case, it takes the global best solution at every global iteration) and executes a set of permutations following a step value. The step value changes from a global iteration to another. This way, even if the global best solution stays the same for many global iterations of the HITS, the new

starting solution generated by this diversification will be different. Algorithm 3 presents the Glover Diversification (GD) pseudo-code.

Algorithm 1 D-HITS for each process

1: **Input**: *perturb*: % perturbation; n: size of solution; *cost*: cost of the current solution; *Fcost*: best cost found; $S_{current}$: current solution; S_{best}: best solution found; *L1,L2*: thresholds for preventive measures;
2: Initialization of the solution for the current process;
3: $w = 0$; /* is the counter to define the search history state */
4: Stagnancy = **false**;
5: **repeat**
6:    TS algorithm /* see Algorithm 2 */
7:    **if** cost < Fcost **then**
8:       /* improvement */
9:       Stagnancy = **false**; $w = 0$; $Fcost = cost$;
10:       Update the S_{best} with $S_{current}$;
11:    **else**
12:       w++;
13:    **end if**
14:    /* Condition of stagnancy */
15:    **if** $w ==$ L2 **then**
16:       Stagnancy = **true**;
17:    **end if**
18:    **if** Stagnancy == **false** **then**
19:       /* no stagnancy */
20:       Update $S_{current}$ with the **Diversification** of S_{best}; /* see Algorithm 3 */
21:    **else**
22:       /* Stagnancy */
23:       **Re-localization** of $S_{current}$;
24:       Stagnancy = **false**;
25:       $w = 0$;
26:    **end if**
27:    **if** $w ==$ L1 **then**
28:       **Perturbation** of $S_{current}$ with the *perturb* parameter;
29:    **end if**
30: **until** (Stop condition)

The preventive perturbation mechanism is based on the history of the search. It is a smart way to explore the search space. The main idea is to find other promising regions of the space for each process if the search stagnates. After some global iterations without improvement, w reaches the first threshold *L1* (Algorithm 1 line 27 to 29). The first measure to unlock the search process is to apply the diversification and to perturb randomly a portion of the solution provided by the diversification. This measure assumes that there is no possible improvement with the current diversification. Hence, a perturbation is applied in a small part of the solution to reuse the structure of the current best solution. If this measure leads to an improvement of the global best solution, then the D-HITS continues its execution until the global stopping criteria are satisfied, otherwise, the counter w continues to grow independently for each process.

When w reaches the second threshold *L2* (Algorithm 1 line 14 to 17), it is assumed that a complete re-localization is required. In this case, the algorithm ignores the diversification part and just perturbs all the current best solutions (Algorithm 1 line 22 to 26). The goal of this second measure is to explore the possibility that a better solution can exist in another region of the search space.

Algorithm 2 The TS framework

1: **Input**: $cost$: cost of the current solution; $S_{current}$: current solution; S_{best}: best solution found; $Tcost$: best cost found inside the TS; $Tsolution$: best solution found inside the TS; $TSiteration$: number of iteration executed by the TS;
2: **for** i = 0 to $TSiteration$ **do**
3:     **if** movement is tabu but meets all aspiration criteria or is not tabu and is a new $Tcost$ **then**
4:         Store the best permutation indexes that meet all conditions;
5:     **end if**
6:     Update tabu list;
7:     Update $S_{current}$;
8:     **if** the $cost$ is better than the $Tcost$ **then**
9:         Update the $Tsolution$ with $S_{current}$;
10:     **end if**
11:     Update the delta matrix of the move costs;
12: **end for**

Algorithm 3 Glover Diversification strategy

1: **Input**: $S_{current}$: current solution; S_{best}: the best solution found; $step$: the step of the permutation; n: the size of the problem
2: position = 0;
3: **for** i = step to 1 /* **decrement i by 1** */ **do**
4:     **for** j = (i-1) to n /* **increment j by $step$** */ **do**
5:         $S_{current}[position] = S_{best}[j]$;
6:         position++;
7:     **end for**
8: **end for**
9: **if** $step$ == n - 1 **then**
10:     reinitialization of $step$
11: **else**
12:     $step$++;
13: **end if**

3.2 Distance Cooperation Hybrid Iterative Tabu Search

In this section, we propose two versions: The DIStance COoperation Hybrid Iterative Tabu Search with the Glover diversification (DISCO-HITS-GD) and with the UX diversification (DISCO-HITS-UX). Different HITS are executed in a distributed environment from different starting solutions. For these two variants, there is an exchange of information between a set of processes in parallel following a ring topology.

The classical exchange of information consists in sending the best solution of the current process to the neighbor process. It allows the neighbor process to improve the search if the best solution received is better than its own best solution.

In our work, we introduce a new mechanism to exchange information for the QAP. The process sends its current solution and receives the current solution of the neighbor process. The idea is to compute the similarity between these two solutions at each position to define the distance (Algorithm 4 line 11-15). According to this distance, each machine takes a decision and follows a specific series of instructions to continue the search (diversification, perturbation or re-localization). Each process executes one HITS and the evolution of each process depends on the search history of its neighbor. The aim is to explore intelligently different regions of the space.
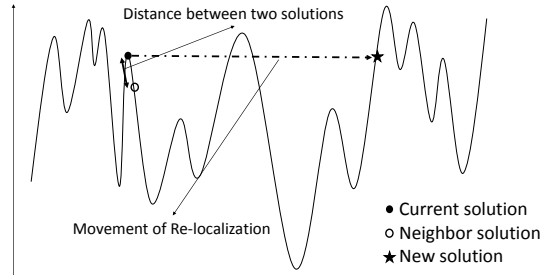
Fig. 1. Movement of re-localization using the distance cooperation

---

Algorithm 4 Distance Cooperation Between Hybrid Iterative Tabu Search

---

1: **Input**: *perturb*: % perturbation; n: size of solution; *cost*: cost of the current solution; *Fcost*: best cost found; $S_{current}$: current solution; S_{best}: best solution found; S_{EX}: solution exchanged;
2: Initialization of the solution for the current process;
3: **repeat**
4:     TS algorithm /* see Algorithm 2 */
5:     **if** cost < Fcost **then**
6:         $Fcost = cost$;
7:         Update the S_{best} with $S_{current}$;
8:     **end if**
9:     level = 0; counter = 0;
10:     Exchange $S_{current}$ between processes (ring topology);
11:     **for** i = 0 to n /* Compute distances */ **do**
12:         **if** $S_{current}[i] == S_{EX}[i]$ **then**
13:             counter ++;
14:         **end if**
15:     **end for**
16:     **if** counter < $\frac{n}{4}$ **then**
17:         level = 0; /* Big distance between the two processes */
18:     **else**
19:         **if** counter < $\frac{3 \times n}{4}$ **then**
20:             level = 1; /* Processes are relatively close */
21:         **else**
22:             level = 2; /* Processes are very close */
23:         **end if**
24:     **end if**
25:     **if** level == 0 **then**
26:         Update $S_{current}$ with the **diversification** of S_{best} (Algorithm 5 or Algorithm 3);
27:     **else**
28:         **if** level == 1 **then**
29:             **Perturbation** of $S_{current}$ with the **perturb** parameter;
30:         **else**
31:             **Re-localization** of $S_{current}$;
32:         **end if**
33:     **end if**
34: **until** (Stop condition)

Figure 1 shows the re-localization movement using our concept of distance between neighbor solutions. If the two solutions (current and neighbor solutions) are very close in the search space, the variable *level* will take the value 2 (Algorithm 4 line 22). This value indicates to the algorithm that the current solution needs to execute a re-localization to discover a new region in the search space (Algorithm 4 line 31).

The algorithm 4 is duplicated for all the processes. It runs a succession of TS (algorithm 2) [16]. After each TS, the algorithm 4 saves the new best solution if there is improvement. The next step of our proposition is to send the current solution and to receive the same information from the neighbor process. This is the information exchange step. The algorithm computes the difference between the two solutions to determine the distance between them. According to the distance, the algorithm takes one decision, to execute a diversification (algorithm 3 for DISCO-HITS-GD or algorithm 5 for DISCO-HITS-UX), to perturb the solution (line 29 algorithm 4) or to make a re-localization of this solution (line 31 algorithm 4).

For the DISCO-HITS-GD, our approach applies the diversification proposed by [7] called the Glover Diversification (GD) in this work. The diversification procedure takes a solution (in our case, it takes the global best solution) and executes a set of permutations following a step value. The step value changes from a global iteration (algorithm 4 from line 3 to line 34) to another. This way, even if the global best solution stays the same for many global iterations of the HITS, the new starting solution generated by this diversification will be different. Algorithm 3 presents the diversification pseudo-code.

For the DISCO-HITS-UX, the diversification applied is the uniform crossover (UX) presented by algorithm 5. The diversification procedure takes a solution (in our case, it takes the global best solution) and executes the UX following the sequence given by the select vector. The select vector is perturbed before each application of the UX. This way, even if the global best solution stays the same for many global iterations of the HITS, the new starting solution generated by this diversification will be different.

The UX algorithm 5 is constituted with three separated loops. The first and the second loops can fuse with each other but the behavior of the UX will be altered. In this proposition, we design Algorithm 5 to be easily implemented in parallel at the solution level.

The following example is an application of the UX (Algorithm 5):

Initialization:

- $select$ = (0,1,1,0,1,0).
- $index$ = (0,0,0,0,0,0).
- $S_{current}$ = (1,2,0,3,5,4).
- S_{best} = (3,5,1,0,4,2).

First loop:

- $S_{current}$ = (1,-1,-1,3,-1,4).
- $index$ = (0,1,0,1,1,0).

Algorithm 5 Uniform Crossover UX:

1: *Input*: $S_{current}$: current solution; S_{Ex}: the best solution found exchanged between process; **n**: the size of the problem; **select**: sequence to define the crossover with a mixing ratio of 0.5; **index**: filter to give a feasible solution initialized to 0;
2: Perturbation of *select*;
3: **for** i = 0 to n /* First loop*/ **do**
4:     **if** *select[i]* == 0 **then**
5:         $S_{current}[i] = S_{current}[i]$;
6:         *index[$S_{current}[i]$]* = 1;
7:     **else**
8:         $S_{current}[i]$ = -1;
9:     **end if**
10: **end for**
11: **for** i = 0 to n /* Second loop*/ **do**
12:     **if** *select[i]* == 1 **and** *index[$S_{Ex}[i]$]* == 0 **then**
13:         $S_{current}[i] = S_{Ex}[i]$;
14:         *index[$S_{current}[i]$]* = 1;
15:     **end if**
16: **end for**
17: **for** i = 0 to n /* Third loop*/ **do**
18:     **if** $S_{current}[i]$ == -1 **then**
19:         **for** k = 0 to n **do**
20:             **if** *index[k]* == 0 **then**
21:                 $S_{current}[i]$ = k;
22:                 *index[k]* = 1;
23:                 break;
24:             **end if**
25:         **end for**
26:     **end if**
27: **end for**

Second loop:

- $S_{current}$ = (1,5,-1,3,-1,4).
- $index$ = (0,1,0,1,1,1).

Third loop:

- $S_{current}$ = (1,5,0,3,2,4).
- $index$ = (1,1,1,1,1,1).

4   Experimental Results

4.1   Platform and Tests

In our experimentation, the algorithm is written in C/C++ and runs on a cluster of 10 machines Intel Core processor i5-3330 CPU (3.00GHz) with 4 GB of RAM. The proposed algorithms are experimented on benchmark instances from the QAPLIB ($http://www.seas.upenn.edu/qaplib/inst.html$) [3]. The size of the instances varies between 20 and 150. All the results are expressed as a percentage deviation from the best known solutions (BKS) (eq 2). All the BKS can be found in the online benchmark library QAPLIB. Each instance is executed 10 times and the average results of these executions are given.

$$deviation = \frac{(solution - BKS) \times 100}{BKS} \qquad (2)$$

The QAPLIB archive comprises 134 instances that can be classified into four types:

- Real life instances (Type 1);
- Unstructured randomly generated instances based on a uniform distribution (Type 2);
- Randomly generated instances similar to real life instances (Type 3);
- Instances in which distances are based on the Manhattan distance on a grid (Type 4);

Only the type 2, 3 and 4 are considered in this work since type 1 is very easy to solve.

4.2   Parameters

Our approaches contained a set of parameters. These parameters are fixed after a set of experimentation to get the best compromise between intensification and diversification. Table 1 shows the parameters used for *D-HITS*, *DISCO-HITS-GD* and *DISCO-HITS-UX*. The TSiteration parameter is the number of iterations executed by each TS inside the ITS (Algorithm 2 from line 3 to 13). The global iteration parameter is between two successive TS (stop condition) and n is the size of the problem. The *rank* value is the number of the current machine going from 0 to 9 in our work.

Table 1. parameters

Parameters	Value
TSiteration	$1000 \times n$
global iteration	200
L1	20+rank
L2	40+rank
aspiration criteria	$n \times n \times 5$
percentage of perturbation	25%

4.3   Experimentation

The first experimentation (Table 2) is focused on the three variants of our work. The same number of objective function evaluations and the same machines are used (equivalent computing power). The time is expressed in minutes. The number within brackets is the number of times each algorithm gets the BKS among the 10 trials. The results are presented for type 2, 3 and 4 respectively.

Table 2 contains the results for the three variants proposed in this work. Our first comparison is between D-HITS and DISCO-HITS-GD. These two variants use exactly the same elements (TS, GD, re-localization and perturbation). The difference is our mechanism to exchange information applied for the DISCO-HITS-GD. This mechanism gives the DISCO-HITS-GD the capacity to explore

Table 2. Distributed and cooperative hybrid iterative tabu search variants

Instances(34)	BKS	D-HITS		DISCO-HITS-GD		DISCO-HITS-UX	
		deviation	times	deviation	times	deviation	times
tai20a	703482	**0.000(10)**	0.41	**0.000(10)**	0.40	**0.000(10)**	0.4
tai25a	1167256	**0.000(10)**	0.81	**0.000(10)**	0.78	**0.000(10)**	0.79
tai30a	1818146	**0.000(10)**	1.40	**0.000(10)**	1.36	**0.000(10)**	1.37
tai35a	2422002	**0.000(10)**	2.18	**0.000(10)**	2.16	**0.000(10)**	2.15
tai40a	3139370	**0.007(9)**	3.25	0.030(6)	3.18	**0.007(9)**	3.22
tai50a	4938796	0.058(7)	6.34	0.062(8)	6.19	**0.048(8)**	6.29
tai60a	7205962	0.369(0)	11.10	0.303(0)	10.71	**0.272(0)**	10.77
tai80a	13515450	0.654(0)	26.58	0.573(0)	25.54	**0.561(0)**	25.62
tai100a	21052466	0.582(0)	54.30	0.552(0)	52.07	**0.359(0)**	52.36
tai20b	122455319	**0.000(10)**	0.31	**0.000(10)**	0.18	**0.000(10)**	0.18
tai25b	344355646	**0.000(10)**	0.75	**0.000(10)**	0.7	**0.000(10)**	0.69
tai30b	637117113	**0.000(10)**	1.36	**0.000(10)**	1.34	**0.000(10)**	1.36
tai35b	283315445	**0.000(10)**	2.14	**0.000(10)**	2.13	**0.000(10)**	2.11
tai40b	637250948	**0.000(10)**	3.18	**0.000(10)**	3.17	**0.000(10)**	3.16
tai50b	458821517	**0.000(10)**	6.19	**0.000(10)**	6.12	**0.000(10)**	6.09
tai60b	608215054	**0.000(10)**	10.64	**0.000(10)**	10.67	**0.000(10)**	10.6
tai80b	818415043	**0.000(10)**	25.36	**0.000(10)**	25.60	**0.000(10)**	25.35
tai100b	1185996137	0.000(6)	52.83	0.000(9)	51.26	**0.000(10)**	53.08
tai150b	498896643	0.076(0)	192.48	**0.015(0)**	214.26	0.027(3)	214.86
sko42	15812	**0.000(10)**	3.68	**0.000(10)**	3.67	**0.000(10)**	3.25
sko49	23386	**0.000(10)**	5.80	**0.000(10)**	5.76	**0.000(10)**	2.67
sko56	34458	**0.000(10)**	8.66	**0.000(10)**	8.63	**0.000(10)**	8.59
sko64	48498	**0.000(10)**	12.99	**0.000(10)**	12.94	**0.000(10)**	7
sko72	66256	**0.000(10)**	18.66	**0.000(10)**	18.53	**0.000(10)**	18.53
sko81	90998	0.001(9)	26.56	**0.000(10)**	26.48	**0.000(10)**	26.62
sko90	115534	**0.000(10)**	37.05	**0.000(10)**	37.17	**0.000(10)**	37.24
sko100a	152002	0.001(9)	51.37	**0.000(10)**	53.65	**0.000(10)**	52.41
sko100b	153890	**0.000(10)**	52.04	**0.000(10)**	51.50	**0.000(10)**	52.63
sko100c	147862	0.000(7)	52.76	**0.000(10)**	51.52	**0.000(10)**	52.04
sko100d	149576	0.001(6)	52.48	**0.000(10)**	54.09	0.000(8)	51.28
sko100e	149150	**0.000(10)**	53.27	0.001(9)	51.42	0.001(9)	51.30
sko100f	149036	**0.001(9)**	52.36	0.001(8)	51.48	**0.001(9)**	51.64
wil100	273038	0.002(0)	53.57	**0.000(9)**	51.60	0.000(8)	51.62
tho150	8133398	0.027(0)	217.34	0.010(0)	218.95	**0.004(0)**	199.36
Average		0.052(262)	32.36	0.046(279)	32.80	**0.038(284)**	31.96

efficiently the search space. It uses the evolution of each process to explore the largest possible search space. Through the 34 benchmark instances presented in this work, the DISCO-HITS-GD gets better results than D-HITS on 9 instances against 3 for D-HITS (tai40a, tai50a and sko100e). The variant with exchange has the capacity to get better results on large size instances like tai150b and tho150. It confirms that our cooperation method between processes is efficient to explore large search spaces. The average results for type 2 are equivalent for the two variants but DISCO-HITS-GD gets better results on type 3 and 4. With a global average of 0.046% the DISCO-HITS-GD is better than D-HITS with a global average of 0.052%.

The second comparison is between DISCO-HITS-GD and DISCO-HITS-UX. The difference between these two variants is only the diversification operator (Glover Diversification against Uniform Crossover). DISCO-HITS-UX gets the best average of 0.038% for the 34 instances thanks to its good results on type 2 and 4. The second average is for DISCO-HITS-GD with 0.046% but this variant gets better results on type 3. DISCO-HITS-UX reaches the BKS 284 times against 279 times for DISCO-HITS-GD which means that the DISCO-HITS-UX variant is more robust. Moreover, it gets better results on 6 instances against only one instances for DISCO-HITS-GD. This comparison reveals that the UX is a more efficient diversification on QAP than the GD.

4.4 Literature Comparison

Table 3 presents several comparisons with leading algorithms from the literature. The two best variants of our work (DISCO-HITS-GD and DISCO-HITS-UX) are compared with four algorithms from the literature.

- Population-based memetic algorithm (BMA) [2] (2015);
- The breakout local search (BLS) [1] (2013);
- Cooperative parallel tabu search (CPTS) [8] (2009);
- Population-based iterated local search (PILS) [15] (2006);

The algorithms of the literature use time as stopping criterion. For fair comparison the same stopping criterion of BLS, BMA and ITS (1 hour for $n <= 100$ and 4 hours for $n > 100$) is used. We can notice that our average of time is lower than BLS, BMA and ITS.

This comparison is focused on the quality of solutions. We use 34 well-known benchmark instances from the QAPLIB which are difficult to solve. The other instances of QAPLIB are easy to solve for our algorithms and the algorithms of the literature except for tai256c. Only the percentage deviation are considered in the comparison. The time is given for information purposes only.

Table 3 presents the experiments of type 2 in the first part. The best average is obtained by the BMA [2] algorithm with 0.1294%. It is followed by DISCO-HITS-UX with a very close average results of 0.1386% . However, our DISCO-HITS-UX outperforms BMA on three instances (tai40a, tai50a and tai100a) against two (tai60a and tai80a). It outperforms the average of the other three algorithms BLS [1], CPTS [8] and PILS [15]. DISCO-HITS-GD is ranked at the fourth place after BMA, DISCO-HITS-UX and BLS.

Table 3 presents the experiments of type 3 in the second part. The best average is obtained by our DISCO-HITS-GD algorithm with 0.0015%. It is followed by DISCO-HITS-UX with an average results of 0.0027% . The most important instance in this type is the tai150b. Our two propositions solve this instance efficiently thanks to the intelligent exploration of the search space.

Table 3 presents the experiments of type 4 in the last part. The best average is obtained by our DISCO-HITS-UX algorithm with 0.0004%. It is followed by DISCO-HITS-GD with an average results of 0.0008%. The most important instance in this type is tho150. Our two propositions are the two best algorithms to solve this instance compared to the other works. Our propositions show high efficiency to solve large size instances of 150.

Table 3. Comparison of DISCO-HITS-GD and DISCO-HITS-UX with BMA, BLS, CPTS and PILS (Type 2, 3 and 4)

Instances(9)	BKS	DISCO-HITS-GD deviation	times	DISCO-HITS-UX deviation	times	BMA deviation	times	BLS deviation	times	CPTS deviation	times	PILS deviation	times
tai20a	703482	0.000(10)	0.40	0.000(10)	0.4	0.000(10)	-	0.000(10)	0	0.000(10)	0.1	0.000(10)	0
tai25a	1167256	0.000(10)	0.78	0.000(10)	0.79	0.000(10)	-	0.000(10)	0	0.000(10)	0.3	0.000(10)	0.2
tai30a	1818146	0.000(10)	1.36	0.000(10)	1.37	0.000(10)	-	0.000(10)	0	0.000(10)	1.6	0.000(10)	0.6
tai35a	2422002	0.000(10)	2.16	0.000(10)	2.15	0.000(10)	-	0.000(10)	0.2	0.000(10)	2.3	0.000(10)	2.3
tai40a	3139370	0.030(6)	3.18	0.007(9)	3.22	0.059(2)	8.1	0.022(7)	38.9	0.148(1)	3.5	0.280(0)	12.0
tai50a	4938796	0.062(8)	6.19	0.048(8)	6.29	0.131(2)	42.0	0.157(2)	45.1	0.440(0)	10.3	0.663(0)	11.2
tai60a	7205962	0.303(0)	10.71	0.272(0)	10.77	0.144(2)	67.5	0.251(1)	47.9	0.476(0)	26.4	0.820(0)	7.4
tai80a	13515450	0.573(0)	25.54	0.561(0)	25.62	0.426(0)	65.8	0.517(0)	47.3	0.691(0)	94.8	0.927(0)	12.7
tai100a	21052466	0.552(0)	52.07	0.359(0)	52.36	0.405(0)	44.1	0.430(0)	39.0	0.589(0)	261.2	1.027(0)	9.8
Average type 2		0.1863(54)	11.38	0.1386(57)	11.44	0.1294(46)	-	0.1530(50)	24.27	0.2604(41)	44.50	0.4130(40)	6.24

Instances(10)	BKS	DISCO-HITS-GD deviation	times	DISCO-HITS-UX deviation	times	BMA deviation	times	BLS deviation	times	CPTS deviation	times	PILS deviation	times
tai20b	122455319	0.000(10)	0.18	0.000(10)	0.18	0.000(10)	-	0.000(10)	0	0.000(10)	0.1	0.000(10)	0
tai25b	344355646	0.000(10)	0.7	0.000(10)	0.69	0.000(10)	-	0.000(10)	0	0.000(10)	0.4	0.000(10)	0
tai30b	637117113	0.000(10)	1.34	0.000(10)	1.36	0.000(10)	-	0.000(10)	0	0.000(10)	1.2	0.000(10)	0
tai35b	283315445	0.000(10)	2.13	0.000(10)	2.11	0.000(10)	-	0.000(10)	0	0.000(10)	2.4	0.000(10)	0
tai40b	637250948	0.000(10)	3.17	0.000(10)	3.16	0.000(10)	-	0.000(10)	0	0.000(10)	4.5	0.000(10)	0
tai50b	458821517	0.000(10)	6.12	0.000(10)	6.09	0.000(10)	1.2	0.000(10)	2.8	0.000(10)	13.8	0.000(10)	0.1
tai60b	608215054	0.000(10)	10.67	0.000(10)	10.6	0.000(10)	5.2	0.000(10)	5.6	0.000(10)	30.4	0.000(10)	0.2
tai80b	818415043	0.000(10)	25.6	0.000(10)	25.35	0.000(10)	31.3	0.000(10)	11.4	0.000(10)	110.9	0.000(10)	1.3
tai100b	1185996137	0.000(9)	51.26	0.000(10)	53.08	0.000(10)	13.6	0.000(10)	16.0	0.001(8)	241.0	0.000(10)	2.3
tai150b	4988966643	0.015(0)	214.26	0.027(3)	214.86	0.060(1)	78.1	0.075(1)	243.6	0.076(0)	7377.8	0.095(0)	36.7
Average type 3		0.0015(89)	31.54	0.0027(93)	31.75	0.006(91)	-	0.0075(88)	27.94	0.0077(88)	778.25	0.0095(90)	4.06

Instances(15)	BKS	DISCO-HITS-GD deviation	times	DISCO-HITS-UX deviation	times	BMA deviation	times	BLS deviation	times	CPTS deviation	times	PILS deviation	times
sko42	15812	0.000(10)	3.67	0.000(10)	3.25	0.000(10)	-	0.000(10)	1.7	0.000(10)	5.3	0.000(10)	2.8
sko49	23386	0.000(10)	5.76	0.000(10)	2.67	0.000(10)	-	0.000(10)	0.5	0.000(10)	11.4	0.000(10)	0.8
sko56	34458	0.000(10)	8.63	0.000(10)	8.59	0.000(10)	-	0.000(10)	1.1	0.000(10)	21	0.000(10)	0.5
sko64	48498	0.000(10)	12.94	0.000(10)	7	0.000(10)	-	0.000(10)	1.3	0.000(10)	42.9	0.000(10)	0.2
sko72	66256	0.000(10)	18.53	0.000(10)	18.53	0.000(10)	3.5	0.000(10)	4.1	0.000(10)	69.6	0.001(8)	6.7
sko81	90998	0.000(10)	26.48	0.000(10)	26.62	0.000(10)	4.3	0.000(10)	13.9	0.000(10)	121.4	0.007(5)	9.6
sko90	115534	0.000(10)	37.17	0.000(10)	37.24	0.000(10)	15.3	0.000(10)	16.6	0.000(10)	193.7	0.006(4)	10.6
sko100a	152002	0.000(10)	53.65	0.000(10)	52.41	0.000(10)	22.3	0.001(9)	20.8	0.000(10)	304.8	0.012(3)	7.9
sko100b	153890	0.000(10)	51.5	0.000(10)	52.63	0.000(10)	6.5	0.000(10)	10.8	0.000(10)	309.6	0.007(5)	7.3
sko100c	147862	0.000(10)	51.52	0.000(10)	52.04	0.000(10)	12.0	0.000(10)	15.5	0.000(10)	316.1	0.002(6)	11.5
sko100d	149576	0.000(10)	54.09	0.000(8)	51.28	0.006(9)	20.9	0.001(5)	38.9	0.000(10)	309.8.6	0.021(0)	11.8
sko100e	149150	0.001(9)	51.42	0.000(9)	51.30	0.000(10)	11.9	0.000(10)	42.5	0.000(10)	309.1	0.001(7)	6.8
sko100f	149036	0.001(8)	51.48	0.001(9)	51.64	0.000(10)	23.0	0.000(10)	17.3	0.003(4)	310.3	0.037(0)	11.7
wil100	273038	0.000(9)	51.60	0.000(8)	51.62	0.000(10)	14.5	0.000(10)	18.9	0.000(10)	316.6	0.004(1)	6.3
tho150	8133398	0.010(0)	218.95	0.004(0)	199.36	0.008(3)	416.4	0.023(1)	268.8	0.013(0)	1991.7	0.068(0)	36.2
Average type 4		0.0008(136)	46.49	0.0004(134)	44.41	0.0009(142)	-	0.0016(135)	31.51	0.0011(134)	308.89	0.0111(79)	8.71

5   Conclusion and Perspectives

In this work, we have presented and validated three variants of a distributed HITS to solve the QAP. Each variant proposes different HITS based on an iterative tabu search and implemented in parallel. The first variant is the D-HITS which executes parallel HITS from different starting solutions and with different parameters. The second variant is the DISCO-HITS-GD algorithm which is a parallel distributed HITS using the distance between processes to select the diversification technique to apply. The last variant is the DISCO-HITS-UX which uses the same concept than the DISCO-HITS-GD variant but using the UX instead of the Glover diversification. All these variants enable a balance between intensification and diversification.

The three approaches demonstrate high-quality results on the set of well-known benchmark instances from QAPLIB. We evaluated our approaches on 34 benchmark instances from the QAPLIB. Indeed, our approaches are very competitive and outperform in many instances the current best approaches solving QAP. The comparison between the variants shows the potential of the exchange of information in the distributed design and the power of this exchange to solve big size instances.

In summary, three main contributions are proposed in this work. The first one is the creation of three new distributed variants to solve the QAP. Each variant is hybridized with a set of different adaptive diversification mechanisms to improve the results in a distributed environment. The second contribution is the creation of a new mechanism which allows the algorithm to compute distance between solutions in order to explore the largest search space possible through the exchange of information. The final contribution is the experimental comparison of two well known diversification techniques (GD and UX).

As a future work, there are several possible ways to extend this work. One possibility is to experiment other parameters to get better results. There is also some instances which are rarely used in literature and they are difficult to solve for metaheuristics, like the instances proposed by [5]. Another possibility is to explore the two other parallel designs (the iteration level and the solution level). For the iteration level we can reduce the execution time with a parallel evaluation and generation of neighborhoods inside the delta matrix. For the solution level we can reduce the execution time with a parallel UX or GD to generate the new starting solution. The best platform to perform these two levels is probably the GPU platform thanks to its single instruction multiple data architecture. We can also use other topologies to exchange information instead of the ring topology. Finally, this approach can be experimented for other combinatorial problems to analyze the behavior of the proposed approach with other kinds of problems.

References

1. Benlic, U., Hao, J.K.: Breakout local search for the quadratic assignement problem. Applied Mathematics and Computation 219(9), 4800–4815 (2013)

2. Benlic, U., Hao, J.K.: Memetic search for the quadratic assignment problem. Expert Systems with Applications 42, 584–595 (2015)
3. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB - A quadratic assignment problem library. journal of global optimization 10(4), 391–403 (1997)
4. Drezner, Z.: Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. Computers and Operations Research 35(3), 717–736 (2008)
5. Drezner, Z., Hahn, P.M., Taillard, E.: Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. Annals of Operations research 139(1), 65–94 (2005)
6. Duman, E., Or, I.: The quadratic assignement problem in the context of the printed circuit board assembly process. Computers and Operations research 34(1), 163–179 (2007)
7. Glover, F.: A template for scatter search and path relinking. Lecture Notes in Computer Science 1363, 1–51 (1998)
8. James, T., Rego, C., Glover, F.: A cooperative parallel tabu search algorithm for the quadratic assignment problem. European Journal of Operational Research 195(3), 810–826 (2009)
9. James, T., Rego, C., Glover, F.: Multistart tabu search and diversification strategies for the quadratic assignment problem. IEEE Transactions on systems Man And Cybernetics part a systems and humans 39(3), 579–596 (2009)
10. Koopmans, T., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica 25(1), 53–76 (1957)
11. Loiola, E.M., de Abreu, N.M.M., Netto, P.O.B., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. European Journal of Operational Research 176(2), 657–690 (2007)
12. Merz, P., Freisleben, B.: Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Transactions on Evolutionary Computation 4(4), 337–352 (2000)
13. Misevicius, A.: An improved hybrid genetic algorithm: New results for the quadratic assignment problem. Knowledge Based Systems 17(2-4), 65–73 (2004)
14. Misevicius, A., Kilda, B.: Iterated tabu search: An improvement to standard tabu search. Information Technology and Control 35(3), 187–197 (2006)
15. Stützle, T.: Iterated local search for the quadratic assignment problem. European Journal of Operational Research 174(3), 1519–1539 (2006)
16. Taillard, E.: Robust taboo search for the quadratic assignement problem. Parallel computing 17(4-5), 443–455 (1991)
17. Talbi, E.G.: Metaheuristics: from Design to Implementation. John wiley and sons Inc, University of Lille - CNRS - INRIA (2009)
18. Ulutas, B.H., Konak, S.K.: An artificial immune system based algorithm to solve unequal area facility layout problem. Expert Systems with Applications 39(5), 5384–5395 (2012)
19. Wu, Q., Hao, J.K.: Solving the winner determination problem via a weighted maximum clique heuristic. Expert Systems with Applications 42(1), 355–365 (2015)
20. Zhang, Q., Sun, J., Tsang, E.: An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Transactions on Evolutionary Computation 9(2), 192–200 (2005)

H-ACO: A Heterogeneous Ant Colony Optimisation approach with Application to the Travelling Salesman Problem

A.T.I Fayeez[1], E.Keedwell[1], M.Collett[2]

[1]College of Engineering, Mathematics and Physical Sciences, Harrison Building, University of Exeter
[2]Animal Behaviour Laboratory, College of Life and Environmental Sciences, Washington Singer Building, University of Exeter
{ab835, E.C.Keedwell, M.Collett}@exeter.ac.uk

Abstract. Ant Colony Optimization (ACO) is a field of study that mimics the behaviour of ants to solve computationally hard problems. The majority of research in ACO focuses on homogeneous artificial ants although animal behaviour research suggests that heterogeneity of behaviour improves the overall efficiency of ant colonies. Therefore, this paper introduces and analyses the effects of heterogeneity of behavioural traits in ACO to solve hard optimisation problems. The developed approach implements different behaviour by introducing unique biases towards the pheromone trail and local heuristic (the next hop distance) for each ant. The well-known Ant System (AS) and Max-Min Ant System (MMAS) are used as the base algorithms to implement heterogeneity and experiments show that this method improves the performance when tested using several Travelling Salesman Problem (TSP) instances particularly for larger instances. The diversity preservation introduced by this algorithm helps balance exploration-exploitation, increases robustness with respect to parameter settings and reduces the number of algorithm parameters that need to be set.

Keywords. Heterogeneity, Heterogeneous, ACO, TSP

1   Introduction

Natural systems provide inspiration for tackling complex tasks by being able to self-organize without the need of a central controller. These behaviours are due to evolution, development and learning thus providing a platform for nature-inspired algorithms to achieve good solutions to complex problems. Another example of such inspiration is the swarm behaviour in which natural organisms behave when they are in groups. As an example, ant collectives are capable of achieving complex tasks such as nest construction and food foraging that would not be possible for individual ants. A colony of ants is capable of finding the shortest path from nest to food in a sophisticated way. Inspiration can be taken from these behaviours and used to tackle optimization problems in the real world where their behaviours have been implemented in ant colony optimization (ACO) research [1]. The main contribution of ants in this research is the foraging behaviour where ants lay pheromone on the ground to mark their path from the nest to food source. This is to guide the ant back to the nest and also guide its colony members towards a food source during the recruitment process. ACO implements similar concepts when optimising combinatorial optimization problems such as the Travelling Salesmen Problem (TSP) [2]. TSP is one of the most widely studied by researchers working on combinatorial optimization problems, is an NP-hard problem and is an interpretation of a salesman requiring to visit n cities via the shortest complete tour.

A significant issue with ACO, as in most other metaheuristic approaches is to find a proper balance between exploitation and exploration. Exploitation is a process of concentration of the algorithm in the areas of the search space where good quality solutions have been previously been found while exploration of the search space denotes action by the search agent in moving towards unexplored areas. Several studies reviewed in [1] show that a proper balance between exploitation and exploration is required in order for a metaheuristic algorithm to achieve good to optimal results. In this paper, we investigate the influence of each ant having different behavioural characteristics or 'traits' in contrast to standard ACO where all ants have the same behavioural traits. In the proposed heterogeneous approach, each ant has individual pheromone (α) and heuristics coefficients (β) where both α and β are parameters that control the relative importance of the pheromone trail and local heuristics used in

transition probability [3]. It is known that too much emphasis on pheromone trail or local heuristics may hinder the performance of the algorithm through over exploration or exploitation. Hence the proposed method can overcome the exploration-exploitation problem thus improving the performance of ACO. The heterogeneous approach implemented in this study stems from the actual behaviour of social insects which are heterogeneous in nature, displaying different traits and in some circumstances behavioural roles within a colony [4] [5]. The paper is structured as follows. In Section 2, ACO is discussed briefly while Section 3 discuss the previous work on heterogeneous approach in ACO. Section 4 describes the methodology of this study and Section 5 explains the experimental setup. Section 6 present the results of the study and the paper is concluded in Section 7 with discussion and conclusion.

2   Ant Colony Optimization

ACO is an optimisation algorithm that takes inspiration from the foraging behaviours of real ants. Some of the most popular conventional ACO are Ant System (AS) [3] and Max Min Ant System (MMAS) [6] that use metaheuristics approach inspired by ant colonies behaviour to find good solutions for an optimization problem. AS was the first ACO algorithm to be developed and acts as proof of ACO concept while MMAS is one of the best performing ACO algorithms in the literature. Both algorithms work through the deposition of pheromone by virtual ants who traverse the set of cities creating a tour, where the level of pheromone deposited on that tour is a function of tour optimality and the pheromone on all paths is evaporated uniformly. Subsequent ants probabilistically choose paths with a preference for those paths with greater pheromone with the goal of converging towards a near-optimal solution. The algorithms differ in that ant system allows all ants to contribute to the deposition of the pheromone, whereas the max-min ant system allows only the best performing ant within a population to contribute and has a lower-bound on pheromone levels.   Both AS and MMAS have been applied to numerous TSP instances, a combinatorial optimization problem that has attracted extensive research [7]. This paper implements the heterogeneous approach on these two ACO variants. Due to limited space, AS and MMAS will not be discussed in detail here and can be referred to [3] and [6] respectively.

3   Heterogeneous ACO

Heterogeneity in swarm intelligence was firstly described in Particle Swarm Optimization (PSO) by Engelbrecht in [8] who proposed that introduction of heterogeneity in a search algorithm can improve the performance. This concept can also be adopted in ACO where artificial ants with different traits of behaviour can help to improve the performance of the ACO algorithm. This mimics the actual behaviour of real ants in a colony in terms of diversity and division of labour [9]. Heterogeneity in ACO can be grouped into *individual* and *colony* level. Artificial ants with different 'behaviours' among them is said to be heterogeneous at the individual level while colonies of ants that differ in behaviour between the colonies is said to be the latter. Heterogeneous individual ants in ACO were first introduced by [10] where the authors used modified ACO with heterogeneity for path planning in mobile robots in order to find obstacle-free path in a certain environment. The author deployed ants with different sight, speed and function behaviours and found that the performance of Heterogeneous ACO (HACO) is better in terms of path planning when compared to conventional ACO. Chira et al. discussed the different sensitivity of the artificial ants to the pheromone trail level in [11]. Ants with higher pheromone sensitivity strongly follow the pheromone trail while ants with lower pheromone sensitivity are more inclined towards random search. In the meantime, Hara et al. [12] proposed the use of classic and exploratory ants where each ant constructs a partial solution which is then combined to produce one single solution. Yoshikawa et al [13] introduces a cranky ant approach to tackle the exploration-exploitation problem which appears to prevent the algorithm from being stuck in local optima. The cranky ants will explore paths with low pheromone level which is the opposite of the behaviour of standard artificial ant. Meanwhile, Zhang et al. [14] proposed colony level heterogeneity where ant colonies have different pheromone updating rules in order to balance exploration and exploitation in the search

process. The authors proposed two colonies where each exhibits behaviour of Elitist Ant System (EAS) and Ant Colony System (ACS) characteristics respectively. They discussed that the algorithm overcomes stagnation and the early suboptimal path convergence problem. Melo et al [15] proposed a multi-caste ant colony in Ant Colony System (ACS) where ants with different preference towards $q0$, parameter that controls the degree of exploration or exploitation in ACS. Many more approaches implement heterogeneity at the colony level, but as this paper study and implementation at individual level, thus colony level heterogeneity will not be discussed in detail here. Each of these algorithms approach the principle of heterogeneity from a different standpoint, either using different ant roles or through the implementation of problem specific heterogeneity. The approach taken in this paper is one of biological plausibility for ants with similar roles, but differing behavioural traits, which would normally be expressed through genetic differences, but here are drawn from a distribution.

4   Methodology

The main motivation of this research work is to study the ant colonies as heterogeneous, multi-behaviours agents that can further improve the performance of the algorithm. The hypothesis is that with heterogeneity, a mixture of ants that are more inclined towards exploration of the search space with other ants that exploit the best path found creates a balance in the search process. This is due to the behaviours of the ants of which are randomly initialized either to be more inclined towards exploration or exploitation. The algorithm proposed a simple heterogeneous method in this study by pre-assigning a random behavioural trait for each of the ants in the population size during initialization that will not change during the iterations, as would be the case with genetic variation in real ants. Each behaviour has a pair of continuous traits that can be related to pheromone trail intensity and visibility or the local heuristic information. The heterogeneous approach in both AS and MMAS platform and comparison were carried out and compared with the original versions of each algorithm.

Algorithm 1: Heterogeneous ACO for TSP

```
1. Begin;
2. Input: Distance Matrix of TSP;
3. Initialize: Max Iteration, m, n, Q, τ, ρ;
4. Initialize ants:
     For i = 1: m
     a=0; b=2;
     Alpha (i) = rand (1) * (b-a) + a;

     c=3; d=5;
     Beta (i) = rand (1) * (d-c) + c;
   End
5. Start Iteration:
     For it=1: Max Iteration
       For k=1: m
       Position each ant on starting node;
          While TourSize < n+1
             Tour Construction;
          End
       End
     End
6. Update solution;
7. Update Pheromone;
8. Pheromone Evaporation;
9. Check if termination criteria = true;
10. End
```

Algorithm 1 depicts the pseudocode of our proposed algorithm and the major difference between this and the base algorithm is that ants will have α and β values that are initialized randomly between a set of pre-determined values rather than identical parameters throughout the run. The range for α and β values were based on experiments

by Dorigo et al in [3] and additional extensive experiments have been conducted to determine the best range for α and β (discussed briefly in section 6.1).

4.1 Travelling Salesman Problem [2]

Travelling Salesman Problem (TSP) is a widely studied combinatorial optimization problem in computer science. The main objective of solving a TSP is to achieve the shortest tour visiting *n* cities and returning to the starting city when there are no more cities left to be visited. TSP is visualized in a graphical format where nodes act as the cities and edges as the link or path between the cities. The edges will have weighting determining the cost of following that edge. TSP has been a popular case study in ACO and other various optimization algorithms.

5 Experimental Setup

The experiments were conducted on an Intel Core i7 CPU-based computer running Windows 7 equipped with 4GB RAM. The base algorithms used are the Ant System (AS) and Max Min Ant System (MMAS) approach developed using the Matlab version R2015a. Each algorithm is tested using several TSP instances taken from TSPLIB [2]. Firstly, the developed AS and MMAS was compared with that of [3] and [6] to show a level of confidence that the developed algorithm is similar to the original version. All the parameters were set according to the authors' recommendations where for AS the parameters were set as follows: α=1, β=5, ρ=0.5 and m = n where m is the number of ants and n is the number of cities related to the TSP. Meanwhile the parameters for MMAS were set as follows: α = 1, β = 2, m = n, ρ = 0.98 and P_{best} is 0.05. The function evaluations for all the experiments were set as *k.n.10000* where k=1 for symmetrical TSPs used, n=number of cities of the TSP instance and 10 000 is the maximum number of iterations. Table 1 shows the comparison between the developed algorithms against its original versions where the results for the developed algorithms are the average of 15 trials. As can be seen, the best cost of the developed algorithm and that of the original developers' are very similar demonstrating that the base algorithm formulations are working appropriately.

Table 1: Developed AS, MMAS vs Original AS, MMAS. Results show the average of the best cost. (*Note: Average of 15 trials*)

TSP	Optimum (Integer Length)	Optimum (Real Length)	Developed A.S	AS	Developed MMAS	MMAS [6]
Oliver 30	420[2]	423.741 [3]	423.7406	423.74 [3]	N.A	N.A
Eil51	426 [3]	428.87 [2]	437.56	437.3 [6]	427.5	427.1
kroA 100	21285 [3]	21285.44 [2]	22451.9819	22471.4 [6]	21299.6	21291.6
D198	15780 [3]	15808.65 [2]	16692.24	16702.1 [6]	15960.2	15956.8

6 Heterogeneous ACO Results

6.1 Exploring the ranges of Alpha and Beta

An extensive experiment based on AS was conducted to find the best range of α and β for our heterogeneous approach where lower and upper bounds of α and β were based on the recommendation of [3]. Both α and β values are varied to create a heterogeneous approach as both α & β plays an important role in exploration and exploitation of the search space. Hence, varying both parameters will introduce more variance in the agents. In addition, Stützle et al [16] suggests that both α and β are good candidates for parameter adaptation in ACO. As can be seen, the recommended range for α is between 0.25 and 1.5 while β has a range of 1 to 5. Therefore, extensive experiments were conducted where the ants were set to have a uniform distribution of α between 0 and 1 and 0 to 2 while a uniformly distributed β was varied between 0 and

5, narrowed down to 4 to 5. The other parameters were set according to [3]: 10 000 iterations, m = n, ρ = 0.5, Q = 100, initial pheromone trail = m/L_{nn} where L_{nn} is the tour length of the tsp instance using nearest neighbour heuristic. 3 tsp instances were used to test the algorithm namely oliver30.tsp (integer length optimum = 420, real length optimum = 423.7406), eil51.tsp (integer length optimum = 426, real length optimum = 428. 8716) and eil101.tsp (integer length optimum = 629). Table 2 and Table 3 summarizes the outcome of our extensive experiment. The results are best tour length found in 15 trials.

Table 2: Results from experimentation where α is uniformly distributed between 0 to 1 and the β distribution varies. Algorithm tested on oliver30.tsp, eil51.tsp and eil101.tsp. Results represent average best cost out of 15 trials while values in bold represents the best average.

α	β	oliver30	eil51	eil101
0 -1	0 -5	427.0934	445.301	699.1238
0 -1	1 -5	425.3379	441.6734	685.7444
0 -1	2 -5	426.0892	439.5271	678.2238
0 -1	3 -5	**423.7406**	**436.2947**	661.9443
0- 1	4 - 5	**423.7406**	436.3278	**659.4744**

Table 3: Results from experimentation whereα is uniformly distributed between 0 and 2 and the β distribution varies. Algorithm tested on oliver30.tsp, eil51.tsp and eil101.tsp for 10 000 iterations with values representing average best cost out of 15 trials while values in bold represents the best average.

α	β	oliver30	eil51	eil101
0- 2	0 -5	427.2749	437.1203	688.2972
0 -2	1 -5	424.6639	442.3749	672.3319
0 -2	2 -5	423.9117	438.0173	665.7093
0 -2	3 -5	**423.7406**	**436.0904**	**645.5318**
0- 2	4- 5	**423.7406**	436.6167	651.2821

Both Table 2 and Table 3 above show that the best range is α: 0 to 2 and β: 3 to 5. The experiment did not include α values greater than 2 because it is proven can lead to stagnation behaviour [3]. Therefore, the following experiments related to heterogeneous AS hereafter will use this parameter range.

6.2   Comparison with Base Algorithms

Next, the Heterogeneous Ant System (HAS) was compared against AS developed by [3] based on several symmetrical tsp instances. The AS (and later MMAS [6]) systems have been subjected to extensive experiments to determine the optimal alpha and beta settings for these problems. The resulting comparisons are therefore made between the heterogeneous system and well-tuned examples of the base ACO algorithms. HAS has the same parameter settings as AS (mentioned in the previous section) expect that α is a uniform distribution between 0 and 2 while β is varied from 3 and 5. The function evaluations for all experiments remain the same as previous section. Table 4 summarizes the comparison of AS and HAS on eil51.tsp for 25 trials. It can be seen that HAS improves on the best cost found by AS where the average is 436 compared to that of AS which is 437.56. Both AS and HAS was not able to find the optimum but it is shown that HAS performs better than AS in terms of 1% deviation and 2% deviation of the optimum. A value is said to be 1% deviation of optimum when it is within the range of 1% to the optimum. In eil51.tsp case, 1% deviation is 1/100 x 426 (optimum value from TSPLIB [2]) =4.26+426 =430.26.

426 < X< 430.26 = 1% deviation of optimum

430.26 < X < 434.52 = 2% deviation of optimum

Table 4: Best, average and worst cost comparison between AS & HAS for eil51.tsp, 10 000 iterations over 25 trials.

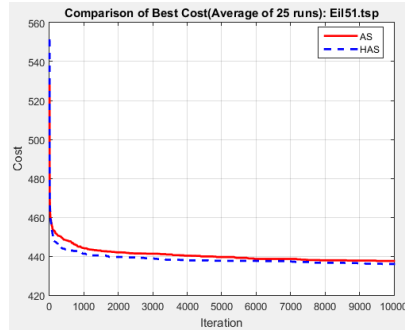| Method	Best	Average	Worst	# Optimum Found	1% dev of opt	2% dev of opt
AS	433	437.56	**441**	0	0	1
HAS	**428**	**436.00**	442	0	**1**	**5**

Figure 2: Comparison of average best cost for 25 independent trials of eil51.tsp with 10 000 iterations for each trial.
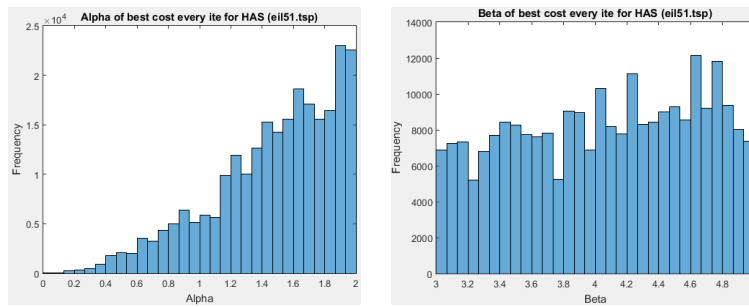

Figure 3 (a)

Figure 3 (b)

Figure 3: Frequency of Alpha & Beta values that managed to find the best cost in every iteration for HAS (eil51.tsp) (*Note: 25 trials x 10 000 iterations each trial = 250 000 iterations*)

Figure 2 shows that the Heterogeneous Ant System (HAS) has a better performance in terms of average best cost compared to AS over the duration of the optimisation. Figure 3 show the frequency of alpha and beta values of ants that found the best cost in every iteration. It can be seen the alpha values that mostly contribute are between 1.9 and 2, with a strong skew towards these values whereas the beta distribution is much more uniform with a small skew towards beta values of 4.6 and 4.75. This shows that heterogeneous approach introduces diversity in the algorithm and suggests the mechanism behind the improved performance over the algorithm with a single 'behavioural trait'.

Table 5: Best, average and worst cost comparison between AS & HAS for kroA100.tsp (optimum: 21282). Results in bold are the best in the table.

Method	Best	Average	Worst	# Optimum Found	5% dev of opt	6% dev of opt
AS	22384	22469.4	22666	0	0	5
HAS	**22215**	**22347.6**	**22487**	0	**22**	**25**

Table 5 shows the comparison between AS and HAS for 100-city tsp, kroA100.tsp. HAS managed to improve on the fitness solution compared to AS where average best cost for HAS is 22347.6 and that of AS is 22469.4. Although both AS and HAS did not manage to find the optimum for 100-city TSP problem, HAS managed to find a best cost that is within 5% of the optimum 22 times compared to none by AS. In addition, HAS found a best cost of 22215 compared to 22384 of AS out of 25 trials.
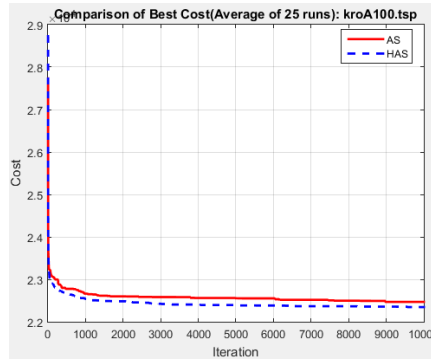

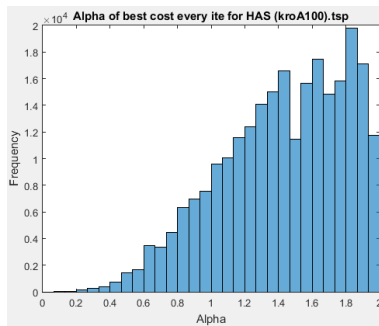Figure 4: Average best cost comparison for AS & HAS (kroA100.tsp) for 10 000 iterations over 25 trials.
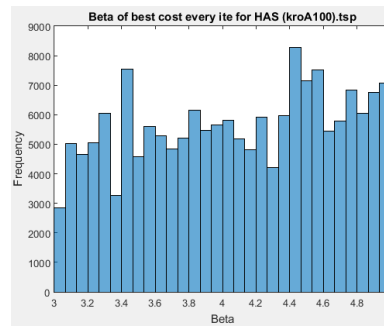

| Figure 5 (a) | Figure 5 (b) |

Figure 5: Frequency of Alpha & Beta values that managed to find the best fitness solution in every iteration for HAS (kroA100.tsp) (*Note: 25 trials x 10 000 iterations each trial = 250 000 iterations*)

Figure 4 shows the improved performance of HAS over AS in terms of average best cost while Figure 5 (a) and 5 (b) show the frequency of alpha and beta values of ants that managed to find best cost in all the 10 000 iterations for 25 trials. The distributions are similar to those of the previous experiments with alpha values peaking at 1.85 while beta has a peak at 4.45.

Table 6: Best, average and worst cost comparison between AS & HAS for d198.tsp (Optimum: 15780). Results in bold are the best in the table.

Method	Best	Average	Worst	# Optimum Found	3% dev of opt	4% dev of opt
AS	16356	16572.48	16724	0	0	3
HAS	**16186**	**16359.04**	**16700**	0	**6**	**19**

Table 6 summarizes the outcome of 25 trials of d198.tsp using both AS and HAS. AS found a best cost of 16356 throughout the 25 trials while HAS found a best cost of 16186. In additionHAS has a lower average compared to AS. Although the optimum is not found by any of the algorithms, HAS managed to find fitness solutions that are 3%

within the optimum range 6 times and 19 times within 4% of the optimum compared to 0 and 3 times respectively by AS.
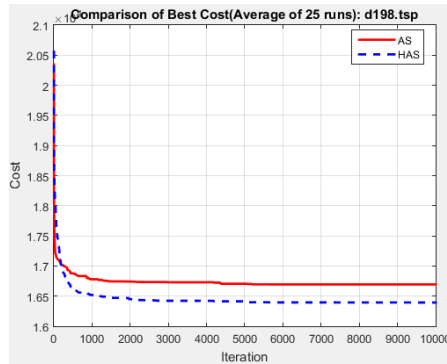

Figure 6: Average of best cost comparison between AS & HAS (d198.tsp) over 25 trials, 10 000 iterations in each trial.
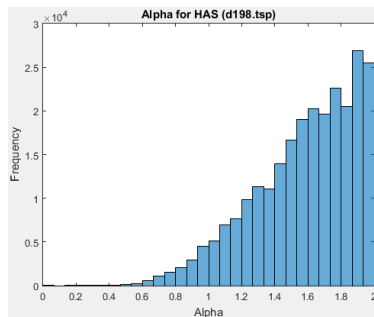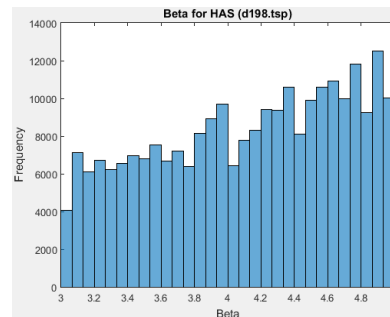

Figure 7 (a)   Figure 7 (b)

Figure 7: Frequency of Alpha & Beta values that found the best fitness solution in every iteration for HAS (d198.tsp) (*Note: 25 trials x 10 000 iterations each trial = 250 000 iterations*)

Figure 6 shows the major improvement in terms of average best cost performance of HAS over AS while it can be seen clearly in Figure 7 that even though both α=1 and β=5 as per suggested in [3] are covered in the pre-determined range for heterogeneous approach, both α and β values that managed to find best cost in every iteration increases rapidly from 0.5 to 2 and a steady increase from 3 to 5 respectively with α values having a peak at 1.9 while β values have a peak of 4.9.

The encouraging results of the heterogeneous approach on Ant System leads to the approach to be implemented on to Max Min Ant System (MMAS) known as Heterogeneous MMAS (HMMAS). All the parameters were set according to [6](discussed in experimental setup) except that of α which was set to 0 to 2 while β is varied between 1 and 3. The same sets of TSP instances were used to compare HMMAS against MMAS. Table 7 summarizes the comparison for eil51.tsp which has an optimum of 426. Although overall average of HMMAS is slightly higher compared to that of MMAS, HMMAS performed much better in relation to the number of times optimum found where both MMAS and HMMAS managed to find the optimum 4 times and 10 times respectively out of 25 trials. Figure 11 shows the comparison of the average best cost of both MMAS and HMMAS for the 51-city TSP problem.

Table 7: Best, average and worst cost comparison between MMAS & HMMAS for eil51.tsp (optimum: 426). Results in bold represents the best value in the table.

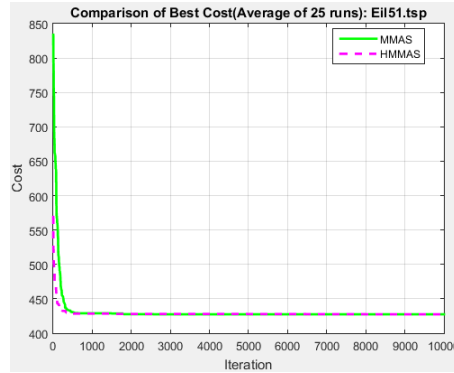| Method	Best	Average	Worst	# Optimum Found	1% dev of opt	2% dev of opt
MMAS	**426**	**427.4**	**430**	4	**25**	**25**
HMMAS	**426**	427.6	431	**10**	23	**25**

Figure 8: Comparison of average best cost for MMAS & HMMAS (eil51.tsp) over 25 trials, each trial = 10 000 iterations.
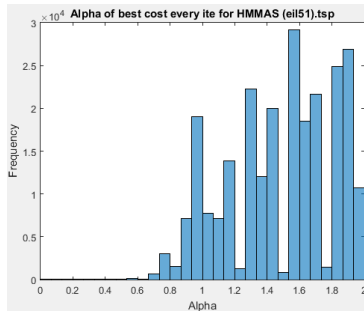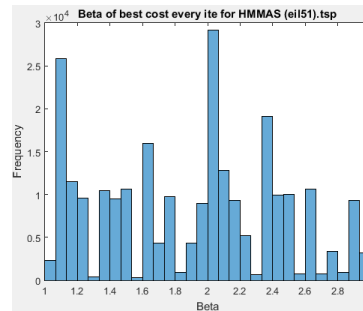

Figure 9 (a)

Figure 9 (b)

Figure 9: Frequency of Alpha & Beta values that managed to find the best fitness solution in every iteration for HMMAS (eil51.tsp) (*Note: 25 trials x 10 000 iterations each trial = 250 000 iterations*)

Figure 8 shows that both MMAS and HMMAS have a similar average best cost. Figure 9 illustrates that both the alpha and beta values of ants that managed to find the best cost in every iteration for HMMAS with alpha has a peak value of 1.55 while beta has a peak of 2.05. The overall distributions are somewhat similar to those from HAS. The diversity in the algorithm helps too as it shows that various alpha and beta values contribute towards finding the best cost.

Table 8: Best, average and worst cost comparison between MMAS & HMMAS for kroA100.tsp (optimum: 21828). Results in bold represent the best value in the table.

Method	Best	Average	Worst	# Optimum Found	1% dev of opt	2% dev of opt
MMAS	**21282**	**21299.6**	21390	4	**25**	**25**
HMMAS	**21282**	21316.6	**21379**	**11**	21	22

Table 8 shows the outcome of experiment on kroA100.tsp where both MMAS and HMMAS managed to find the optimum of 21282 while MMAS has an average of

21294.4 and HMMAS has an average of 21316.6. This can be due to several trials producing fitness solutions out of the 1% and 2% range of optimum thus causing the HMMAS to have a higher average. Although MMAS have a lower average best and lower worst cost, HMMAS still outperforms MMAS by finding the optimum 11 times compared to 4 times for MMAS. Figure 10 shows the comparison of the average best cost between MMAS and HMMAS.
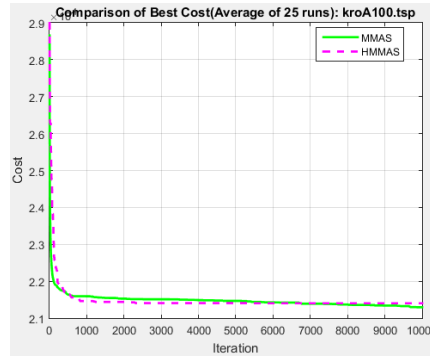

Figure 10: Comparison of MMAS & HMMAS of average best cost (kroA100.tsp) over 25 trials, each trial = 10 000 iterations.
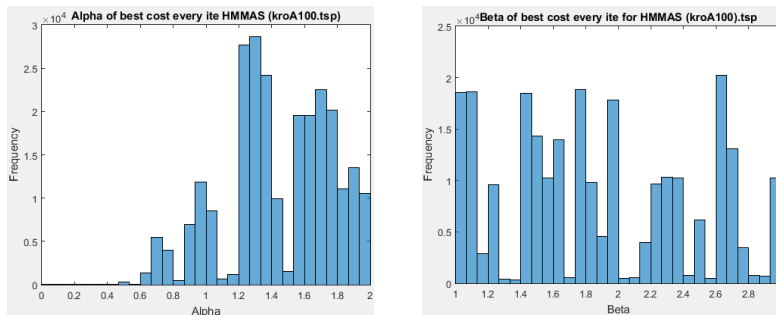

Figure 11 (a)   Figure 11 (b)

Figure 11: Frequency of Alpha & Beta values that managed to find the best fitness solution in every iteration for HMMAS (kroA100.tsp) (*Note: 25 trials x 10 000 iterations each trial = 250 000 iterations*)

Figure 10 shows that the average best cost of MMAS is slightly better compared to HMMAS for 100-city problem. Both Figure 8 and Figure 10 suggest that MMAS performs considerably well for eil51.tsp and kroA100.tsp due to the small problem size. Figure 11 illustrates the alpha and beta values related to the best cost in every iteration over 25 trials. Alpha values peak around 1.3 and beta has a peak of 2.65 respectively.

Table 9: Best, average and worst cost comparison between MMAS & HMMAS for d198.tsp (optimum: 15780). Results in bold represents the best value in the table.

Method	Best	Average	Worst	# Optimum Found	1% dev of opt	2% dev of opt
MMAS	15846	15961.12	16137	0	10	22
HMMAS	**15795**	**15871.68**	**16006**	0	**21**	**25**

Table 9 summarizes the comparison made between MMAS and HMMAS for 198-city TSP. HMMAS has a best cost of 15795 compared to 15846 of MMAS and HMMAS also has a lower average and lower worst cost compared to MMAS. Meanwhile,

HMMAS also managed to find fitness solutions 21 times within the 1% range of optimum compared to that of MMAS of 10 times.
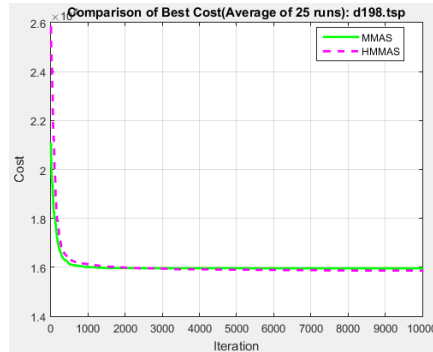
Figure 12: Comparison of MMAS & HMMAS of average best cost (d198.tsp) over 25 trials, each trial = 10 000 iterations.
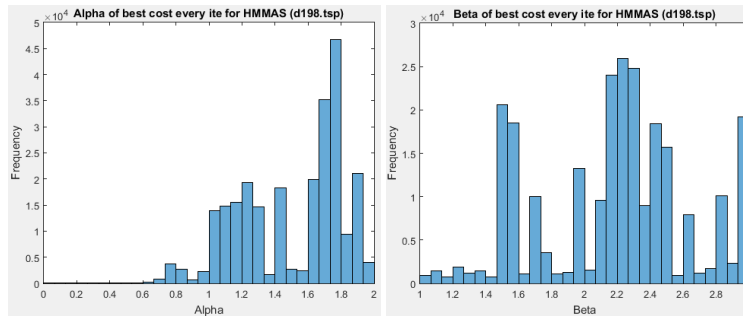
Figure 13 (a)   Figure 13 (b)

Figure 13: Alpha & Beta values that managed to find the best fitness solution in every iteration for HMMAS (eil51.tsp) (*Note: 25 trials x 10 000 iterations each trial = 250 000 iterations*)

Figure 12 shows the comparison of the average best cost between MMAS and HMMAS over 25 trials for eil51. HMMAS have a better average best cost compared to MMAS in a medium-sized tsp. Figure 13 shows the alpha and beta values with a peak of 1.7 and 2.2 respectively.
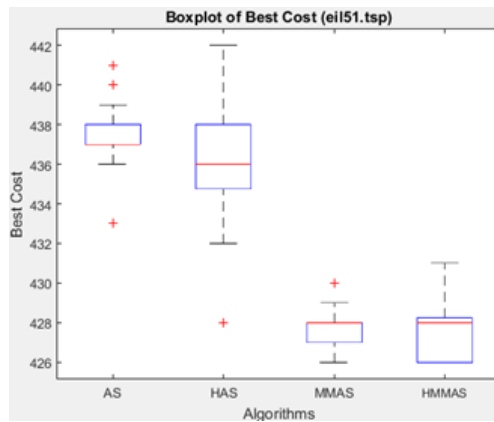
Figure 14: Boxplot of best cost for 25 independent trials of 4 different algorithms namely AS, HAS, MMAS and HMMAS for eil51.tsp. Each trial were conducted for 10 000 iterations.

Figure 14 shows that both HAS and HMMAS have a better performance compared to its base algorithm in terms of best cost found in each of the 25 independent trials for eil51.tsp. HAS has a lower median and lower inter-quartile (IQR) values compared to AS. Furthermore, HMMAS has a worst cost larger than MMAS, but more of the best costs are at the optimum of 426 for eil51.tsp.

Figure 15: Boxplot of best cost for 25 independent trials of 4 different algorithms for AS, HAS, MMAS and HMMAS for kroA100.tsp.

Figure 15 shows the boxplot for the best cost found by all 4 algorithms in test in each of the 25 trials. It can be seen that HAS has a better performance compared to AS in terms of best cost with a lower median as well. On the other hand, HMMAS has a slightly higher median compared to MMAS. It can also be observed from Figure 14 that both HAS and HMMAS have a larger IQR and this can be attributed to the variance in terms of best cost found caused by the heterogeneous approach introduced.

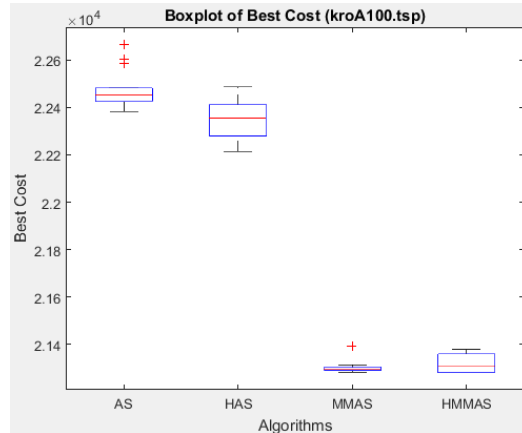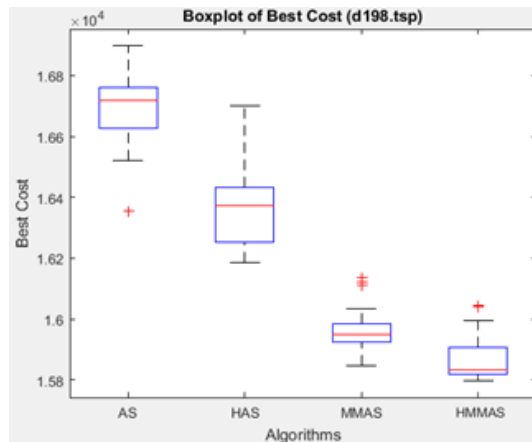Figure 16: Boxplot of best cost for 25 independent trials of 4 different algorithms namely AS, HAS, MMAS and HMMAS for d198.tsp.

Figure 16 shows the improvement of HAS and HMMAS over its base algorithms. Both heterogeneous algorithms have a lower median compared to AS and MMAS. In both cases, the improvements are statistically significant thus the algorithms clearly benefitting from the heterogeneous approach.

Table 10: p-values and z-values of Wilcoxon rank sum test for best cost of HAS and HMMAS against its respective base algorithm.

TSP	HAS vs AS	HMMAS vs MMAS
eil51	0.0143	0.8796
kroA100	2.03E-0.6	0.3078
d198	2.87E-08	1.27E-04

A two-tailed Wilcoxon rank sum test with confidence level of 95% was conducted for HAS against AS and HMMAS against MMAS with $p < 0.05$ as the threshold level where the difference is significant. The table above shows that the best cost found by HAS for the 25 trials are significantly better when compared to AS for all the three instances. The test shows that the best cost of HMMAS is not significant over its base algorithm for eil51 and kroA100. First of all, these two tsp instances fall under the category of small instance problem where even the base algorithm performs moderately. Secondly, the effect of individual variance or heterogeneity is limited in HMMAS due to algorithm's limitation of only a single agent to modify the pheromone limiting the overall heterogeneity advantage. Furthermore the performance of the base algorithm MMAS is clearly superior to that of AS meaning that it is also more difficult to for heterogeneity to show an improvement. However, despite this, HMMAS is statistically significant when compared to MMAS in terms of best cost found for d198.tsp.

7   Discussion, Conclusion & Future Work

In summary, a heterogeneous ACO has been introduced which implements artificial ants that have different 'behavioural traits' compared to the traditional homogeneous approach. This computational work in ACO is in relation to the biological aspect of real ants where ants are known to have diversity in their population. The results clearly show that the heterogeneous approach in ACO produce improved performance over the standard, parameter tuned algorithms on which they are based. The performance difference was particularly marked when implemented on Ant System. This is likely to be due to the greater contribution of each ant to the pheromone trail, highlighting the effect of diversity. The smaller gains made with HMMAS can be explained by the increased performance of the base algorithm, locating solutions closer to the optimum and also that only the best ant contributes to the pheromone update reducing the effect of population diversity on algorithm progression.

The implemented approach, by varying the alpha and beta values shows that even though prior work [3] suggests a range of optimal α and β values to choose from, determining a certain value is not easy as the parameters are problem-dependant. The results here show that the heterogeneous approach is able to overcome this problem by being robust to parameter settings by effectively exploring the parameter space in conjunction with optimising the problem. Having a variety of 'behavioural traits' rather than a single behaviour   shows the advantage in the performance of the algorithm. Recording the best performing alpha and beta values provides some support for the parameter values suggested by both Dorigo [3] and Stützle [6], but also highlighted instances where these parameter settings were not optimal. The discovery of distinct distributions of parameter settings for alpha and beta is interesting and demonstrates the algorithms' sensitivity to these parameters. These distributions remained stable despite being tested on multiple problem sizes. The work here has explored the hypothesis that heterogeneity is able to improve the performance of an algorithm and the results have gone some way to showing that heterogeneity applied to ACO can improve performance on the TSP and robustness to parameter settings. The next focus is on implementing Gaussian distribution towards heterogeneity and greater biological plausibility.

Acknowledgement

References

[1]   Christian Blum, "ACO Applied to Group Shop Scheduling: A Case Study on Intensification and Diversification," in *ANTS '02 Proceedings of the Third International Workshop on Ant Algorithms*, 2002, pp. 14–27.

[2]   G. (Gerhard) Reinelt and Gerhard, *The traveling salesman : computational solutions for TSP applications*. Springer-Verlag, 1994.

[3]   M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.

[4]   A. P. Modlmeier and S. Foitzik, "Productivity increases with variation in aggression among group members in Temnothorax ants," *Behav. Ecol.*, vol. 22, no. 5, pp. 1026–1032, Sep. 2011.

[5]   M. Collett and T. S. Collett,"*Spatial aspects of foraging in Ants and Bees*," Cold Spring Harbor Monograph Series *49*, pp 467 - 502, 2007.

[6]   T. Stutzle and H. Hoos, "MAX MIN Ant System and Local Search for the Traveling Salesman Problem," *Ieee Int. Conf. Evol. Comput.*, pp. 309–314, 1997.

[7]   G. Gutin and A. P. Punnen, Eds., *The Traveling Salesman Problem and Its Variations*, vol. 12. Boston, MA: Springer US, 2007.

[8]   A. Engelbrecht, "Heterogeneous particle swarm optimization," *Swarm Intell.*, pp. 191–202, 2010.

[9]   O. Blight, G. Albet Díaz-Mariblanca, X. Cerdá, and R. Boulay, "A proactive–reactive syndrome affects group success in an ant species," *Behav. Ecol.*, vol. 27, no. 1, pp. 118–125, Jan. 2016.

[10]   J. W. Lee and J. J. Lee, "Novel ant colony optimization algorithm with path crossover and heterogeneous ants for path planning," in *Proceedings of the IEEE International Conference on Industrial Technology*, 2010.

[11]   C. Chira, D. Dumitrescu, and C. M. Pintea, "Heterogeneous sensitive ant model for combinatorial optimization," *Genet. Evol. Comput.*, p. 163, 2008.

[12]   A. Hara, S. Matsushima, T. Ichimura, and T. Takahama, "Ant colony optimization using exploratory ants for constructing partial solutions," in *2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010*, 2010.

[13]   M. Yoshikawa, "Adaptive Ant Colony Optimization with Cranky Ants," Springer Netherlands, 2009, pp. 41–52.

[14]   P. Zhang and J. Lin, "An adaptive heterogeneous multiple ant colonies system," in *Proceedings - 2010 International Conference of Information Science and Management Engineering, ISME 2010*, 2010.

[15]   L. Melo, F. Pereira, and E. Costa, "Extended experiments with ant colony optimization with heterogeneous ants for large dynamic traveling salesperson problems," *Proc. - 14th Int. Conf. Comput. Sci. Its Appl. ICCSA 2014*, pp. 171–175, 2014.

[16]   T. Stutzle *et al.*, "Parameter Adaptation in Ant Colony Optimization IRIDIA – Technical Report Series Parameter Adaptation in Ant Colony Optimization," 2010.

SESSION 6 - APPLICATIONS

Evolutionary learning of fire fighting strategies

Martin Kretschmer and Elmar Langetepe

University of Bonn, Department of Computer Science, D-53113 Bonn, Germany

Abstract. The dynamic problem of enclosing an expanding fire can be modelled by a simple discrete variant in a grid graph. While the fire expands to all neighbouring cells in any time step, the fire fighter is allowed to block c cells in the average outside the fire in the same time interval. It was shown that the success of the fire fighter is guaranteed for $c > 1.5$ but no strategy can enclose the fire for $c \leq 1.5$. For achieving such a critical threshold the correctness (sometimes even optimality) of strategies and lower bounds have been shown by integer programming or by direct but often very sophisticated arguments. We investigate the problem whether it is possible to find or to approach such a threshold and/or optimal strategies by means of evolutionary algorithms, i.e., we just try to learn successful strategies for different constants c and have a look at the outcome. We investigate the variant of protecting a highway with still unknown threshold and found interesting strategic paradigms.
Keywords: Dynamic environments, fire fighting, evolutionary strategies, threshold approximation

1 Introduction

In the field of motion planning, online algorithms or Computational Geometry (and of course in many other areas) there are many examples of (somewhat annoying) gaps between upper and lower bounds of interesting and important constants. For example in the field of online algorithms for the famous k-server problem in almost all metric spaces the best known lower bound on the competitive ratio is k whereas the best known upper bound is $2k-1$, which gives a blind interval of $[k, 2k-1]$ for this value. The conjecture is that k is the tight bound; see for example [2]. Similarly the VC-dimension of L_2-visibility in simple polygons currently lies in the interval $[6, 14]$; see for example [8]. The threshold has to be somewhere in between.

A challenging approach might be to close or reduce such gaps (or only get some more insight w.r.t. a tendency) by means of rather simple but efficient evolutionary or genetic approaches. In this paper for a suitable scenario in the context of motion planning in grid environments we would like to find out how far this general idea might work. We make use of an *Evolutionary Computation* approach and manipulate a population of solutions by natural selection and mutation such that a fitness gradually increases; see also [7,9,12,13]. Rather than analysing evolutionary algorithms theoretically as for example given in [1,3], we would like to analyse the power of such simple algorithms for getting insight in well-defined theoretical questions.

In this paper we concentrate on the context of discrete *fire fighting* in different theoretically motivated variants. An overview of theoretical results in this context is given by Finbow & MacGillivary [5]. Assume that in a grid-cell environment a cell that is on fire expands the fire from one cell to its four neighbouring cells in one time step. On the other hand the fire fighter can block some of the cells outside the fire in any time step. The number of cells that can be blocked is given by an asymptotic budget $c \geq 1$ such that at any time step t we could have made use of $\lfloor c \times t \rfloor$ blocked cells in total.

We examine two questions. It is well-known that for $c > 1.5$ an expanding fire in the above model can be enclosed; see [11]. The result is obtained by a sophisticated recursive strategy idea. Optimality (minimum number of burned cells) can be obtained for example for $c = 2$ by making use of ILP formulations; see [14]. This does not work well for smaller c because of the running times. On the other hand for $c \leq 1.5$ no strategy can stop the fire, shown by a tricky proof in [4]. Therefore $c = 1.5$ is the fixed threshold for this case.

For this well-understood scenario we make use of simple evolutionary rules and show that for $c = 2$ we obtain the optimal strategy extremly fast. For $c \geq 1.7$ we still obtain enclosement results that seem to be close to the optimal. For c less than 1.6 our approach fails. The results are presented in Section 2.

The above first results might be seen as a test scenario for a new question considered in Section 3. For a protection budget c the task is rather than enclosing the fire, we would like to prevent a highway from being reached by the fire soon. Theoretical results and a fixed threshold for this setting are still unknown. We try to get an impression how reasonable strategies look like for different values of bugdet $c < 1.5$. It is more likely to make use of a single barrier close to the fire or is it recommendable to build (multiple) barriers away from the fire close to the highway? The focus here is that we get some ideas or insights by the use of evolutionary methods. In contrary to the former enclosement problem we first make experiments and an ongoing task is to find formal proofs. The results and the corresponding conjectures are presented in Section 3.

The main conclusion of our work is that simple, goal oriented evolutionary strategies could help to give insight into the solutions of dynamic motion planning problems. Especially, if such problems come along with a threshold question. The hope is that such approaches can also be used for similar problems.

2 Fire enclosement in a discrete grid setting

Given an infinite grid graph with vertex set $\mathbb{Z}^2$. Each vertex represents a cell in a grid graph. In the following vertices and cells are handle as synonyms. The set of edges is given by $\{((u,v),(x,y)) \mid |u - x| + |v - y| = 1\}$, i.e. each cell is neighbour to the cell directly above, below, left, and right. A fire starts at $(0,0)$ and spreads over time. After each time step, all cells with a burning neighbour start burning as well.

2

In the first setting the goal is to enclose the fire, such that only a finite (minimal) number of cells is lost. To achieve this, a certain number of non burning cells can be protected at each time step, which will then never catch fire.

The number of cells that can be blocked is given by an asymptotic budget $c \geq 1$ such that at any time step t we could have made use of $\lfloor c \times t \rfloor$ blocked cells. A simple example for $c = 2.7$ is shown in Figure 1. In the first step the fire fighter blocks $\lfloor 1 \times 2.7 \rfloor = 2$ two cells outside the fire. After the fire spreads in the next step the fire fighter blocks $\lfloor 2 \times 2.7 - 2 \rfloor = \lfloor 3.4 \rfloor = 3$ non-burning cells. Then the fire spreads again and in step 3 again $\lfloor 3 \times 2.7 - 5 \rfloor = \lfloor 3.1 \rfloor = 3$ cells can be blocked by the fire fighter outside the fire. The fire spreads for the last time and by blocking $\lfloor 4 \times 2.7 - 8 \rfloor = \lfloor 2.8 \rfloor = 2$ cells in the fourth step the fire is enclosed.
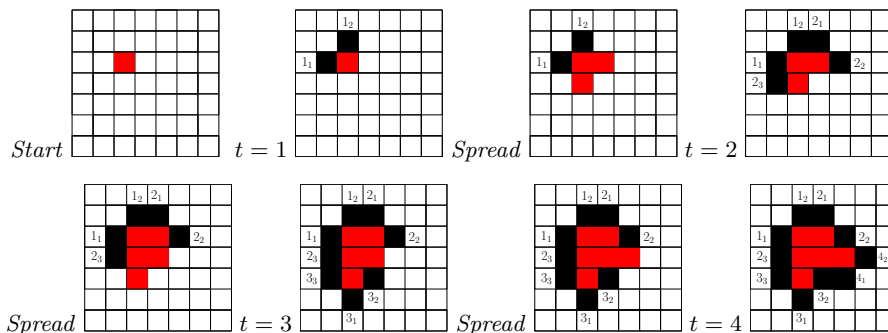
Fig. 1. An example for threshold $c = 2.7$. The fire starts at a single cell. At any time step, the fighter blocks the remaining cells of its overall budget $\lfloor t \times 2.7 \rfloor$ outside the fire. Then the fire spreads. The protected (black) cells are labelled by time parameters. After 4 time steps the fire is enclosed.

It has been shown that a fire can always be enclosed protecting $c = 2$ cells at each time step and it is impossible to do so with only one [6,14]. Finally, it was proved that a fire can always be enclosed when the average number c of protected cells exceeds 1.5 [11]. This bound is tight as shown by [4].

In the case of $c = 2$ even an optimal solution (i.e. minimal number of burning cells) has been found by using Integer Linear Programming [14]. Compared to that, in the following we want to investigate how good a simple evolutionary inspired algorithm can solve this task and how close we can get to the thresholds. The first experiments also can be seen as a test scenario for the question of protecting a highway considered in Section 3.

2.1 A goal oriented evolution model

To use an evolutionary method, we require a formal description of a general strategy, which can be modified (mutation) and recombined (inheritance) to

3

obtain a new strategy. Additionally, we have to define a fitness function for the comparison of strategies. Intuitively (and also driven by the known theoretical results) it seems to be a good idea for a strategy to

- start close to the fire
- build a (more or less) connected chain of protected cells, trying to surround the fire

Remark We further confirm these intuitions by having tried other variants as well. Our evolutionary experiments showed, that strategies which start protecting vertices further away from the origin perform worse than strategies that start close to the origin, some results for this are presented in Table 1. Analogously, the experiments showed that multiple disconnected barriers (that finally might be connected) do not work well. We omit to show the corresponding experiments due to space constraints. We refer to Section 3 where we have similar results for the problem of protecting a highway. I.e., general disconnected genomes tend to run in a connected barrier construction. The following definition is designed to follow the above simple principles.

Definition 1. *A strategy consists of*

- *a starting point (i, j)*
- *a sequence of directions $\{North, NorthEast, East, \ldots\}$ and each direction is combined with the information whether to extend the front (F) or the back (B) of the chain*

For short the strategy is given by the starting point and a list of pairs (X, Y) with $X \in \{N, NE, E, \ldots\}$ and $Y \in \{F, B\}$.

An example of a strategy (without a fire spread) is given in Figure 2. For the fixed starting cell $(0, -1)$ the sequence $((N, F), (NE, F), (SE, B), (SE, F), (E, B))$ is applied as follows. By (N, F) we extend $(0, -1)$ *forward* by the cell $(0, 0)$ in the north which now is the *new* front cell of the barrier. Then by (NE, F) relative to the new front cell we block the cell in direction north-east, which is cell $(1, 1)$. After that we apply (SE, B) for the current back end of the barrier which still is $(0, -1)$. The new back end cell is $(1, -2)$ which lies south-east from $(0, -1)$ and so on.

Notice, that such a strategy does not contain the information of the time at which the next vertex is protected. Instead, the next tuple of the sequence is applied, whenever we are allowed to protect an additional vertex.

The number of vertices that are protected per step is based on a bank account idea. We start with an initial budget and each time a vertex is protected, the budget decreases by 1. The budget has to remain positive but is always fully exhausted. After the fire has spread by one step, the budget increases by the fixed amount c. E.g. $c = 2$ means we can protect exactly two vertices in any step. For $c = 1.5$, the number of protected vertices alternates between 1 and 2.

Handling illegal genomes The above genome design does not pay any attention to the restriction that we cannot protect already burning cells. To deal with

4
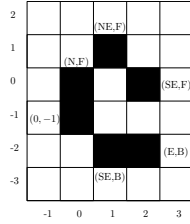
Fig. 2. Example of a strategy starting at $(0, -1)$ with sequence $((N, F), (NE, F), (SE, B), (SE, F), (E, B))$. Each protected vertex is labelled by the tuple that caused its protection.

that we decide to use the following behaviour: Whenever the sequence tries to protect a cell that is already burning, we start a search for the next non-burning cell in clockwise or counter clockwise order, depending whether we want to extend the front or the back of the barrier, starting at the direction that is given by the sequence. For example in Figure 2, if $(0, 0)$ is burning in the beginning, the application of (N, F) from $(0, -1)$ results in blocking the cell $(-1, 1)$, which gives the new front.

Fitness Evaluation In order to determine the fitness of a strategy two values seem to be important. The time needed to enclose the fire and the total number of burning vertices. Since randomly initialized sequences will most likely not enclose the fire, we use the total number of burning vertices after a fixed simulation time t. This also gives rise to gradual improvements. For example in Figure 1 for a simulation time $t = 3$ the given strategy has fitness 5, since 5 cells are burning at time $t = 3$. Note that we run arbitrary strategies with different simulation times (or steps).

2.2 Evolutionary Algorithm

The following algorithm keeps improving a randomly initialized set of strategies until it is manually stopped. Besides the budget c, it has several parameters which determine its behaviour.

- Input:
  - c budget income per time step
  - n population size
  - t number of simulation steps
  - p mutation probability
  - r ratio of parents kept after external selection
- Initialization: A population P of n randomly generated strategies (except the start point which is fixed to $(0, 1)$), each strategy needs to have a sequence of length at least $t \cdot c$

5

– Repeat
- • simulate any strategy of P for t time steps and determine its fitness
- • order P by fitness in increasing order and keep only the best $\lfloor r \cdot n \rfloor$ strategies as parents
- • restock P again to size n by selecting two parent strategies and combining their sequences via single-point crossover
- • for each tuple in each sequence of P, change it with probability p to a new random direction and extension side (mutation)

Note that for speeding up the results of our simulation as presented in the next section for $c < 2$ we decided to start the algorithm with an initial budget of 2. This allows us to protect two cell in the first step. Our experiments showed that this allows our algorithm to find successful strategies much faster and therefore also for smaller values of c. Asymptotically, there is no difference for the threshold. This small artefact might also be interpreted as a goal oriented approach.

2.3 Experimental results
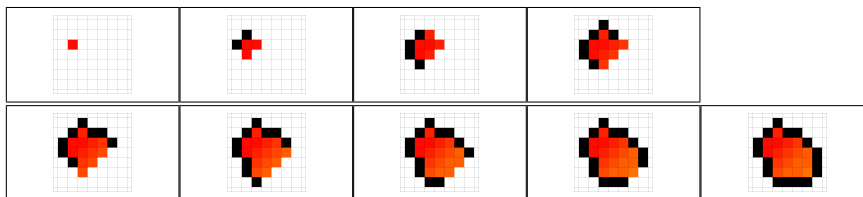
Fig. 3. Example of a strategy found when protecting exactly $c = 2$ vertices per step. Enclosed after 8 steps with 18 burning vertices.

Figure 3 shows an optimal strategy that was found by evolution for the case $c = 2$. It takes 8 steps to enclose the fire and in the end 18 vertices are on fire. This is optimal for both time and number of burning vertices as shown in [6]. Surprisingly, it tooks only 84 generations in total until this strategy was found.

An example of a strategy that was found for $c = 1.7$ after 1002 generations is depicted in Figure 4. A video of the successful strategy is shown in
.

For even smaller values of c, our algorithm starts failing to find enclosing strategies. An example for $c = 1.6$ is given in Figure 5. It seems that the strategy might be able to enclose the fire after a longer time, but even increasing the simulation time t did not lead to success.

Figure 6 shows for which values of c we were able to find enclosing strategies. For constructing the figure we choose a simulation time of $t = 80$. As mentioned above increasing the simulation time did not help. One can see that for values

6
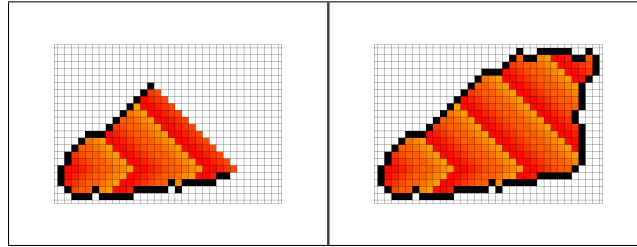
Fig. 4. Strategy found for $c = 1.7$. Enclosed after 46 steps with 371 burning vertices. The colored shading indicates how the fire spreads over time.
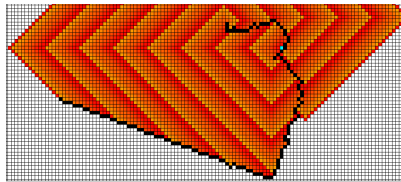
Fig. 5. Extract of a failing strategy for $c = 1.6$. Note that the fire expands on both sides of the barrier.
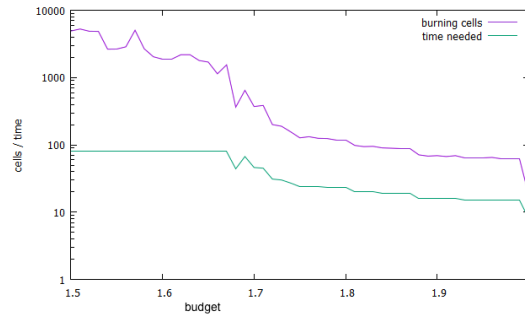
Fig. 6. An overview of the results. For convenience (results for different c were similar for any number of simulation steps) we used simulation time $t = 80$. For a given budget the green curve shows the time required for enclosing the fire and the blue curve shows the number of burning cells. We obtain positive (enclosement) results up to budget c slightly larger than 1.68. After that the green curve is just fixed to the restricted simulation time which indicates that the fire was not enclosed.

Start	Enclosing Time	Burning Vertices
$(0, 1)$	8	18
$(0, 4)$	23	156
$(1, 3)$	15	68
$(2, 2)$	24	161

Table 1. Fitness of best strategies found for $c = 2$ and different starting points.

7

smaller than $c \approx 1.68$ the building of the barrier continued until the simulation ended. This means that the fire was not enclosed.

So far, any strategy presented had a fixed start point neighbouring the origin of the fire. As an example we compare the strategy for $c = 2$ mentioned above to strategies whose start point is fixed to a vertex four steps away from the origin. Up to symmetry there are three different coordinates for this. $(0, 4)$, $(1, 3)$ and $(2, 2)$. Table 1 shows the times required to enclose the fire using these different starting points, compared to the optimal strategy shown above. Starting further away from the fire takes longer to enclose the fire. We have similar results for other values of c.

2.4 Fire enclosement conclusion

At least for values of c a bit away from the overall tight threshold, the simple evolutionary goal oriented algorithm was able to find successful (and in the case of $c \geq 2$ even optimal) strategies surprisingly fast. Successful strategies close to the threshold $c = 1.5$ are not easy to find, even by the use of very general genomes and many simulation steps. This seems to be clear for the following reason. The corresponding successful connected barrier solution presented in [11] for any $c = 1.5 + \epsilon$ makes use of four rounds. For any round the strategy behaves analogously but the next round starts with a 90 degree rotation. It is unlikely that a random approach will find the appropriate time for starting the next round and rotation.

3 Protection of a highway

Here we consider a different and new question. Conversely to the previous section we did not have any idea for a reasonable strategy and/or a threshold. The question is how long can we protect a highway (modeled by a line of cells) from the fire, if some budget $c < 1.5$ is given. We would like to avoid that the fire touches a *line* very early? What is a reasonable strategy? Should we start close to the fire or close to the highway? Should we design a single connected barrier or more barriers which are partly disconnected?

In Figure 7 we give an example for a strategy for $c = 1.2$. This means that in the first 4 time steps the fire fighter makes use of a single blocking cell. In step $t = 5$ the fire fighter can block two cells for the first time since $\lfloor 5 \times 1.2 - 4 \rfloor = 2$ holds. Similar to the previous section we can also assume that in the start situation some constant cells are already blocked, this is indicated by the blocked cell of label 0 in Figure 7. Figure 7 has to be interpreted as follows. If c_{t-1} cells were used from the budget of the fire fighter after step $t - 1$, at the next time step t, the fighter first blocks $\lfloor t \times c - c_{t-1} \rfloor$ cells outside the fire and then the fire spreads. After 7 time steps and the corresponding spread the fire reaches the highway. Note that the strategy stops in this moment.

8
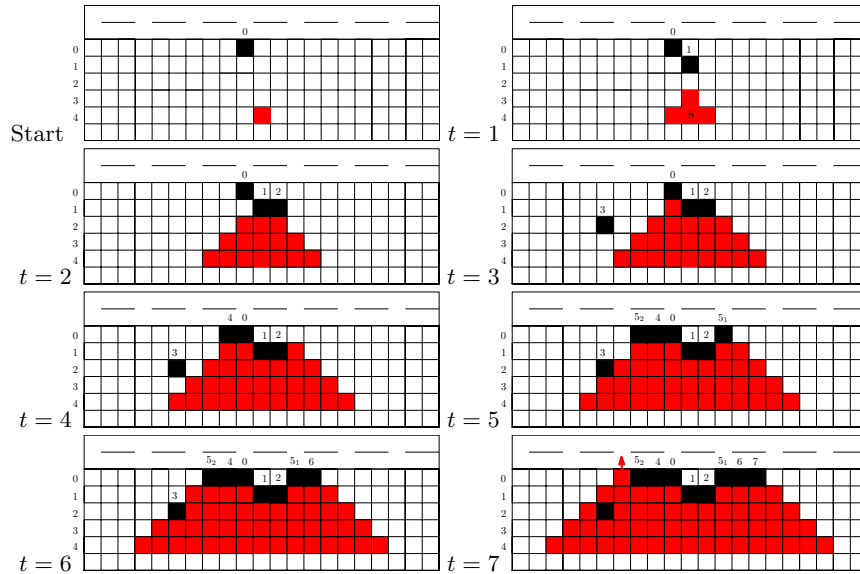
Fig. 7. A fire fighter strategy for protecting a highway with budget $c = 1.2$. The fire starts at a single cell, one cell is initially protected. At any time step, the fighter first blocks the remaining cells of its overall $\lfloor t \times c \rfloor$ budget outside the fire and then the fire spreads. After 7 time steps the fire reaches the highway.

3.1 Evolution models

Since the given problem was not theoretically analysed before, we first had to test several ideas experimentally in order to achieve a more goal oriented model. In contrary to the enclosement scenario discussed before we do not know whether a connected strategy will lead to optimal or efficient solutions. So we first tried to allow general strategies that could protect arbitrary cells. We made use of a very simple coordinate based genome model, such that a strategy is simply defined by a set of cell coordinates defining which cells should be protected.

For such a set of cells, the cells are protected in their L_1-distance order from the origin of the fire, i.e., cells closer to the fire origin will be protected first. In the evolution process this behaviour forces that useless protections far away from the origin will be cancelled out more quickly. In principle the above principles allow us to define arbitrary strategies.

Altogether, we either make use of

- a connected genome (as in the previous section) or
- a coordinate genome described by a set of cell coordinates (as just mentioned)

3.2 Evolutionary Algorithm

We noticed that usually we do not get any improvements by the recombination of strategies. Therefore we changed the framework used in Section 2.2 and

9

restrict the algorithm to mutation only. This also means that we do not need to have a large population, instead we only initialize a single randomly generated strategy that will keep mutating. If a mutation leads to an improvement, the strategy keeps that mutation, otherwise it is undone. Altogether, this is a so-called (1+1)EA; see [12].

This process of improving a single strategy can easily be parallelized such that a larger set of single strategies keeps improving over time. This is very beneficial, because the final result often depends on the initialization and not every run leads to the best result. Another difference to the previous section is the fitness evaluation, which obviously has to be adjusted with respect to the problem definition.

Fintness evaluation For the enclosement problem considered in Section 2.1 we tried to minimize the total number of burning cells. In this case we have used exactly this number for determining the fitness. Now we want to maximize the time the fire requires to reach the highway. It turns out that increasing this time value directly by a random mutation or recombination is very unlikely. Therefore we require a fitness evaluation that also allows for smaller and gradual improvements. To attain this we take into account how many vertices are burning and also their corresponding distance to the highway. A formal definition is given below.

For letting the algorithm run, actually we only need to be able to compare strategies pairwise. Fortunately, this can also be realized by our fitness function.

Definition 2. *Let S be a protection strategy for a given highway. By $r(S)$ we denote the first moment in time when the fire reaches the highway, if S is applied. By $d(S)_i$ we denote the number of burning cells with distance i to the highway after $r(S)$ simulation steps.*

Strategy S_1 has a larger fitness *than S_2 if $r(S_1) > r(S_2)$ holds or for $r(S_1) = r(S_2)$ if $d(S_1)_i < d(S_2)_i$ holds for the smallest index i where $d(S_1)_i \neq d(S_2)_i$.*

For example in Figure 7 the given strategy S has value $r(S) = 7$. We also have $d(S)_0 = 1$, $d(S)_1 = 9$ and $d(S,9)_2 = 12$ and so on. So another strategy S' would have larger fitness, if $r(S') > 7$ holds or for $r(S') = 7 = r(S)$, if we have for example $d(S')_0 = 1$, $d(S')_1 = 9$ and $d(S')_2 = 11 < 12 = d(S)_2$.

The main idea is that by trying to keep the fire farther away from the highway, finally also the overall time where the fire reaches the highway can be increased.

3.3 Experimental results

Similar to the enclosement problem our implementation allows us to set or manipulate many different parameters and options for a goal oriented evolutionary process, such as the budget c, the strategy design (general genome or connected barriers), the population size (number of strategies optimized in parallel), the mutation rate, the fire source (distance to the highway), starting positions (for connected barriers), optional initial budget and so on.

10

Videos Finally and interestingly we mainly found two different strategic behaviours depending on the corresponding genomes, they will be explained precisely below. For convenience for $c = 1.2$ we prepared two animations that show the finally attained best strategies for

1. **General genomes** Symmetric and alternating strategy:
   http://tizian.informatik.uni-bonn.de/Video/1.2SymAlt.mp4
2. **Connected barriers** Asymmetric and diagonal strategy:
   http://tizian.informatik.uni-bonn.de/Video/1.2AsymDiag.mp4

where in the second case of connected barriers sometimes also symmetric and alternating strategies were attained under circumstances explained below. The above strategies have been found after 156925 (1.) and 34226 (2.) generations.

1. General genomes First, we found out that the use of general genomes always (for different settings) mutate toward connected barriers; Figure 8 shows some of the finally attained strategies. All strategies show a similar behaviour. They start somewhere between the origin of the fire and the highway, usually a bit closer to the fire. Then any strategy continues to protect cells alternating between left and right, trying to keep the fire as long and as far away from the highway as possible. In the following we refer to such strategies as *symmetric and alternating*.

Note that any of the given strategies can be reconstructed such that the symmetric and alternating process is performed directly at the highway. The time where the fire reaches the boundary will not change in this case. Our fitness function simply prefers to shift the fire away from the highway.
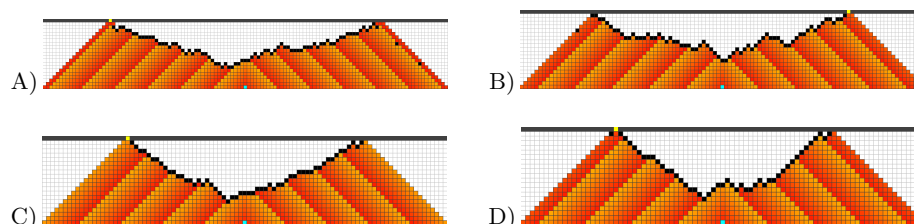
Fig. 8. Resulting strategies using the general coordinate genome for different values of the budget: A) $c = 1.4$, B) $c = 1.3$, C) $c = 1.2$ D) $c = 1.1$. The fire starts 20 steps away from the highway. The highway was reached after 61, 54, 48 and 43 steps, respectively.

2. Connected barriers After that we again considered connected barriers with different starting positions below the origin. Depending on the distance between the start and the fire, we observed two different strategic behaviours which can be categorized as follows.

If the starting position is somehow chosen *too close* to the fire origin or *too close* to the highway we obtain strategies that behave in a symmetric and

11

alternating way as before. On the other hand if we somehow start at the *right* distance, the attained strategies suddenly performed different and a lot better. An example of such a strategy for $c = 1.2$ is given in Figure 9, the behaviour of the strategy can be subdivided into three different phases which will be explained below. In contrast to the symmetric and alternating strategy which only kept the fire for 48 steps away from the highway with the same budget, the alternative strategy increases this time to 92!
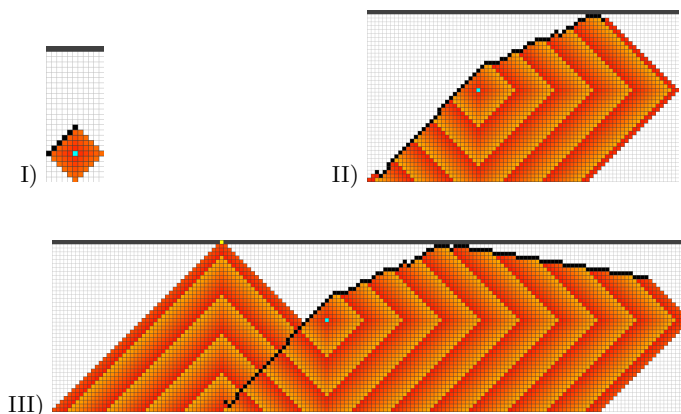
Fig. 9. Three phases of a connected strategy protecting the highway for $c = 1.2$. I) Reach the level of the starting position of the fire diagonally from the center. II) Extend this diagonal, but simultaneously also try to shift the fire away from the highway on the other side until the fire reaches the highway there. III) Use the full budget to keep the fire away at this side, the fire *runs* to the highway at the other side. The strategy starts 5 steps above the origin. The fire reaches the highway after 92 steps.

In general these strategies can be subdivided into three phases. Figure 9 shows the end of each phase.

I) Protect a diagonal downwards until a cell at the same level as the origin is reached. Starting n cells above the origin, this requires the protection of $n + 1$ cells and this needs to be done before the n−th time step, because otherwise the fire would reach that cell first. This in turn requires c to be large enough. Or the other way round, given $1.0 < c < 1.5$, n needs to be large enough such that $n+1$ cells can be protected after n steps $\Leftrightarrow cn \geq n+1$ $\Leftrightarrow n \geq \frac{1}{c-1}$.

II) Continue the diagonal downwards by one cell in every second step. Use the rest of the budget to keep the fire at the other end of the barrier as far away from the highway as possible. This procedure ends when the fire gets close to the highway.

III) In order to protect the highway, from now on we are forced to protect at least one cell per step at the end close to the highway. Since protecting one

12

cell at every step at one end and one cell at every second step at the other end would require a budget of $c \geq 1.5$, the diagonal part of the barrier will be overrun by the fire making it impossible to continue this end at all. So the strategy will simply continue to hold the fire back at the upper part of the barrier until the fire reaches the highway on the other side. Again, because the fitness evaluation prefers fewer burning vertices close to the highway, the slope of the part built in this phase occurs.

We will refer to this behaviour as an *asymmetric and diagonal* strategy. Notice that for $c = 1.5$ this leads immediately to a strategy that protects the highway infinitely. Furthermore this strategy can only be applied if the fire starts far enough away from the highway. The closer the budget c gets to 1.0, the more distance is required. If this distance is not available, there seems to be no better strategy than the symmetric and alternating one.

3.4   Highway protection conclusion

Using the evolutionary algorithm, we gained helpful insights into the highway protection problem. Both the symmetric and alternating and the asymmetric and diagonal strategy are promising candidates for optimal solutions. The choice between the two alternatives seem to depend on the possibility of building the diagonal of phase I). We found out that with one additional initial budget, the connected genome always run into the asymmetric and diagonal variant. By protecting two cells in the beginning, we can immediately finish the first phase by starting directly above and to the left of the fire. Without an initial budget the strategy has to fight for reaching the starting level of the fire source from the left. This can only happen if in comparison to the budget, the source lies sufficiently far away from the highway.

Considering phase III), there seems to be some room for a further recursive improvement. When the fire has overcome the diagonal part of the barrier in phase II) it will take the direct way to the highway. For a while we shift the fire away from the other side by using the full budget. The barrier is build with a given slope; see Figure 9 III). But this part could have been build also with budget 1 along the highway. Therefore the remaining budget can be used to protect the highway at the left hand side. Therefore we can consider the situation with a budget $c' = c - 1$. We found out that in this case the best strategy builds a symmetric and alternating barrier directly at the highway.

For values $c \approx 1.5$ we never observed a strategy that was able to protect the highway infinitely long. We think that 1.5 is the threshold.

Conjecture For $c < 1.5$ there is no strategy that protects an arbitrary highway from the fire. The best protection strategy either builds a single connected barrier symmetrically and alternating close to the highway or first the asymmetric and diagonal connected barrier strategy is applied. This depends on the relationship between the distance of the fire source to the highway and the given budget.

13

4 Future work on theoretical threshold questions

Besides proving and analysing the above conjecture theoretically, we finally would like to mention that there are other interesting upper and lower bounds which we analogously would like to attack by a goal oriented evolutionary approache. For example, similar to the subjects presented here there are other scenarios in discrete and continuous fire fighting settings that come along with a threshold. An interesting overview for such gaps is given in the CG Column by Klein and Langetepe[10]. Alternatively, one might also think of the protection for different objects, also formalized by a set of cells. Additionally, among many others, the two blind intervals (VC-dimension, k-server conjecture) mentioned in the very beginning are also worth considering.

References

1. H.-G. Beyer, H.-P. Schwefel, and I. Wegener. How to analyse evolutionary algorithms. *Theoretical Computer Science*, 287(1):101 – 130, 2002.
2. M. Chrobak and L. L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM J. Comput.*, 20(1):144–148, Feb. 1991.
3. S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1):51 – 81, 2002.
4. O. N. Feldheim and R. Hod. 3/2 firefighters are not enough. *Discrete Applied Mathematics*, 161(1-2):301 – 306, 2013.
5. S. Finbow and G. MacGillivray. The firefighter problem: a survey of results, directions and questions. *Australasian Journal of Combinatorics*, 43(57-77):6, 2009.
6. P. Fogarty. *Catching the fire on grids*. PhD thesis, The University of Vermont, 2003.
7. D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.
8. A. Gilbers and R. Klein. A new upper bound for the vc-dimension of visibility regions. *Comput. Geom.*, 47(1):61–74, 2014.
9. J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.
10. R. Klein and E. Langetepe. Computational Geometry Column 63. *SIGACT News*, 47(2):34–39, June 2016.
11. K. Ng and P. Raff. Fractional firefighting in the two dimensional grid. Technical report, DIMACS Technical Report 2005-23, 2005.
12. I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Number 15 in Problemata. Frommann-Holzboog, Stuttgart-Bad Cannstatt, 1973.
13. H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981.
14. P. Wang and S. A. Moeller. Fire control on graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 41:19–34, 2002.

14

Evolutionary Optimization of Tone Mapped Image Quality Index

Xihe Gao, Jeremy Porter, Stephen Brooks, and Dirk V. Arnold

Faculty of Computer Science, Dalhousie University
Halifax, Nova Scotia, Canada B3H 4R2
`{xgao,jporter,sbrooks,dirk}@cs.dal.ca`

Abstract. The development of reliable image quality measures for the assessment of tone mapped images constitutes a significant advancement in high dynamic range imaging. The ability to objectively assess the quality of tone mapped images allows treating tone mapping as an optimization problem that can be solved by automated algorithms, without the need for human input. The most prominent quality measure for tone mapped images is the Tone Mapped Image Quality Index. An optimization approach has been proposed in connection with the introduction of that measure that operates in a high-dimensional search space and is computationally expensive. In this paper, we propose an evolutionary algorithm to solve the tone mapping problem using a generic tone mapping operator and the Tone Mapped Image Quality Index as the objective to be maximized in a much lower dimensional solution space. We show that the evolutionary approach results in significantly reduced computational effort.

Keywords: High dynamic range imaging, tone mapping, optimization, evolution strategy, image quality assessment

1   Introduction

High dynamic range (HDR) images provide the capacity to represent a greater range of luminance values than standard image formats do. Sources of HDR images include digital cameras as well as photorealistic rendering algorithms. With advances in hardware, they are rapidly becoming more commonplace in digital imaging. To display HDR images on contemporary display devices, the dynamic range needs to be adapted to the much smaller range of the device. This computational task is called tone mapping.

Numerous tone mapping operators (TMOs) have been proposed (see Banterle et al. [2] for an overview), and many of those depend on parameters that significantly impact the appearance of the tone mapped images. Choosing an appropriate TMO and setting its parameters for a particular HDR image often requires careful tuning. Chisholm et al. [3] have proposed an interactive approach to tone mapping, where the quality of tone mapped images is optimized iteratively, using an evolutionary algorithm (EA) that relies on human input for

selection. More recently, significant progress has been made in the development of objective quality measures for tone mapped images, potentially opening up the possibility of generating optimized tone mapped images without the need for user input. Algorithms for the automatic optimization of tone mapped images have been proposed by Gao et al. [5, 6], who employ saliency based image quality measures in connection with simple EAs for optimization. Their work shows that visually acceptable tone mapped images can often be obtained by automatically tuning the parameters of various TMOs and blending their results.

A milestone in the development of objective quality assessment measures for tone mapped images is the introduction of the Tone Mapped Image Quality Index (TMQI) by Yeganeh and Wang [14]. TMQI combines measures of structural fidelity and statistical naturalness in a single numerical score. Structural fidelity is derived from structural similarity (SSIM) [13] and measures spatial dependencies of pixels between test and reference images. Statistical naturalness measures overall brightness and contrast for "natural" appearance. Ma et al. [9] have proposed TMQI-II as an improved variant of that quality measure that is sufficiently robust to allow for the automatic optimization of tone mapped images. They also present an optimization method that searches for images with optimal TMQI-II scores. Their algorithm interleaves the optimization of structural fidelity using gradient ascent with that of statistical naturalness. Experimental results demonstrate that their method can significantly improve the TMQI-II score and appearance of tone mapped images, albeit usually at a high computational cost.

The purpose of this paper is to compare the approach to the optimization of TMQI-II used by Ma et al. [9] with an evolutionary approach similar to those used by Chisholm et al. [3] and Gao et al. [5]. In contrast to the approach by Ma et al. [9], we do not operate in the high-dimensional search space resulting from considering all pixel intensities as variables, but instead we employ the generic TMO by Mantiuk and Seidel [10]. The parameters of that operator define a low-dimensional space in which the EA can find good solutions with relatively little computational effort. Our algorithm differs from that of Gao et al. [5, 6] in the use of the generic TMO as opposed to a blend of results from multiple operators, but also in that we use TMQI-II as the quality measure in order to be able to compare different optimization approaches using a common platform. The effectiveness of the proposed approach is validated using an HDR image benchmark set. The remainder of the paper is organized as follows. In Section 2 we briefly discuss related work. Section 3 describes our algorithm. An experimental comparison of our algorithm with that by Ma et al. [9] is presented in Section 4. Section 5 concludes with a brief summary and proposed future work.

2 Related Work

With the increasingly widespread use of HDR images, tone mapping has attracted much attention from researchers. During the past two decades, many TMOs have been developed. Most of the current operators are derived from the

human visual system and try to preserve perceptual factors, such as brightness, contrast, and visibility during tone mapping. Operators can coarsely be classified into global and local operators. The compression curve of global operators is the same over the entire image, while local operators are adaptive to each pixel. Local adaptation can better preserve image details, but it is also computationally more expensive and not guaranteed to give better results. A thorough review of existing TMOs can be found in a book by Banterle et al. [2].

The appearance of tone mapped images depends on the choice of TMOs and the setting of their parameters. Notable progress has been made towards the objective quality assessment of tone mapped images. Yeganeh and Wang [14] have proposed TMQI, which combines measures of structural fidelity and statistical naturalness in a single numerical score. Ma et al. [9] have proposed an improved variant of that measure. Nafchi et al. [12] have presented a feature similarity index based on local phase information of images. Gu et al. [7] have proposed a no-reference quality measure that estimates the amount of local detail in images. Gao et al. [5] have introduced an image quality measure that measures visual saliency distortion caused in the process of tone mapping. Gao et al. [6] have expanded on that work by developing a perceptual quality measure to capture the reproduction of perceptual features including brightness, visual saliency, and details during tone mapping. Of the proposed measures, TMQI and its immediate successor, TMQI-II, are the most prominent. A systematic comparison of existing image quality measures for tone mapped images remains to be performed.

Progress in the objective quality assessment of tone mapped images has opened up a new approach to tone mapping: using any of the proposed quality measures as the objective, tone mapping can be solved as an optimization problem. Notably, Ma et al. [9] have proposed an optimization algorithm to iteratively improve TMQI-II scores of tone mapped images. Structural fidelity is improved using a gradient ascent method and statistical naturalness is enhanced with a point-wise intensity transformation. The algorithm operates in a high-dimensional solution space. In contrast, Chisholm et al. [3] and Gao et al. [5, 6] employ EAs to optimize the parameters of TMOs, allowing optimization to proceed in much lower dimensional spaces. As they have used different quality measures, no immediate comparison of the two approaches to the optimization of tone mapped images has been performed.

3   Algorithm

The solution space available for our algorithm to search is that defined by the generic tone mapping operator. The optimization algorithm we adopt is an evolution strategy. Both are described in this section.

3.1   Tone Mapping

The generic tone mapping operator by Mantiuk and Seidel [10] aims to provide the ability to emulate a wide range of tone mapping operators using computa-
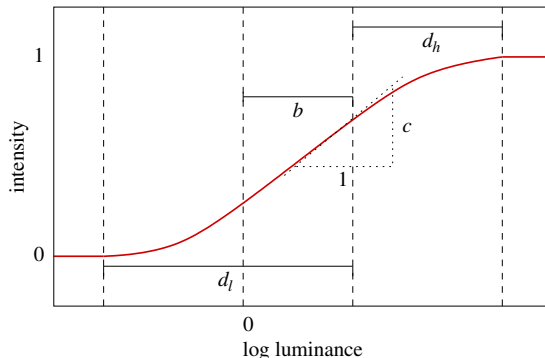
Fig. 1. Tone curve used in the generic TMO and its parameters (adapted from [10]). Parameter b allows for brightness adjustment, d_l and d_h determine the lower and higher midtone ranges, and c governs contrast.

tionally inexpensive image processing operations. We choose it for its low computational cost as well as due to its capacity to generate a wide range of tone mapped images using a relatively small set of parameters that can be used for optimization. The operator maps intensity values as

$$C_{\text{LDR}} = f_{\text{MT}}(f_{\text{TC}}(L_{\text{HDR}})) \cdot \left(\frac{C_{\text{HDR}}}{L_{\text{HDR}}}\right)^s, \tag{1}$$

where C_{HDR} and C_{LDR} are the colour channels of the HDR image and the tone mapped image, respectively, L_{HDR} is the luminance of the HDR image, f_{TC} denotes the tone curve, f_{MT} represents the modulation transfer function, and s is a parameter used for saturation adjustment. The tone curve is defined as a sigmoidal function, with parameters b, d_l, d_h, and c provided for tuning the curve shape as illustrated in Fig. 1. The modulation transfer function allows specifying several parameters that determine a 1D function of spatial frequency and allows the tuning of blurring and sharpening operations applied to an image. The function involves band-pass filtering implemented with difference of Gaussian operators, with parameters m_1, m_2, and m_3 for the adjustment of different frequency components; see [10] for details.

Since TMQI-II scores are computed on the basis of luminance values, without taking colour information into account, we choose not to modify the value of the saturation adjustment parameter s. Thus, a total of seven parameters are available for tuning.

3.2 Evolutionary Optimization

For the optimization of TMQI-II scores in the seven-dimensional parameter space thus defined, we use the $(1 + \lambda)$-ES[1] employed by Chisholm et al. [3] for the

---

[1] See Hansen et al. [8] for an overview of evolution strategy related terminology.

Table 1. Parameters of the generic TMO.

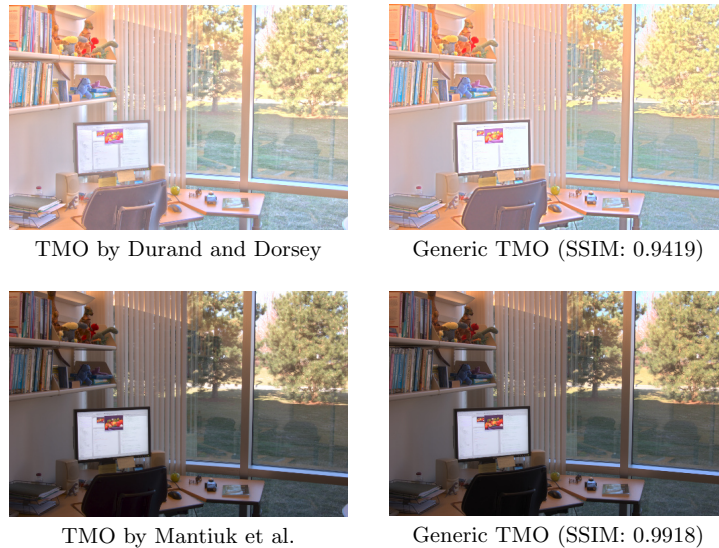| Parameter	Range	Description
b	$[-2.0, 2.0]$	brightness factor
d_l	$[0.0, 2.5]$	lower midtone range factor
d_h	$[0.0, 2.5]$	higher midtone range factor
c	$[0.2, 1.5]$	contrast factor
m_1	$[-2.0, 2.0]$	high frequency factor
m_2	$[-2.0, 2.0]$	medium frequency factor
m_3	$[-2.0, 2.0]$	low frequency factor

interactive evolutionary optimization of tone mapped images. The use of the evolutionary approach to optimization is motivated by the lack of availability of analytical gradients and the potential for ruggedness resulting from the choice of quality criterion.

There are no restrictions regarding the setting of parameters of the generic TMO. However, we find that allowing parameters to grow without bounds may result in very marginal changes to TMQI-II scores and thus in ill-conditioning that negatively impacts the ability of the simple evolution strategy to optimize image quality. We thus impose boundary constraints that prevent the parameters from moving past their useful ranges. In order to be able to define image-independent ranges, we calibrate the logarithmic luminance values of the pixels in an HDR image by subtracting the mean logarithmic luminance. Ranges and short descriptions of the parameters are listed in Table 1.

Candidate solutions are seven-dimensional real vectors comprised of the parameters of the generic TMO. In each iteration of the algorithm, $\lambda > 1$ offspring are generated from the parental candidate solution $\mathbf{x} \in \mathbb{R}^7$ as

$$\mathbf{y}_i = \mathbf{x} + \sigma \mathbf{z}_i \qquad i = 1, \ldots, \lambda \qquad (2)$$

where $\sigma \in \mathbb{R}$ denotes the step size parameter and the $\mathbf{z}_i \in \mathbb{R}^7$ are independent, standard normally distributed mutation vectors. Out-of-range values of variables are clamped to the boundaries. In light of potential issues with decreasing step size as a result of constraint handling such as discussed by Arnold [1], we have also experimented with an exterior penalty approach, but not observed a significant difference in performance. In each iteration, the candidate solution that leads to the highest TMQI-II value among the union of the parent and the set of all offspring is selected and adopted as the parent for the next iteration. The offspring number λ is set to 10 throughout, and the step size parameter is initialized to 0.5 at the start of a run. That parameter is decreased by multiplication with 0.8 in each iteration where the parental candidate solution is superior to all of its offspring; it is unchanged in those iterations where an offspring candidate solution is successful. We terminate a run when the change in the best TMQI-II value has been less than 10^{-4} for six consecutive iterations and return the best candidate solution found as the result.

TMO by Durand and Dorsey	Generic TMO (SSIM: 0.9419)
TMO by Mantiuk et al.	Generic TMO (SSIM: 0.9918)

Fig. 2. Comparison between images tone mapped using the TMOs by Durand and Dorsey (top left) and Mantiuk et al. (bottom left) using default parameter settings, and corresponding images generated using the generic TMO (right) with parameters chosen to maximize SSIM scores.

4   Experimental Results

To evaluate the performance of the evolutionary approach, we conduct a comparison with the algorithm by Ma et al. [9] for the optimization of TMQI-II scores. We carry out the comparison on a set of sixteen HDR images. This set is identical with that used by Ma et al. in the evaluation of their approach, with the exception of one missing image to which we do not have access.

For each HDR image, we employ three starting points in the search for optimal tone mapped images. Ma et al. use starting points generated by various TMOs with default parameter settings, including the the logarithmic operator, the operator by Durand and Dorsey [4], and that by Mantiuk et al. [11]. In order to ensure comparable starting points for both algorithms, we determine parameter settings for the generic TMO such that the resulting tone mapped images closely match those generated by the various TMOs. Figure 2 shows a typical example of the generic TMO's ability to emulate other TMOs and find matches that are visually nearly indistinguishable.

We use the implementation by Ma et al. [9] for TMQI-II as well as of their optimization algorithm. That implementation is in *Matlab* and, according to the authors, not optimized for speed. However, as the same implementation of TMQI-II is used for image quality assessment in both of the optimization algo-
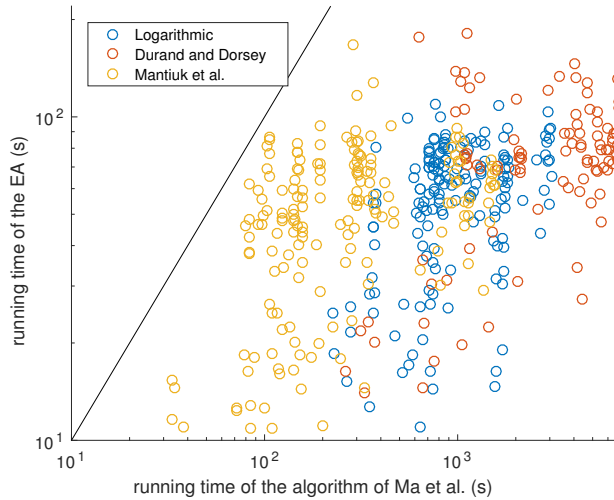
Fig. 3. Running times of the EA plotted against running times required by the algorithm of Ma et al. [9] to reach equivalent TMQI-II scores. Data are shown for starting points matching those generated by the logarithmic operator, the operator by Durand and Dorsey, and that of Mantiuk et al. The black line indicates the identity.

rithms, the comparison is meaningful for establishing relative performance. The EA as well as the generic TMO are implemented in *Matlab* and not optimized for speed either. Running times reported are for a PC with Intel Quad-Core 2.66GHz CPU with 4GB of RAM. As the optimization approach by Ma et al. does not exploit parallelism, we have chosen not to make use of more than one CPU core in the implementation of the EA. Making use of parallel computational resources by evaluating offspring simultaneously on multiple cores is straightforward. On our hardware, the computation of a single TMQI-II score for images of the size considered here (approximately 360×500 pixels) takes about 0.3 seconds.

For each of the sixteen HDR images and three starting points, we have conducted eleven independent runs of the EA for the optimization of TMQI-II, for a total of 528 runs. All of the TMQI-II scores are calculated from tone mapped images stored in PNG format (i.e., with lossless compression, but with only eight bits per colour channel). Table 2 shows TMQI-II scores of the starting points as well as median and standard deviation of the scores obtained after evolutionary optimization. Also shown are median and standard deviation of the computational times. We have then run the algorithm of Ma et al. [9] and recorded the time it requires to reach the TMQI-II scores obtained by the EA. Median and standard deviation of those times are shown in the last column of the table. Running time data from all 528 runs are represented graphically in Figure 3.

Table 2. Comparison between evolutionary TMQI-II optimization and optimization using the algorithm of Ma et al. [9]. The table lists TMQI-II scores of starting points, scores after evolutionary optimization, the computational time (in seconds) required to generate those results, and the computational time (in seconds) for the algorithm of Ma et al. to reach equivalent TMQI-II scores. That computation time is omitted (—) when an equivalent score could not be reached within 7200 seconds. Shown are median values of eleven independent runs, with standard deviations given in parentheses.

HDR Image	Starting Point	Initial TMQI-II Score	Optimized TMQI-II Score	Running Time (EA)	Running Time (Ma et al.)
Foggy Night	Logarithm	0.3807	0.9780 (0.0047)	58.5 (23.6)	756.0 (39.0)
340 × 512	Durand/Dorsey	0.3844	0.9766 (0.0020)	71.0 (40.4)	1263.7 (304.9)
	Mantiuk et al.	0.9091	0.9779 (0.0011)	74.6 (26.3)	357.6 (39.2)
Clock Building	Logarithm	0.4409	0.9757 (0.0029)	80.6 (22.6)	1035.7 (288.3)
384 × 512	Durand/Dorsey	0.4415	0.9752 (0.0040)	68.4 (24.2)	5748.9 (2126.1)
	Mantiuk et al.	0.9692	0.9769 (0.0031)	38.0 (16.2)	83.9 (22.8)
Dani Cathedral	Logarithm	0.3999	0.9704 (0.0099)	76.3 (27.8)	2835.3 (1079.6)
384 × 512	Durand/Dorsey	0.4186	0.9703 (0.0011)	62.9 (17.3)	— (—)
	Mantiuk et al.	0.4616	0.9700 (0.0100)	79.0 (24.4)	293.0 (46.9)
Kitchen	Logarithm	0.3651	0.9714 (0.0002)	52.1 (12.0)	1756.7 (51.6)
342 × 512	Durand/Dorsey	0.3804	0.9715 (0.0160)	73.3 (22.8)	6760.6 (2120.2)
	Mantiuk et al.	0.8896	0.9712 (0.0033)	56.9 (20.9)	147.4 (19.1)
Memorial Church	Logarithm	0.4442	0.9795 (0.0041)	63.0 (20.5)	1716.9 (65.7)
340 × 512	Durand/Dorsey	0.4520	0.9804 (0.0046)	63.7 (24.6)	— (—)
	Mantiuk et al.	0.9086	0.9798 (0.0015)	43.5 (16.3)	116.0 (11.3)
Women	Logarithm	0.4151	0.9806 (0.0074)	64.0 (26.1)	1015.0 (336.7)
342 × 512	Durand/Dorsey	0.4135	0.9809 (0.0077)	96.2 (27.1)	5283.8 (1771.6)
	Mantiuk et al.	0.5189	0.9808 (0.0120)	67.4 (41.6)	299.6 (61.0)
Desk 1	Logarithm	0.3881	0.9801 (0.0002)	83.9 (8.4)	857.7 (7.9)
512 × 384	Durand/Dorsey	0.4256	0.9800 (0.0028)	75.1 (18.0)	2079.3 (431.1)
	Mantiuk et al.	0.7600	0.9800 (0.0067)	56.2 (25.0)	192.9 (31.4)
Desk 2	Logarithm	0.3765	0.9654 (0.0045)	54.0 (17.8)	800.8 (29.9)
512 × 384	Durand/Dorsey	0.4031	0.9655 (0.0083)	76.4 (47.9)	1088.4 (236.4)
	Mantiuk et al.	0.8178	0.9652 (0.0069)	51.5 (27.0)	102.6 (14.1)
Display1000	Logarithm	0.4004	0.9649 (0.0038)	59.6 (31.0)	2810.3 (1033.1)
512 × 384	Durand/Dorsey	0.4220	0.9648 (0.0070)	99.3 (36.1)	6983.4 (—)
	Mantiuk et al.	0.7236	0.9649 (0.0059)	63.2 (25.3)	985.1 (351.5)
Belgium House	Logarithm	0.4096	0.9778 (0.0005)	73.0 (14.3)	707.7 (19.4)
512 × 384	Durand/Dorsey	0.4186	0.9777 (0.0030)	91.8 (20.2)	5986.0 (1400.7)
	Mantiuk et al.	0.8552	0.9774 (0.0052)	47.8 (18.8)	141.3 (25.9)
Woods	Logarithm	0.0708	0.9844 (0.0056)	75.3 (24.3)	— (—)
512 × 340	Durand/Dorsey	0.3541	0.9845 (0.0069)	77.3 (30.3)	— (—)
	Mantiuk et al.	0.4957	0.9843 (0.0024)	54.7 (14.7)	1078.2 (356.3)
Lawn	Logarithm	0.4434	0.9864 (0.0069)	68.1 (17.8)	1318.3 (317.5)
512 × 381	Durand/Dorsey	0.4585	0.9861 (0.0043)	89.2 (35.9)	4276.2 (1558.4)
	Mantiuk et al.	0.9689	0.9861 (0.0019)	39.3 (19.1)	278.4 (69.2)
Bristol Bridge	Logarithm	0.3968	0.9843 (0.0128)	52.0 (27.0)	746.5 (191.7)
512 × 384	Durand/Dorsey	0.3891	0.9845 (0.0001)	83.8 (23.0)	— (—)
	Mantiuk et al.	0.9520	0.9841 (0.0071)	54.3 (25.7)	887.5 (364.3)
Office	Logarithm	0.4477	0.9729 (0.0012)	54.3 (18.7)	368.2 (4.2)
512 × 340	Durand/Dorsey	0.4245	0.9738 (0.0020)	71.2 (24.5)	— (—)
	Mantiuk et al.	0.9503	0.9742 (0.0048)	44.1 (16.7)	158.2 (36.1)
Vine Sunset	Logarithm	0.4246	0.9636 (0.0051)	50.6 (21.3)	613.3 (123.6)
512 × 345	Durand/Dorsey	0.4440	0.9639 (0.0007)	89.7 (27.8)	— (—)
	Mantiuk et al.	0.4617	0.9641 (0.0003)	73.9 (12.6)	1498.9 (56.8)
Wreathbu	Logarithm	0.4910	0.9652 (0.0010)	58.8 (18.0)	1204.1 (354.4)
512 × 384	Durand/Dorsey	0.4365	0.9655 (0.0010)	88.6 (34.0)	3814.1 (716.4)
	Mantiuk et al.	0.7982	0.9660 (0.0010)	49.0 (21.0)	406.1 (82.1)

It can be seen from the table that compared with the starting points, the EA can significantly improve TMQI-II scores. Final scores are within a narrow range, both across test images and across starting points. Standard deviations are such that the empirical coefficient of variation of TMQI-II scores rarely exceeds 0.01. Running times for the EA range from well under a minute to no more than three minutes in the longest of the 528 runs. In comparison, the algorithm of Ma et al. [9] requires significantly more time to generate tone mapped images with equivalent TMQI-II scores, and in a number of instances remains unsuccessful even after two hours (where runs are terminated)

It is worth noting that Ma et al. [9] report TMQI-II scores in excess of most of the values attained by the EA and reported in Table 2. The EA operates in a low-dimensional search space that may implicitly limit the quality of the tone mapped images that can be achieved by parameter optimization. However, TMQI-II scores in excess of the values achieved by the EA are of little use as they require that intensity values be represented with more than eight bits per pixel and colour channel. As the tone mapped images will almost always be stored using image file formats with limited intensity resolution, any further improvements in TMQI-II scores are likely to disappear due to quantization noise.

Clearly, the running times of both the EA and the algorithm by Ma et al. are impacted by the size of the images being processed. However, the approach relying on the generic TMO and the EA for optimization admits a simple technique for reducing running time: rather than performing the optimization on potentially sizable images, shrink the images before applying the EA to obtain parameter settings for the generic TMO. Then use the parameter settings obtained on the small images to tone-map the full-sized images. Two examples of results from this approach can be found in Figure 4. The images on the left have been obtained through optimization using the full-sized images of size 1024×768. The optimization took 344 seconds for the "Bristol Bridge" image and 200 seconds for the "Belgium House" image. The images on the right are the result of computing parameter settings on images of size 256×192 and applying those settings to the full-sized images. The TMQI-II scores somewhat decrease in both cases, but the results are visually nearly indistinguishable, and optimization in the latter case is accomplished in 11 and 9 seconds, respectively. A straightforward technique for reducing running time while not having to contend with reduced TMQI-II scores is to interleave increasing the size of the images being processed with the running of the EA.

Finally, in addition to the reduced running time, we have found our method to often be preferable to the algorithm by Ma et al. [9] in that it generates images with more consistent appearance across starting points. Figure 5 shows examples where the appearance of the images generated using the algorithm by Ma et al. differs from starting point to starting point and suffers from artifacts, such as over- and under-saturation, while the results generated using the EA look comparatively uniform. Further examples can be found in the complete set of experimental data which is available at `www.cs.dal.ca/~xgao/EAdata.rar`.

TMQI-II: 0.9794　　　　　　　　　　　　　TMQI-II: 0.9759

TMQI-II: 0.9716　　　　　　　　　　　　　TMQI-II: 0.9660

Fig. 4. Comparison between images with parameters of the generic TMO obtained through evolutionary optimization on smaller versions of the images. The images on the left have been obtained from optimization on the full-sized images of size 1024×768. The images on the right are the result of shrinking the images to size 256×192, solving the optimization problem using the EA, and then using the TMO parameter settings obtained on the full-sized images.

Figure 6 shows an example for a deliberately poorly chosen starting point for the search. The starting point is encoded with eight bits per colour channel and pixel, and the dark regions in it are solidly black. The algorithm by Ma et al. is not able to restore the image content in those regions and thus converges to a suboptimal solution while the EA generates a satisfactory solution.

5　Conclusion

To conclude, we have used an EA to solve the tone mapping problem based on maximization of TMQI-II scores. Compared to TMQI-II optimization by interleaving gradient based maximization of structural fidelity with optimization of statistical naturalness, we observe significantly reduced running times. The reduced amount of computational effort is due to performing the optimization in a much lower dimensional parameter space. By distributing the computation of the TMQI-II values of the offspring across multiple cores or obtaining parameter settings for the generic tone mapping operator by (initially) optimizing using
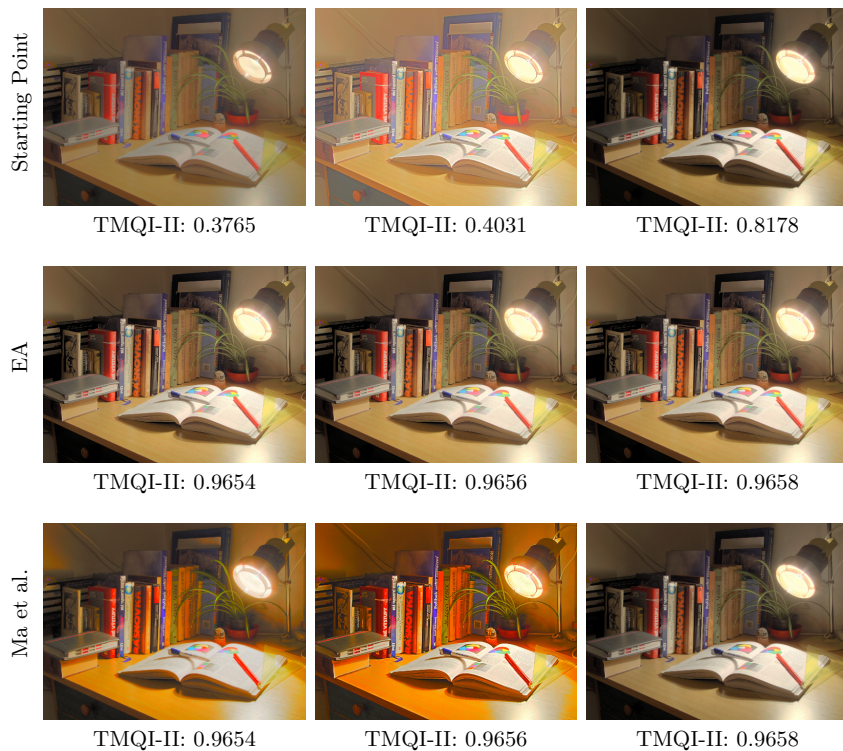
Fig. 5. Comparison of results for HDR image "Desk 2". First row: starting points for the search; second row: images optimized with the EA; third row: images optimized with the algorithm of Ma et al. [9].

reduced-size images, obtaining optimized tone mapped images with the simple EA requires but a few seconds.

In future work, we will consider the suitability of other image quality assessment techniques for tone mapped images as well as other image processing tasks that are commonly performed using gradient based techniques in high-dimensional spaces, but that may conceivably be solved using much lower-dimensional parametric models. Also, we would like to further analyze the influence of strategy selections, and more standard strategies such as CSA-ES will be explored.

References

1. D. V. Arnold. Resampling versus repair in evolution strategies applied to a constrained linear problem. *Evolutionary Computation*, 21(3):389–411, 2013.

Starting Point	EA	Ma et al.
TMQI-II: 0.4429	TMQI-II: 0.9847 Time: 49.63 seconds	TMQI-II: 0.93119 Time: 43203.33 seconds

Fig. 6. Comparison of results for HDR image "Woods" and a poorly chosen starting point. First image: starting point for the search; second image: image optimized with the EA; third image: image optimized with the algorithm of Ma et al. [9].

2. F. Banterle, A. Artusi, K. Debattista, and A. Chalmers. *Advanced High Dynamic Range Imaging: Theory and Practice*. AK Peters (CRC Press), 2011.
3. S. B. Chisholm, D. V. Arnold, and S. Brooks. Tone mapping by interactive evolution. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, 515–522, 2009.
4. F. Durand, and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transaction on Graphics*, 21(3):257-266, 2002.
5. X. Gao, S. Brooks, and D. V. Arnold. Automated parameter tuning for tone mapping using visual saliency. *Computer & Graphics*, 52:171–180, 2015.
6. X. Gao, S. Brooks, and D. V. Arnold. Automatic blended tone mapping through evolutionary optimization. In *Proceedings of the IEEE World Congress on Computational Intelligence*, 3855–3862, 2016.
7. K. Gu, G. Zhai, M. Liu, X. Yang, and W. Zhang. Details preservation inspired blind quality metric of tone mapping methods. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 518–521, 2014.
8. N. Hansen, D. V. Arnold, and A. Auger. Evolution Strategies. In J. Kacprzyk and W. Pedrycz (eds.), *Handbook of Computational Intelligence*, 871–898, Springer, 2015.
9. K. Ma, H. Yeganeh, K. Zeng, and Z. Wang. High dynamic range image compression by optimizing tone mapped image quality index. *IEEE Transactions on Image Processing*, 24(10):3086–3097, 2015.
10. R. Mantiuk, and H.-P. Seidel. Modeling a generic tone-mapping operator. *Computer Graphics Forum*, 27(2):699–708, 2008.
11. R. Mantiuk, S. Daly, and L. Kerofsky. Display adaptive tone mapping. *ACM Transactions on Graphics*, 27(3):68:1–68:10, 2008.
12. H. Z. Nafchi, A. Shahkolael, R. F. Moghaddam and C. Mohamed. FSITM: A Feature Similarity Index For Tone-Mapped Images. *IEEE Signal Processing Letters*, 22(8):1026–1029, 2015.
13. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
14. H. Yeganeh and Z. Wang. Objective quality assessment of tone-mapped images. *IEEE Transactions on Image Processing*, 22(2):657–667, 2013.

LIDeOGraM : An interactive evolutionary modelling tool.

Thomas Chabin, Marc Barnabé, Nadia Boukhelifa, Fernanda Fonseca, Alberto Tonda, Hélène Velly, Benjamin Lemaitre, Nathalie Perrot, Evelyne Lutton

UMR 782 GMPA, Agroparistech, INRA, Université Paris-Saclay
1 Av. Lucien Brétignères, 78850 Thiverval-Grignon, FRANCE

Abstract. Building complex models from available data is a challenge in many domains, and in particular in food science. Numerical data are often not enough structured, or simply not enough to elucidate complex structures: human choices have thus a major impact at various levels (data and model structuration, choice of representative scales, parameter ranges, uncertainty assessment and management, expert knowledge). LIDeOGraM is an interactive modelling framework adapted to cases where numerical data and expert knowledge have to be combined for building an efficient model. Exploiting both stand-alone evolutionary search and visual interaction with the user, the proposed methodology aims at obtaining an accurate global model for the system, balancing expert knowledge with information automatically extracted from available data. The presented framework is tested on a real-world case study from food science: the production and stabilisation of lactic acid bacteria, which has several important practical applications, ranging from assessing the efficacy of new industrial methods, to proposing alternative sustainable systems of food production.

Keywords: Complex systems, Lactic acid bacteria, Interactive modelling, Symbolic regression, Living food system

1   Introduction

Agri-food processes can be regarded as complex systems, as they are characterised by uncertain and intricate interaction effects between physical, chemical, and biological components [10,11].

In this context, modelling techniques drawn from complexity science prove especially advantageous for dealing with the co-existing multiscale inter-dependencies, uncertainty, partial knowledge and sparse experimental data. Models are a mean to gather available knowledge about a process. Additionally model exploration allow a better understanding via simulated experiments. Expert knowledge yields additional, precious information [14,1]. Indeed, building a model in these conditions is a complex optimisation task: learning from data sets, dealing

190

with sparsity of data, possible overfitting issues, and complexity of the models. At the same time expert knowledge can drastically modify the shape of the search space, the relative impact of some data, or even the optimisation aims.

In this paper, we propose an interactive modelling approach based on a two-level evolutionary optimisation scheme, *local* and *global*. *Local* corresponds to local possible dependencies between variables, while *global* corresponds to a model that represents the system as a whole. Users can interact with the constructed models via a graphical user interface, run various optimisation steps, revisit optimisation results, restart the process, add constraints, and take decisions.

The system and dataset considered in this study concern the full process of bacteria production and stabilisation, with 49 variables measured at 4 different steps (fermentation, concentration, freeze-drying and storage), at 4 different fermentation conditions (22°C and 30°C, evaluated at the beginning of the stationary growth phase and 6 hours later). The considered variables range from transcriptomic data to fatty acid membrane composition, from acidification activity to viability [15].

The paper is organised as follows: Section 2 provides background on complex systems approaches in food science, on symbolic regression and on the modelled process. *LIDeoGraM* is detailed in section 3. Section 4 describes experimental results from a preliminary user evaluation, comments, conclusions and future developments are given in sections 5 and 6.

2 Background

2.1 Food complex systems

A complex system[1] is a collection of multiple processes, entities, or nested subsystems, where global properties emerge as the result of an imbrication of phenomena occurring at different scales. For these systems, there is a need for appropriate descriptions for the underlying mechanisms with high expressiveness and little uncertainty. Building complex system models is essential, but highly difficult; it is usually necessary to have a robust framework, with strong iterative interaction combining computational intensive methods, formal reasoning and experts from different fields. As shown in the rest of the paper, optimisation plays an important part in this context[8].

The specifics of the food domain (uncertainty and variability, heterogeneity of data, coexistence of qualitative and quantitative information, conjunction of different perspectives) raise the focus on another crucial issue, that can be called the *human factor*. Human expertise and decision making are of major importance for a better understanding of food systems, and should thus be integrated into computing approaches[7].

---

[1] Complex Systems Society, see `http://cssociety.org` or `http://www.mathinfo.inra.fr/en/community/complexsystems/presentation` for an introduction to the topic.

2.2 Symbolic regression

Symbolic regression, based on a genetic programming approach, is a technique able to extract free-form equations that expose correlations in a given experimental dataset. The original idea is presented in [6], and the technique has been applied to a vast array of real-world problems [2,12,5]. Candidate solutions are encoded as trees, with terminal nodes encoding constants and variables of the problem, whereas intermediate nodes correspond to mathematical functions such as $\{+, -, *, /, ...\}$. In most implementations, the fitness function is proportional to the absolute or squared error between experimental data, with parsimony corrections to reward simpler solutions. *Eureqa Formulize*[2] is one of the most notable symbolic regression tools. Eureqa deals with the issue of overfitting by returning a Pareto front of candidate solutions, each one presenting a compromise between fitting and complexity [13], leaving the final choice to the user.

2.3 Production and stabilisation process of lactic acid bacteria

Concentrates of Lactic Acid Bacteria (LAB) are widely used in food applications, ranging from yoghurt and cheese to fermented meat, from vegetables to fruit beverages. In industry, these bacterial starters are produced in large quantities by fermentation and must therefore undergo a preservation procedure, called stabilisation. Both production and stabilisation processes aim at protecting the quality of bacterial starters, characterised by their cell viability and their acidification activity. The full process involves numerous control parameters across its different steps (Fermentation, Concentration, Freeze-Drying and Storage) [3]. Moreover, the process is a multi-scale system. Indeed, the quality of the starters can be explained by the cellular composition in fatty acid which is in turn explained by the genomic expression in each cell. This latter only depends on the parameters of fermentation and concentration.

3 Proposed approach

Experts in the process of production and stabilisation of lactic acid bacteria have numerous questions about how a given bacteria strain draws its resistance to the process. Different mathematical tools, including mathematical formulas are generally used to help them to answer these questions with more or less success. Finding reliable formulas linking the different variables of such a system is indeed challenging [9]. In biological data, a high level of variability is often encountered for repetitions of a given experimental condition. Moreover, experiments are usually time-consuming and expensive – only a few experiments are thus performed – which makes the task of characterising the existing variability difficult.

LIDeOGraM (*Life-based Interactive Development Of Graphical Models*) tries answering these challenges with an original approach of semi-automatic modelling.

---

[2] http://www.nutonian.com/

The goal of LIDeOGraM is to help experts build a global model of their complex process by characterising each non-input variable by a mathematical formula that depends on the other variables in the system. Finding the right equation in a context with high variability in the dataset is an ambitious task. Indeed, it is easy to come up with over-fitted equations that perfectly model a dataset including its noise. However, over-fitted equations do not generalise well.

In order to rule out over-fitted equations, a solution is to involve experts in the course of the modelling process. The expectation is that they will be able, thanks to their knowledge of the process, to identify over-fitted or under-fitted equations.

Symbolic regression using a Pareto-like approach such as the one implemented in Eureqa, constitutes a compelling approach to take advantage of the expert's insight. Indeed, by providing a set of formulas according to different compromises between fitness and complexity, the approach allows the experts to filter out incoherent equations or even designate the most suitable one.

Therefore, as a first optimisation step, LIDeOGraM uses Eureqa runs on each variable, in order to get a set of candidate equations. For automatic learning purposes, the dataset is separated into training and test sets. Moreover, some constraints in the search are defined beforehand by the user, using the interface presented in Figure 2. This tool allows attributing each variable to a given class, and defining authorized links between them. This means that only the variables from a parent class can be used in the equations for determining the variables of the child class. This also means that dependencies will be searched only with variables of other classes and that no intra-class dependencies will be considered. This structure of classes can be used to distinguish between scales and steps in the studied process. Variables measured at a macro-scale, like the viability of the population of bacteria could, for example, be only explained with variables from a micro-scale, such as the composition in fatty acids. Similarly variables measured in a given step could only be explained by variables from previous steps.

A qualitative view of these results is presented to the user in the form of a graphical network (See Figure 1). The goal of this display is to help the user focus on the critical variables, i.e. where expert feedback is most needed. In this prospect, variables are represented as nodes in the graph. The colour of the nodes depends on its attributed class. A link between two variables shows that the parent node is used at least once in the set of equations attributed to the child node. The colour of a link represents the mean value of a given criteria on the equations involving the parent node in the child node. The criteria can be chosen by the user as the fitness of the equation or its complexity.

Additionally, since the displayed graphical network can have a lot of links, making the network hard to read, a slider allows the filtering of links based on their level of importance. The importance of a link is defined by the number of equations in the child node that use the parent node, divided by the total number of equations in the child node.
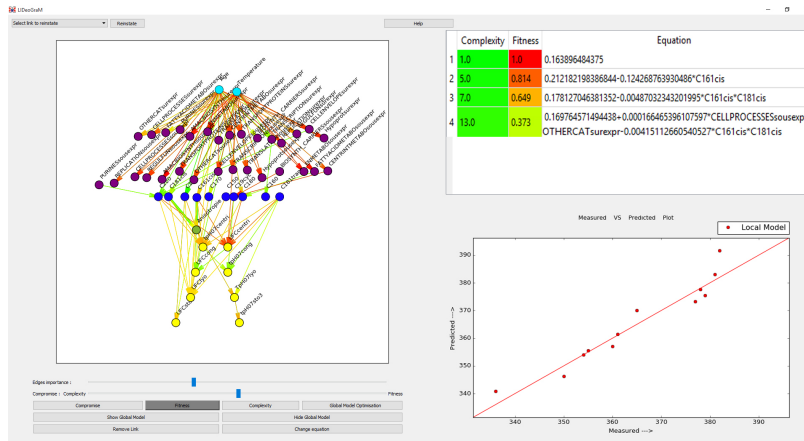
Fig. 1. Screenshot of LIDeOGraM. The left side shows a graphical model representing the mean fitness of the local models obtained by symbolic regression. The top-right part is the list of equations proposed by Eureqa for the selected node, and the bottom-right part shows a plot of the measured versus predicted data associated to the selected equation.

By clicking on a node, the equations found by Eureqa are displayed to the user on the top-right side (See Figure 1). Similarly, a click on an equation provides a plot of the experimental measures versus what is predicted by the corresponding equation. The user can then interact with the system by deleting an equation, deleting a link between a parent node and a child node (i.e. all equations using the parent node in the child node are deleted), or deleting a variable (i.e. all equations using the deleted variable are deleted). After this, few or no equations may remain for some nodes, the user can choose to restart a symbolic regression on any node.

The user can iterate the process for as long as desired: add or suppress constraints, restart symbolic regression on any node. Once the user is satisfied with local models, a global model can be built.

For the global model, one equation only is kept for each node. However, choosing the most reliable ones is a challenging task. Contrarily to the local models, where the experimental measures are used to predict a variable, in a global model the value predicted by an equation depends on the value predicted for the variables used in that equation. For this reason, each choice of equation for a given variable will influence the quality of the prediction of other equations that use the variable. To tackle this challenge, evolutionary optimisation is used to build a global model.

The evolutionary algorithm has the following features. Its fitness function, to be minimised for the global model, is the mean of the fitness calculated on each non-input nodes. The fitness function of a single node computes a value based on the Pearson correlation coefficient of the measured versus predicted

194

data. Such a fitness function does not take into account the complexity of the equations. The reason behind this choice is that over-fitted equations will be naturally discarded during the learning of the global model, as they will likely create noise for their children variables.

After the evolutionary optimisation process, remaining incoherence in the choice of equations for the global model can still be edited by the user. For this purpose, a qualitative view of the global model is displayed as a new visualisation (See Figure 4). Contrarily to the graphical network displayed for the local models, here only the variables used in the chosen equation of a child node are considered as parents of the node. A link from a parent node to a child node in this graph represents the fact that the chosen equation for the child node contain the parent variable, its colour depends on the fitness of the child node in the global model. Green represents a good fitting, and red a bad one. The goal of this view is to help the user focus on the nodes with bad predictions. A user change on the selected equation of a node impacts the predicted value of its children nodes. The graphical model is therefore automatically updated, and the update shows the consequence of the change in term of fitness on the other nodes of the graph.

After this step, the user has the possibility to go back to local view and make changes before restarting a new global optimisation. A global model is thus iteratively built via user interaction, local and global optimisation.

4 Experimental results

4.1 The dataset

The case study is based on the work of H. Velly et al. [15][16] about the resistance of *Lactococcus lactis* subsp. *lactis TOMSC161* to freeze-drying. This bacteria is used in the production of *Tomme de Savoie*, a french cheese, for its interesting texturing and acidification properties, but exhibits a high sensitivity to freeze-drying. The resistance of the bacteria is studied for 4 different conditions of fermentation: 22°C and 30°C, evaluated at the beginning of the stationary growth phase and 6 hours later.

The dataset featured 12 data points, with 3 biological repetitions of each experimental condition. The dataset features 2 input variables, the temperature of fermentation and the time at which the fermentation is stopped and 49 variables measured at 4 different steps (fermentation, concentration, freeze-drying and storage) for 3 biological scales (Genomic, Cellular and Population).

4.2 Search with Eureqa

The 51 variables described above are first separated into 9 classes of variables: **Inputs**, **Genomic** for **overexpressed** and **underexpressed** genes, **Cellular**, **Anisotropy**, **Population** at the end of the **Concentration** step, **Population** at the end of the **Congelation**, **Population** at the end of the **Drying** step and the **Population** after 3 months of **Storage**. Each class of variables can only be
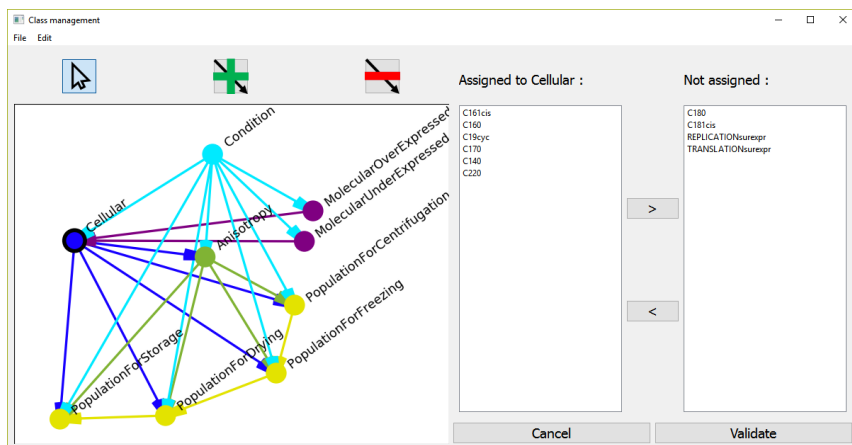
Fig. 2. Screenshot of the interface allowing the users to choose the authorised links between the defined classes. A link between two classes means that all variables associated to the parent class can be used in the equations for all variables associated to the child class. The displayed graph represents the selected contraints chosed for this experiment.

explained by user specified classes. The possible links between classes are shown in Figure 2.

The dataset is also separated into a training dataset (66%) made of two out of the three repetitions for each experimental condition, and into a test dataset (33%) with the remaining repetition.

The authorised mathematical operators for the Symbolic regression using Eureqa are: Constants, Input variables, Addition, Subtraction, Multiplication, Division, Exponential, and the Natural logarithm. For each non-input variables, 3 minutes of computation were allowed on an Intel(R) Core(TM) i7-4790 CPU. A total of 232 equations were obtained for the variables.

4.3 Optimisation of the Global Model

A ($\mu + \lambda$)-evolutionary algorithm was taken from the Python DEAP package[4] to optimise the global model. The genome of a candidate global model is a string of integers, of size equal to the number of variables in the process. Each gene is associated to a variable, and can assume a value between 1 and the number of equations available to describe that variable, thus representing an index for a candidate equation in that node.

The parameters of the evolutionary optimization algorithm used for the global model are reported in Table 1.

The mutation function takes complexity information into account. It has been experimentally shown to be more efficient than a mutation which randomly picks an equation in the list of candidate equations.

Table 1. Parameters of the evolutionary algorithm used during the optimization process for the global model.

μ	100
λ	80
Number of generations	100
Probability of crossover	0.8
Probability of mutation	0.2
Selection	Tournament of size 2
Crossover function	Uniform
Mutation function	With a probability 0.05 for each gene, change the selected equation to the previous or the next one by order of complexity.

The graphical model associated to one of the optimisation runs is shown in Figure 4.

The creation of a global model does not involve only an automatic optimisation, but also requires experts knowledge, obtained via interaction with the software. An informal evaluation of the software was performed by a researcher with 20years of expertise in the bacteria freeze-drying process. For this purpose the expert gave us feedback on the proposed local models, presented in Figure 1, during a 20 minutes exploration. The expert chose to remove 5 equations. Some equations were removed for using both variables from the Cellular scale with the Anisotropy variable. The reason is that the Anisotropy is an emergent property of the fatty acid composition at the Cellular scale and it is not straightforward to make sense of such an equation. Similarly, an equation using the viability at both the centrifugation step and the drying step was removed. The reason is that the viability at the centrifugation step is used to predict the viability at the drying step, therefore, it is hard to understand the necessity of using both steps since the obtained data values are dependent. The expert also chose to remove 2 nodes, after observing that those nodes were used repeatedly in the equations of many nodes. Indeed, due to their insignificant measured quantities, they were not expected to be important variables, rather, they were deemed useful for refining some models. Therefore, they were considered as creators of overfitted equations. The deletion of those two variables removed 14 more equations. With such major deletions, some variables were left with only a few equations, therefore, the expert chose to restart a symbolic regression on 3 nodes, obtaining 12 new equations in total. To reveal the contribution of the expert, the global model optimisation was performed 10 times using expertise, and 10 times without. The fitness evolution of these runs are shown in Figure 3. To obtain an accurate comparison of the models, the fitness computed for optimisation without the expertise did not take into account the two removed nodes. The global models obtained using expertise have a median fitness of 0.787 with a standard deviation of 0.010 whereas the global models obtained without expertise have a median fitness of 0.801 with a standard deviation of 0.013. The expert was
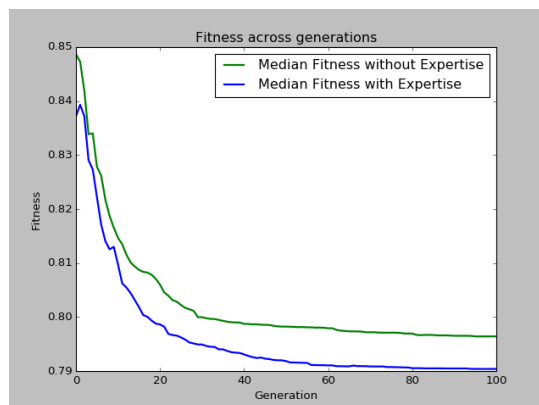
Fig. 3. Comparison of the evolution of the minimum fitness across generations for 10 runs, with and without the expert's contribution.

asked to provide feedback for the last step of the modelling process in which one of the global model obtained was submitted to his expertise. The results were explored during 10 minutes, and the equations for three node were changed. Two of the modified equations were indeed overfitted, and the last one was an underfitted. For example, one of the equations selected by LIDeOGraM, at the cellular scale, the variable C18:0 was defined as being equal to the duration of fermentation, which seemed a rather drastic choice. The expert chose to select a more reasonable equation presenting a linear dependency involving the duration of fermentation. The obtained graphical model is presented in Figure 4. The fitness of the final global model was slightly degraded, changing from a fitness of 0.789 to a fitness of 0.801, but the produced model is able to better reflect the underlying reality of the process.

5   Discussion

We proposed a time-saving modelling tool for the experts, allowing them to design a better global model of their process by a semi-interactive approach. Figure 3 shows that the resulting models are "better", not only according to the expert requirements, but also with respect to the numerical data (faster and better convergence). Above all, this method offers tools for domain experts to design and test different hypothesis, using different datasets and class constraints. The complexity of the modelled process and the scarcity of the dataset is taken into account by allowing the expert to interact with the results all along the optimisation process. The expert who tested the software mentioned it is easy to question her hypothesis, to keep an open-mind and to find new mechanisms.

Nevertheless, the approach has some drawbacks. Since the predictions of each node are propagated, only the inputs are indeed used in a global model to determine every other variables. Knowing this, a natural question should be why
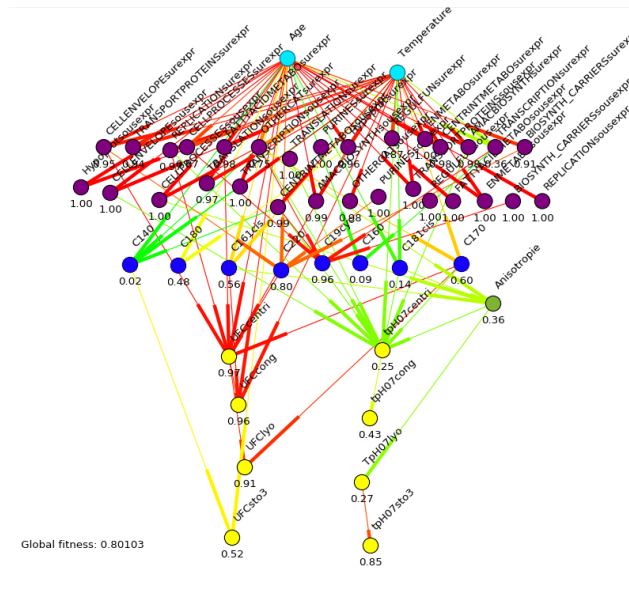
Fig. 4. Graphical model representing a global model. The Global fitness of the model is indicated at the bottom-left. The fitness value of each node is indicated under them.

all variables are not directly linked to the inputs, and why intermediary variables exist. A reason behind this is that the goal is not only to get the best prediction out of every variable, but also to help the expert understand mathematically the existing dependencies between variables and the multi-scale / multi-step organisation of the process.

Besides this, we should mention that the current results remain not fully satisfying for the genomic scale. The hypothesis made by the expert was that the genomic scale is only explained by the conditions of fermentation (temperature and time at which the fermentation is stopped). This hypothesis needs additional verifications, as the relation with genomic scale might not be so straightforward. Other variables, not measured during the experiments, may be involved. A more refined work on the expressed genes and their classification is necessary. A future study will explore in more details different hypothesis about the possible links between classes of variables.

Another point is that some expert-defined variables used in the literature are the sum of some measured variables. For example, the Saturated Fatty Acids variable, is defined as the sum of 6 variables at the cellular scale. New tools could be designed to incorporate this kind of knowledge and allow the user to create "hierarchic" variables. Such variable would allow taking into account different levels of details in the modeling process and would allow to easily test various hypotheses for the computation of variables at the genomic scale.

Finally, creating a global model from local hypothesis means the creation of a global hypothesis. With some local hypothesis being thought of as equally plausible for a node, a lot of equally plausible global hypotheses could be constructed. Moreover, since the proposed local models of different nodes are not equivalent, a notion of confidence could be associated to each node of a global hypothesis, and each node would be associated with a set of equally probable local models.

6 Conclusions

In this paper, we proposed a new approach to semi-automatic modelling allowing users to design complex models for multi-scale and multi-steps processes. Using expert's knowledge integrated during the optimisation process, the proposed approach is able to tackle challenges such as scarcity in a dataset, high dimensionality and high variability. According to experts guidelines, a set of local models are proposed for each variable, using symbolic regression. The local models form a Pareto front of candidate solutions to compromise between model fitness and complexity. These local models are then used to automatically construct a global model where each variable is defined by a given equation from the local models. In a global model, the multi-scales multi-steps process is taken into account by classifying the variables into different classes and by forwarding the predicted value of a variable to equations that use this variable to predict other ones. An expert is able to contribute to the automatic design of a global model in many ways, by acting on the proposed local models and by correcting the global model. The approach was applied to the production and stabilisation process of lactic acid bacteria. The contribution of the expert was shown to be useful to provide a more accurate global model. Future improvements will involve new tools to create and manage hierarchic variables and associate a level of confidence for each variable. These improvements will allow producing a full and efficient study of the production and stabilisation process of lactic acid bacteria.

Aknowledgements

References

1. Allais, I., Perrot, N., Curt, C., Trystram, G.: Modelling the operator know-how to control sensory quality in traditional processes. Journal of food engineering 83(2), 156–166 (2007)
2. Babovic, V., Keijzer, M.: An evolutionary approach to knowledge induction: Genetic programming in hydraulic engineering. In: Proceedings of the World Water & Environmental Resources Congress

3. Champagne, C., Gardner, N., Brochu, E., Beaulieu, Y.: freeze-drying of lactic acid bacteria. a review. Canadian Institute of Food Science and Technology journal: Journal de l'Institut canadien de science et technologie alimentaire (1991)

4. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research 13, 2171–2175 (jul 2012)

5. Gaucel, S., Keijzer, M., Lutton, E., Tonda, A.: Learning dynamical systems using standard symbolic regression. In: European Conference on Genetic Programming. pp. 25–36. Springer (2014)

6. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press (1992)

7. Lutton, E., Perrot, N.: Complex systems in food science: Human factor issues. 6th International Symposium on Delivery of Functionality in Complex Food Systems Physically-Inspired Approaches from the Nanoscale to the Microscale. (2015)

8. Lutton, E., Perrot, N., Tonda, A.: Evolutionary Algorithms for Food Science and Technology. John Wiley & Sons (2016)

9. Passot, S., Fonseca, F., Cenard, S., Douania, I., Trelea, I.C.: Quality degradation of lactic acid bacteria during the freeze drying process: Experimental study and mathematical modelling (2011)

10. Perrot, N., De Vries, H., Lutton, E., Van Mil, H.G., Donner, M., Tonda, A., Martin, S., Alvarez, I., Bourgine, P., Van Der Linden, E., et al.: Some remarks on computational approaches towards sustainable complex agri-food systems. Trends in Food Science & Technology 48, 88–101 (2016)

11. Perrot, N., Trelea, I.C., Baudrit, C., Trystram, G., Bourgine, P.: Modelling and analysis of complex food systems: state of the art and new trends. Trends in Food Science & Technology 22(6), 304–314 (2011)

12. Pickardt, C.W., Hildebrandt, T., Branke, J., Heger, J., Scholz-Reiter, B.: Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. International Journal of Production Economics 145(1), 67–77 (2013)

13. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science 324(5923), 81–85 (2009)

14. Sicard, M., Baudrit, C., Leclerc-Perlat, M., Wuillemin, P.H., Perrot, N.: Expert knowledge integration to model complex food processes. application on the camembert cheese ripening process. Expert Systems with Applications 38(9), 11804–11812 (2011)

15. Velly, H., Bouix, M., Passot, S., Penicaud, C., Beinsteiner, H., Ghorbal, S., Lieben, P., Fonseca, F.: Cyclopropanation of unsaturated fatty acids and membrane rigidification improve the freeze-drying resistance of lactococcus lactis subsp. lactis tomsc161. Applied microbiology and biotechnology 99(2), 907–918 (2015)

16. Velly, H., Fonseca, F., Passot, S., Delacroix-Buchet, A., Bouix, M.: Cell growth and resistance of lactococcus lactis subsp. lactis tomsc161 following freezing, drying and freeze-dried storage are differentially affected by fermentation conditions. Journal of applied microbiology 117(3), 729–740 (2014)

Automatic configuration of `GCC` using irace

Leslie Pérez Cáceres, Federico Pagnozzi, Alberto Franzin, and Thomas Stützle

IRIDIA, CoDE
leslie.perez.caceres@ulb.ac.be, federico.pagnozzi@ulb.ac.be,
alberto.franzin@ulb.ac.be, stuetzle@ulb.ac.be
Université Libre de Bruxelles,
Brussels, Belgium

Abstract. Automatic algorithm configuration techniques have proved to be successful in finding performance-optimizing parameter settings of many search-based decision and optimization algorithms. A recurrent, important step in software development is the compilation of source code written in some programming language into machine-executable code. The generation of performance-optimized machine code itself is a difficult task that can be parametrized in many different possible ways. While modern compilers usually offer different levels of optimization as possible defaults, they have a larger number of other flags and numerical parameters that impact properties of the generated machine-code. While the generation of performance-optimized machine code has received large attention and is dealt with in the research area of auto-tuning, the usage of standard automatic algorithm configuration software has not been explored, even though, as we show in this article, the performance of the compiled code has significant stochasticity, just as standard optimization algorithms. As a practical case study, we consider the configuration of the well-known GNU compiler collection (`GCC`) for minimizing the run-time of machine code for various heuristic search methods. Our experimental results show that, depending on the specific code to be optimized, improvements of up to 40% of execution time when compared to the `-O2` and `-O3` optimization flags is possible.

Keywords: irace, automatic configuration, parameter tuning, gcc

1 Introduction

The performance of computational procedures such as optimisation algorithms is commonly a major concern for both developers and users. Algorithm performance depends on several elements related to the characteristics of the algorithm itself, the particular problem being solved and the environment in which the execution will be performed. Normally, algorithms expose parameters that allow the user to adjust their behaviour to the problem solved and the execution circumstances. Generally, parameter settings or *configurations* have a strong impact on the performance of algorithms and, consequently, finding high-performing configurations is essential to reach peak algorithm performance. The process of finding good configurations can be challenging, requiring significant expertise, consuming large amounts of time and be computationally expensive given the large

number of parameters that some algorithms have. To alleviate this situation, a number of automatic algorithm configuration tools, also called *configurators*, have been proposed in the literature. Examples include ParamILS [15], SMAC [14], GGA [2] and irace [17]. These tools have shown to be able to obtain very good parameter settings when configuring different types of algorithms, in some cases strongly improving the performance obtained by expert-defined default parameter settings [12,24]. Configurators aim at efficiently using the available computational resources to search the parameter space for high-performance parameter settings. The use of such tools can enable developers to test and make available more design features, and provides a simple and general approach to obtain the best performance of an algorithm.

Very often the configuration of optimisation algorithms is limited to their parameter settings, overlooking the code compilation as an element that can affect algorithm performance. On the contrary, in contexts where producing high-performance and portable code is very important, this issue is widely acknowledged. Several projects define application-specific "autotuners" that adjust the program produced to the system on which it will be installed. Examples of such work include ATLAS [25], Spiral [21], FFTW3 [6] and Patus [5]. On the other side, compilers such as GCC [8], provide several optimisation options that aim at improving the quality of the generated executable. Default levels of optimisation that activate different sets of options are defined in GCC using the -Ox settings. These optimisation levels are defined in a general fashion and further improvements are possible depending on the operations performed by the compiled code and the architecture in which the algorithm is executed. In this sense, GCC is like any other computational procedure whose performance can be optimised by setting its parameters. Selecting the optimisation options of GCC, or any other compiler, to obtain the best executable performance is itself an algorithm configuration problem, which has been tackled by methods such as OpenTuner [1] and COLE [13], or in more specific approaches [4], [7] and [20]. Consequently, when configuring optimisation algorithms, it is possible to optimise the compilation of the code as part of the algorithm configuration process.

The irace package [17] is an automatic algorithm configurator that provides an implementation of iterated F-race [3] and other approaches to automatic configuration. Irace is freely available as an R package, it provides several options to adjust the configuration process (e.g. parallel execution) and it does not require knowledge of R or about the inner workings of irace itself. Irace has been widely applied in the literature and is a state-of-the-art algorithm configurator. In this work, we exploit irace to configure the optimisation options of GCC for minimising the execution time on six different benchmarks that execute different optimisation algorithms. This work shows that (i) the running time of optimisation algorithms can be further improved by configuring the compilation options, (ii) the best options depend on the benchmark and the machine used, and (iii) irace is a suitable configurator to configure the compilation process. Beyond these contributions, irace can be used to improve the execution time of any other compiled code and we will make our procedure, which is tested here on the most recent stable version of GCC, version 7.1, available for other users.

The remainder of this article is organised as follows. First, section 2 describes the algorithm configuration problem, gives details of the configuration process

performed by irace and discusses related work. The details of the compilation configuration benchmarks used are provided in Section 3 and Section 4 gives an analysis of the base performance of the compilation of GCC for the different benchmarks. In Section 5, we provide the experimental results obtained by irace when configuring the compilation options of GCC and we analyse them. Finally, we discuss future work and conclude in Section 6.

2 Automatic algorithm configuration

The configuration of algorithms, is the task of finding a set of algorithm parameter values, also called *algorithm configuration*, that exhibit good empirical performance for a particular class of problem instances. Algorithm configuration can be defined as an optimization problem over a parameter search space, which has the goal of identifying parameter settings that maximize algorithm performance. In general, an algorithm configuration scenario defines: a *parameter search space* consisting of the algorithm parameters defined as variables and their domains, a set of *training* and *test* instance sets, and a total configuration *budget*. Broadly, parameters can be classified in two types: (i) parameters that indicate the selection of algorithmic components (e.g. the crossover operator for an evolutionary algorithm, or the branching strategy for an exact algorithm), and (ii) parameters that control the behavior of algorithmic components (e.g. the length of a tabu list, or the size of a perturbation) and whose domain commonly correspond to numbers of a discrete or continuous domain. The type of parameters are commonly best represented by *categorical* or *ordinal* variables, while the former as *numerical* ones. Additionally, parameters can have *conditional* relations that is, their use depends on the value of other parameters (e.g. the use of the tabu list length parameter is conditional to the selection of tabu search as local search). The *homogeneity* of a scenario indicates the degree of consistency of the relative performance of configurations in the parameter space across the instance set. Highly homogeneous scenarios, have configurations that are consistently good (or bad) for all problem instances, while in heterogeneous scenarios certain configurations perform best in a subset of the instances and poorly in other instances. An algorithm can be configured for optimizing different performance measures e.g. solution quality, running time, SAT count, etc. The evaluation of the performance of a configuration is commonly defined as the aggregation of the selected performance measure over the instance set (commonly the mean or median). Given that optimization algorithms are often stochastic, the real performance of configurations can be only estimated and several repetitions are required in order to have a precise estimation.

The irace package [17] is an algorithm configurator that implements configuration procedures based on iterated racing, e.g. iterated F-race [3]. Irace is a general-purpose configurator and it only requires the definition of a configuration scenario as described above. The supported parameter types are *categorical*, *integer*, *real* and *ordered* (a categorical parameter that defines a precedence of values in its domain). Irace iteratively applies a racing procedure in which several configurations are incrementally evaluated on bigger subsets of the training instance set. Statistical tests (the Friedman test by default) are performed to

identify configurations that obtain poor performance. These poor configurations are eliminated from the race and the execution continues with the surviving configurations until the termination criterion of the iteration is met. After each iteration, new configurations are sampled from a probabilistic model that is updated to be centered around the best configurations obtained in the previous iteration (*elites*). This way, irace iteratively converges to high-performing areas of the parameter search space, while increasing the precision of the performance estimation by increasing the number of instances in which the elite configurations are evaluated. For more details about irace we refer to [17].

As already mentioned, the configuration of algorithms is a main concern when developing and applying algorithms. The use of automatic configuration tools not only facilitates the algorithm configuration process but also allows developers and researchers to focus on proposing and improving techniques for a wide scope of scenarios, while delegating the tedious configuration task to specialized tools. Several approaches can be found in the literature to automatically generate code and optimize compilation options. These approaches optimize programs to obtain the best performance in particular architectures, multicore or cluster environments, GPU, etc. Most of these methods are program-specific techniques that apply expert knowledge to implement procedures designed for particular scenarios. Examples of such techniques are ATLAS [25] for linear algebra software, Spiral [21] for digital signal processing algorithms, FFTW3 [6] for discrete Fourier transform computation and Patus [5] for stencil computations. As most computational procedures, compilers have parameters that can be optimised thus, a number of approaches have been developed to configure compiler options. An example of such initiatives is ACOVEA [16], an open-source project that currently is not in development. ACOVEA implemented a genetic algorithm based approach to configure the GCC compiler options to obtain lower execution times. Similarly, Milepost GCC [7] is an open-source project that aims at using machine learning techniques to learn high-performing GCC settings, with the aim of reusing this knowledge to improve the performance of programs in particular architectures. The *Tool for automatic compiler tuning* (TACT) [20] is a genetic-based compiler configurator. TACT supports the configuration for single and multiple objectives by obtaining a pareto-optimal set of configurations.

A general autotuning framework, called OpenTuner, is proposed in [1] and applied to configure the options of GCC. OpenTuner provides a framework where domain-specific configuration procedures can be instantiated, while also offering several general-purpose features to configure computational programs. OpenTuner was used to instantiate a specialized autotuner to configure the optimization options of GCC, the approach obtained considerable speed ups of execution performance in several scenarios. A drawback of OpenTuner, from the perspective of the automatic configuration of optimization algorithms, is that it does not provide an explicit method to handle the stochastic behavior of algorithms or the evaluation of problem instances. The evaluation of the configurations is fully delegated to the user and therefore, the use of problem instances and repetitions in the evaluation must be handled by the user. COLE [13], is a compiler optimizer that implements an evolutionary algorithm based on SPEA2 [27]. COLE is able to optimize the compilation for mutiple objectives (e.g. running time and memory use) by searching pareto-optimal sets of compilation options. In [13],

COLE was used to configure the optimization flags of `GCC` (version 4.1.2), compiling the SPEC CPU2000 benchmarks [11]. The results showed that the default optimization levels of `GCC` could be strongly improved. In this work, we evaluate the use of irace to configure the optimization parameters of `GCC` for optimizing the performance obtained by 6 optimization algorithms. We do not apply any `GCC`-dependent processing mechanism to the evaluation of configurations; in this regard, we apply irace like for any other configuration scenario. We argue that, since the evaluation of the compilation with `GCC` is a stochastic procedure, irace is an adequate method to perform the configuration of the `GCC` optimization options. The experiments show that irace can significantly improve the performance obtained by the tested optimization algorithms without requiring any specific knowledge about the compilation process or the targeted algorithms.

3 Configuration scenarios

The experiments presented in this paper configure the optimization options of `GCC` [8] to compile a set of optimization algorithms benchmarks. We use `GCC` version 7.1, the optimization options considered in the configuration were obtained from the documentation of `GCC`[1]. The total number of `GCC` parameters selected are 367 categorical and integer parameters. Enabling the optimization options in `GCC` requires to select an optimization level, thus, the set of `GCC` parameters includes a parameter to select the optimization level (`-O1`, `-O2` or `-O3`). We define two types of `GCC` configuration scenarios by making two sets of parameters:

- GCC_{flags}: 171 categorical parameters consisting of only options that enable/disable main optimization options.
- $GCC_{flags+num}$: 366 parameters, 173 categorical and 193 integer.

The domains in the `GCC` parameter definition do not provide the upper bound of some numerical parameter domains. In these cases we set as upper bound the default value of the parameter multiplied by a constant (4 in this work). Configurations that generate invalid executables or failed in the compilation were penalized returning a large numerical value to irace. Other types of specific error handling were not implemented. The compilation performed by `GCC` is optimized using the GCC_{flags} and $GCC_{flags+num}$ parameter sets in 6 optimization algorithms benchmarks, resulting in 12 different configuration scenarios. The optimization algorithms were executed with fixed parameter settings and fixed evaluation budgets (e.g. fixed number of iterations or solutions generated) and each benchmark defines a set of training and test instances. The goal of the configuration process is to find `GCC` parameter settings that minimize the execution time of the benchmark algorithms (using the defined fixed settings). The optimization algorithm benchmarks are the following:

ACOTSP: framework of ant colony optimization algorithms [23] for solving the traveling salesman problem (TSP). ACOTSP is implemented in `C`. The

---

[1] The `GCC` optimization options are available at `https://gcc.gnu.org/onlinedocs/gcc-7.1.0/gcc/Optimize-Options.html` and the parameter definition can be obtained in the `params.def` file in the source code of `GCC`.

compilation is configured on a training set of 20 TSP instances and evaluated on 100 TSP instances of sizes 1000 and 1500. We configure two versions of ACOTSP, one that applies 3-opt local search to each tour built (ACOTSP ls3) and a version without local search (ACOTSP ls0).

ILS: iterated local search implementation for solving TSP. ILS is implemented in `C`. The compilation is configured on a training set of 10 TSP instances and evaluated on 50 TSP instances of size 1500.

LKH: state-of-the-art Lin-Kernighan heuristic implementation by K. Helsgaun [9], [10]. LKH is implemented in `C`. The compilation of this algorithm is configured on a training set of 10 TSP instances and evaluated on 50 TSP instances of size 1000.

TS: tabu search implementation for solving the quadratic assignment problem (QAP). TS is implemented in `C`. The compilation is configured on a training set and test set of 50 QAP instances.

EMILI: Iterated greedy algorithm instantiated with the EMILI framework for solving the permutation flowshop problem (PFSP). EMILI is implemented in `C++`. The compilation was configured on a training set of 30 PFSP instances with 20 machines and 50 to 100 jobs and evaluated on 120 PFSP instances with 5 to 20 machines and 20 to 500 jobs.

The execution of irace was given a configuration budget of 10 000 evaluations, the statistical test used was the t-test and the performance of the candidate configurations corresponds to the execution time of the optimization algorithm benchmarks. We have chosen not to provide an initial `GCC` configuration to irace, which is commonly done when good parameter settings are known (for `GCC`, the `-O3` or `-O2` options are possible initial settings). We omit the initial configurations to evaluate the ability of irace to find high-performing configurations without additional information of the parameter space. The main tests were run under Cluster Rocks 6.2, which is based on CentOS 6.2. The machines used were:

- *m1*: 2 AMD Opteron (2.4 GHz), 2 cores, 1 MB cache and 4 GB RAM.
- *m2*: 2 Intel Xeon (2.33 GHz), 4 cores, 6 MB cache and 8 GB RAM.
- *m4*: 2 AMD Opteron (2.1 GHz), 16 cores, 16 MB cache and 64 GB RAM.
- *m5*: 2 Intel Xeon (2.5 GHz), 12 cores, 16 MB cache and 128 GB RAM.

The *m2* machine was used by default to perform the experiments unless specified otherwise. More details about the configuration scenarios are available in the supplementary material provided with this paper [19].

4   `GCC` configuration scenarios analysis

Our premise is that the optimization options of `GCC` can improve greatly the performance of the described benchmarks. In order to prove this, we perform experiments to compare the performance of the benchmarks when compiled by `GCC` with and without optimization options enabled. Table 1 gives the speed up obtained by using the options `-O3`, `-O2` and `-O1` for `GCC`, respectively. We can observe that the speed up obtained by the optimization options is strongly influenced by the benchmark. Nevertheless, all benchmarks improve their performance by using the optimization options. EMILI is the benchmark that shows

Table 1. Speed up of 10 executions of the benchmarks, by setting `GCC` to use `-O3`, `-O2` and `-O1`, compared to `GCC` with no optimization options.

speed up	ACOTSP ls0	ACOTSP ls3	ILS	LKH	TS	EMILI
`-O3`	1.52	1.66	1.72	1.57	3.19	10.31
`-O2`	1.47	1.67	1.68	1.59	2.98	10.54
`-O1`	1.35	1.54	1.66	1.57	2.88	7.20

the biggest improvement in performance by reducing 10 times its running time compared to the one obtained by the executable generated without optimization. This could be related to the fact that the code of EMILI is written in `C++`, which allows more optimization. All the benchmarks obtain their best performance using `-O3` or `-O2`, showing that that those optimization levels, which are commonly advised for compilation, already lead to significant improvements.

The homogeneity of configuration scenarios regarding the problem instances is an important element for the algorithm configuration procedure. Very homogeneous scenarios allow to estimate the performance of a configuration based on less problem instances, more budget can be then used to explore the parameter search space. In terms of compilation, the homogeneity quantifies how consistent is the relative performance of executables obtained by different settings of optimization options across the different instances. In order to investigate the homogeneity of the benchmarks, we sampled uniformly 1 000 random configurations of `GCC` and we evaluate their performance over the training set. We removed the data of the configurations that produced failed compilations or executions. With this data, we calculate the Kendall concordance coefficient (W) [22] considering problem instances as blocks and configurations as groups. W is a normalization of the Friedman statistic and can be interpreted as a measure of how consistent is the ranking of the performance of the configurations over the instances. Table 2 gives the W coefficients for the different scenarios, the higher the number obtained the more homogeneous is the scenario. In addition, we also report the number of not failing configurations and the number of these configurations that obtain significantly better performance compared with `-O3`. If few configurations were used to calculate W, this indicates that the benchmark has many `GCC` settings that produce invalid executables or failed compilations. The number of configurations that are statistically better than `-O3`, gives an indication of how difficult is to optimize the performance for each benchmark. If many configurations showed better performance than `-O3`, good settings will be easy to find in the parameter space. On the contrary, if only few configurations are better than `-O3`, finding good configurations will be more challenging. The values of W indicate that in general, the GCC_{flags} scenario is more homogeneous than the $GCC_{flags+num}$. TS-GCC_{flags} shows to be a perfectly homogeneous scenario, indicating that the compilation could be optimized evaluating configurations in only one or very few instances and the results could be generalized to the rest of them. On the contrary, LKH has the lowest homogeneity evidencing that different instances benefit of different compilation options. This is surprising giving that the LKH scenario uses only one instance size, and therefore such variability between instances was not expected. The $GCC_{flags+num}$ scenarios are less homo-

Table 2. Kendall concordance coefficient W of uniformly sampled configurations of the 12 different `GCC` configuration benchmarks. Numbers in parenthesis indicate the number of configurations used (from the $1\,000$ uniformly generated) to compute W and the number of those configurations that have a significantly better performance compared with the mean performance of 10 executions with `-O3` (comparison performed by paired t-test with significance level 0.05).

W	ACOTSP ls0	ACOTSP ls3	ILS
GCC_{flags}	0.92 (747|387)	0.97 (747|564)	0.97 (426|418)
$GCC_{flags+num}$	0.75 (466| 91)	0.58 (466| 32)	0.64 (318|224)
	LKH	TS	EMILI
GCC_{flags}	0.89 (472| 0)	1.00 (534| 8)	1.00 (332| 0)
$GCC_{flags+num}$	0.79 (352| 0)	0.81 (350| 0)	1.00 (304| 1)

geneous (with the exception of EMILI) indicating that some parameter values might be better for certain instances. This could be related to the instance size or type. These results confirm the differences between the configuration scenarios and therefore the potential benefits configuring their compilation.

Another important aspect of configuration scenarios is the stochastic behavior of algorithms. Algorithms that are strongly affected by stochasticity make the estimation of the performance of configurations more difficult. We explore the stochasticity effects on the evaluation of the performance of the `GCC` compilation using the `-O3` option by calculating a confidence interval based on 20 executions of the benchmarks. Figure 1 gives the confidence intervals of 20 evaluations over the instance test set. The confidence intervals clearly show the different effect of the stochasticity in the measured performance of these two benchmarks. ILS presents high variability of the intra-instance execution times, showing that the stochasticity has a great effect on this scenario. On the contrary, the LKH benchmark has a very low intra-instance variability but shows a high inter-instance variability. This is, again, surprising and indicates that the execution time of some components of this algorithm are affected by other instances features.

We presume that the best settings of the optimization options depend on the characteristics of the operations performed by the code to be compiled, but also of the machine in which the algorithm will be executed. To observe this, we report in Figure 2 the confidence intervals of the speed ups obtained by compiling the benchmarks with `-O3` on different machines, compared with `GCC` without optimization. These results indicate that the improvements in the performance depend on both the machine and the benchmark. For example, while ACOTSP ls3 obtains the biggest improvement in performance on $m2$, ILS obtains the biggest improvement on $m5$. This indicates that some of the optimization options activated by using `-O3` are very favorable for ILS when executed on $m5$, while these same settings seem to be less favorable for ACOTSP ls3 on the same machine. Moreover, different machines exhibit different variability of the results, which can indicate that some machines could be more affected by the stochastic behavior of the algorithms. We can derive from these results the strong impact the system can have on the performance of a compilation process, even if it

Fig. 1. Confidence intervals of the running time obtained by 20 executions of the benchmarks compiled with `GCC` using `-O3` over the instances test set.
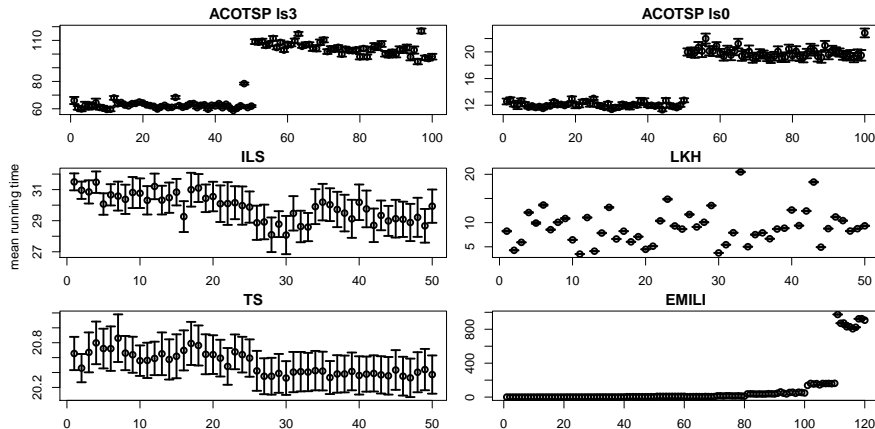

has been adjusted to the particular scenario. Good configurations are then not always portable between systems.

5   Experimental results

We configure the benchmarks described in Section 3 using only the optimization flags of `GCC` (GCC_{flags}) and the flags with the numerical parameters ($GCC_{flags+num}$). Figure 3 gives the mean speed up of 10 `GCC` settings obtained by 10 executions of irace compared to the execution times obtained by `GCC` using `-O3`.

In general, irace is able to obtain speed ups over `-O3` for all the benchmarks. While for some scenarios the improvements are greater than for others, the executions of irace are on average better than `-O3`. When configuring only the flags (GCC_{flags}), all the benchmarks clearly improve their performance with the exception of ACOTSP ls3, for which two executions of irace obtain slightly worse performance than `-O3` (0.98 and 0.99). Regardless of this, the mean speed up obtained by irace in this benchmark is of 1.03. The benchmarks for which the largest speed up is obtained on the GCC_{flags} scenario are ILS and EMILI.

The configuration of the optimization flags together with the numerical parameters ($GCC_{flags+num}$) obtains mixed results across the different benchmarks. The speed up obtained by irace on the ACOTSP ls3, ILS and LKH scenarios is increased, while the speed up is reduced (compared to GCC_{flags}) for ACOTSP ls0, TS and EMILI. For EMILI, the speed up obtained for $GCC_{flags+num}$ is only slightly smaller compared with the one obtained for GCC_{flags}. For TS, the mean speed up is reduced compared to the results obtained for GCC_{flags}. Nevertheless, all the configurations obtained by irace have better mean performance than `-O3`. The results obtained for $GCC_{flags+num}$ on ACOTSP ls0, are worse than the results obtained when configuring GCC_{flags}; despite this, the mean speed up obtained over `-O3` is 1.03 and of the 10 executions the worst speed up is only 0.99.
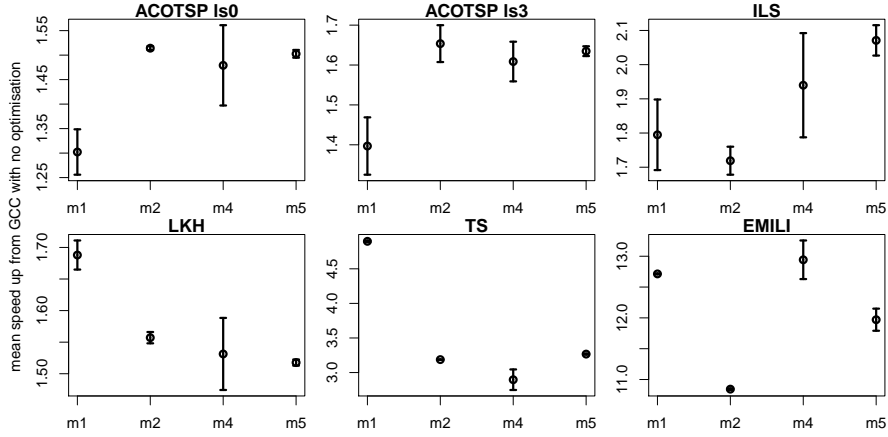
Fig. 2. Confidence intervals of the mean speed up of the running time of 20 executables compiled with `GCC` using `-O3` from `GCC` with no optimization on 4 different machines.
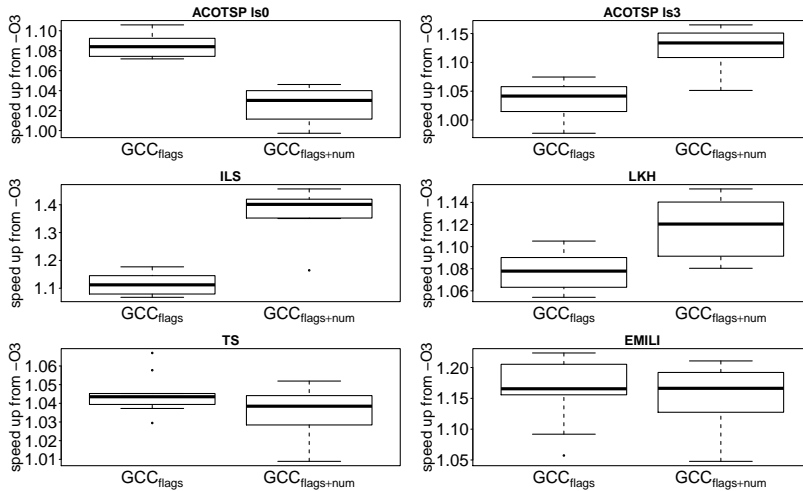
Fig. 3. Mean execution time speed up of 10 `GCC` configurations found by 10 `irace` executions from the mean running time of 10 compilations performed by `GCC` using `-O3`.

The reduced performances when configuring $GCC_{flags+num}$ for ACOTSP ls0, EMILI and TS can be explained by the size and difficulty of the configuration space. The $GCC_{flags+num}$ scenarios have considerably more parameters (366) and, as we maintained the same configuration budget as in the GCC_{flags} scenarios, the search may be too limited. Additionally, the $GCC_{flags+num}$ scenarios seem to be (see Table 2) less homogeneous, and they also produce more failed executions than the GCC_{flags} scenarios. This might actually help the configuration process, in some cases, by easily reducing the search space of interest. The possible ap-

proaches to tackle the configuration of such scenarios with irace are: (i) increasing the configuration budget (if feasible), (ii) providing `-O3` or `-O2` as initial configurations and/or (iii) increasing the number of instances required to start the elimination of configurations. The last one would allow better estimation of the performance of the configurations, which may account for the higher variability of the intra- or inter-instance execution times as seen in Figure 2 for some of the benchmarks.

The performance data obtained from the executions of irace can be useful in order to analyze the characteristics of the configuration scenarios. Such analysis can provide guidelines to set up the compilation with `GCC` on scenarios with similar characteristics. We trained random forest models with the performance data obtained by the 10 executions of irace on each benchmark. The procedure and the settings used to train the random forest model are described in [18]. For the random forest implementation we use the **`ranger`** `R` package [26]. When building the training data set, the configurations that produced failed compilations or executions were removed and thus only valid execution data points are included in the data set. As is common for these performance models, the instances are included as a variable in the training data. Additionally, instance features could be added to the data set to provide more information on the impact of the instances on performance.

Table 3 gives the five most important parameters for each of the benchmarks. Note that since the instances are included in the data set as a variable, for some of the benchmarks the instances are the variable that explains most of the variability. This is the effect of the variability of the execution times required for different instances, which can be observed in Figure 1. The instances variable is therefore more important in benchmarks that have greater inter-instance variability, such as LKH, and less important when the running time for all the instances is similar, as for TS. Even if there are parameters that are important for more than one algorithm, it is not possible to find a parameter that consistently has the same impact on the execution time of all the algorithms. For example, EMILI has a completely different set of important parameters compared to the rest of the algorithms. This difference can be explained by the fact that EMILI is the only program among the benchmarks written in `C++` with an Object Oriented philosophy, while the others are written in `C`. For the benchmarks with algorithms written in `C` (ACOTSP, ILS, TS and LKH), the most important options are the ones that attempt to optimize memory allocation and the use of the registers (`falign-labels`, `fstrict-aliasing`, `fcaller-saves`, `falign-function`, `fomit-frame-pointer`), the linking process (`flto-partition`) and the optimization of the internal representation of the source code used by the compiler (the `ftree` flags). On the contrary, for EMILI, the optimization seems to be more focused on inlining (`finline`), branching optimization(`fif-conversion`, `fcode-hoisting`) and trying to avoid unnecessary function calls (`fipa-pure-const`), which is consistent with an object oriented code where it is common to have a large number of very small functions. Most of the parameters retain a similar importance when we extend the tuning to the numerical parameters. In fact, the list of parameters does not change for ACOTSP ls3 and for the others, with the exception of ILS, for which the changes are minimal. In the case of ILS, the numerical parameter `max-unswitch-insns`

Table 3. Variable importance % obtained by random forest models trained using data from 10 executions of irace.

$\mathrm{GCC_{flags}}$					
ACOTSP ls0		ILS		TS	
instance	89.9	falign-labels	39.1	falign-labels	45.1
falign-labels	2.5	instance	15.3	fguess-branch-probability	6.2
fstrict-aliasing	1.1	fcaller-saves	4.4	fstrict-aliasing	5.4
ftree-ch	0.8	ftree-pre	3.2	ftree-loop-im	5.2
flto-partition	0.5	falign-functions	2.0	ftree-ter	5.0
ACOTSP ls3		LKH		EMILI	
instance	87.3	instance	95.4	finline	50.3
falign-labels	5.5	falign-labels	2.0	instance	40.0
fcaller-saves	0.6	flto-partition	0.2	fif-conversion	3.3
fstrict-aliasing	0.5	finline-limit	0.2	fcode-hoisting	2.4
falign-functions	0.4	fomit-frame-pointer	0.1	fipa-pure-const	1.3
$\mathrm{GCC_{flags+num}}$					
ACOTSP ls0		ILS		TS	
instance	90.5	max-unswitch-insns	29.4	falign-labels	38.9
falign-labels	1.9	falign-labels	13.6	ftree-ter	7.0
ftree-ch	1.3	instance	10.1	ftree-loop-optimize	3.8
fstrict-aliasing	0.9	ftree-dominator-opts	5.3	ftree-loop-im	2.3
ftree-ter	0.5	ftree-loop-optimize	2.9	fomit-frame-pointer	2.2
ACOTSP ls3		LKH		EMILI	
instance	87.4	instance	93.8	finline	44.1
falign-labels	3.5	falign-labels	2.0	instance	40.2
fcaller-saves	1.0	flto-partition	0.5	fif-conversion	3.0
fstrict-aliasing	0.9	parloops-schedule	0.3	sccvn-max-scc-size	2.4
ftree-ch	0.5	finline-limit	0.3	fipa-pure-const	1.2

is the most important one according to our analysis. This parameter controls the threshold used by the compiler to decide if to unswitch a loop, that is moving a loop invariant condition outside of the loop.

Further optimization on the execution time could be then achieved using this information by restricting the configuration process to the set of variables that show to have great impact in the execution times.

6 Conclusion and future work

Reducing the execution time of optimization algorithms, even in cases in which the main objective is not fast execution, is of great interest for developers and users. We have shown that significant reductions of computation times can be obtained by optimizing compilation options. The optimization of compilation options may either be considered as an extra step to improve execution times after an algorithm configuration process, but also as part of the full configuration process to address interactions between algorithm components and compilation options. In preliminary experiments with GCC, we showed that, for the optimization algorithms used, the execution times are stochastic and depend on the computational platform and the characteristics of the code itself. Here, we configured the GCC compiler options, without including any particular knowledge of the compilation process itself, using a general-purpose algorithm configuration software, irace. The experimental results with irace are encouraging and show that irace can find settings that significantly improve over the default optimization flags available in GCC, the commonly recommended -O2 and -O3 settings.

In future work we will extend these experiments to new benchmarks with algorithms that present different characteristics and also to standard compiler benchmark sets used for autotuning methods. For optimization algorithms, it is also of interest to study the effect of instance types and size and their relationship with the GCC options. The information obtained in these experiments could be then further analyzed to extract features from the benchmarks that cause certain optimization flags to have greater effect over performance. Another direction for future work is to specialize the settings of irace to configure compilation options such as a set of initial promising configurations or additional pruning techniques to improve the search. Additionally, we will investigate the use of other automatic configuration tools such as SMAC and compare our approach to other methods specifically designed for the optimization of compiler flags such as OpenTuner and COLE. Finally, we plan to make available the configuration files used in this work to configure the options of GCC, so that other researchers can attempt to configure their own algorithms to obtain better performance.

Acknowledgments.

References

1. Ansel, J., Kamil, S., Veeramachaneni, K., Ragan-Kelley, J., Bosboom, J., O'Reilly, U.M., Amarasinghe, S.: Opentuner: An extensible framework for program autotuning. In: Proceedings of the 23rd International Conference on Parallel Architectures and Compilation. pp. 303–315. ACM New York, NY, USA (2014)
2. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of algorithms. In: Gent, I.P. (ed.) CP 2009, LNCS, vol. 5732, pp. 142–157. Springer (2009)
3. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-race and iterated F-race: An overview. In: Bartz-Beielstein, T., et al. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 311–336. Springer (2010)
4. Blackmore, C., Ray, O., Eder, K.: Automatically tuning the GCC compiler to optimize the performance of applications running on the ARM cortex-M3. Tech. rep., CoRR, `https://arxiv.org/abs/1703.08228` (2017)
5. Christen, M., Schenk, O., Burkhart, H.: Patus: A code generation and autotuning framework for parallel iterative stencil computations on modern microarchitectures. In: Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium. pp. 676–687. IPDPS '11, IEEE Computer Society (2011)
6. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE 93(2), 216–231 (2005), special issue on "Program Generation, Optimization, and Platform Adaptation"
7. Fursin, G., Kashnikov, Y., Memon, A.W., Chamski, Z., Temam, O., Namolaru, M., Yom-Tov, E., Mendelson, B., Zaks, A., Courtois, E., Bodin, F., Barnard, P., Ashton, E., Bonilla, E., Thomson, J., Williams, C.K.I., O'Boyle, M.: Milepost gcc: Machine learning enabled self-tuning compiler. International Journal of Parallel Programming 39(3), 296–327 (2011)

8. GNU Project, Free Software Foundation: GCC, the GNU compiler collection. `https://www.gcc.gnu.org` (1987)

9. Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research 126, 106–130 (2000)

10. Helsgaun, K.: General k-opt submoves for the Lin-Kernighan TSP heuristic. Mathematical Programming Computation 1(2–3), 119–163 (2009)

11. Henning, J.L.: Spec cpu2000: measuring cpu performance in the new millennium. Computer 33(7), 28–35 (2000)

12. Hoos, H.H.: Automated algorithm configuration and parameter tuning. In: Hamadi, Y., et al. (eds.) Autonomous Search, pp. 37–71. Springer (2012)

13. Hoste, K., Eeckhout, L.: Cole: Compiler optimization level exploration. In: Proceedings of the 6th Annual IEEE/ACM International Symposium on Code Generation and Optimization. pp. 165–174. CGO '08, ACM Press, New York, NY (2008)

14. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello Coello, C.A. (ed.) LION 5, LNCS, vol. 6683, pp. 507–523. Springer (2011)

15. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306 (2009)

16. Ladd, S.R.: ACOVEA (Analysis of compiler options via evolutionary algorithm). `https://github.com/Acovea/libacovea` (2000)

17. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58 (2016)

18. Pérez Cáceres, L., Bischl, B., Stützle, T.: Evaluating random forest models for irace. In: GECCO'17 Companion. ACM Press (2017)

19. Pérez Cáceres, L., Pagnozzi, F., Franzin, A., Stützle, T.: Automatic configuration of gcc using irace: Supplementary material. `http://iridia.ulb.ac.be/supp/IridiaSupp2017-009/` (2017)

20. Plotnikov, D., Melnik, D., Vardanyan, M., Buchatskiy, R., Zhuykov, R., Lee, J.H.: Automatic tuning of compiler optimizations and analysis of their impact. In: Alexandrov, V., et al. (eds.) 2013 International Conference on Computational Science. Procedia Computer Science, vol. 18, pp. 1312–1321. Elsevier (2013)

21. Püschel, M., Franchetti, F., Voronenko, Y.: Spiral. In: Padua, D. (ed.) Encyclopedia of Parallel Computing, pp. 1920–1933. Springer, US (2011)

22. Siegel, S., Castellan, Jr, N.J.: Non Parametric Statistics for the Behavioral Sciences. McGraw Hill, 2 edn. (1988)

23. Stützle, T.: ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem (2002), `http://www.aco-metaheuristic.org/aco-code/`

24. Stützle, T., López-Ibáñez, M.: Automatic (offline) configuration of algorithms. In: Laredo, J.L.J., et al. (eds.) GECCO (Companion), pp. 681–702. ACM Press (2015)

25. Whaley, C.R.: Atlas (automatically tuned linear algebra software). In: Padua, D. (ed.) Encyclopedia of Parallel Computing, pp. 95–101. Springer, US (2011)

26. Wright, M.N., Ziegler, A.: ranger: A fast implementation of random forests for high dimensional data in C++ and R. Arxiv preprint arXiv:1508.04409 [stat.ML] (2015)

27. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., et al. (eds.) EUROGEN. pp. 95–100. CIMNE, Barcelona, Spain (2002)

Session 7 - Learning

Reinforcement Learning as a Model of Aposematism

Jan Teichmann[1], Eduardo Alonso[2] and Mark Broom[1]

[1]Department of Mathematics, City University London, London EC1V 0HB, UK
[2]Department of Computer Science, City University London, London EC1V 0HB, UK
`E.Alonso@city.ac.uk`

Abstract. Natural environments are intrinsically complex. This complexity derives on the one hand from the high entanglement of organisms interacting in competitive relationships with each other: the prey is part of the predator's environment and vice versa. On the other hand, natural environments are also defined by their dynamics of constant change. Thus, evolution in natural environments is defined by the dynamic competitive relationships of organisms and, typically, results in multiple species, which successively adapt in response to their adaptations. In particular, predator-prey co-evolution has been identified as an influential factor in the evolution of aposematism. In this paper we address the problem of formalizing predator aversive learning in the presence of aposematism by applying a reinforcement learning algorithm on a biologically plausible predator lifetime model.

Keywords: Predator-prey, Aposematism, Reinforcement Learning.

1   Introduction

The majority of species are at risk of predation in their natural habitat and are targeted by predators as part of the food web. Through the process of evolution by natural selection a predator is confronted with manifold mechanisms that have developed to avoid predation. So-called secondary defenses commonly involve the possession of toxins or deterrent substances which are not directly observable by predators. However, many defended species use conspicuous signals as warning flags in combination with their secondary defenses –what we call aposematism. There is a wide body of theory addressing the emergence and evolution of aposematism (Broom et al., 2006; Broom et al, 2008; Lee et al., 2010; Lee et al., 2011; Leimar et al., 1986; Marples et al., 2005; Ruxton et al., 2004; Ruxton et al., 2009; Yachi and Higashi, 1998). Within this context, the role of the predator as the selective agent and the mechanisms of the predator's aversive learning process are at the heart of current research (Barnett et al., 2007; Hagen et al., 2009; Sherrat, 2003). Nevertheless, there is no accepted formal model of aversive learning in foraging literature.

A normative modern framework for aversive learning can be found within the field of reinforcement learning, which provides a mapping of environmental states to an individual's action in order to maximize a reward signal in an unsupervised manner (Barto et al., 1990; Watkins, 1989; Sutton and Barto, 1998). Solving the underlying

reinforcement learning problem is crucial since natural environments are too complex to learn from examples of desired behaviour –such examples will not be representative for all states of an individual's environment. But it is in these unknown situations when learning is most beneficial and an individual has to rely on its own generalized experience from interactions with its environment. This is where a classical trade-off arises between exploration and exploitation: to maximize a reward an individual should perform the actions that it knows to be rewarding from previous experience. However, to find such actions in the first place an individual had to explore actions with unknown outcome. Therefore, the mapping of states to actions has to be obtained through trial-and-error, or goal directed learning where actions have subsequent effects on future rewards. The last decade has seen a proliferation of research on the neural and psychological mechanisms of reinforcement learning. We know from studies of neural correlates in behaving animals that reinforcement signals in the brain represent reward prediction error rather than a direct reward-reinforcement relation (Daw and Doya, 2006; Dayan and Balleine, 2002; Dayan and Daw, 2008; Dayan and Niv, 2008; Doya, 2007; Maia, 2009; Montague et al., 1996; Montague et al., 2004; Rangel et al., 2008; Schultz, 2002, 2007, 2008; Schultz et al., 1997). Temporal difference learning is a reinforcement learning methodology that reflects these insights by representing states and actions in terms of predictions about future rewards where the learning objective is to iteratively update the target values of future rewards towards their true values based on experience from interactions with the environment. However, apart from in Teichmann et al. (2014a) reinforcement learning has not been proposed to formalize aposematism within the foraging problem.

In this paper, we introduce a predator lifetime model where an individual's payoff is both dependent on the environment and additional aspects of an individual's behaviour, metabolism, and lifetime traits, which are usually abstracted away in reinforcement learning formalisms. This approach will allow us to investigate the cost of learning and the interactions of behaviour and metabolism on the learning outcome. In our approach the learning problem is to find an optimal foraging strategy under the aspects of maximizing the predator's payoff in an episodal task such as a day of foraging. Importantly, the predator's behaviour has delayed effects on its rewards so that a trajectory is not only dependent on its initial conditions but also on all the predator's actions and the subsequent state transitions. Therefore, we have to choose an episodal learning algorithm, which considers the entirety of actions and state transitions of a trajectory within its learning updates. We have used back-propagation through time (BPTT) as an efficient method to calculate the derivative of the predator's payoff function in the policy space for episodal tasks. The rest of the paper is structured as follows: in the next section we describe how reinforcement learning operates. The predator life model is introduced in detail in Section 3. Section 4 describes the reinforcement learning algorithm used to calculate optimal trajectories, and the derivatives used are formulated in Section 5. The results of the simulations are presented in Section 6. We shall conclude with a discussion of the results and further work.

2   Reinforcement Learning

A typical reinforcement learning scenario is an animal inhabiting a state space $S \subset \mathbb{R}^n$, such that at k iteration it "lives" in state $s_k \in S$. The state space represents any features the modeller considers relevant, typically a collection of stimuli, but can also include internal constructs. At each iteration, the animal chooses an action u_k (from an action space $u_k \in U$), which takes it to the next state according to a *model function*

$$s_{k+1} = f(s_k, u_k) \tag{1}$$

and gives it an immediate scalar r_k, represented by the *reward function*

$$r_{k+1} = r(s_k, u_k) \tag{2}$$

The animal keeps acting, forming a trajectory of states $(s_0, s_1, ...)$ indefinitely or until a given terminal state is reached. In this problem the animal must learn to choose actions that maximize the expectation of the total long-term reward, the return, received from any given start state s_0. Formally, the problem is to find an *policy* $\pi(s, z)$, where z is the parameter of a function approximator (typically, a neural network), which calculates actions

$$u_k = \pi(s_k, z) \tag{3}$$

such that the following *value function* is maximized

$$V(s_0, z) = \sum_k \gamma^k r_k \tag{4}$$

subject to Equations (1), (2) and (3), where $\gamma \in [0,1]$ is a constant discount factor that specifies the relative importance of long-term costs over short-term ones.

3   The Predator Lifetime Model

This section introduces the lifetime model of an individual predator and the definition of the individual's payoff based on its environment and additional aspects of its behaviour, metabolism, and lifetime traits. In this model an individual predator is characterized by its state s_k at iteration k. The state is given by

$$s_k = \{T, A, X, Y\} \tag{5}$$

with T being the time of an iteration k within an episode, A the age of the predator, and X and Y the axes of the spatial location of the predator within its environment at iteration k. The predator finds itself in an environment defined by the availability of different food sources. The dispersion of each prey population i within the environment is described by a well-understood Gaussian distribution function

$$g_i(X,Y) = p_i \exp\left(-\left(\frac{(X-x_{i,0})^2}{2\sigma_{i,x}^2} + \frac{(Y-y_{i,0})^2}{2\sigma_{i,y}^2}\right)\right) \tag{6}$$

with $(x_{i,0}, y_{i,0})$ being the center of the prey population with density p_i and $(\sigma_{i,x}, \sigma_{i,y})$ the spread of the prey. The model assumes that the prey is aposematic with both models (venomous animals) and potential mimics (non-venomous animals which mimic the defenses of venomous animals). The predator feeds on prey it encounters, as it cannot distinguish between models and mimics based on their appearance. However, the predator has the option to move around freely in its environment to avoid encounters with possibly aversive prey based on its experience. The predator's locomotion is defined by its action vector u_k, given by

$$u_k = \{e_x, e_y\} \tag{7}$$

with e_x, e_y representing the energy invested into locomotion at iteration k. The value function V describes the total payoff of a predator at the end of an episode and is the result of the predator's interaction with the environment. Thereby, the predator's actions have subsequent effects on its environment through locomotion and the predator's spatial location within the environment according to the reinforcement learning model. The subsequently received payoff for the predator being in a specific state s_k and taking action u_k at iteration k is given by the payoff function as follows

$$r_{k+1} = \dot{V} = \underbrace{\lambda(A_k)R(s_k) - t_0\dot{T}}_{state\,dependent} \overbrace{-\left|E(u_k)\right|}^{action\,dependent} \tag{8}$$

where $t_0\dot{T}$ is the metabolic cost of the predator, $-\left|E(u_k)\right|$ the absolute energy expenditure of a predator's actions, and $R(s)$ the state specific payoff defined as

$$R(s_k) = \sum_i g_i(s_k) d(t_i)(r - t_i^2) \tag{9}$$

where

$$d(t) = \frac{1}{1 + d_0 t} \tag{10}$$

is the probability of ingesting a prey individual of toxicity t after taste sampling. The model has the option to include age related effects such as an age dependent agility of the predator given, for example, by

$$\lambda(A) = \frac{1}{1 + A} \tag{11}$$

The environment of this model is Markovian defined by the state transition function

$$f(s, u)_{k \to k+1} = \begin{pmatrix} \dot{T} = 1 + \sum_i g_i(s_k)\left(d(t_i)\left(t_h + t_t t_i^2\right) + t_s\right) \\ \dot{A} = (1/\lambda_0)\dot{T} \\ \dot{X} = \tanh(c_0 e_x) \\ \dot{Y} = \tanh(c_0 e_y) \end{pmatrix} \tag{12}$$

The transition of time $(\dot{T})$ between iterations occurs in unit time steps reflecting a basal metabolic expenditure and the additional costs of foraging such as the sampling of prey items t_s, the handling of prey t_h, and the recovery from ingested toxins $t_t t^2$. The predator ages $(\dot{A})$ linearly with time. The predator's locomotion results in a change of its spatial location $(\dot{X}, \dot{Y})$ depending on the predator's energy investment e_x, e_y with the maximal spatial displacement per iteration being a unit step of one. The functions of the model are governed by single parameters, which allow the trade off between the different aspects of the predator's behaviour, lifetime traits, and environment $(x_0, y_0, t_0, d_0, \lambda_0, c_0)$. These parameters define the lifetime model. The only term missing is the subjective payoff. The assumption is that this subjective payoff can be reverse engineered from the observed foraging behaviour of the predator using a reinforcement learning algorithm. The aim is to find the subjective value of the payoff for prey type i such as to reproduce the observed foraging behaviour of the predator. Additionally to the lifetime model, we define a final instantaneous cost Ψ of the terminal state s_l with l being the final iteration of an episode based on the spatial distance of the predator from its den at $(X = 0, Y = 0)$:

$$\Psi(s)_l = \begin{cases} -r_l \sqrt{X^2 + Y^2} & \text{if} \quad \sqrt{X^2 + Y^2} > \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

with $-r_l$ being a punishment for not returning to the den at the end of an episode. Within a biological context such a final cost can be thought of as a step-like around the predator's den. If a predator has to feed offspring staying behind in the den the cost of almost returning will not decrease smoothly within the proximity of the den.

4   Reinforcement Learning using BPTT

As discussed previously, the control problem in reinforcement learning is about finding an optimal behaviour policy (aka an actor). Reformulating reinforcement learning using an actor becomes the problem of finding the parametrization $\vec{z}$ of the actor $\pi(\vec{s},\vec{z})$ that maximizes the total value –or minimizes the overall prediction error for a complete trajectory based on $V(\vec{s},\pi(\vec{s},\vec{z}))$. This can be achieved using hill climbing on the total value of a complete trajectory itself with respect to $\vec{z}$, i.e. $\Delta\vec{z} = \alpha(\partial V / \partial \vec{z})$, which is also called a policy gradient with back-propagation through time (BPTT) being an efficient implementation of the optimization problem for episodal tasks (Werbos, 1974). BPTT uses the actor $\pi(\vec{s},\vec{z})$, traditionally implemented as an artificial neural network with weights $\vec{z}$ as a universal function approximator and which is equivalent to the behaviour policy. As such, BPTT is an off-line learning algorithm that issues a weight update $\Delta\vec{z}$ at the end of an episode. The delayed effects of actions in the reinforcement learning problem means that the outcome of a trajectory is not only dependent on its initial conditions but also on all the actions of an individual and the subsequent state-transitions. Therefore, an episodal learning algorithm has to consider the entirety of actions and state-transitions of a trajectory within its updates. Thereby, the trajectory of a complete episode is unrolled backwards using the Markovian properties of the environment with the component $(\partial V / \partial \vec{z})_k$ being computed from the prevailing quantity $(\partial V / \partial \vec{z})_{k+1}$, i.e. the policy gradient of the value function is computed backwards in time starting at the end of an episode (Eq. 14). This property gives the methodology its name. It is well-known that back-propagation is an efficient way of calculating the derivative of the network function in artificial neural networks. Back-propagation through time is the extension of that methodology to efficiently calculate the derivative of the network function in episodal tasks where the neural network has been applied multiple times to create a trajectory of states and payoffs – similarly to recurrent neural network problems – including the concepts of discounting and bootstrapping. Therefore, the derivative of the overall network function is the sum of the discounted incremental gradients at each iteration of the trajectory with their calculation expanding as follows: at the beginning of the BPTT algorithm the partial gradients of the value function are initialized: $(\partial V / \partial \vec{z})_l \leftarrow \vec{0}$ and $(\partial V / \partial \vec{s})_l \leftarrow (\partial \Psi / \partial \vec{s})_l$, with Ψ (Eq. 13) being the final instantaneous cost of the terminal state s_l, and l being the final iteration in an episode of finite length.

Following the initialization the algorithm processes the trajectory of an episode backwards in time starting from the second last iteration to the first iteration in the episode. At each step the algorithm adds the partial policy gradient of the current

iteration to the overall policy gradient of the value function for an episode $\left(\partial V / \partial \vec{z}\right)_k$ as follows:

$$\left(\frac{\partial V}{\partial \vec{z}}\right)_k \leftarrow \left(\frac{\partial V}{\partial \vec{z}}\right)_{k+1} + \gamma^k \underbrace{\left(\frac{\partial \pi(\vec{s},\vec{z})}{\partial \vec{z}}\right)_k}_{actor} \underbrace{\left(\left(\frac{\partial r}{\partial \vec{u}}\right)_k + \gamma \left(\frac{\partial f}{\partial \vec{u}}\right)_k \left(\frac{\partial V}{\partial \vec{s}}\right)_{k+1}\right)}_{behavioral\ gradient} \tag{14}$$

with the following state dependent value contribution deriving from the Markovian properties of the environment

$$\left(\frac{\partial V}{\partial \vec{s}}\right)_k = \underbrace{\left(\left(\frac{\partial r}{\partial \vec{s}}\right)_k + \gamma \left(\frac{\partial f}{\partial \vec{s}}\right)_k \left(\frac{\partial V}{\partial \vec{s}}\right)_{k+1}\right)}_{environmental\ gradient} + \underbrace{\left(\frac{\partial \pi(\vec{s},\vec{z})}{\partial \vec{z}}\right)_k}_{actor} \underbrace{\left(\left(\frac{\partial r}{\partial \vec{u}}\right)_k + \gamma \left(\frac{\partial f}{\partial \vec{u}}\right)_k \left(\frac{\partial V}{\partial \vec{s}}\right)_{k+1}\right)}_{behavioral\ gradient} \tag{15}$$

The final weight update gives the implementation of hill climbing on the value function V with respect to the policy gradient of $\pi(\vec{s},\vec{z})$ using

$$\vec{z} \leftarrow \vec{z} + \alpha \frac{\partial V}{\partial \vec{z}} \tag{16}$$

with α representing a learning rate. Summarizing, the BPTT algorithm can be understood as propagating the policy gradient of the value function with respect to a future state $\left(\partial V / \partial \vec{s}\right)_{k+1}$ backwards in time through the actor, the state-transition function, and the payoff function to obtain the policy gradient of the value function $\left(\partial V / \partial \vec{s}\right)_k$ of the previous state. As BPTT utilizes the Markovian properties of the environment using the state-transition function for the propagation of the state-dependent gradient backwards through time it is a model-based methodology. BPTT as a simple hill-climbing algorithm on the value function has robust convergence proofs (Werbos, 1990).

5   Derivatives Used by the BPTT Algorithm

As BPTT is model-based it requires a number of derivatives of the underlying lifetime model. The lifetime model is implemented as a Markovian decision process and the propagation of incremental gradients backwards through time in the algorithm requires the state $\partial f(\vec{s}_k,\vec{u}_k) / \partial \vec{s}$ and action $\partial f(\vec{s}_k,\vec{u}_k) / \partial \vec{u}$ dependent derivatives of the state-transition function f, where

$$\frac{\partial f(\vec{s}_k,\vec{u}_k)}{\partial \vec{u}} = \left\{\frac{\partial \dot{T}}{\partial \vec{u}}, \frac{\partial \dot{A}}{\partial \vec{u}}, \frac{\partial \dot{X}}{\partial \vec{u}}, \frac{\partial \dot{Y}}{\partial \vec{u}}\right\} \tag{17}$$

and

$$\frac{\partial f(\vec{s}_k, \vec{u}_k)}{\partial \vec{s}} = \left\{ \frac{\partial \dot{T}}{\partial \vec{s}}, \frac{\partial \dot{A}}{\partial \vec{s}}, \frac{\partial \dot{X}}{\partial \vec{s}}, \frac{\partial \dot{Y}}{\partial \vec{s}} \right\} \tag{18}$$

Furthermore, BPTT requires the state and action dependent derivatives, respectively, $\partial r_{k+1}(\vec{s}_k, \vec{u}_k)/\partial \vec{s}$ and $\partial r_{k+1}(\vec{s}_k, \vec{u}_k)/\partial \vec{u}$ of the payoff function r_{k+1}. In regards to the lifetime model, the derivatives of the state transition function f are

$$\frac{\partial \dot{T}}{\partial \vec{s}} = \sum_i p_i \frac{\partial g_i(X,Y)}{\partial \vec{s}} \left(t_s + d(t_i)(t_{h,i} + t_t t_i^2) \right) = \tag{19}$$

$$\begin{cases} \dfrac{\partial \dot{T}}{\partial T} = 0 \\[2mm] \dfrac{\partial \dot{T}}{\partial A} = 0 \\[2mm] \dfrac{\partial \dot{T}}{\partial X} = \sum_i p_i \left(-g_i(X,Y)(X - x_{i,0})/\sigma_{i,x}^2 \right)\left(t_s + d(t_i)(t_{h,i} + t_t t_i^2) \right) \\[2mm] \dfrac{\partial \dot{T}}{\partial Y} = \sum_i \underbrace{p_i \left(-g_i(X,Y)(X - y_{i,0})/\sigma_{i,y}^2 \right)}_{\text{encounter with prey}} \underbrace{\left(t_s + d(t_i)(t_{h,i} + t_t t_i^2) \right)}_{\text{prey handling}} \end{cases}$$

where the state dependent time transition $\partial \dot{T}/\partial \vec{s}$ is defined by the predator's spatial location within the environment and its interactions with the present prey. Otherwise time progresses constantly and independently of age and time. The state dependent derivative of the predator's ageing follows directly from the time transition. Other relevant derivatives of the state transition function f are the action dependent changes in the predator's spatial location within the environment. The predator can invest energy e_x, e_y into locomotion with respect to X and Y respectively. By definition of the lifetime model the remaining derivatives of the state transition function f are independent of the state or the predator's actions, i.e.

$$\frac{\partial \dot{T}}{\partial \vec{u}} = \frac{\partial \dot{A}}{\partial \vec{u}} = \frac{\partial \dot{X}}{\partial \vec{s}} = \frac{\partial \dot{Y}}{\partial \vec{s}} = \vec{0} \tag{20}$$

with the spatial location of the predator being solely affected by the predator's action. Additionally, time and age progress independently from the predator's investment into locomotion within each iteration. Next, consider the state and action dependent derivatives related to the value function V, which are given by the sum of discounted payoffs along the trajectory of an episode. The derivatives of the incremental changes to the value of an episode along a trajectory are

$$\frac{\partial \dot{V}}{\partial \vec{s}} = \begin{cases} \dfrac{\partial \dot{V}}{\partial T} = 0 \\[2mm] \dfrac{\partial \dot{V}}{\partial A} = \dot{\lambda} R(s_k) \\[2mm] \dfrac{\partial \dot{V}}{\partial X} = \lambda(A)\dfrac{\partial R(s_k)}{\partial X} - \left(t_0 \dfrac{\partial \dot{T}}{\partial X}\right) \\[2mm] \dfrac{\partial \dot{V}}{\partial Y} = \lambda(A)\dfrac{\partial R(s_k)}{\partial Y} - \left(t_0 \dfrac{\partial \dot{T}}{\partial Y}\right) \end{cases}$$

(21)

where $\partial R(\vec{s})/\partial \vec{s}$ is the derivative of the state dependent payoff from interacting with prey given by

$$\frac{\partial R(\vec{s})}{\partial \vec{s}} = \sum_i p_i \frac{\partial g_i(X,Y)}{\partial \vec{s}} d(t_i)(r_i - t_i^2) =$$

(22)

$$\begin{cases} \dfrac{\partial R(\vec{s})}{\partial T} = 0 \\[2mm] \dfrac{\partial R(\vec{s})}{\partial A} = 0 \\[2mm] \dfrac{\partial R(\vec{s})}{\partial X} = \sum_i p_i \left(-g_i(X,Y)(X - x_{i,0})/\sigma_{i,x}^2\right) d(t_i)(r_i - t_i^2) \\[2mm] \dfrac{\partial R(\vec{s})}{\partial Y} = \sum_i \underbrace{p_i \left(-g_i(X,Y)(X - y_{i,0})/\sigma_{i,y}^2\right)}_{\text{encounter with prey}} \underbrace{d(t_i)(r_i - t_i^2)}_{\text{prey payoff}} \end{cases}$$

and

$$\dot{\lambda}(A) = -\frac{1}{(A+1)^2}$$

(23)

The absolute amount of energy invested into locomotion each iteration affects the value of an episode as follows:

$$\frac{\partial \dot{V}}{\partial \vec{u}} = \begin{cases} \dfrac{\partial \dot{V}}{\partial e_x} = -\text{sgn}(e_x) \\[2mm] \dfrac{\partial \dot{V}}{\partial e_y} = -\text{sgn}(e_y) \end{cases}$$

(24)

225

6   Results

The results in Fig. 1 and Fig. 2 depict the trajectories which are close to an optimal trajectory and which were found running the learning algorithm continuously saving trajectories that increased the overall payoff V for an episode. The environment is composed of an aposematic prey population and a population of Batesian mimics. The predator cannot distinguish between them *visually* and has to utilize experience from ingesting prey individuals to find a rewarding feeding ground. The trajectory of a predator which is not utilizing taste-sampling (Fig. 1a) shows avoidance of the aversive prey population taking a non-direct route to the population of mimics. The pre-condition of exploration for successful aversion formation and the non-direct route result in a very low value for the locomotion parts of the trajectory. In order to make the predator exploit the population of mimics the length of an episode had to be high with $l = 80$ in the simulation. The taste-sampling predator takes a more direct route towards the population of mimics and experiences a much higher value for the locomotion parts of the trajectory (Fig. 1b).

Fig. 2 shows the locomotion profile for the non-taste sampling predator. The predator's locomotion in general is optimized towards efficiency maximizing the displacement per energy expenditure $\max_{e_x} d\dot{X}/de_x$ and $\max_{e_y} d\dot{Y}/de_y$, which is at $e_x = e_y = 0.5$ in the simulation with a diagonal locomotion of length 0.7 being therefore most efficient. There is a trade-off in this simulation as the population of mimics is not located on the diagonal and due to the presence of an aposematic prey population. Additionally, the predator is over-staying in the feeding grounds with the second half of the trajectory showing a more rapid locomotion than the first half.
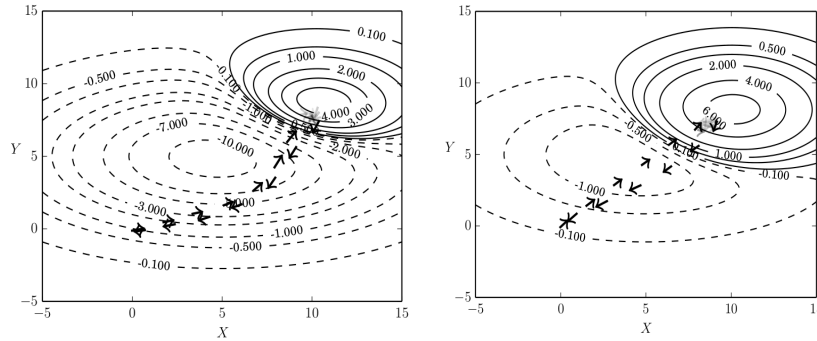

(a) The state dependent reward payoff $R(s_k)$ for a predator not utilizing taste sampling: $d_0 = 0$ and $t_s = 0$.

(b) The state dependent reward payoff $R(s_k)$ for a predator utilizing taste sampling: $d_0 = 1$ and $t_s = 0.1$.
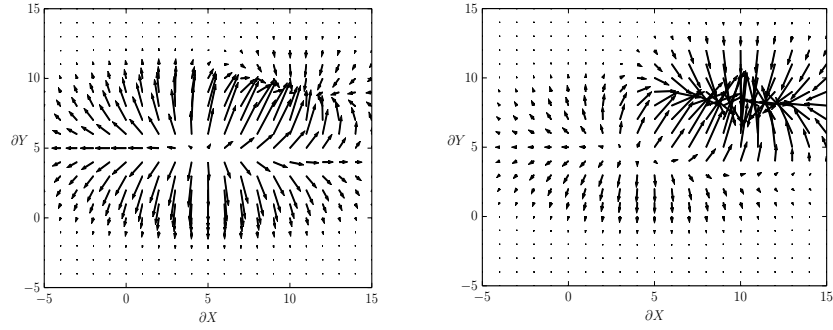
(c) The partial derivative of the reward with respect to the spatial position of the predator not utilizing taste sampling: $d_0 = 0$ and $t_s = 0$.

(d) The partial derivative of the reward with respect to the spatial position of the predator utilizing taste sampling: $d_0 = 1$ and $t_s = 0.1$.

Fig. 1. The state dependent reward of an exemplary environment with aposematic prey and Batesian mimics. All with $t_h = 0.1$, $t_t = 0.2$, $p_1 = p_2 = 0.5$, $x_{1,0} = y_{1,0} = 5$, $\sigma_{1,x} = 5$, $\sigma_{1,y} = 2.55$, $t_1 = 5$, $r_1 = 1$, $x_{2,0} = 10$, $y_{2,0} = 8$, $\sigma_{2,x} = 2$, $\sigma_{2,y} = 2$, $t_2 = 0$, $r_2 = 15$, $l = 80$.
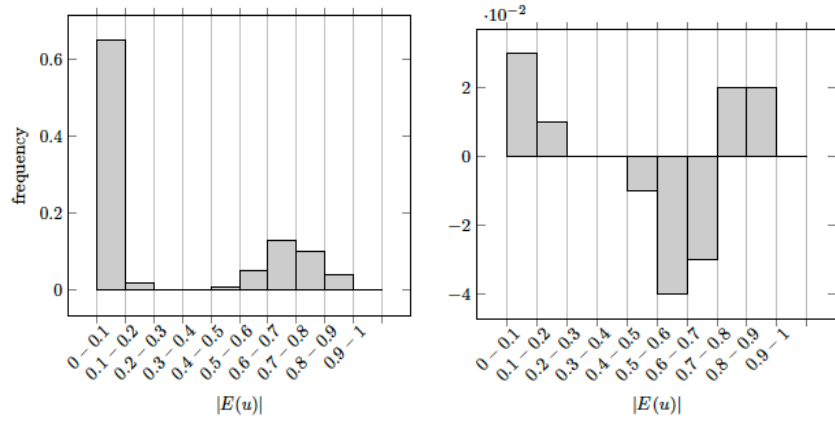
(a) The predator's locomotion optimizes the energy expenditure with $\max_e d\dot{X} / de = 0.5$ and $\sqrt{\dot{X}(0.5)^2 + \dot{Y}(0.5)^2} = 0.7$.

(b) This plot shows the difference of the second half compared to the first half of an episode. The predator prefers to feed longer and return to its den using a greater step size than the optimal of 0.7.

Fig. 2. The locomotion profile of a predator not utilizing taste sampling with an episode of length $l = 80$.

These results show some interesting properties: (a) In a biological context the trajectories of the predator are unstable; (b) the element of the model which is generally optimized is the efficiency of locomotion (the behavioural expenditure); (c) a non-taste sampling predator avoids the aposematic prey population in order to minimize its metabolic costs from toxin ingestion; and (d) The predator shows a tendency to over-stay in the feeding grounds and returns to the den with above optimal energy expenditure for locomotion.

7   Discussion

In this paper we have presented a predator lifetime model including life history traits that have been traditionally abstracted away in the literature such as metabolic costs, locomotion, prey handling, and toxin recovery. The model was defined in such a way that it can be interpreted in a psychological context of subjective behaviour driven by reward motivated objectives and an evolutionary context of a behavioural repertoire which is driven by fitness and co-evolution between predator and prey.

We applied a reinforcement learning algorithm trained using back-propagation through time (BPTT), to simulate behaviour of single individuals driven by rewards. BPTT address learning in episodal tasks based on experience including discounted future rewards. We used an artificial neural network as a universal function approximator in order to implement the policy. The learning task for the simulator is defined in a way to address the discussion of when behaviour is optimal (Teichmann et al., 2014b). On the one hand, the environment in the simulation contains rewards and punishment and optimal behaviour should maximize positive reinforcement. On the other hand, the environment contains a fitness related element in the form of an instantaneous final cost in case the predator does not return to its den at the end of an episode. From a biological context this cost function is a steep step-like function: if the predator has to feed offspring in its den then being close to the den will not gradually reduce the cost of not returning. The trajectories from the simulator show instability due to the interference of maximizing positive reinforcement along the trajectory (excluding the fitness cost) and maximizing the value of a complete trajectory (including the fitness cost). The simulator oscillates between two states: (i) a state of maximizing rewards along the trajectory excluding the final cost where the predator stays in the feeding ground and does not return to its den and (ii) a state of maximizing the value of the complete trajectory including the final cost where the predator successfully returns to its den. However, the simulation shows that the predator generally optimizes the efficiency of its behavioural expenditure. That the rewards interfere with the optimal behaviour as the predator overstays in the feeding grounds and uses above optimal energy for its locomotion on its return to the den corresponds in fact with biological findings (Nonacs, 2001). Summarizing, as far as we know the paper presents the first attempt to formalize foraging behaviour as a learning problem. In order to do this we have extended the traditional reinforcement learning framework with relevant ethological features, making it biologically plausible (à la Berridge 1996, 2003; Berridge et al., 2009). Hence our model is a

contribution to foraging theory as well as to reinforcement learning research. Of course, our results are preliminary, and future work will include investigating the effects of alternative terminal cost functions and refining the lifetime model. For instance, the model could be further developed by integrating a Darwinian fitness function. Finally, we will also consider whether we can eliminate, at least partially, fluctuations around optimal trajectories by using variations of the learning algorithm as such.

References

Barnett, C.A, Bateson, M., Rowe, C.: State-dependent decision making: educated predators strategically trade off the costs and benefits of consuming aposematic prey. Behavioral Ecology 18, 645–651 (2007).

Barto, A.G., Sutton, R.S., Watkins, C.J.C.H.: Learning and sequential decision making. In: Gabriel, M., Moore, J.W. (eds.) Learning and Computational Neuroscience: Foundations of Adaptive Networks, pp. 539-602, MIT Press, Cambridge, Mass (1990).

Berridge, K.C.: Food reward: brain substrates of wanting and liking. Neuroscience & Biobehavioral Reviews 20(1), 1–25 (1996).

Berridge, K.C.: Pleasures of the brain. Brain and Cognition 52(1), 106–128 (2003).

Berridge, K.C., Robinson, T.E., Aldridge, J.W.: Dissecting components of reward: 'liking', 'wanting', and learning. Current Opinion in Pharmacology 9(1), 65–73 (2009).

Broom, M., Ruxton, G.D., Speed, M.P.: Evolutionarily stable investment in anti-predatory defences and aposematic signaling. In: Deutsch, A., Bravo de la Parra, R., de Boer, R.J., Diekmann, O., Jagers, P., Kisdi, E., Kretzschmar, M., Lansky, P., Metz, H. (eds.) Mathematical Modeling of Biological Systems, Volume II: Epidemiology, Evolution and Ecology, Immunology, Neural Systems and the Brain, and Innovative Mathematical Methods, pp. 37–48, Springer, Berlin (2008).

Broom, M., Speed, M.P., Ruxton, G.D.: Evolutionarily stable defence and signalling of that defence. Journal of Theoretical Biology 242, 32–34 (2006).

Daw, N.D., Doya, K.: The computational neurobiology of learning and reward. Current Opinion in Neurobiology 16, 199–204 (2006).

Dayan, P., Balleine, B.W.: Reward, motivation, and reinforcement learning. Neuron 36(2), 285–298 (2002).

Dayan, P., Daw, N.D.: Decision theory, reinforcement learning, and the brain. Cognitive, Affective & Behavioral Neuroscience 8, 429–453 (2008).

Dayan, P., Niv, Y.: Reinforcement learning: the good, the bad and the ugly. Current Opinion in Neurobiology 18, 185–196 (2008).

Doya, K.: Reinforcement learning: Computational theory and biological mechanisms. Human Frontiers Science Program 1, 30–40 (2007).

Guilford, T.: "Go-slow" signalling and the problem of automimicry. Journal of theoretical biology 170, 311–316 (1994).

Hagen, E.H., Sullivan, R.J., Schmidt, R., Morris, G., Kempter, R., Hammerstein, P.: Ecology and neurobiology of toxin avoidance and the paradox of drug reward. Neuroscience 160, 69 –84 (2009).

Lee, T.J., Marples, N.M., Speed, M.P.: Can dietary conservatism explain the primary evolution of aposematism?. Animal Behaviour 79, 63 –74 (2010).

Lee, T.J., Speed, M.P., Stephens, P.A.: Honest signaling and the uses of prey coloration. The American Naturalist 178(1), E1–E9 (2011).

Leimar, O., Enquist, M., Sillen-Tullberg, B.: Evolutionary stability of aposematic coloration and prey unprofitability: A theoretical analysis. The American Naturalist 128(4), 469–490 (1986).

Maia, T.V.: Reinforcement learning, conditioning, and the brain: Successes and challenges. Cognitive, Affective, & Behavioral Neuroscience 9, 343–364 (2009).

Marples, N.M., Kelly, D.J., Thomas, R.J.: Perspective: The evolution of warning coloration is not paradoxical. Evolution 59, 933–940 (2005).

Montague, P.R., Dayan, P., Sejnowski, T.J.: A framework for mesencephalic dopamine systems based on predictive Hebbian learning. The Journal of neuroscience 16, 1936–1947 (1996).

Montague, P.R., Hyman, S.E., Cohen, J.D.: Computational roles for dopamine in behavioural control. Nature 431, 760–767 (2004).

Nonacs, P.: State dependent behavior and the marginal value theorem. Behavioral Ecology 12(1), 71–83 (2001).

Rangel, A., Camerer, C., Montague, P.R.: A framework for studying the neurobiology of value-based decision making. Nature Reviews Neuroscience 9, 545–556 (2008).

Ruxton, G.D., Sherratt, T.N., M.P. Speed, M.P.: Avoiding attack: The evolutionary ecology of crypsis, warning signals and mimicry. Oxford University Press, Oxford (2004).

Ruxton, G.D., Speed, M.P., Broom, M.: Identifying the ecological conditions that select for intermediate levels of aposematic signaling. Evolutionary Ecology 23, 491–501 (2009).

Schultz, W.: Getting formal with dopamine and reward. Neuron 36, 241–263 (2002).

Schultz, W.: Multiple dopamine functions at different time courses. Annual Review of Neuroscience 30, 259–288 (2007).

Schultz, W.: Neuroeconomics: the promise and the profit. Philosophical Transactions of the Royal Society B: Biological Sciences 363, 3767–3769 (2008).

Schultz, W., Dayan, P., Montague P.R.: A neural substrate of prediction and reward. Science 275, 1593–1599 (1997).

Sherratt, T.N.: State-dependent risk-taking by predators in systems with defended prey. Oikos 103, 93–100 (2003).

Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. Cambridge University Press, Boston (1998).

Teichmann, J., Broom, M., Alonso, E.: The application of temporal difference learning in optimal diet models. Journal of Theoretical Biology 340, 11–16 (2014a).

Teichmann, J., Broom, M., Alonso, E.: The evolutionary dynamic of aposematism: a numerical analysis of co-evolution in finite populations. Mathematical Modelling of Natural Phenomena 9, 148-164 (2014b).

Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. Thesis. King's College, Cambridge University, London (1989).

Werbos, P.J.: Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. Thesis. Harvard University, Cambridge, Mass (1974).

Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78(10) 1550–1560 (1990).

Yachi, S., Higashi, M.: The evolution of warning signals. Nature 394, 882–884 (1998).

Offline Learning for Selection Hyper-heuristics with Elman Networks

W. B. Yates and E. C. Keedwell

Computer Science, College of Engineering, Mathematics and Physical Sciences,
University of Exeter, Exeter, EX4 4QF, UK.
`wy254@exeter.ac.uk` `E.C.Keedwell@exeter.ac.uk`

Abstract. Offline selection hyper-heuristics are machine learning methods that are trained on heuristic selections to create an algorithm that is tuned for a particular problem domain. In this work, a simple selection hyper-heuristic is executed on a number of computationally hard benchmark optimisation problems, and the resulting sequences of low level heuristic selections and objective function values are used to construct an offline learning database. An Elman network is trained on sequences of heuristic selections chosen from the offline database and the network's ability to learn and generalise from these sequences is evaluated. The networks are trained using a leave-one-out cross validation methodology and the sequences of heuristic selections they produce are tested on benchmark problems drawn from the HyFlex set. The results demonstrate that the Elman network is capable of intra-domain learning and generalisation with 99% confidence and produces better results than the training sequences in many cases. When the network was trained using an inter-domain training set, the Elman network did not exhibit generalisation indicating that inter-domain generalisation is a harder problem and that strategies learned on one domain cannot necessarily be transferred to another.

Keywords: Hyper-heuristics, Elman networks, Offline learning.

1   Introduction

Hyper-heuristics are heuristic methods that are employed to solve computationally hard problems for which no known effective algorithmic solution exists. Typically such problems are presented as optimisation problems where the goal is to minimise an *objective function* defined on a space of solutions. Such methods have proved effective on a number of real world problems (see [1]).

A *selection hyper-heuristic* selects heuristics from a given set of low level heuristics and applies them sequentially to optimise a particular problem. Many hyper-heuristics employ learning algorithms in order to improve optimisation performance, and this learning may be classified as either *online* or *offline*. Online learning is based on the low level heuristic selections and resulting objective function values computed during the execution of a hyper-heuristic. In contrast,

offline learning is performed on a database of low level heuristic selections and objective function values computed by a hyper-heuristic on a fixed number of benchmark problems. This paper is concerned with offline learning for selection hyper-heuristics.

A variety of machine learning algorithms have been proposed for offline learning (see for example [2], [3], and [4]). In [2] *classifier systems* are applied to the 1D bin packing problem. Here the system learns a set of rules which associate characteristics of the current problem state with specific heuristics. Heuristics are selected and applied sequentially, thus gradually altering the characteristics of the problem. The system when trained on several problems, generalises by also performing well on unseen problems. In [3] *case based reasoning* (CBR) is applied successfully to exam timetabling problems. The assumption underlying CBR is that "similar problems will have similar solutions". Previous problems and their "good" solutions (called source cases) are collected and stored. A similarity based retrieval process compares the source cases with the problem at hand, and selects heuristics that were employed successfully in similar situations. Here the authors employ a two-stage learning process, one for the case representation (or feature selection) and another for source case selection. In [4], *messy genetic algorithms* are used to evolve combinations of condition-action rules which represent problem states and associated heuristics. Each chromosomes represents a hyper-heuristic and contains the set of rules that determine which heuristic should be applied to which problem state. When tested, these hyper-heuristics generalised well and solved many of the test problems efficiently.

In each case, learning is used to improve optimisation performance by improving the selection of *individual* heuristics at particular points in the search process across a number of training problems. In contrast, recent research (see [5] and [6]) has argued that heuristic selections should be understood as part of a *sequence* of selections. The concept of heuristic sequences is intuitive, certain heuristic orderings make sense (e.g. an explorative mutation followed by an exploitative local search) whereas others (e.g. the reverse of the previous example) do not.

The objective of this study is to test the thesis that subsequences of heuristics can be found in the offline learning database that are effective across a number of problems and (it is hoped) problem domains. A selection hyper-heuristic is executed on the well known HyFlex set of benchmark problems (see [7]) and the resulting sequences of low level heuristic selections and objective function values are used to construct an offline learning database. An Elman network (see [8]) is used to extract effective subsequences of heuristics automatically by learning from suitable sets of sequences chosen from the offline database. Elman networks are recurrent neural networks which naturally learn from, process and produce sequences of data. After training, the Elman network is used to compute new sequences of heuristics which are then evaluated on unseen HyFlex example problems. The aim is to determine if the network has generalised from the training sequences. In this context, generalisation means that the network is

able to produce a sequence of heuristic selections which, when evaluated on the unseen examples, outperform the training sequences.

The benchmark problems are drawn from 4 distinct *problem domains*. Offline learning can be classified as either *intra-domain* or *inter-domain*. In intra-domain learning, the training sequences and the test optimisation problem are drawn from the same problem domain. In inter-domain learning, the training sequences and test problem can be drawn from different domains.

The results presented here demonstrate that an Elman network is capable of intra-domain learning and generalisation with 99% confidence when trained on suitable sequences of heuristic selections. When trained using an inter-domain training set, the Elman network did not exhibit generalisation indicating that inter-domain generalisation is harder, and the methodology used to choose the training sets is unsuitable in this case.

This paper is structured as follows. Section 2 details the methodology and describes the construction of the offline learning database, the structure of the Elman networks and their training sets, and the hyper-heuristic used to evaluate the sequences produced by the trained Elman networks. Section 3 contains the results of two experiments designed to test the suitability of Elman networks for offline intra-domain and inter-domain learning. Finally, Section 4 presents the conclusions of this study.

2 Methodology

Section 2.1 contains a description of the HyFlex benchmark problems and the DBGen hyper-heuristic used to generate the offline learning database. In Section 2.2 the mathematical concept of a logarithmic return is introduced and used to quantify hyper-heuristic performance, and to select training sequences from the database. Section 2.3 details the architecture of the Elman network used in this study, while Section 2.4 describes the construction of the intra-domain and inter-domain training sets. Finally, in Section 2.5, the BLIND hyper-heuristic that is used to evaluate the sequences produced by the trained Elman networks is presented.

2.1 HyFlex and the Offline Learning Database

The Hyper-heuristics Flexible framework (or HyFlex[1], see [7]) is a set of benchmark problems that has been used in a number of studies. See for example [9], [10], [11], [12], [5], and [13]. HyFlex contains an implementation of four computationally hard problem domains:

1. 1D bin packing (BP),
2. permutation flow shop (PFS),
3. boolean satisfiability (SAT), and

---

[1] HyFlex, Cross-domain Heuristic Search Challenge (CHeSC 2011) is used in this study (see http://www.asap.cs.nott.ac.uk/chesc2011/).

    4. personnel scheduling (PS).

Each problem domain contains 10 distinct problems of varying complexity. HyFlex hides all problem specific information such as the solution representations, the solution constructions, and the low level heuristic implementations. Each HyFlex problem has four general *heuristic classes*:

1. parameterised mutation (`M`) which perturbs a solution randomly,
2. crossover (`C`) which constructs a new solution from two or more existing solutions,
3. parameterised ruin and recreate (`R`) which destroys a given solution partially and then rebuilds the deleted parts, and
4. parameterised hill climbing or local search (`L`) that incorporates an iterative improvement process and returns a non-worsening solution.

The actual number and implementation of the low level heuristics in each class differs between problem domains. As a result, it is not possible to directly compare sequences of low level heuristics from different domains. Instead, sequences of heuristic classes are compared.

Algorithm 1 The DBGen hyper-heuristic in pseudocode.

```
 1. ITERATIONS ← 150;
 2. new-sol ← initialiseSolution();
 3. new-obj ← f(new-sol);
 4. cross-sol ← initialiseSolution();
 5. cross-obj ← f(new-sol);
 6. while (ITERATIONS−− > 0) do
 7.    cur-sol ← new-sol;
 8.    cur-obj ← new-obj;
 9.    Heuristic h ← selectHeuristic();
10.    new-sol ← apply( h, new-sol, cross-sol );
11.    new-obj ← f(new-sol);
12.    double r ← ran();
13.    if (new-obj < cross-obj or r < 0.5) then
14.        cross-sol ← new-sol;
15.        cross-obj ← new-obj;
16.    end if
17.    if (new-obj ≥ cur-obj and r ≥ 0.5) then
18.        new-sol ← cur-sol;
19.        new-obj ← cur-obj;
20.    end if
21. end while
```

    The random, unbiased, single selection hyper-heuristic DBGen used to generate the offline learning database is shown in listing 1. The function *select()* (line 9) selects a single low level heuristic class at random from the set $\{C, L, R, M\}$. The function *apply()* (line 10) takes the heuristic class and chooses, again at random, an actual low level heuristic and its parameters from the available heuristics of that class. The actual heuristic is then applied to the current solution *cur-sol*, and if the class is `C`, to the current crossover solution *cross-sol*. An objective function evaluation (line 11) and an acceptance check (lines 12–20) are then

performed. The function *ran()* (line 12) returns a uniformly distributed pseu-dorandom number in the interval $(0, 1)$. If a new solution's objective value is less than the current solution's objective value *cur-obj* or *ran()* < 0.5 then it is accepted. Otherwise the new solution is rejected. The random term allows new solutions to be accepted regardless of their objective function approximately 50% of the time. Accepting states that may lead to a large increase in objective function value forces the DBGen hyper-heuristic to explore the space of low level heuristic selections instead of optimising the problem efficiently.

The DBGen hyper-heuristic is executed 40 times, for 150 selections, on the 10 problems in each of the 4 HyFlex domains. The resulting 1600 sequences of low level heuristic selections and associated objective function values are used to construct an offline learning database. The number of 40 trials was chosen because for a sufficiently large number (say $n > 30$) the central limit theorem ensures that the arithmetic mean of any observed values will be approximately normally distributed, regardless of the underlying distribution. This allows robust statistics to be calculated for each problem. The number of 150 selections was chosen after experimental observations indicated that no major improvements in objective function occurred beyond this point.

2.2   Final Log Returns and the BEST Sequences

In this study, logarithmic returns are used to measure the performance of a hyper-heuristic. The *final log return* α_f of a hyper-heuristic run or sequence s is the log return between the initial solution of a run x_0, which has an objective function value o_0, and the best final solution $x_{\min}$ found during the run, which has an objective function value of $o_{\min}$. In symbols

$$\alpha_f(s) = \log_{10}\left(\frac{o_{\min}}{o_0}\right).$$

Logarithmic returns allows us to easily compare the objective function values produced by a hyper-heuristic executing on a number of distinct problems or problem domains.

The *mean final log return* of a set of N sequences is

$$\overline{\alpha}_f(\{s_1, \ldots, s_N\}) = \frac{1}{N}\sum_{i=1}^{N}\alpha_f(s_i).$$

The function $\overline{\alpha}_f$ is the mean of log values. The anti-log of the mean of the logs is equivalent to the geometric mean. In symbols

$$\log^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\log(x_i)\right) = \sqrt[N]{x_1 \cdot x_2 \cdots x_N}$$

assuming the values x_i all have the same sign. The geometric mean is always less than or equal to the arithmetic mean, and is employed to average values

which have very different ranges. The geometric mean normalises the ranges, so that no range dominates the average. Although the use of log returns normalises the ranges of different objective functions, the log return values can still differ significantly, as some problems are harder to optimise than others. For this reason, in this study, the arithmetic mean of the final log returns $\overline{\alpha}_f$ is used in preference to the arithmetic mean of the decimal returns.

The *final unit log return* β_f is the final log return α_f divided by the sequence's length up to (and including) the minimum objective function value. That is

$$\beta_f(s) = \frac{\alpha_f(s)}{\min}.$$

The length of a sequence is important because for many real world optimisation applications the execution times of the low level heuristics and objective function evaluations can be non-trivial.

The HyFlex benchmark problems set consists of 4 problem domains, each one containing 10 problems. The set of the 40 "best" sequences in the offline database, denoted BEST, consists of the sequences with the lowest final unit log return β_f for each problem. These sequences are the shortest sequences that produce the largest decrease in the objective function value for each problem. As the offline database was generated by executing the DBGen hyper-heuristic 40 times on each of the 40 HyFlex problems, the "best" sequence for each problem is selected from a pool of 40 sequences.

2.3 Elman Networks

Elman networks are examples of *simple recursive neural networks*. They are typically applied to problems which express themselves naturally as temporal sequences such as natural language processing applications (see [8] and [14]). Such networks learn from, process, and produce sequences of data.

The training sequences are sequences of low level heuristics selections chosen from the offline learning database. Each such sequence is encoded using a *field* representation so that it can be processed by the Elman network. Specifically, each low level heuristic selection $\{M, C, R, L\}$ is encoded as a vector in $\{0, 1\}^4$ where

$$M = (1, 0, 0, 0)$$
$$C = (0, 1, 0, 0)$$
$$R = (0, 0, 1, 0)$$
$$L = (0, 0, 0, 1),$$

and $X = (0, 0, 0, 0)$ denotes a missing or unknown selection. These vectors are then concatenated to form an input pattern. For example, given the sequence MCRLR, an input pattern of 4 low level heuristic selections, corresponding to the current selection L and the three past selections MCR is

$$(\overbrace{1, 0, 0, 0}^{M}, \overbrace{0, 1, 0, 0}^{C}, \overbrace{0, 0, 1, 0}^{R}, \overbrace{0, 0, 0, 1}^{L})$$

while the output pattern corresponding to the next selection in the sequence is

$$\overbrace{(0,0,1,0)}^{\text{R}}.$$

The number of selections to be used as an input is termed the memory length of a selection strategy (see [15]). Using the current heuristic selection and those prior to it as inputs provides context for the next selection.

Initial experiments with memory length show that Elman network learning improves significantly as the number of past selections increases. Figure 1 shows the results of training an Elman network with a memory length of 1, 2, 3, 4 and 5, on the INTRA training sequences for each domain (see Section 2.4). It should be noted that increasing the number of past selections also increases the number of weights which also improves learning.

In this study, a memory length of 4 is used because, with this number, the Elman network learns 80% (or more) of each training set. Thus, the 3-layer Elman network used in this experiment has 16 input units, 16 hidden units (and therefore 16 context units), 4 output units, and 596 weights. The hidden and output units employ the sigmoid activation function. The number of 16 hidden units was chosen arbitrarily.

Fig. 1: The percentage of LLH training errors for an Elman network with 4, 8, 12, 16 and 20 inputs, 16 hidden units, and 4 output units, for each domain.
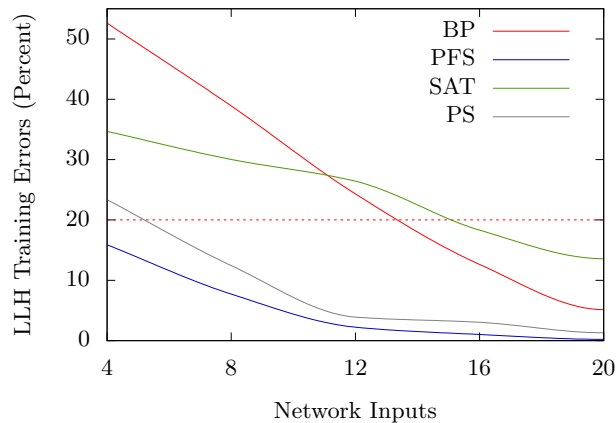

After training, given some initial input, an Elman network produces a sequence of outputs. The output sequence may converge to a single point, a limit cycle of repeating values, or produce a chaotic non-repeating sequence.

2.4 Training Sets

This study is concerned with offline intra-domain and inter-domain learning of heuristic classes. In intra-domain learning, the training sequences and the test optimisation problem are drawn from the same problem domain. This simplifies the learning task considerably as the low level heuristics in each class are identical for each problem and so the heuristic classes will have similar statistical characteristics across the problems of the domain. This is not generally the case for inter-domain learning where the training sequences and test problem can be drawn from different domains. These different domains will have different low level heuristic implementations and so the heuristic classes can have different statistical characteristics in each domain (see figure 2). However, the general underlying principles of each heuristic class should remain similar, for example a mutation operation should make small random changes, while a local search operation will greedily search the surrounding space.

Fig. 2: The scaled mean log returns $\overline{\alpha}$ of the heuristic classes C, L, M, and R for each domain. In each domain the $\overline{\alpha}$ values have been scaled by the largest absolute $\overline{\alpha}$ value into the interval $[-1, 1]$.
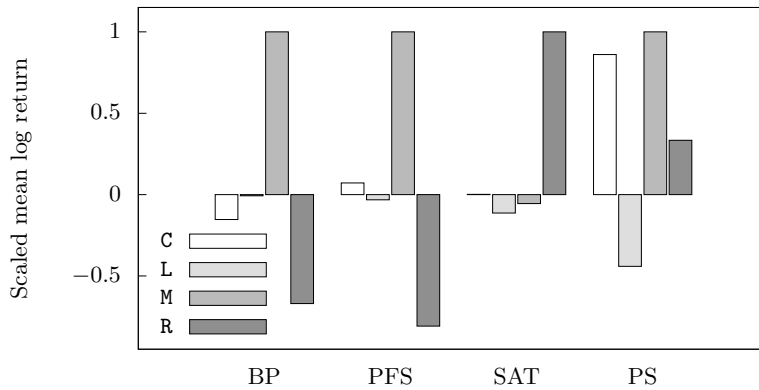
The training sets for intra-domain and inter-domain learning are constructed from the BEST heuristic class sequences. As these sequences are the most efficient optimisations of each problem available they contain the most "useful information" regarding that problem and therefore they are prime candidates for inputs to a machine learning algorithm. In this study, *leave-one-out* cross-validation (see [16]) is employed to determine whether the Elman network sequences are able to outperform the BEST training sequences.

For intra-domain learning, the BEST subsequences are divided by domain into 4 sets of 10 sequences. For each problem in a domain, the sequence for that problem is left out of the training set and the remaining 9 sequences are used to train a network. The sequence produced by the trained network is then evaluated on the problem that was left-out. Thus the sequence generated by the network is always evaluated on a problem that the network has not been trained on. Applying this methodology gives rise to 40 training sets of 9 sequences, one for each problem, constructed from the 10 sequences selected for each domain.

For inter-domain learning, the BEST subsequences are again divided by domain into 4 sets of 10 sequences. For each domain, 3 sequences are selected from each of the 3 remaining domains. These sequences correspond to the problems with the lowest β_f in those domains. Applying this methodology gives rise to 4 training sets of 9 sequences, one for each domain, constructed from the 9 sequences selected from the other domains.

In each case, for each problem, the Elman network is trained with 9 sequences drawn from the set BEST. It should be noted that for network training, only the accepted selections of each sequence up to (and including) the minimum objective function value are used. Rejected selections, and those selections that occur after the minimum objective function value are not used.

2.5 The BLIND Hyper-heuristic

The BLIND hyper-heuristic is used to evaluate sets of heuristic sequences on the HyFlex problems. It is intended to serve as a simple test bed and a "level playing field", in order to evaluate and compare the performance of sequences. The sequence based hyper-heuristic BLIND used in these experiments blindly applies a given sequence, one low level heuristic class after another to a HyFlex problem, accepting every selection. The actual low level heuristics and their parameters are chosen at random.

3 Results

Section 3.1 presents the results of training the Elman networks with the intra-domain and inter-domain training sequences. In Section 3.2 the sequences that are generated by the trained networks are evaluated on the HyFlex problems using the BLIND hyper-heuristic.

3.1 Network Training

An Elman network is trained with the intra-domain and inter-domain training sets using stochastic Backpropagation with early stopping over a maximum of 1000 epochs (see [16]) using the parameters shown in table 1. The learning rate, momentum term, and the number of training epochs have not been optimised.

The results of network training are summarised in table 2 and figure 3. Table 2 shows the results of training the Elman network with the 40 intra-domain

Table 1: The Elman network structure and training parameters.

Input	Hidden	Out	Learn	Momentum	Epochs
16	16	4	0.1	0.25	1000

training sets. The results are averaged over the 10 training sets in each domain. The columns show the average number of low level heuristics in each set, the average percentage of low level heuristics incorrect after training, the average network root mean square error, and the average number of epochs. Low level heuristic correctness is determined by applying a winner-take-all strategy to the network's output units and comparing the network's choice of heuristic with the target heuristic. Figure 3a shows the percentage of low level heuristic errors dur-

Table 2: The averaged training results of the Elman network on the intra-domain training sets.

| Dom.	Num.	Wrong (%)	Error	Epochs
BP	369.0	12.6407	4.2958	907.7
PFS	94.5	1.0260	1.0491	328.9
SAT	288.2	18.3158	4.1991	918.9
PS	121.5	3.0474	0.9290	947.3

ing intra-domain training for 4 representative problems (number 7, 19, 34, and 14) chosen from the BP, PFS, SAT and PS domains. These results demonstrate that the difficulty of learning intra-domain sequences of heuristic selections varies by domain. For example, the SAT domain sequences are much harder to learn than the training sequences of the other domains.

Table 3: The averaged training results of the Elman network on the inter-domain training sets.

Dom.	Num.	Wrong (%)	Error	Epochs
BP	151	1.7391	1.0443	999
PFS	224	1.0638	1.3208	994
SAT	175	0.7194	0.8031	616
PS	221	1.0810	0.9290	739

Fig. 3: The Elman network training results for the intra-domain and inter-domain sets. In figure (a) the training sequences are drawn from the BP, PFS, SAT and PS domains. In figure (b) the training sequences are drawn from the {PFS SAT PS}, {BP SAT PS}, {BP PFS PS}, and {BP PFS SAT} domains.

(a) Intra-domain training results.   (b) Inter-domain training results.
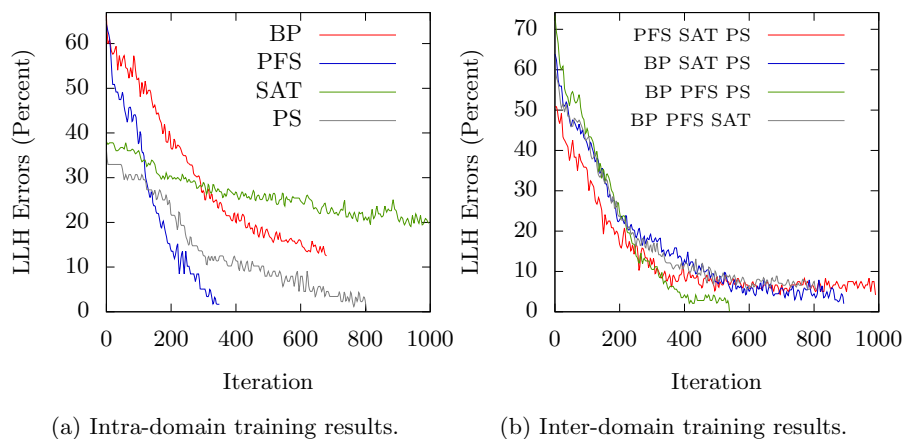
Similarly, table 3 and figure 3b show the results of training the Elman network with the 4 inter-domain training sets. These results demonstrate that intra-domain learning is harder than inter-domain learning.

After training, the Elman network is then given the initial "blank" input XXXX. As Elman networks are deterministic, the intra-domain trained networks produces a set of 40 sequences, one for each problem, while the inter-domain trained networks produce a set of 4 sequences, one for each domain.

3.2   Evaluating the Elman Network Sequences

The BLIND hyper-heuristic is parameterised with three sets of sequences denoted BEST, INTRA, and INTER and then executed 40 times on each of the HyFlex problems. The INTRA sequence set is generated by the intra-domain trained Elman networks, while the INTER sequence set is generated by the inter-domain trained Elman networks. It should be noted that the pseudorandom number seeds and therefore the initial solutions used for the INTRA, INTER, and BEST evaluation runs presented here are identical and distinct to the pseudorandom number seeds used by DBGen to generate the offline database from which the BEST sequences are selected.

When parameterised with the BEST sequences the BLIND hyper-heuristic applies *all* the accepted selections including those *after* the minimum objective function value. This is done because some sequences in BEST find a minimum quickly, in some cases after only 9 selections. Using all accepted selections gives

the BLIND hyper-heuristic a larger number of iterations/selections to better optimise a problem. The length of the BEST sequences also dictate the number of selections used by the INTRA and INTER parameterisations. The results of evaluating the INTRA and INTER sequence sets on the HyFlex problems are compared to the BEST sequences (see table 4). The intention of the comparison is to determine whether the network has learned anything over and above the information contained in the BEST sequences. The INTRA sequences outperform the BEST sequences overall and on each domain, while BEST outperforms INTER overall, and on each domain except the PFS domain. The best generalisation is observed between INTRA and BEST on the SAT domain (which was the hardest to learn). The overall averages are calculated over 1600 sequences, and the domain averages are calculated over 400 sequences.

Table 4: A domain by domain and overall comparison of the mean final log return $\overline{\alpha}_f$ of BEST, INTRA and INTER.

Dom.	BEST	INTRA	INTER
BP	-0.2172	**-0.2202**	-0.0375
PFS	-0.0043	-0.0049	**-0.0051**
SAT	-0.4345	**-0.6919**	-0.2313
PS	-1.7912	**-1.8042**	-1.5560
All	-0.6118	**-0.6803**	-0.4575

A paired t-test is used to establish whether the difference observed in the mean final log returns of BEST and INTRA is statistically significant. Formally, the null hypothesis

$$\overline{\alpha}_f(\text{BEST}) \geq \overline{\alpha}_f(\text{INTRA})$$

is rejected if t lies outside the interval $[-2.3287, \infty)$ and the alternative hypothesis

$$\overline{\alpha}_f(\text{BEST}) < \overline{\alpha}_f(\text{INTRA})$$

is accepted with 99% confidence. The results of the t-test are shown in table 5. The difference in mean is statistically significant overall, and for the PFS and SAT domains with 99% confidence. For the BP and PS domains the difference in mean is not statistically significant.

4 Conclusions

The sequence set BEST consists of the sequences with the lowest final unit log return β_f for each HyFlex problem. An intra-domain training set INTRA and an inter-domain training set INTER are constructed from the BEST sequences and used to train an Elman network. In order to estimate the Elman network's

Table 5: The domain, the sample mean difference, the standard deviation, the t score, and the interval within which the population mean difference falls with 99% confidence.

Dom.	Diff.	SD	t	Conf. Int.
BP	-0.0030	0.0821	-0.7214	[-0.0136, 0.0077]
PFS	-0.0006	0.0024	-5.2796	[-0.0009, -0.0003]
SAT	-0.2573	0.1085	-47.4485	[-0.2714, -0.2433]
PS	-0.0130	0.1225	-2.1289	[-0.0289, 0.0028]
All	-0.0685	0.1424	-19.2384	[-0.0777, -0.0593]

capacity for generalisation the network is evaluated using a *leave-one-out cross-validation* methodology. The first result presented in this study demonstrates that the Elman network is capable of intra-domain generalisation with 99% confidence. This result is notable because the Elman network is able to significantly outperform the sequences on which it was trained. The process of generalisation across the training problems within a domain has generated a network that is able to perform better on unseen test problems in that domain. This shows that useful information can be learned about the problems in a domain from the sequences of heuristic selections used to optimise them. The second result shows that the Elman network is not capable of inter-domain generalisation using the training set INTER in spite of the fact that the training sets are easier to learn. This suggests that inter-domain generalisation is harder than intra-domain generalisation, and that low training errors need not translate into good generalisations. This was generally to be expected, the sequences of heuristics learned on one domain are not expected to be applicable to another. However, there are exceptions, for example the performance on PFS domain from the INTER trained network performed well and indicates perhaps that a more general strategy for solving the PFS domain would be successful.

Overall, the Elman network proved to be able to generalise the training sequences for intra-domain learning which opens up the possibility of the use of bespoke learned algorithms for particular problems. Inter-domain generalisation was more difficult, as expected, and more work would need to be conducted to determine whether a different methodology would allow domains with similar sequences to be identified.

References

[1] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward, *A Classification of Hyper-heuristic Approaches.* Springer US, 2010.

[2] P. Ross, S. Schulenburg, J. G. Marín-Bläzquez, and E. Hart, "Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO

2002, (San Francisco, CA, USA), pp. 942–948, Morgan Kaufmann Publishers Inc., 2002.

[3] E. K. Burke, S. Petrovic, and R. Qu, "Case-based heuristic selection for timetabling problems," *Journal of Scheduling*, vol. 9, no. 2, pp. 115–132, 2006.

[4] H. Terashima-Marín, J. C. Ortiz-Bayliss, P. Ross, and M. Valenzuela-Rendón, "Hyper-heuristics for the dynamic variable ordering in constraint satisfaction problems," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO 2008, (New York, NY, USA), pp. 571–578, ACM, 2008.

[5] A. Kheiri and E. Keedwell, "A sequence-based selection hyper-heuristic utilising a hidden Markov model," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO 2015, pp. 417–424, ACM, 2015.

[6] W. B. Yates and E. C. Keedwell, "Clustering of hyper-heuristic selections using the Smith-Waterman algorithm for offline learning," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, (Berlin), pp. 119–120, ACM, 2017.

[7] G. Ochoa, M. Hyde, T. Curtois, J. A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. J. Parkes, S. Petrovic, and E. K. Burke, "HyFlex: A benchmark framework for cross-domain heuristic search," in *Evolutionary Computation in Combinatorial Optimization* (J. K. Hao and M. Middendorf, eds.), pp. 136–147, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[8] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[9] J. D. Walker, G. Ochoa, M. Gendreau, and E. K. Burke, "Vehicle routing and adaptive iterated local search within the HyFlex hyper-heuristic framework," in *Learning and Intelligent Optimization - 6th International Conference, LION 6, Paris, France, January 16-20, 2012, Revised Selected Papers*, pp. 265–276, 2012.

[10] J. H. Drake, E. Özcan, and E. K. Burke, "An improved choice function heuristic selection for cross domain heuristic search," in *Parallel Problem Solving From Nature (PPSN XII), Lecture Notes in Computer Science* (C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, eds.), vol. 7492, pp. 307–316, 2012.

[11] M. Mısır, K. Verbeeck, P. D. Causmaecker, and G. V. Berghe, "A new hyper-heuristic as a general problem solver: an implementation in HyFlex," *Journal of Scheduling*, vol. 16, no. 3, pp. 291–311, 2013.

[12] J. H. Drake, E. Özcan, and E. K. Burke, "A comparison of crossover control mechanisms within single-point selection hyper-heuristics using HyFlex," in *IEEE Congress on Evolutionary Computation (CEC)*, (Sendai, Japan), pp. 3397–3403, May 2015.

[13] P. Dempster and J. H. Drake, "Two frameworks for cross-domain heuristic and parameter selection using harmony search," in *Harmony Search Algorithm: Proceedings of the 2nd International Conference on Harmony Search Algorithm (ICHSA2015)* (H. J. Kim and W. Z. Geem, eds.), (Berlin, Heidelberg), pp. 83–94, Springer Berlin Heidelberg, 2016.

[14] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," *Machine Learning*, vol. 7, pp. 195–224, 1991.

[15] R. Bai, E. K. Burke, M. Gendreau, G. Kendall, and B. McCollum, "Memory length in hyper-heuristics: An empirical study," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling*, pp. 173–178, 2007.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

POSTERS

Reinforcement learning is an effective strategy to create phenotypic variation and a potential mechanism for the initial evolution of learning.

Jan Teichmann[1], Eduardo Alonso[2], and Mark Broom[1]

[1] Department of Mathematics, City University of London, Northampton Square, London EC1V 0HB jan.teichmann@city.ac.uk Mark.Broom@city.ac.uk
[2] Department of Computer Science, City University of London, Northampton Square, London EC1V 0HB E.Alonso@city.ac.uk

Abstract Evolution leads to animals being well-adapted to their environment, in terms of physical abilities and behaviours. Many animal behaviours, however, are not simply genetically determined, but are a result of learning. Learning is generally assumed to be an adaptation to environmental change, but the relationship between environmental factors, learning, and evolution is complex and not fully understood. Here, the role of reinforcement learning is of increasing interest. We present a general model inspired by evolutionary games which analyses and compares fitness distributions of individuals in a changing environment which either learn or evolve through mutation. We show that reinforcement learning offers a potential mechanism for the initial evolution of learning irrespective of any technical parameters and confirm previous findings of other established models of learning.

1 Introduction

Through evolution, animals are generally very well-adapted to their environment. Phenotypic plasticity allows for suitable adaptations even in the face of changing environments [12]. Thus both physical abilities and behaviours of animals are generally appropriate to their environment. Nevertheless many animal behaviours are not solely genetically determined, though some are, but the response of the animal's learning capabilities. Hence a key question arises: under which conditions is the ability to learn beneficial? From a biological perspective, learning is a mechanism for rapid adaptation (modification) of behaviour during the individual's lifetime and a distinct adaptation to changing environments in particular [7]. The main line of argument is that learning incurs some cost, so that a constant environment should select for a genetically fixed pattern of behaviour over learned behaviour. But the relationship of learning and evolution is complex and an important aspect of learning is environmental predictability (commonly referred to as regularity) [15]. Clearly, there is nothing to learn in an environment which is absolutely unpredictable. So far both factors, environmental change and regularity, have been discussed in the literature as selective

factors in the evolution of learning. A contradiction at first sight, but a solution to the paradox would be that learning is in fact an adaptation to intermediate levels of environmental change [8].

Associative learning is a fundamental cognitive process observed across species (including mollusks, insects, birds and mammals) [11] that affects a wide variety of behaviours ranging from colour recognition [2, 9, 13, 20] and spatial representation [1], to causality judgements [14] and goal-directed behaviour [21]. Of course, animals use other types of learning (e.g. social learning or perceptual learning) and ontogenetic mechanisms (e.g. habituation and phenotypic plasticity) to adapt their behaviour to the environment. Nonetheless, the pervasiveness and relevance of associative learning makes it the ideal candidate to investigate when learning is most effective. We are interested in the initial evolution of learning [3, 17] and in particular how reinforcement learning [4, 5] could evolve within a population of generally well adapted animals. To answer this question we focus on a simple model of learning where individuals learn to associate events that occur together, for instance two stimuli, a stimulus and a response, or a response and its outcome [10].

2 Model Definition

Our work develops that of [18, 19], which investigated the effects of aversive learning in a changing environment on a predator's diet choice and energy intake. Here we describe fitness distributions of learning individuals in changing environments more generally and compare them with a simplistic mutation process to understand the relationship between evolution and learning.

In our model the learning individual uses Q-learning [22], chosen for the simplicity of its implementation of real-time error-correction learning and as it is increasingly supported by both behavioural and neural data. In Q-learning an individual uses experience following its interactions with the environment to infer optimal decisions. The learning individual utilises an action-value function to build a representation of the environment which describes the expected future payoff following a specific action in a specific state of the environment. It then minimises the error of the function's future payoff prediction building on a growing amount of evidence from past trial-and-error interactions with the environment. These payoff predictions are discounted by a factor γ, representing future uncertainty. The prediction error is modulated by a learning rate α. Finally, the individual translates the action-value function predictions into a decision following a stochastic policy, e.g. Gibb's soft-max policy.

An individual of the mutating population has a genetically determined decision policy chosen randomly from a uniform distribution at the beginning of each generation. The important differences between the two populations are: (i) the mutation process is random and not adaptive, operating on fixed phenotypes and (ii) learning is adaptive but incurs an exploration cost, where suboptimal decisions are made to learn about the environment. Selection is not included as we are only interested in both populations' fitness distributions.
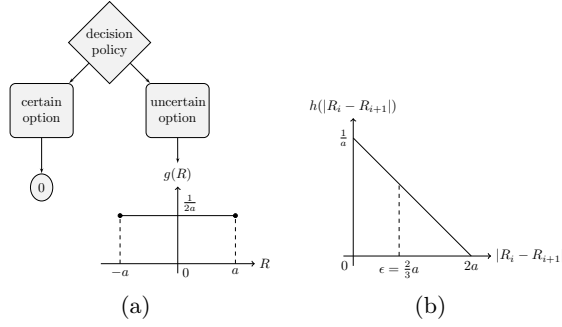
Figure 1. The environment with the choice of a certain and an uncertain option. **(a)** The two options of the environment with the certain option being equal to zero and the uncertain option following a uniform fitness payoff distribution $g(R)$ with limits $-a$ and a as given by Equation (1). **(b)** The distribution of absolute fitness change follows a triangular distribution $h(|R_i - R_{i+1}|)$ with $\epsilon = (2/3)a$ being the average absolute fitness change given the uniform distribution of fitness payoff $g(R)$ with $\beta = 1$.

We define the environment to be stationary and ergodic, consisting of two options, a certain and an uncertain one, as shown in Figure 1. The model parameters are described as follows: (i) R: the fitness payoff following an interaction with the environment, (ii) F: the fitness of an individual at the end of a generation, (iii) $\hat{F}$: the scale-free fitness of an individual, (iv) l: the length of a generation in interactions with the environment, (v) β: the number of environmental changes per generation time, (vi) ϵ: the extent of environmental change per generation time, (vii) α: the learning rate of the learning individual and (viii) γ: the discount rate of future payoffs. The certain option gives a constant fitness payoff $R = 0$ and the uncertain option returns a uniformly distributed fitness payoff $g(R)$ with zero mean. The value of 0 for the fitness of the constant option and the mean of the variable option is chosen for simplicity (it is possible to add an arbitrary constant to both and not qualitatively change our results).

The learning individual cannot draw from any secondary source of information such as a correlation between environmental states which motivates the choice of the uniform distribution $g(R)$. The term regularity refers to the predictability of an environment within models of learning. A perfectly regular environment is one that is constant throughout the lifetime of an individual, with irregularity increasing as the number of changes per lifetime increases. We define irregularity as the expected number of environmental changes in a lifetime, given by β/l. This is perhaps a simplistic definition, and we do not make direct use of it, except that the environment becomes increasingly irregular with greater (smaller) values of β (l). We define the limits $[-a, a]$ of the Uniform distribution $g(R)$ as follows:

$$a = \frac{3}{2\beta}\,\epsilon, \tag{1}$$

where ϵ is the absolute average fitness change per generation derived from the triangular distribution $h(|R_i - R_{i+1}|)$ of the absolute difference of the uniform fit-

ness payoff $g(R)$ as shown in Figure 1. We assume that an increased frequency of environmental change β results in smoother and less pronounced single changes as reflected in Equation (1). The fitness F of an individual is the sum of the fitness payoffs from its interactions with the environment $F = \sum_{t=1}^{l} R_t$.

3   Results

We present the distributions of $n = 5000$ generations interacting with their environment using a scale free variant of the fitness $\hat{F} = \beta F / l\epsilon$, which will allow a more intuitive comparison of the two populations. We will present the results for each population respectively in the form of box-plots and associated Kolmogorov-Smirnov significance tests.

Figures 2a and 2b show the main characteristic of the mutation process: as the process is random and not adaptive it is independent of the number of inter-actions per generation l and of the extent of environmental change per generation ϵ. The fitness distributions are also symmetric with mean zero. Additionally, α and γ do not apply to the mutation process. Figure 2c shows the effects of the frequency of environmental changes β. The fitness distribution is unaffected for $\beta \leq 1$, i.e. when mutations occur more frequently than changes in the environ-ment. If $\beta > 1$ the fitness distribution of the population of mutating individuals becomes increasingly narrow. This is a direct result of the mutation process be-ing non-adaptive and therefore it is less likely that individuals are well suited (or poorly suited) for a number of consecutive environmental states.
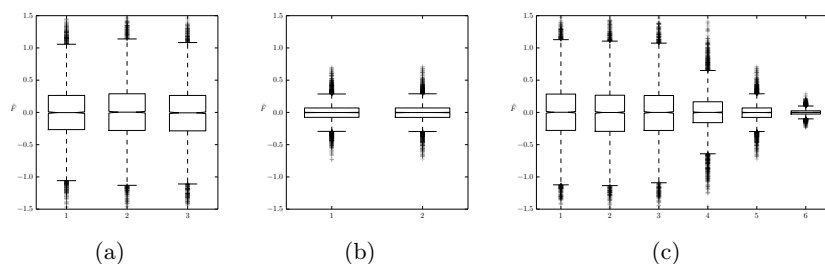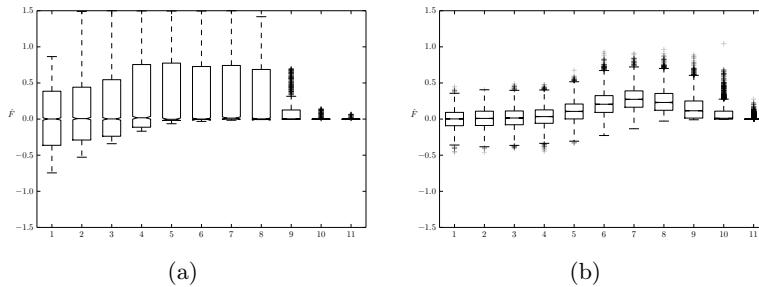

Figure 2. Scale-free fitness distributions of the mutating population, all with $n = 5000$ generations. **(a)** The fitness distribution is independent of ϵ and l. (1) $\epsilon = 0.1$, $\beta = 1$, and $l = 1000$. (2) $\epsilon = 100$, $\beta = 1$, and $l = 1000$. (3) $\epsilon = 10$, $\beta = 1$, and $l = 10$. Distributions are not significantly different using a Kolmogorov-Smirnov test, all with $p > 0.1$. **(b)** Fitness distributions scale equally with β independently of ϵ and l. (1) $\epsilon = 10$, $\beta = 10$, $l = 1000$. (2) $\epsilon = 1$, $\beta = 10$, $l = 10$. Distributions are not significantly different using a Kolmogorov-Smirnov test, all with $p > 0.1$. **(c)** Fitness distributions become narrower with increasing β. Distributions are not significantly different for $\beta \leq 1$ using a Kolmogorov-Smirnov test, all with $p > 0.1$. (1) $\beta = 0.1$, (2) $\beta = 0.5$, (3) $\beta = 1$, (4) $\beta = 2$, (5) $\beta = 10$, (6) $\beta = 100$. In all cases $\epsilon = 1$, $l = 1000$.
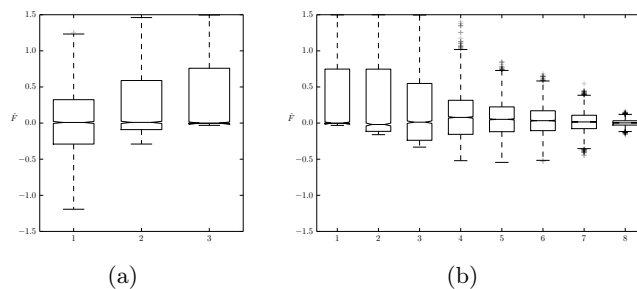
Figure 3. Scale-free fitness distributions of a learning individual. All cases are with $n = 5000$ generations. **(a)** Learning requires a certain amount of environmental change to occur to be beneficial: (1) $\epsilon = 0.1$, $l = 10$ vs. (2) $\epsilon = 10$, $l = 10$. Learning benefits from longer generation times: (2) vs. (3) $\epsilon = 10$, $l = 1000$. In all cases $\beta = 1$, $\alpha = 0.5$, $\gamma = 0.9$. **(b)** Learning incurs exploration cost and learning benefits diminish in unreliable environments. (1) $\beta = 0.1$, (2) $\beta = 0.5$, (3) $\beta = 1$, (4) $\beta = 2$, (5) $\beta = 3$, (6) $\beta = 5$, (7) $\beta = 10$, (8) $\beta = 100$. In all cases $\epsilon = 1$, $l = 1000$, $\alpha = 0.5$, $\gamma = 0.9$.

The population of learning individuals uses Q-learning to adapt to the current environmental state. This requires exploration which is the sole cost of learning in our model. Additional costs of learning are difficult to quantify and we assume that during the initial evolution of learning these were relatively small [7]. Figure 3a shows that learning requires certain environmental conditions; in particular learning benefits from a changing environment (Figure 3a1 vs. 3a2). The cost and benefits of exploration in the learning population can be seen (Figure 3a1) versus the mutating population (Figure 2a) where the learning behaviour cuts off both tails of the fitness distribution and does not produce the outliers as in the mutation process. Additionally, learning benefits from longer generation times to exploit experience (Figure 3a3).

In Figure 3b we see that learning is directly affected by β compared to the population of mutating individuals which is unaffected for $\beta \leq 1$ (Figure 2c). The effect of β on the fitness distribution of the learning individuals is not linear, and there are multiple underlying factors. In environments with very rare changes an increasing majority of the population benefits from learning (Fig 3b1). At first, an increasing frequency of environmental change increases the fraction of individuals benefiting less from learning with the fitness distribution developing a more pronounced tail of learning individuals having negative relative fitness (Figure 3b1-5). Nevertheless, environmental change benefits learning at the same time with the median of the population increasing (Figure 3b4). Secondly, a further increase of β results in the cost of consecutive exploration and consequent errors outweighing this initial benefit and the distribution increasingly aligns with the fitness distribution of the mutating population. Finally, learning does not provide any benefits in highly irregular environments interfering with any possibility of exploitation. Therefore the only difference in the fitness distributions is the shorter tails for the learning individuals, which is the result of continuous exploration (Figure 3b8 vs. Figure 2c6).

250

Figure 4. Scale-free fitness distributions of the learning individuals showing the effects of the extent of environmental change (ϵ). Learning benefits from a certain extent of environmental change but too severe changes incur a high cost of mistakes during the required exploration. Additionally, there is a combined effect of regularity and change. (1) $\epsilon = 0.1$, (2) $\epsilon = 0.5$, (3) $\epsilon = 1$, (4) $\epsilon = 2$, (5) $\epsilon = 5$, (6) $\epsilon = 10$, (7) $\epsilon = 20$, (8) $\epsilon = 50$, (9) $\epsilon = 100$, (10) $\epsilon = 500$, and (11) $\epsilon = 1000$. All cases are with $n = 5000$ generations, $l = 1000$, $\alpha = 0.5$, and $\gamma = 0.9$. **(a)** $\beta = 1$. **(b)** $\beta = 10$.

Figure 4 shows the combined effect of ϵ and β. We have already discussed the individual effects of β relating to Figure 3b. For ϵ we can see that in environments with small values of absolute environmental change throughout a generation the fitness distribution of the learning individuals aligns with the fitness distribution of the population of mutating individuals. A learning individual prioritises continuous exploration if the environmental change is small which is a consequence of learning being an adaptation to changing environments. The shorter tails in the fitness distribution of the learning population compared to the mutating population are the result of this continuous exploration as discussed previously (Figure 4a1 vs. Figure 2a and Figure 4b1 vs. Figure 2b). An increase of ϵ has beneficial effects for learning individuals as learning requires a certain extent of environmental change to exploit. A further increase of ϵ makes mistakes during exploration more expensive which can potentially neutralise the benefits of exploiting beneficial states of the environment. The important difference between a severe extent of environmental change (ϵ) and an irregular environment (β) is that mistakes in the case of ϵ are extremely aversive and stop any further costly exploration. This is why the fitness distribution of learning individuals in violently changing environments loses the negative tail compared to rapidly changing environments (Figure 4a11 vs. Figure 3b8). Combining frequency and extent of environmental change shows that there is a specific combination of these factors which hugely benefits the learning individuals (Figure 4b7).

Our results also show that the fitness distribution of the learning population is independent of the learning rate α and the discount factor γ, within a meaningful range (clearly α cannot be too low; $\alpha = 0$ corresponds to no learning at all). This result should not be misinterpreted: there are specific values of α and γ best suited for achieving optimality in a specific state of the environment. But within a changing environment the distribution of fitness is independent of the specific choice of α and γ.

4 Discussion

In this paper we look at the fitness distribution of individuals using reinforcement learning, i.e. Q-learning, in a changing environment. Our model confirms previous findings: (i) learning requires environmental change and longer generation times to be beneficial, (ii) learning is optimal for specific combinations of regularity and size of environmental change, and (iii) regularity is the key environmental factor which impacts whether learning is advantageous. In addition, we show that (iv) the fitness distribution of learning individuals in changing environments is independent of the learning rate α and the discount factor γ.

We do not present an evolutionary theory of learning in itself, but show that a simple reinforcement strategy (increasingly backed by experimental studies of neural correlates) is beneficial for a vast range of environmental parameters. In particular, the fact that the success of reinforcement learning is independent of technical parameters of learning α and γ, is a new and reassuring insight. These are technical parameters which allow the tuning of over-fitting and the extent of exploration for a specific learning task and have great importance in the field of computing. But in a biological context of changing environments these technical learning parameters become negligible. This significantly reduces the complexity of the initial evolution of reinforcement learning in comparison to the combinatorial explosion of the parameter space in models of connectionism [6].

Our model does not include interactions between individuals. Nevertheless, it reproduces many widely accepted theories of learning in the context of change and regularity [17]. In addition, it has been widely acknowledged that the benefit of learning is the ability to adapt to a changing environment faster than the time scale on which evolution operates [7]. This is an important evolutionary dynamical argument, but is distinct from those of our paper; here the benefits of learning lie in exploitation ability rather than adaptation speed itself.

Considering the effects of selection the mutating population in our model has a constant relative arithmetic mean fitness of zero. The environmental changes only affect the fitness variability of the mutating population in a symmetric fashion. As the arithmetic payoff of both options in our model are equal, selection would increase long-term fitness of the mutating population by discarding the uncertain option from the action space of the mutation process in order to reduce fitness variability [16, 18]. This provides an alternative interpretation of why learning is a distinct adaptation to changing environments alongside the cost argument: a simplistic mutation process cannot exploit environmental change without the introduction of increased fitness variability at the same time.

Taking selection into account, our results show that reinforcement learning is a promising starting point for the initial evolution of learning. The key environmental factor here is regularity. If selection cannot discard the uncertain option from the action space of the mutational process, learning is always beneficial as it has lower fitness variability even in extremely irregular environments when compared to the mutating population. If selection can in fact discard the uncertain option in the case of the mutating population then learning becomes disadvantageous in highly irregular environments.

Acknowledgements

References

1. Albasser, M.M., Dumont, J.R., Amin, E., Holmes, J.D., Horne, M.R., Pearce, J.M., Aggleton, J.P.: Association rules for rat spatial learning: The importance of the hippocampus for binding item identity with item location. Hippocampus 23, 1162–1178 (2013)
2. Carew, T.J., Hawkins, R.D., Kandel, E.R.: Differential classical conditioning of a defensive withdrawal reflex in aplysia californica.. Science 219, 397–400 (1983)
3. Chalmers, D.J.: The evolution of learning: An experiment in genetic connectionism. In: Proceedings of the 1990 connectionist models summer school. pp. 81–90 (1990)
4. Dayan, P., Daw, N.D.: Decision theory, reinforcement learning, and the brain. Cognitive, Affective, & Behavioral Neuroscience 8, 429–453 (2008)
5. Doya, K.: Reinforcement learning: Computational theory and biological mechanisms. Human Frontiers Science Program Journal 1, 30–40 (2007)
6. Hinton, G.E., Nowlan, S.J.: How learning can guide evolution. Complex systems 1(3), 495–502 (1987)
7. Johnston, T.D.: The selective costs and benefits of learning: an evolutionary analysis. Adv. Stud. Behav 12, 65–106 (1982)
8. Johnston, T.D., Turvey, M.T.: A sketch of an ecological metatheory for theories of learning. The psychology of learning and motivation 14, 147–205 (1981)
9. Leimar, O., Enquist, M., Tullberg, B.S.: Evolutionary stability of aposematic coloration and prey unprofitability: A theoretical analysis. American Naturalists 128, 469–490 (1986), http://www.jstor.org/stable/2461331
10. Mackintosh, N.J.: The psychology of animal learning. Academic Press (1974)
11. Macphail, E.M.: Brain and intelligence in vertebrates. Claredon (1982)
12. Pigliucci, M.: Phenotypic plasticity: beyond nature and nurture. JHU Press (2001)
13. Ruxton, G.D., Sherratt, T.N., Speed, M.P.: Avoiding Attack: The Evolutionary Ecology of Crypsis, Warning Signals and Mimicry. Oxford University Press (2004)
14. Shanks, D.R.: The psychology of associative learning. Cambridge University Press (1995)
15. Staddon, J.E., Simmelhag, V.L.: The" supersitition" experiment: A reexamination of its implications for the principles of adaptive behavior. Psychological Review 78, 3–43 (1971)
16. Starrfelt, J., Kokko, H.: Bet-hedging—a triple trade-off between means, variances and correlations. Biological Reviews 87(3), 742–755 (2012)
17. Stephens, D.W.: Change, regularity, and value in the evolution of animal learning. Behavioral Ecology 2, 77–89 (1991)
18. Teichmann, J.: Models of aposematism and the role of aversive learning. Ph.D. thesis, City, University of London (2014)
19. Teichmann, J., Broom, M., Alonso, E.: The application of temporal difference learning in optimal diet models. Journal of theoretical biology 340, 11–16 (2014)
20. Teichmann, J., Broom, M., Alonso, E.: The evolutionary dynamic of aposematism: a numerical analysis of co-evolution in finite populations. Math. Model. Nat. Phenom. 9, 148–164 (2014)
21. Valentin, V.V., Dickinson, A., O'Doherty, J.P.: Determining the neural substrates of goal-directed learning in the human brain. The Journal of Neuroscience 27, 4019–4026 (2007)
22. Watkins, C., Dayan, P.: Q-learning. Machine learning 8, 279–292 (1992)

Learning new Term Weighting Schemes with Genetic Programming

Ahmad Mazyad, Fabien Teytaud, and Cyril Fonlupt

LISIC, ULCO, Université du Littoral Côte d'Opale

Abstract. Text Classification (or Text Categorization) is a popular machine learning task which consists in assigning categories to documents. Feature weight methods are classic tools that are used in text categorization in order to assign a score to each term of a document based on a mathematical formula. In this paper, we are interested in automatically generating these formulas based on genetic programming. We experiment the generated formulas on three well-known benchmarks and state of the art classifiers.

1 Introduction

Text classification is an important task, with numerous applications, such as web search, document classification, information retrieval [5, 16, 15]. Over time, numerous text classification methods have appeared [6], such as k-nearest neighbor [22], Naïve Bayes [13], decision trees [1], neural networks [14], boosting methods [18] and Support Vector Machines [2].

For text classification a crucial point is to represent text documents in a suitable format recognizable by a classifier. In the Vector Space Model (VSM) representation, a document is represented by a vector of terms and each term is associated with a weight. This weight represents how informative/discriminative the correspondent term is. The method which assigns a weight to a term is called Term Weighting Scheme (TWS).
Numerous TWS exist and the most used are presented in Section 2. They are generated according to human a priori and mathematical rules. TWS are usually simple mathematical expressions. Unfortunately, depending on the application, it is not easy to know a priori which TWS will be effective.

As expression discovery may naturally be addressed by genetic programming, we are interested in this paper to study whether it is possible to generate TWS automatically using this programming approach. We are interested to know if a stochastic evolutionary process with no information about the complexity, the shape and the size of the expression can find at least competitive discriminative TWS.

The paper is organized as follows : Section 2 presents TWS and the main state-of-the-art methods. In section 3 we present Genetic Programming. Section 3.2 presents how we apply genetic programming to TWS. Section 4 presents the experiments and the results, and then we conclude in section 5.

2 Term weighting scheme

First, we present Term Frequency-Inverse Document Frequency ($tf.idf$) which is the most popular TWS method, proposed by Jones in [19]. This method is unsupervised (does not take into account the class of a document) and can be formally defined as :

$$w_{t,d} = tf_{t,d} \times \log \frac{N}{N_t} \quad , \tag{1}$$

where, $w_{t,d}$ is the weight of term t in document d, $tf_{t,d}$ is the raw count of t in d, and $\log(N/N_t)$ is the inverse document frequency (idf) where N is the total number of documents in the document set and $N_t = |\{d' \in D | t \in d'\}|$ is the number of documents that contains the term t. Beside the raw count ($f_{t,d}$) representation of tf, there exit numerous other variants such as binary weight, $log(f_{t,d}) + 1$, $f_{t,d}/\sum_{t' \in d} f_{t',d}$. idf has also a number of variants such as $\log(N/N_t) + 1$, $\log((N - N_t)/N_t)$, for instance.

The intuition behind this formula is to have a large weight for terms which are frequent in only a few number of documents (having a term present in all documents is not informative).

In [17], Salton *et al.* pointed out three main considerations for a text retrieval system that are believed to improve both recall and precision:

- *Term Frequency (TF) factor*: The TF factor is used to capture the relative importance of terms in a document. This is the first term in the previous formula.
- *Collection Frequency (CF) factor* : Also called term discrimination. The importance of words in a document (TF factor) does not provide enough discrimination ability. A common word like 'The' is frequent in almost all documents, and then it could not separate a group of documents from the remainder of the collection. Hence a discrimination factor is needed to favor those terms that are concentrated in a few documents of the collection. This corresponds to the second part of the previous formula. Main known CF factors are presented in Table 1.
- *Normalization factor*: In text retrieval, all documents are considered equally important, however TF factor will favor large documents over shorter ones, as large documents contains more unique terms and/or greater occurrence values. The normalization factor can eliminate this length effect.

Text classification is a supervised learning task, consequently a document class is known in advance. To compute weights some methods used this information. Such methods are called Supervised Term Weighting (STW). In that context and based on the considerations discussed above, researchers proposed various supervised term weighting to use in feature selection methods namely, χ^2 [7, 4], information gain [7, 4], gain ratio [4], odds ratio [7], relevance frequency [11], and recently inverse category frequency [21].

Table 1. Four traditional CF factors. Given a term *t* and a category *cat*, N stands for the total number of documents, a is the number of documents that contain t and belong to *cat*, c is the number of documents that contain t and do not belong to *cat*, b is the number of documents that do not contain t and belong to *cat* and d is the number of documents that do not contain t and do not belong to *cat*.

CF	Defined by
idf	$\log(\frac{N}{N_t})$
χ^2	$\frac{N*(a*d-b*c)*(a*d-b*c)}{(a+c)*(b+d)*(a+b)*(c+d)}$
or	$\log(2 + \frac{a*d}{b*c})$
rf	$\log(2 + \frac{a}{\max(1,c)})$

3   Genetic programming

In this section, we first introduce the Genetic Programming model, and then we present how we adapt the GP model to the TWS.

3.1   Presentation

Evolutionary computing is based on the Darwin's theory of "survival of the fittest". The main scheme of evolutionary algorithms is to evolve a population of individuals that are randomly generated. Each individual represents a candidate solution that undergoes a set of genetic operators that allow to mix and alter partial solutions. One of the key features of evolutionary algorithms is that they are stochastic approaches.

Genetic Programming (GP) belongs to the family of evolutionary algorithms. It was first proposed by Cramer in [3] and then popularized by Koza [10]. Unlike genetic algorithms where the aim is to find out a solution, the goal of GP is to find out a computer program that is able to solve a problem.

In GP, a set of random expressions that usually represent computer programs is generated. As in all evolutionary computation algorithms, this set of programs will evolve and change dynamically during the course of evolution. What makes GP suitable for a number of different applications is that these computer programs can represent many different structures, such as mathematical expressions for symbolic regression, decision trees, programs that control a robot to fulfill a certain task or programs that are able to predict defibrillation success in patients ans so on.

The quality of a candidate solution (i.e. a program) is usually assessed by confronting it with a set of fitness cases. This step is usually the most time-consuming step as the programs may get huge and several thousands of candidate programs are usually evaluated at each generation. These computer programs

will undergo one or several evolutionary operators that will alter in a hopefully beneficial way. The most classical evolutionary operators are usually the crossover operator that allows the exchange of genetic material (in our case subtrees) and the mutation operator that allows a small alteration to the program.

In the most conventional GP approach, programs are usually depicted by trees. In GP terminology, the set of nodes are split into two sets, inner nodes of the tree are drawn from a set of functions while the terminal nodes (leaves) are drawn from a so-called terminal set. Depending on the problem, the set of functions can be mathematical functions, Boolean functions, control flow functions (if,...), or any functions that may be suitable to solve the given problem. The terminal set is usually the set of inputs of the problem, e.g., parameters and constants for symbolic regression problems, sensors for robot planning...

If the stopping criteria is reached, then the best individual is returned, otherwise, the loop continues and the best individuals are selected (according to their fitness). There exist numerous ways for selecting the population, the mutation and the crossover operators. This is beyond the scope of this paper and the reader can refer to [10, 9] for more information.

3.2 Term Weighting Scheme using genetic programming

A CF factor is a combination of statistical information. It is intended to measure the discriminative power of a term, that is, it tells how much a term is related to a certain category. These statistical information are combined by means of mathematical operators and functions.

We are interested in automatically evolving a CF factor (an individual) using GP (the learned CF factor combined to the TF factor form a term weighting method)

In our context of automatically evolving term weighting methods, an individual is a combination of the function set that is built with simple arithmetical operators (+,-,*,/,log,...) and the terminal set (constant values and inputs to our problem). A presentation of the implementation and the parameters used in this study will be presented in Section 4.1.

Tables 2 and 3 show the statistical information used as terminal set, and the operators for generating formulas (the function set) which represent CF factors. As it can be seen, the function set is made of very simple arithmetical functions while the terminal set includes to the best of our knowledge all the statistical information used to build a TWS.

4 Experiments and results

In order to validate our approach, we compare the four most well-known traditional weighting methods with GP evolved formulas on the ten most frequent categories of the popular dataset Reuters-21578 using linear Support Vector Machine (SVM) [8].

In all our experiments, we perform a cross-validation process.

Table 2. Statistical information (Terminals) used to evolve a TWS.

Label	Description
N	Total number of documents
C	Number of categories
C_t	Number of categories that contain the term t
N_{cat}	Number of documents in the positive category cat
$\overline{N_{cat}}$	Number of documents that do not belong to cat
N_t	Number of documents that contain t
$\overline{Nt}$	Number of documents that do not contain t
a	Number of documents that contain t and belong to the positive category cat
b	Number of documents that do not contain t and belong to cat
c	Number of documents that contain t and do not belong to cat
d	Number of documents that do not contain t and do not belong to cat

Table 3. Arithmetic operators and functions used to evolve a TWS.

Operator	Description
$+$	Arithmetic addition operator
$-$	Arithmetic subtraction Operator
$*$	Arithmetic multiplication operator
$/$	Arithmetic division operator
$\log 1p(x)$	Natural (base e) logarithm of 1 plus ($\ln(1+x)$)
$\log 2p(x)$	Natural (base e) logarithm of 2 plus ($\ln(2+x)$)

4.1 Experimental setup

Reuters-21578 is a multi-labeled dataset and one of the most frequently used test collections for text classification research.

We use the traditional "ModApte" split which contains 90 categories. The dataset is transformed into multiple single-label binary classification tasks using the binary relevance transformation strategy.

The point is that the ten binary tasks corresponding to the ten most frequent categories are used. No preprocessing steps or feature selection are performed on the resulting dataset.

For our experiments, the Liblinear classification library [8] is used with a L2-regularized L2-loss Support Vector classification (dual) solver. The other parameters are set to their default values.

Concerning the GP, we use a population of 100 individuals (candidate solutions, i.e. formulas), and 100 generations (corresponding to the stopping criterion).

The crossover probability is set to 0.85 and the mutation probability is defined as one over the number of terminals and operators.

A validation phase is needed as one cannot learn directly over the test set. Thus, the population of formulas are evaluated by means of stratified 3-fold cross validation over the training dataset.

To assess the performance of STW, we report in the result tables the precision p, the recall r and the standard $F1$ measure.

The precision is the true positive tp over the true positive plus the false positive fp. The recall is defined as the true positive over the true positive plus the false negative fn.

The $F1$ measure is defined as

$$F1(tp, fn, fp) = \frac{2 * tp}{2 * tp + fn + fp} \ .$$

In order to obtain one general score for the multiple tasks, the F1 measure is usually averaged using micro-/macro-averaging methods. Macro-averaging gives equal weight to each category, thus, it gives an insight on the efficiency/effectiveness of a classifier on small categories. In contrast, micro-averaging gives weight depending on the category size (the larger the category, the larger the weight), thus results are dominated by large categories.

The micro-$F1$ ($\mu - F1$) is defined as

$$\mu - F1 = F1(\sum_{l=1}^{C} tp_l, \sum_{l=1}^{C} fp_l, \sum_{l=1}^{C} fn_l) \ .$$

The macro-$F1$ ($m - F1$) is defined as

$$m - F1 = \frac{1}{C} \sum_{l=1}^{C} F1(tp_l, fp_l, fn_l) \ .$$

It is important to understand that the macro measure m gives equal weight to each category. The micro measure μ gives equal weight to each per-document classification decision. The main problem of the F1 measure is that it ignores true negatives and its magnitude is mostly determined by the number of true positives, then large classes dominate small classes in terms of micro measure [12, 20].

4.2 Results

Table 4 shows the results of the four traditional TWS on the ten categories. The CF column is the best of the four traditional TWS used in this paper (the best between idf, rf, or, χ^2). We report the precision, the recall and the $F1$-score for the ten categories experimented.

In comparison, in Table 6 we report the same measures but for formulas generated with GP. For each category, the corresponding formulas generated by GP are presented in Table 5.

First, we can notice that for each category the generated formulas are different. The generated formulas (f_1 to f_{10}) perform better on seven out of ten tasks, and the traditional TWS performed better on three tasks (interest, wheat and corn).

In Table 7 we compare $\mu - F1$ and $m - F1$ for the best validated TWS between idf, rf, or, χ^2 for each category against the generated formulas f_1 to f_{10} based on GP. In terms of micro score (i.e. μ) the performance is slightly better for the generated formula, and for the macro score (i.e. m) we are able to improve the performance by 1%. Even if the results seem close, it is impressive that we are able to have such performances with automatically generated TWS.

In Table 8 we compare the performances of each TWS with f_{11} in terms of micro-/macro-averaged $F1$ scores over all categories. f_{11} is a formula which has been generated by GP by considering all ten categories. The goal is to experiment whether GP is able to generate a formula which is good on all ten datasets. In term of micro-averaged score, the generated f_{11} formula is better than all the other formulas. For the macro-averaged score, results are similar.

Finally, in Table 9 it is interesting to see that five from ten generated formulas (f_4, f_5, f_6, f_8 and f_{10}), which have been generated on one specific category, generalize well on the nine other categories, and are slightly better than the four traditional TWS, (see Table 8).

Table 4. Best traditional scheme for each category in top 10 Reuters-21578 categories evaluated by 3-fold cross validation over the training set.

	CF	Precision	Recall	F1
earn	idf	98.98%	97.79%	98.38%
acq	idf	98.71%	95.55%	97.10%
money-fx	idf	80.23%	77.09%	78.63%
grain	rf	96.43%	90.60%	93.43%
crude	rf	89.19%	87.30%	88.24%
trade	rf	82.08%	74.36%	78.03%
interest	idf	88.00%	67.18%	76.19%
ship	or	95.24%	67.42%	78.95%
wheat	chi	84.15%	97.18%	90.20%
corn	chi	91.53%	96.43%	93.91%

Table 5. Formulas generated by GP. f_{11} is a formula generated on all the ten categories.

	Denoted by	Defined by
earn	f_1	$b * (N_t - \overline{N_c})$
acq	f_2	$C_t - (b/(\log 1p((N_t * d * a * \log 2p(\log 2p(\overline{N_t}))) + b)/d))$
money-fx	f_3	$(N + c) - (\log 1p(N_t) * d)$
grain	f_4	$N_t/(c - C_t)$
crude	f_5	$d * (\overline{N_t} + N + C + C_t - (d * d))$
trade	f_6	$N * (\overline{N_t} - N_t + d)$
interest	f_7	$d - (((\overline{N_c} + C) + ((c * ((\overline{N_c} * N_c) + a))/\overline{N_t}))/\overline{N_t})$
ship	f_8	$d * N_t * (\overline{N_t} + N_c + C_t + b)/c$
wheat	f_9	$\log 2p(\overline{N_c} + ((\overline{N_c} * \overline{N_t})/(\log 2p((b + C)/C_t) * b))$
corn	f_{10}	$(c * (N_c + \overline{N_c}) + \overline{N_t} * a * a)/c$
All	f_{11}	$\log 1p(d/c) + (N_t/(\log 1p(\log 2p(\overline{N_c})) * c))$

Table 6. Formulas generated by GP and evaluated by 3-fold cross validation for each category of the ten biggest categories in Reuters-21578.

	CF	Precision	Recall	F1
earn	f_1	98.70%	98.16%	98.43%
acq	f_2	98.85%	95.83%	97.32%
money-fx	f_3	81.82%	80.45%	81.13%
grain	f_4	94.04%	95.30%	94.67%
crude	f_5	90.22%	87.83%	89.01%
trade	f_6	83.49%	77.78%	80.53%
interest	f_7	87.63%	64.89%	74.56%
ship	f_8	92.00%	77.53%	84.15%
wheat	f_9	86.84%	92.96%	89.80%
corn	f_{10}	86.15%	100.00%	92.56%

Table 7. $\mu - F1$ and $m - F1$ for the best validated TWS between idf, rf, or, χ^2 for each category against the generated formulas f_1 to f_{10} based on GP. Results are really close, especially for the μ score. It is interesting to see that GP is able to compete with *a priori* usual known TWS.

	Best rules	GP
μ-F1	93.19%	93.53%
m-F1	87.31%	88.42%

Table 8. $\mu - F1$ and $m - F1$ for all TWS over all categories against the generated formula f_{11} validated over all categories.

	$\mu - F1$	$m - F1$
idf	92.52%	84.90%
rf	92.56%	85.68%
or	92.54%	85.93%
χ^2	91.59%	87.18%
f_{11}	93.13%	86.55%

Table 9. $\mu - F1$ and $m - F1$ over all categories for the generated formula f_1 to f_{10}. We can note that the generated formula reach good performance even on the other categories, which is important in term of finding a global rule.

	$\mu - F1$	$m - F1$
f_1	91.51%	80.77%
f_2	91.06%	79.40%
f_3	91.77%	83.32%
f_4	92.13%	86.30%
f_5	92.82%	85.59%
f_6	92.68%	85.24%
f_7	92.44%	84.65%
f_8	92.39%	86.96%
f_9	89.47%	75.78%
f_{10}	91.10%	86.82%

5 Conclusion

The aim of this paper is to test the ability of GP in automatically generating efficient TWS.

Experiments are conducted and the generated TWS are compared with four traditional TWS. In the experiments, the generated formulas are able to compete with the traditional schemes and are even slightly better in some cases.

We also find that the generated formulas have a good generalization ability.

Our future work will focus on generating a TWS that can generalize even better over different datasets. That can be done by learning directly over the test data of different datasets using the GP.

References

1. Apte, C., Damerau, F., Weiss, S., et al.: Text mining with decision rules and decision trees. Citeseer (1998)
2. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning 20(3), 273–297 (1995)
3. Cramer, N.L.: A representation for the adaptive generation of simple sequential programs. In: Proceedings of the First International Conference on Genetic Algorithms. pp. 183–187 (1985)
4. Debole, F., Sebastiani, F.: Supervised term weighting for automated text categorization. In: Text mining and its applications, pp. 81–97. Springer (2004)
5. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American society for information science 41(6), 391 (1990)
6. Deng, Z.H., Tang, S.W., Yang, D.Q., Li, M.Z.L.Y., Xie, K.Q.: A comparative study on feature weight in text categorization. In: Asia-Pacific Web Conference. pp. 588–597. Springer (2004)

7. Deng, Z.H., Tang, S.W., Yang, D.Q., Li, M.Z.L.Y., Xie, K.Q.: A comparative study on feature weight in text categorization. In: Advanced Web Technologies and Applications, pp. 588–597. Springer (2004)
8. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. Journal of machine learning research 9(Aug), 1871–1874 (2008)
9. Karakus, M.: Function identification for the intrinsic strength and elastic properties of granitic rocks via genetic programming (gp). Computers & geosciences 37(9), 1318–1323 (2011)
10. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press (1992)
11. Lan, M., Tan, C.L., Su, J., Lu, Y.: Supervised and traditional term weighting methods for automatic text categorization. Pattern Analysis and Machine Intelligence, IEEE Transactions on 31(4), 721–735 (2009)
12. Manning, C.D., Raghavan, P., Schütze, H., et al.: Introduction to information retrieval, vol. 1. Cambridge university press Cambridge (2008)
13. McCallum, A., Nigam, K., et al.: A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization. vol. 752, pp. 41–48. Citeseer (1998)
14. Ng, H.T., Goh, W.B., Low, K.L.: Feature selection, perceptron learning, and a usability case study for text categorization. In: ACM SIGIR Forum. vol. 31, pp. 67–73. ACM (1997)
15. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using em. Machine learning 39(2-3), 103–134 (2000)
16. Pang, B., Lee, L., et al.: Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval 2(1–2), 1–135 (2008)
17. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management 24(5), 513–523 (1988)
18. Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine learning 39, 135–168 (2000)
19. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of documentation 28(1), 11–21 (1972)
20. Van Asch, V.: Macro-and micro-averaged evaluation measures [[basic draft]] (2013)
21. Wang, D., Zhang, H.: Inverse category frequency based supervised term weighting scheme for text categorization. preprint arXiv:1012.2609v4 (2013)
22. Yang, Y.: Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 13–22. Springer-Verlag New York, Inc. (1994)

The Ant Reconciliation Algorithm (ARA): Ant-hill learning for label matching

Pierre Parrend[123], Camille Maller[2], and Etienne Dietrich[2]

[1] ICube Laboratory, Université de Strasbourg, France
[2] ECAM Strasbourg-Europe, Schiltigheim, France
pierre.parrend@ecam-strasbourg.eu
[3] Complex System Digital Campus (UNESCO Unitwin)
http://unitwin-cs.org/

Abstract

Reconciliation of billing and payments is a critical process in the processing of medical reimbursement data for which no satisfactory solution exists so far. Such reconciliation aims at alleviating the manual work for operator, and is characterized by a high expectation with regard to the algorithm effectiveness: automated reconciliation can only be proposed if it is almost certain, so as to avoid costly and un-ergonomic undo process. A promising solution for performing automated reconciliation is the use of Ant-Colony Optimisation (ACO)-based learning, which has a strong ability of finding sequences of evaluation criteria, here in the context of label matching, while taking into account search history weighted by success scores and history. The ant reconciliation algorithm (ARA) extends and implements the Ant-Miner algorithm by defining specific computation parameters as well as a rule model for text matching. We define the ARA model, explicit its implementation, and provide evidences of the relevance of the proposal drawn from evaluation on real world medical billing and payment data.

1 Introduction

Ant-hill algorithms are typically used as automated heuristics for exploring optimisation landscapes, addressing in particular the risk of combinatorial explosion through emerging auto-organisation of the 'colonies' of artificial ants, communicating through the exchange of messages transmitted by the environment, a phenomenon called stigmergy.

We claim that such properties can be leverage to dramatically enhance learning time of any machine learning system where a meaningful feedback from the human operator can be obtained. To evaluate this hypothesis, we define the ARA Ant Reconciliation Algorithm (ARA), which is an Ant-hill learning scheme for label matching. ARA explores label matching rules for optimising coverage of the

dataset under study, and integrates a built-in mechanism for tuning or disprove rules through targeted expert feedback. ARA is an extension of the Ant-Miner Algorithm which specifies specific pheromones types and a dedicated rule system for label matching.

ARA is implemented and evaluated in the context of a software for managing medical expenses and reimbursement at the level of mutual health insurances and health professionals. The efficiency and performance of ARA is computed against a production dataset which is kindly made available through the industrial partner sponsoring this research.

This paper is organised as follows: section 2 presents the state of the art. Section 3 presents our ant-hill model for label matching. Section 4 gives the experimental setup of the proposed approach and related observations. Section 5 analyses and evaluate the output of the experiment. Section 6 concludes the study.

2 State of the art

This section provides an overview of topics relevant for our proposal. First, the exploitation of the ant-hill approach to human traces rather than to artificial ants is reviewed. Then the adaptation of this approach for pattern matching, at the example of the Ant-Mining algorithm, is discussed and compared to the original Ant-System algorithm.

2.1 Ant-hill approaches from human traces

This powerful scheme has been the subject of significant research efforts since its inception in 1996 [DMC96,DB05]. One particular approach is the exploitation of actual traces let by agents in their digital environment: in a context where social networks claims to have 500.10^6 users, for LinkedIn or even $1,86.10^9$ active users each month, for Facebook, such behaviour can reasonably be expected to emerge. With the notable exception of supercolonies which have been discovered and studies since the early 2000's [VWTT10], ant colonies do not entail more that a couple of million users – far less than their human counterparts. The ant-hill approach has therefore been applied to human traces [GVCK07], and proves to bring significant benefits in the construction of learning path in online learning environments. Its deployment has been evaluated both for secondary school home support [VLC06] and university MOOCs [CSSB$^+$15].

2.2 Ant-System and Ant-Miner

The learning system for reconciliation rules is based on a variation of the Ant-System algorithm [DMC96] which was adapted to sorting rules, Ant-Miner [LAM02]. In Ant-Miner, ants are traveling through a graph, which nodes represent attributes to be added to a given rule. Ant movements thus tend towards the rule which can at best explain the data to be analysed. Ant-miner has two main differences *wrt.*Ant-System:

- Heuristic information (the desirability) is a characteristic of each node and does not depend on previous positions of the ant. Heuristic information describes the capacity of an attribute to discriminate efficient and inefficient rules for sorting. It is a real value comprised between 0 and 1, with 1 presenting the best possible attributes. Heuristic information is defined by:

$$\eta_k = 1 + x.log_2 x + (1-x).log_2(1-x) \tag{1}$$

where x is the probability for a relevant sorting when the attribute is verified. This probability is evaluated on the training dataset
- Quality of the rule created by each ant through its path in the graph is evaluated by calcutating the fitness score of this rule with:

$$fitness = \frac{TP}{(TP+FN+1)} * \frac{TN}{(TN+FP+1)} \tag{2}$$

The quantity of pheromones deposited on each attribute building the rule is proportional to the fitness value. After a significant number of ants have traveled through the graph, the rule with the best fitness is kept.

Several extensions of Ant-Miner have been proposed by the community: for coping with continuous attributes [OFJ08], for unordered rule sets [SF06].

This process generates a rule which describes a subset of the population of data entries. The entries which are correctly described by this rule are removed from the dataset, pheromones are reinitialised and the heuristic information for each attribute is calculated again according to remaining entries: a new rule is generated through a new iteration of ant traveling through the graph, and the process is repeating until a stop criteria is matched. This stop criteria can be any of the following:

- a new generated rule has a fitness of δ, with $\delta << 1$
- the remaining dataset entails a negligible fraction of the entries of the original dataset.

In any case, ants traveling through an attribute graph stop adding new attributes as far as the fitness of generated rules decreases as a consequence of the addition of these attributes.

3 An ant-hill model for label matching

ARA, the Ant Reconciliation Algorithm, aims at providing a solution for enabling user-guided learning through an ant-hill approach where the pheromones are set according to both rule coverage and expert feedback. It is applied to label matching for reconciliation, in the context of medical expense reimbursement for professional users such as mutual health insurances and medical professionals. Reconciliation of label is the process of performing label matching through numerous and heterogeneous data fields. In our evaluation dataset, which represents production data of health professional expenses and thus provides a representative set of medical expense data, reconciliation is to be performed between

the reimbursement requests, which can be considered as standardised, and the bank transfer information, which are highly heterogeneous: expenses can be reimbursed in isolation or in bulk, reimbursement request references are included or not, the paying body may be or not be the same organisation to which the request has been emitted. To make the problem still more challenging, bank transfer information data structure includes various fields which are completed manually by the insurance and banking organisations with no coherence between the different actors, including redundant fields for payment emitter and several free text fields. The presentation of the model will focus on the ARA algorithm, which is the contribution of this paper, and not on domain specific data formats to ensure reusability of the scheme for other domains.

3.1 Defining ARA pheromones for label matching

In order to refine the heuristic information, we introduce two complementary types of pheromones for tracking both the matching efficiency and the expert feedback. The coexistence of several pheromone types is a usual approach in ant-hill optimisation algorithms, as well as for ant-hill based mining [SAF11].

Success pheromones The pheromones for matching efficiency are simply named 'success pheromones'. We define here the fitness as the probability of a path N to be chosen. Pheromones are computed for each iteration i, and writen as φ.

$$fitness_N = \frac{\varphi_N}{\sum_{l=1}^{k} \varphi_l} \tag{3}$$

$$\varphi_0 = 0 \tag{4}$$

$$\varphi_{N,i+1} = 1 \ if \ \varphi_{N,i} = 0 \ and \ match \ found \ for \ path \ N \tag{5}$$

$$\varphi_{N,i+1} = \varphi_{N,i} * 1,1 \ if \ \varphi_{N,i}! = 0 \ and \ match \ found \tag{6}$$

$$\varphi_{N,i+1} = \varphi_{N,i} * 0,9 \ if \ \varphi_{N,i}! = 0 \ and \ match \ NOT \ found \tag{7}$$

A stable reconciliation occurs when a new rules appears to be more efficient than a previous one for performing label matching, without later regression to previous rules. Figure 1 shows the values of success pheromones for a stable reconciliation.

History pheromones The pheromones for expert feedback are named 'history pheromones', and track both validation of performed reconciliation and sudden de-reconciliation. Such de-reconciliation typically means that a matching rule is no longer valid. In our medical expense reimbursement example, for instance, the name of and insurance organisation has changed, or a given service is covered by a different insurance than previously. For 'History pheromones', the fitness function as computed as follows:

$$fitness_N = 100\% \ if \varphi_{h,N,i} > P \tag{8}$$
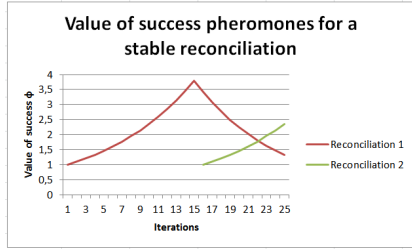
Figure 1. Values of success pheromones for a stable reconciliation
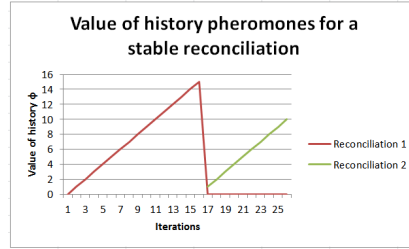
Figure 2. Values of history pheromones for a stable reconciliation

$$\varphi_{hi+1} = \varphi_{hi} + 1 \ if \ match \ found \ for \ path \ N \qquad (9)$$

$$\varphi_{hi+1} = 0 \ if \ match \ NOT \ found \ for \ path \ N \qquad (10)$$

Figure 2 shows the values of history pheromones for a stable reconciliation: an explicit de-reconciliation by the expert, followed by the identification of new reconciliation rules, leads to a sudden modification of the rules to be applied by the reconciliation engine.

3.2 The ARA rule system

Label matching rules learnt by ARA are bulding a set of tests, bound together with a logical AND. When the test is valid, the relevant 'class' to which the considered data entry pertains is provided as output. Each test is called an attribute, and matches fields from the two domains to be reconciliated. In our case, it matches field from the medical bill with a field from bank transfer extraction. For instance:

- An attribute can be: 'the amount of the bill is egal to the amount of the transfer'
- a rule can be: IF 'the amount of the bill is egal to the amount of the transfer' AND 'the bill data and the transfer data are separated by less than 30 days' THEN 'perform reconciliation'
- another rule can be: 'the bill data and the transfer data are separated by more than 30 days' THEN 'DO NOT perform reconciliation'

Attributes can be bound together in any possible combination for creating new rules. Each attribute is created twice: one performs the test, the other is its logical opposite. This enables to take decision when a given amount is NOT egal to another, for instance, and thus to remove non-relevant rules when a rule is invalidated by a manual reconciliation by the expert.

An attribute is built by three components:

The left accessor, responsible for extracting the data from the left dataset; in our case, the left accessor is the 'bill accessor', and extracts information related to the medical expense bill

The right accessor, responsible for extracting the data from the right dataset; in our case, the right accessor is the 'transfer accessor', and extracts information related to the bank transfer.

The evaluator, responsible for performing a test between these values provided by the accessors.

This model is therefore easily extensible.

3.3 Leveraging expert feedback

The ARA algorithm is designed for quickly integrating expert feedback after the initial learning phase. The objective is to provide a built-in mechanism for tuning or disprove extracted rules. The interaction occurs in two ways:

- The expert reviews performed reconciliations. A very low acceptance readiness of the user for false positive urges us to tune the rule selection so that only label matching which are quasi certain are enforced.
- The expert performs manual reconciliation. Users are used to perform fully manual matching, and shall not be exposed with approximation or uncertainty. For this reason, likely reconciliations are shown to the user, but without providing her with quantitative information about the computed fitness value. These reconciliations are simply provided by decreasing fitness value. Pheromone information is kept intern to the matching engine.

This approach enables to fully exploit the feedback of the user, without exposing the internals of the algorithm, which would rather disturb them than help them make better manual choices. In our case, users are typically administrative agents of the insurance organisations or of medical centers, and shall be given all tools to focus on the service they are committed to, not to understanding the reconciliation logic.

4 The experiment

This section introduces the ARA implementation, specifies the required building bricks, and explain how the current version of the tool can easily be extended for still better performances in the evaluation application, or for application to label matching in different domains. The evaluation process is then presented, and results are provided to benchmark the relevance of our proposal, which proves to provide very promising efficiency and performance.

4.1 Implementation

The development of the ARA tool and its evaluation entail following mandatory building bricks:

- A system of dynamic rules which support the definition of complex rules from simple tests

- A mechanism for the evaluation of the performance of a rule on a real and representative training dataset from a production environment, including fitness value, true negative and false negative rates
- An ant-hill algorithm for ordering a given dataset
- A graphical interface to visualise the output and evaluation of the algorithm
- The extraction of a basic dataset to benchmark the early versions of rule learning. A given number of valid reconciliations and of invalid potential reconciliations are taken at random from the reference dataset.

Figure 3 presents the algorithm workflow for ant-hill learning, with initialisation of the dataset, initialisation of the rule graph including nodes and edges, iterative construction of a new rule by the ants, and extraction of the final rule set.
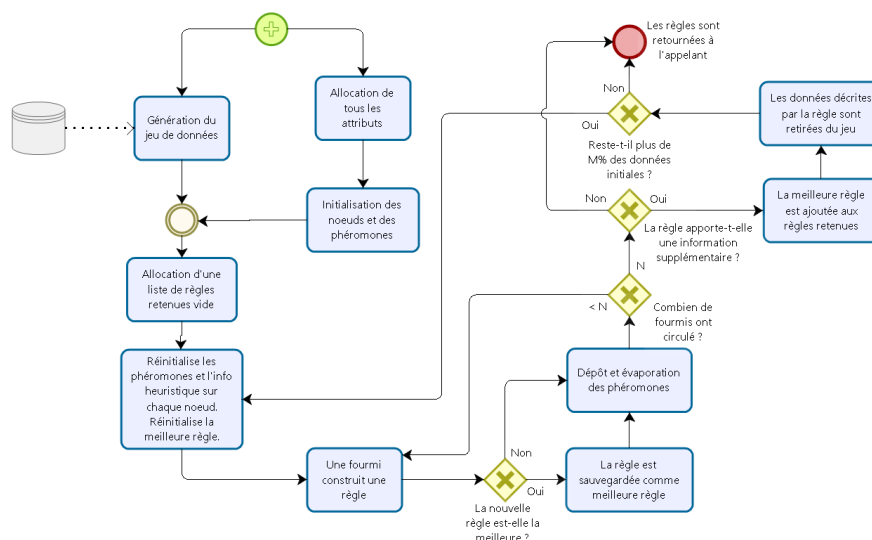
Figure 3. Algorithm workflow for ant-hill learning

The ARA tool is therefore a complete library for label matching applied to reconciliation of medical expense reimbursement bills and bank transfers. Moreover, it is designed to be easily extendible to ensure its seamless implementation in any production environment.

4.2 Implementing new attributes

A new attribute can easily be generated by creating a dedicated subclass implementing accessor methods, and by instantiating these classes in the attribute allocator. This process occurs in three phases:

1. Identify the Accessor or Evaluator class matching the datatype of the considered attribute. Such classes are for instance DecimalAccessor, DateTimeEvaluator, etc.. If the attribute needs to operate on a new datatype, a new Accessor or Evaluator subclass can simply be created. Methods to be implemented are: expectedType() or returnedType().
2. Define a subclass of Accessor[Type] or Evaluator[Type]. Methods to be implemented are: value() or test().
3. Instanciate the newly created class in the constructor of the AttributeAllocator class. The AttributeAllocator the creates all possible combination of compatible accessors and evaluators.

Figure 4 presents the software architecture of the solution, including named methods (here in French).
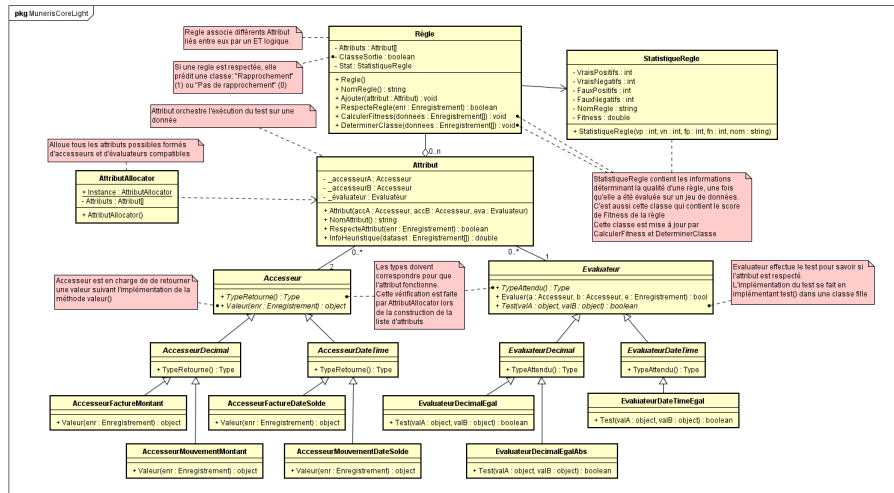
Figure 4. Software architecture of the solution

ARA is not only a full-fledge label matching environment, it is also very easily extensible and adaptable to other environments.

4.3   Experiment results

The ARA algorithm is evaluated *wrt.*its efficiency and performance as a standalone, autonomous algorithm. The efficiency of the expert feedback still requires integration work from the industrial partner, and is therefore not yet available. ARA is evaluated on a dataset containing 2000 entries, 1000 of them being valid reconciliations and 1000 being invalid ones. The automated generation of matching rules provides following results:

- 3 rules are generated,
- 1927 entries are correctly classified,
- 26 are incorrectly classified, *i.e.* 1,3%.

If we consider that the characteristics of the data being analysed are stable over time, this is a very satisfactory result. Figure 5 shows the evaluation results, as depicted on the console. The first learnt rules can be read from it. The first rule compares the payment dates (DateSolde) on the bill excerpt and on the bank transfer statement. If a match occurs, the rule indicates that the bill and the transfer can be considered as matching. This rule classifies correctly 971 entries, for only 2 false positives. It has therefore a fitness score of 0,967, which is nearing 1. Following rule entails 3 attributes, *i.e.* it performs three tests. It states that is the dates do not match, and no mention of the amount in the 'wording' and the 'amount' field of the back transfer, then no reconciliation can be made between bill and transfer. Such a rule enables to eliminate matching that are known they can't be conclusive, so as to reduce the potential solutions which are shown to the final user.


```
 => Jeu de règles apprises:
AccesseurFactureDateSolde EvaluateurDateIdentique AccesseurMouvementDateSolde
Classe décrite: Rapprochement
VraisPositifs = 971
VraisNégatifs = 998
FauxPositifs = 2
FauxNégatifs = 29
 => Fitness = 0,967122787302607
AccesseurFactureDateSolde EvaluateurDateIdentique_Inversé AccesseurMouvementDateSolde
AccesseurFactureMontantTexte EvaluateurTexteSimilaire_Inversé AccesseurMouvementMontantTexte
AccesseurFactureMontantTexte EvaluateurTexteSimilaire_Inversé AccesseurMouvementLibelle
Classe décrite: Pas de rapprochement
VraisPositifs = 644
VraisNégatifs = 21
FauxPositifs = 8
FauxNégatifs = 356
 => Fitness = 0,45034965034965
```
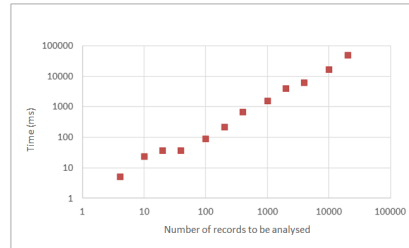
Figure 5. Evaluation results, shown on the console

One can notice that the number of false positives tend to increase when more rules are learnt: the second rule generates 8 false positives, and this number still raises with later rules. Figure 6 presents the evaluation results, as shown on the graphical interface developed for the evaluation of ARA. Three graphs are made available: the first one shows the number of true and false positives and negatives for each rule; the second one, on the bottom left corner, shows the evolution of the global fitness value as a ROC curve when more rules are learnt; the third graph, on the upper right corner, shows the fitness function for each individual rule.

The ARA algorithm has been tested for evaluating its performance on dataset from some dozens to 20000 elements. It presents a linear execution time for this range of dataset size. Figure 7 shows analysis duration, according to the dataset size.
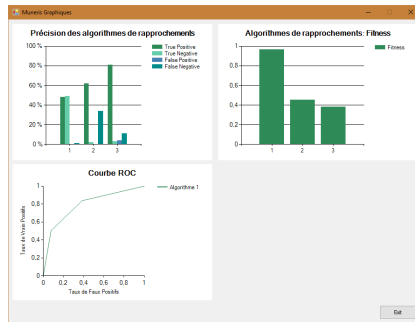
Figure 6. Performance, precision and ROC curve for the algorithm to be evaluated

Figure 7. Analysis duration, according to the dataset size

5 Discussion

The evaluation results show that ARA is able, at least for the target dataset, to perform very efficient and performant label matching proposals with a reduced set of rules. With 1,3% of failing reconciliations, its proves to be very promising, at least for a dataset which characteristics are stable over time. If we moreover consider that the highly probable evolution of the data characteristics can be handled in a very reactive manner through the ARA expert-in-the-loop scheme, ARA can be considered as a theoretically very promising approach. More experimentations in production environment is of course required to ensure that the user interaction flow is actually ergonomic and enables to leverage these theoretical possibilities.

An extension of the ARA model can be envisioned, to still increase proposed results of this ant-hill algorithm:

- The fitness score of the rules used for performing matching proposals can be exploited to detail the trust level one ca have in this matching. These approach would enable to propose the most probable combinations to the user, while letting her manual valide the matching itself. This feature is specified, but still needs to be implemented
- Fine-grained tuning of the parameters of the ant-hill algorithm can still be performed to ensure that the best possible results are provided. The effect of this possible tuning is still to be evaluated so as to provide guidance for configuration or further development of ARA. Two parameters are serious candidates for tuning:
  - The rate of unclassified dataset entries which is considered acceptable. A lower rate leads the algorithm to identify more complex rules to classifying last cases, but typically leads to increase the false positive rate
  - The number of ants traveling for the generation of a rule. Once the ants converge toward a given solution, it is of no use to carry on the compu-

274

tation. However, if the number of ants is too restricted, the algorithmic has no time to converge toward an optimal solution.

- the computation of the fitness value could be tuned so that false positives are more disadvantageous. These two expressions of the fitness value could be evaluated, as well as the impact of the parameter $a > 1$. Actually, when a increases, false positives tend to let the fitness value decrease more significantly. TP are true positives, FP are false positives, TN are true negatives, FN are false negatives.

$$fitness = \frac{TP}{(TP + FN + 1)} * \frac{TN}{(TN + a * FP + 1)} \tag{11}$$

$$fitness = \frac{TP}{(TP + FN/a)} * \frac{TN}{(TN + aFP + 1)} \tag{12}$$

Based on these qualitative and quantitative evaluation of the ARA proposal for ant-hill drive label matching, we can therefore conclude that, in agreement with our working hypothesis, stigmergetic properties can be leverage to dramatically enhance learning time of any machine learning system. Moreover, the availability of a meaningful feedback from the human operator is identified a key enabler for a rapid adaptation to a changing environment. Detailed evaluation of the effect of such a feedback is out of the scope of this paper.

6   Conclusions and perspectives

In this paper, we introduce the ARA Ant Reconciliation Algorithm for optimising label matching through an innovative Ant-hill learning approach. ARA applies the ant-hill optimisation scheme to human trace to leverage the focused feedback of an expert for guided machine learning. It dramatically improves learning time, and support efficient correction of learnt rules when they appear to be suboptimal or erroneous. The current implementation of ARA is dedicated to label matching applications, and is evaluated to consolidate information on reimbursement of medical expenses. ARA is based on the Ant-Mining algorithm. It entails specific pheromones for quickly learning from expert feedback, as well as a dedicated rule system for maximising rule coverage. In our first experiment, the efficiency and performance of the algorithm show promising results.

Further evaluation is still required to evaluate the ergonomy of the developed solution for the expert. The interaction is meant to be fully transparent for the user, by proposing actual as well as potential label matching occurrences and hiding the underlying quantitative evaluation. Although the ARA algorithms provides very satisfactory results on production dataset, and thus shows a great potential for an efficient integration in the expense reconciliation software, an evaluation of the integrated solution is not yet available.

ARA opens very rich perspective both in the label matching domain, as well as in the domain of machine-learning guide by human experts. It provides a very frugal scheme capable of greatly improving numerous applications while implying very limited impact to the user with regard to interaction overhead or system performance.

References

[CSSB$^+$15] P. Collet, R. Seereekissoon, A. Scius-Bertrand, R. Stein, and P. Parrend. Evaluation pair-à-pair participative (p3e) de la plate-forme poem. In *E-Formation*, Jun 2015.

[DB05] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2):243–278, 2005.

[DMC96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, 1996.

[GVCK07] Sergio Gutiérrez, Grégory Valigiani, Pierre Collet, and Carlos Delgado Kloos. Adaptation of the aco heuristic for sequencing learning activities. In *Proceedings of the EC-TEL*, pages 17–20, 2007.

[LAM02] Bo Liu, Hussein A Abbass, and Bob McKay. Density-based heuristic for rule discovery with ant-miner. In *The 6th Australia-Japan joint workshop on intelligent and evolutionary system*, volume 184, 2002.

[OFJ08] Fernando EB Otero, Alex A Freitas, and Colin G Johnson. cant-miner: an ant colony classification algorithm to cope with continuous attributes. In *International Conference on Ant Colony Optimization and Swarm Intelligence*, pages 48–59. Springer, 2008.

[SAF11] Khalid M Salama, Ashraf M Abdelbar, and Alex A Freitas. Multiple pheromone types and other extensions to the ant-miner classification rule discovery algorithm. *Swarm Intelligence*, 5(3-4):149–182, 2011.

[SF06] James Smaldon and Alex A Freitas. A new version of the ant-miner algorithm discovering unordered rule sets. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 43–50. ACM, 2006.

[VLC06] Grégory Valigiani, Evelyne Lutton, and Pierre Collet. Adapting the elo rating system to competing subpopulations in a 'man-hill'. In *Proceedings of the 2006 conference on Leading the Web in Concurrent Engineering: Next Generation Concurrent Engineering*, pages 766–773. IOS Press, 2006.

[VWTT10] Ellen Van Wilgenburg, Candice W Torres, and Neil D Tsutsui. The global expansion of a single ant supercolony. *Evolutionary Applications*, 3(2):136–143, 2010.

An Improved Particle Swarm Optimization Algorithm for MRI Image Segmentation

Thuy Xuan Pham, Patrick Siarry, and Hamouche Oulhadj

Laboratory Images, Signals, and Intelligent Systems (LiSSi),
University Paris-Est Créteil, 94400 Vitry sur Seine, France
`thuy.pham@univ-paris-est.fr,{siarry,oulhadj}@u-pec.fr`

Abstract. In this paper, an improvement method for segmentation of Magnetic Resonance Imaging (MRI) images using particle swarm optimization (PSO) is proposed. We introduce a new objective function, based on kernelization of fuzzy entropy clustering algorithm with local spatial information and bias correction. By utilizing the state-of-the-art development of PSO in optimization, accurate and robust segmentation results are achieved. The proposed algorithm has been evaluated on several benchmark images. Experimental results show that the proposed method performs better than competing algorithms.

Key words: Image segmentation, particle swarm optimization, entropy fuzzy clustering, objective function, Magnetic Resonance Imaging

1 Introduction

Medical image segmentation is one of the most important problems in medical imaging analysis [1]. Input images may be corrupted by noise and intensity non-uniformity (INU) artifact due to various factors, such as spatial variations in illumination or radio frequency coil used in image acquisition [2]. This makes segmentation a challenging task. Therefore, various methods for segmentation of medical images have been proposed [3]. They can be divided into four groups: region-based techniques, pixel-based classification techniques, threshold-based techniques, and model-based techniques. In this work, the pixel-based classification techniques will be focused on.

Among the pixel-based classification methods, fuzzy clustering is widely used because of their simplicity and applicability. However, the major drawbacks of this approach are: (i) they are very sensitive to noise and INU artifact, since no local spatial information in the image is considered; (ii) the results depend not only on the choice of the initial clustering centroids but also on the high vulnerability of the algorithms to be trapped into local minima.

In this work, a new method for segmentation of MRI images is proposed. This method uses low-discrepancy sequence initialized particle swarm optimization with high-order non-linear time varying inertia weight (LHNPSO) algorithm introduced by Yang et al [5] with a new fitness function. The LHNPSO algorithm, a global optimization algorithm, has two main advantages, namely, (i) the method can converge faster and give an accurate global solution [6] (ii) it is

277

easy to implement. The new fitness function takes advantage of the kernelized fuzzy clustering approach; in addition, it also takes into account local spatial information and bias field correction. Thus, the proposed method may tackle simultaneously the two main problems mentioned above. Experiments using both simulated MRI brain images and real MRI brain images are reported and the results are compared with those from recent segmentation methods.

The rest of the paper is organized as follows. Section 2 presents background information. Section 3 introduces the proposed method: LHNPSO algorithm with a new fitness function. The performance of the proposed method is evaluated and its comparison with a set of algorithms from the literature is reported in Section 4. Finally, Section 5 concludes this paper.

2 Methods

2.1 Kernelized fuzzy entropy clustering with spatial information and bias correction

In this section, a new objective function based on fuzzy entropy clustering algorithm is designed. Let $\mathbf{X} = (x_1, x_2, \ldots, x_N)$ be an image with N pixels, where x_j represents the feature of jth pixel, a partition of the image can be obtained by minimizing an objective function with respect to the partition matrix $\mathbf{U} = (u_{ij})_{C \times N}$ and the set of C cluster prototypes $\mathbf{C} = (c_i)_C$ with $1 < C < N$ as follows:

$$
\begin{aligned}
J_{\text{KFECSB}}\left(\mathbf{X,C,B,U}\right) = & \sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}\left(1 - K(x_j, c_i b_j)\right) \\
& + n\sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}\log\left(u_{ij}\right) + \eta\sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}\left(1 - K(\overline{x}_j, c_i b_j)\right) \quad (1)
\end{aligned}
$$

It should satisfy the following conditions:

$$
\mathbf{U} \in \left\{ u_{ij} \in [0,1] \,\middle|\, \sum_{i=1}^{C} u_{ij} = 1, \quad \forall j \quad \text{and} \quad 0 < \sum_{j=1}^{N} u_{ij}, \quad \forall i \right\} \quad (2)
$$

where $K\left(\cdot, \cdot\right)$ is the Gaussian kernel function defined by $K(x_j, c_i) = \exp\{-||x_j - c_i||^2/\sigma^2\}$; The median of the neighbours within a window of size 3×3 around the pixel x_j is used to represent $\overline{x}_j$, which can be computed in advance. The bias field estimation $\mathbf{B} = (b_j)_N$ based on Li's work [7] is computed as follows:

$$
\mathbf{B} = \left[\left(\sum_{i=1}^{C}\mathbf{G} \cdot \mathbf{X} \sum_{j=1}^{N} c_i u_{ij}\right)^{-1}\left(\mathbf{G} \cdot \mathbf{G}^T \sum_{j=1}^{N} c_i^2 u_{ij}\right)\right]^T \cdot \mathbf{G} \quad (3)
$$

where $\mathbf{G} = (g_1, g_2, \ldots, g_m)^T$ is a set of $2D$ orthogonal Legendre polynomials.

Regarding the objective function, J_{KFECSB} (1), the first term on the right-hand side controls the shape and size of the clusters. In addition, if we consider clusters as fuzzy sets, the second term expresses the average degree of non-membership of members in a fuzzy set [4]. The third term is the local spatial constraint term, in which the parameter η controls the effect of the penalty. In essence, this term, equivalently, aims at guaranteeing noise and INU artifact insensitiveness, and image detail preservation. Thus, minimizing J_{KFECSB} means simultaneously minimizing the dispersion within clusters, maximizing the degree of membership of members, and suppressing noise and INU artifact.

By using the Lagrange multiplier method, the necessary conditions for the minimization of the objective function in Eq. (1) with the constraints in Eq. (2) can be found. Specifically, taking the first derivatives of J_{KFECSB} with respect to u_{ij} and c_i, and zeroing them respectively, two necessary but not sufficient conditions for J_{KFECSB} to be local optimal solution will be obtained as follows:

$$u_{ij}^{-1} = \sum_{r=1}^{C} \frac{\exp\left\{\frac{1}{n}\left[(1 - K(x_j, c_i b_j)) + \eta\left(1 - K(\overline{x}_j, c_i b_j)\right)\right]\right\}}{\exp\left\{\frac{1}{n}\left[(1 - K(x_j, c_r b_j)) + \eta\left(1 - K(\overline{x}_j, c_r b_j)\right)\right]\right\}} \tag{4}$$

$$c_i = \sum_{j=1}^{N} u_{ij}\left[x_j K(x_j, c_i b_j) + \eta \overline{x}_j K(\overline{x}_j, c_r b_j)\right] / \sum_{j=1}^{N} b_j u_{ij}\left[K(x_j, c_i b_j) + \eta K(\overline{x}_j, c_r b_j)\right] \tag{5}$$

2.2 The low-discrepancy sequence initialized PSO

LHNPSO is a population based stochastic optimization technique and is regarded as a global search strategy, designed by Yang et al [5]. In LHNPSO, each member of the population called a particle represents a potential solution to the optimization problem; and the population or swarm P is evolved through successive iterations. In addition, in this algorithm, the initial population of particles is generated by using the Halton sequence to cover the search space efficiently. Each particle, denoted by i, has a position vector $\mathbf{X}_i = (x_{ir})_P$, a velocity vector $\mathbf{V}_i = (v_{ir})_P$, its own best position P_{best} found so far, and interacts with neighbouring particles through the best position G_{best} discovered in the neighbourhood so far. At iteration k, each particle is moved according to equations (6) and (7):

$$\mathbf{V_i}(k+1) = w\mathbf{V_i}(k) + c_1 r_1 \left[P_{best}(k) - \mathbf{X_i}(k)\right] + c_2 r_2 \left[G_{best}(k) - \mathbf{X_i}(k)\right] \tag{6}$$

$$\mathbf{X_i}(k+1) = \mathbf{X_i}(k) + \mathbf{V_i}(k+1) \tag{7}$$

where c_1 and c_2 are acceleration coefficients, equal to 2, that scale the influence of the 'cognitive' and 'social' components; r_1 and r_2 are two random values, uniformly distributed in $[0, 1]$; w is inertia weight, which is updated as follows:

$$w(k+1) = w_{max} - (w_{max} - w_{min}) \cdot (k/k_{max})^{\frac{1}{\pi^2}} \tag{8}$$

where (w_{min}, w_{max}) is the range of inertia weight, with $w_{min} = 0.4$ and $w_{max} = 0.9$, k and k_{max} are the iteration number starting from iteration one and the maximum number of allowable iterations, respectively.

This law of variation of w increases the exploration of the search space in the first iterations of the algorithm, and the exploitation of the best solutions found so far towards the end of the algorithm.

3 The proposed algorithm

The kernelized fuzzy entropy clustering with spatial information and bias correction (KFECSB) model is a complex non-linear model. Using local search method (gradient method) for solving this model, solution of minimisation process can be trapped into local minima. By contrast, the algorithm has the advantage of fast converging. On the other hand, LHNPSO is a global optimization method, which can provide global optimum solution, but with a long convergence time. Thus, LHNPSO algorithm is used with minor improvements to optimize J_{KFECSB}, reaching global optimum solution by escaping local optima trap. The gradient method of KFECSB is used concurrently to guide the LHNPSO research process in order to converge faster and anticipate more accurate solutions. As such, a new MRI image segmentation algorithm based on KFECSB and LHNPSO, called PSO-KFECSB, is proposed.

To make sure that all particles are moving within the search space and avoiding divergent behaviour, their position and velocity are limited as follows:

$$v_{ir}(k+1) = \begin{cases} + \operatorname{rand}() \cdot v_{max}, & \text{if } v_{ir}(k+1) > v_{max} \\ - \operatorname{rand}() \cdot v_{max}, & \text{if } v_{ir}(k+1) < -v_{max} \\ + v_{ir}(k+1), & \text{otherwise} \end{cases} \qquad (9)$$

$$x_{ir}(k+1) = \begin{cases} \frac{1}{2} \cdot \operatorname{rand}() \cdot (x_{max} - x_{min}), & \text{if } x_{ir}(k+1) < x_{min} \\ & \text{or } x_{ir}(k+1) > x_{max} \\ x_{ir}(k+1), & \text{otherwise} \end{cases} \qquad (10)$$

where v_{max} is the largest allowable step size in any dimension; and, $[x_{min}, x_{max}]$ are the bounds of the search space in each dimension. In image clustering, commonly, v_{max} is set to 1 and (x_{min}, x_{max}) are the minimum and maximum of the feature (intensity or gray value) of the image.

The PSO-KFECSB algorithm is shown as Algorithm 1: In this work, the number of non-significant improvements of the partition matrix and the maximum number of iterations are used as the stopping criteria of the algorithm. Thus, if the condition $(\max\{\|\mathbf{U}_{Gbest,new} - \mathbf{U}_{Gbest,old}\|\} < \varepsilon)$ is completed k_{stop} times or the condition $(k > k_{max})$ is reached, the algorithm is immediately stopped.

4 Experimental results

This section presents the results of the experimental evaluation of the proposed algorithm. The algorithm is compared with five well-known image clustering algorithms in the literature: FEC [4], KFCMS2 [8], FLICM [9], MICO [7], and

---

Algorithm 1: PSO-KFECSB

Initialization: Set $k = 1$; Initialize all parameters for the LHNPSO algorithm:
$\{c_1, c_2, k_{max}, P, \ldots\}$ and the objective function according to Eqs. (1):
$\{C, n, \eta, \ldots\}$;

Results   : The cluster centres $\mathbf{C}$ and the partition matrix $\mathbf{U}$

1 repeat

2   **for** *each particle* **do**

3   Calculate kernel distance K; Calculate the partition matrix $\mathbf{U}$ using Eq. (4);

4   Calculate the objective function using Eq. (1);

5   Update the P_{best}, G_{best}, and $\mathbf{B}$ by using Eq. (3), G_{best}, and $\mathbf{U}$;

6   Update the inertia weight w using Eq. (8);

7   Update the positions $\mathbf{X}_i$ and the velocities $\mathbf{V}_i$ using Eqs. (6) and (7);

8   Map $\mathbf{X}_i$ and $\mathbf{V}_i$ into search space using Eqs. (9) and (10);

9 until *the stopping criteria are met*;

---

csFCM [10]. Two image datasets: simulated MRI brain images from BrainWeb [11] and real MRI brain images [12], with different levels of corrupting noise and INU artifact, are used to evaluate the performance of the proposed algorithm. The values of the parameters of the proposed method are based on conclusions in the relative literature and the trial and error technique. Parameter settings are given in Table 1.

Table 1: Setting of specific parameters in the PSO-KFECSB algorithm

Parameters	Values
Population size	40
$c_1 = c_2$	2.0
Degree of fuzzy entropy, n	20
Spatial constraint parameter, η	2.6
Controlling Gaussian kernel parameter, λ	$1/2\pi$
Number of non-significant improvement, k_{stop}	5
Terminate criterion parameter, ε	0.0001
Maximum number of iterations, k_{max}	100

In order to compare the results of different segmentation algorithms, supervised evaluation methods were used. In supervised evaluation methods, Jaccard index [13] (JAC) and Hausdorff distance [14] (HD) were used because current research [15] reports that they are suitable metrics for evaluation of different segmentation methods when there exist outliers with or without small segments, complex boundaries, low densities in the image.

4.1 Experiments on simulated MRI images

The proposed PSO-KFECSB and 5 other algorithms have been tested on normal brain images which have characteristics of Mobility T1, slice thickness 1mm, with different levels of noise (0%, 1%, 3%, 5%, 7%, 9%), and different levels of non-uniformity (0%, 20%, 40%). These images were segmented with 4 cluster centroids: background, cerebral spinal fluid (CSF), gray matter (GM), and white matter (WM).

Table 2: Segmentation evaluation in average of the simulated MRI image

Regions	Metrics	FEC	KFCMS2	FLICM	MICO	scFCM	proposed
CSF	JAC	0.9427	0.9598	0.9541	0.9549	0.9464	**0.9634**[*]
	HD	**73**	136	145	74	**73**	125
GM	JAC	0.7845	0.8482	0.8466	0.7504	0.8055	**0.8738**
	HD	72	**61**	**61**	85	72	**61**
WM	JAC	0.8460	0.9010	0.9039	0.7975	0.8680	**0.9243**
	HD	106	85	241	208	89	**81**

[*]The bold numerical values indicate the best performance.

Fig. 1: Simulated brain MRI images: (a) original T1-weighted axial image with 9% noise and 40% intensity of non-uniformity; (b) original image after skull stripping; (c) ground truth images; (d) FEC results; (e) KFCMS2 results; (f) FLICM results; (g) MICO results; (h) scFCM results; (i) PSO-KFECSB results.
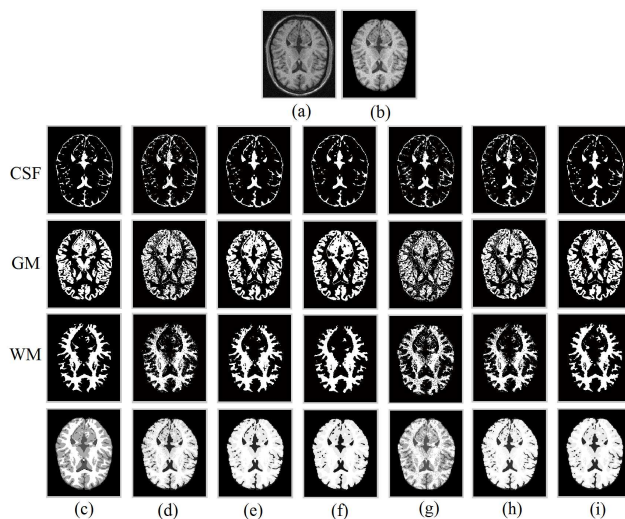

Figure 1 shows an example of segmentation of a simulated brain image (slice 80) with 9% noise and 40% intensity of non-uniformity. Visually, performance comparison of the six competitive algorithms is inexplicit. The quantitative performance evaluation (average in 10 program runs) is given in Table 2. In this table, it can be seen that the proposed PSO-KFECSB algorithm outperforms all competing algorithms, except for CSF with Hausdorff distance.

4.2 Experiments on real MRI images

The performance of the PSO-KFECSB algorithm has been also tested on the Internet Brain Segmentation Repository (IBSR) database [12]. Figure 2 shows an example of segmentation of a real brain MRI image, the 25_{th} plane slice of the 4_{th} volume. The number of tissue classes in the segmentation was set

to three, which corresponds to cerebrospinal fluid (CSF), gray matter (GM), and white matter (WM). Background pixels are ignored in the computation. A careful investigation of the results presented in Fig. 2 suggests that the proposed algorithm provides superior results as compared to the other five competing algorithms. Table 3 shows the quantitative evaluation (average in 10 program runs) of the performance. The results presented here validate the efficiency of the proposed PSO-KFECSB algorithm and also demonstrate its superiority over the FEC, KFCMS2, FLICM, MICO, and csFCM algorithms.

Fig. 2: Real brain MRI images: (a) original image; (b) manual segmented image; (c) ground truth images; (d) FEC results; (e) KFCMS2 results; (f) FLICM results; (g) MICO results; (h) scFCM results; (i) PSO-KFECSB results.
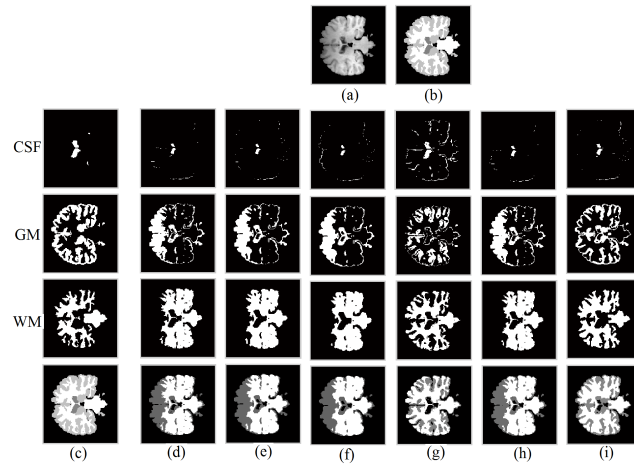
Table 3: Segmentation evaluation in average of the real MRI image

Regions	Metrics	FEC	KFCMS2	FLICM	MICO	scFCM	proposed
CSF	JAC	0.9696	0.9696	0.9659	0.9432	0.9696	**0.9738**[*]
	HD	**1405**	1700	1753	1640	**1405**	1700
GM	JAC	0.6863	0.6835	0.6756	0.7590	0.6835	**0.7872**
	HD	296	**290**	297	296	296	**290**
WM	JAC	0.6804	0.6777	0.6733	0.7800	0.6799	**0.7953**
	HD	293	360	377	85	361	**73**

[*]The bold numerical values indicate the best performance.

5 Conclusion

In this paper, an improved particle swarm optimization with a new fitness function, kernelized fuzzy entropy clustering with spatial information and bias correction, for segmentation of MRI images is proposed. Two drawbacks of fuzzy clustering algorithms, which are the trapping of the solution into local minima

and sensitivity to noise and INU artifact, have been partially overcome. The experimental results show that the proposed method is more effective in comparison with five competitive methods of the literature. However, there is still much ground work to cover. In particular, the determination of the parameters: n (degree of fuzzy entropy) and η (controlling parameter of spatial information) is an open question. In addition, only one criterion (J_{KFECSB}) is used to guide the search process of the solution, which may lead to the situation where the solution is a global optimum of the criterion, but may not be the optimum for the segmentation. To improve the performance, we plan to apply a multiobjective optimization approach, in order to exploit the strengths of other criteria.

References

1. Duncan, J.S., Ayache, N., Medical image analysis: progress over two decades and challenges ahead, IEEE Trans. Pattern Anal. Mach. Intell, 22, 85-106, 2000.
2. Simmons, A., Tofts, P.S., Barker, G.J., Arridge, S.R., Sources of intensity nonuniformity in spin echo images at 1.5 T, Magnetic Resonance in Medicine, 32(1), 121-128, 1994.
3. Gordillo, N, Montseny, E., Sobrevilla, P., State of the art survey on MRI brain tumor segmentation. Magnetic Resonance Imaging, 31(8), 1426–38, 2013.
4. Tran, D., Wagner, M., Fuzzy entropy clustering, IEEE International Conference Fuzzy Systems, 1, 152-157, 2000.
5. Yang, C., Gao, W., Liu, N., Song, C., Low-discrepancy sequence initialized particle swarm optimization algorithm with high-order nonlinear time-varying inertia weight, Appl. Soft. Comput., 29, 386-394, 2015.
6. Harrison, K.R., Engelbrecht, A.P., Ombuki-Berman, B.M., Inertia weight control strategies for particle swarm optimization, Swarm Intelligence, 10(4), 267-305, Dec 2016.
7. Li, C., Gore, J.C., Davatzikos, C., Multiplicative intrinsic component optimization (MICO) for MRI bias field estimation and tissue segmentation, Magnetic Resonance Imaging, 32, 913-923, 2014.
8. Chen, S.C, Zhang, D.Q, Robust image segmentation using FCM with spatial constraints based on new kernel-induced distance measure, IEEE Trans. Syst. Man. Cybern, Part B: Cybern, 34(4), 1907-1916, 2004.
9. Krinidis, S., Chatzis, V., A robust fuzzy local information c-means clustering algorithm, IEEE Trans. Image Process, 19, 1328-1337, 2010.
10. Adhikari, S.K., Sing, J.K., Basu, D.K., Nasipuri, M., Conditional spatial fuzzy c-means clustering algorithm for segmentation of MRI images, Appl. Soft Comput., 34, 758-769, 2015.
11. BrainWeb: Simulated Brain database, http://brainweb.bic.mni.mcgill.ca/brainweb/, 2017.
12. The Internet Brain Segmentation Repository, http://www.nitrc.org/projects/ibsr, 2017.
13. Jaccard, P., The distribution of the flora in the alpine zone, New Phytologist, 11(2), 37-50, 1912.
14. Beauchemin, M., Thomson, K.P.B, Edwards, G., On the Hausdorff distance used for the evaluation of segmentation results, CJRS, 24(1), 3-8, 1998.
15. Taha, A.A., Hanbury, A., Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool, BMC Medical Imaging, 15-29, 2015.

Crowd-Sourced Optimisation of Procedural Animation Systems

Gareth I. Henshall, William J. Teahan, and Llyr ap Cenydd

Bangor Univeristy, Wales, UK,
g.i.henshall@bangor.ac.uk

Abstract. Complex procedural animation systems are capable of synthesising vivid and organic character motion automatically. However due to potentially hundreds of interlinked parameters, the search space can be very large. Furthermore automatic optimisation of such systems is notoriously difficult as the results can be very subjective.

We start by describing issues with evolving and tuning procedural animation systems and survey related work. We then describe a web-based simulation which allows users to interactively rate a population of virtual creatures to a prescribed criteria. Each individual rating is stored and used to seed subsequent generations using a genetic algorithm, thereby facilitating crowd-sourced guiding of the system towards a desired motion or behaviour.

The results demonstrate that our system can successfully tune animation systems to a required specification. The next steps of our research include the optimisation of complex procedurally animated dolphins in both 2D and VR.

Keywords: procedural animation, parameter optimization, genetic algorithms

1 Introduction

Advances in computer graphics continue to permit the development of increasingly realistic real-time virtual characters. However, while rendering techniques allow for life-like visuals, the animation of the vast majority of complex characters are still heavily reliant on pre-created data like key-frame or motion capture sequences. While these techniques are very effective in specifically designed scenes, sequences or environments, the interaction between the character and its environment is primarily an illusion. In recent years, we have seen the increasing use of ancillary techniques like Inverse Kinematics (IK) for limb placement, and hair and cloth simulation, in order to provide a layer of dynamic simulation to augment the underlying data-driven sequences. None-the-less, there remains a disconnect between the character and virtual world.

The advent of presence inducing VR gives us for the first time the opportunity to exist in the same virtual world as virtual creatures with tangible volume. This opens challenges in many areas of animation, as now even the slightest hitch or

erroneous motion can be enough to break immersion. The intimate nature of such experiences also requires greater attention to behavioural realism, player interaction and an emphasis on micro animation such as eye contact.

In procedural animation, motion is calculated live at run-time using algorithms. This synthesis of motion through simulation potentially allows for much more interesting, complex and realistic behaviour to form. Procedural animation facilitates limitless permutations of motion in real time, allowing for a reactive system with a much wider range of actions and characteristics than would be possible by blending together predetermined sequences. Animating in this way is also just as viable for non-human characters, especially as some of the most widely used techniques such as motion capture do not apply as easily to non-humanoid forms.

Most non-trivial procedural animation systems will consist of interdependent components controlled by a large set of parameters. A walking system for example could have parameters for controlling the timing, speed, rotation and sequencing of all joints in an articulated figure. One of the main challenges when developing a procedural animation system is how to tweak potentially hundreds of interconnected parameters to produce desired results. Furthermore such complex systems can have emergent phenomenon, where the simple behaviours of small components combine to produce much more complex behaviour. A great example of this is Craig Reynolds Boids system [1], where three basic rules (cohere, separate, align) combine to synthesize complex flocking behaviour such as the murmuration of starlings.

As procedural animation systems become increasingly complex, it becomes harder for a single person or a development team to optimise the parameter space to a desired level, due to the vast number of permutations. Furthermore, the appearance and anthropomorphisation of a virtual creature is difficult to automate as the outcome can be very subjective, especially when the system is still in an embryonic state and being tuned to evoke emergent behaviour.

Ultimately, if the total number of parameters in a data set is n and the value range for this is r then the gross number of variations is r^n. For example, assuming a parameter has an integer value range of 1–15, then even if there are only five parameters, there would be 15^5 or 759,375 possible variations. With ten parameters this would increase to 5.77e+11 possible variations. A trial and error approach for searching a parameter space would not be efficient for a more complex creature animation system, as they contain hundreds of parameters with larger ranges. Another issue is that the vast majority of parameters are implemented using a continuous scale rather than a discrete integer scale. This means that the possible number of variations increases exponentially as accuracy improves. A more effective method of annealing and exploring wide animation parameter spaces could potentially allow for faster and more malleable development of complex animation systems.

Our approach aims to make parameter optimisation abstract, where the user makes their judgement based purely on the perceived animation, and never actually sees the numbers being adjusted or how they relate to one another. Further-

more, our system is automatically updated based on crowd-sourced data, and takes advantage of both human and computer based optimisation strategies.

Our long-term goal is to develop a system that allow for individuals or a group of users to search through the vast parameter space of complex procedural animation systems (and therefore the gamut of virtual creature behaviour) using powerful and user friendly techniques that are not reliant on knowledge of the underling structure of the system.

1.1 Related Work

A persistent challenge in real-time computer animation research is the creation of self-driven characters which have the ability to navigate around virtual environments. How natural a character's movement appears versus how much control can be exerted also continues to be a core issue, and there is always a balance to be found between animation accuracy and ease of control. As realistic as a motion synthesising system can potentially be, there will remain a desire to have as much control over the global movement and behaviour of the character as possible.

Data-driven constraint based approaches for blending a large library of motion captured sequences is still one of the most common approaches within both commercial software and research [2, 3]. A motion database is used to house a set of particular actions or movements for a character which are then blended together to create the illusion of natural movement [4, 5].

Motion primitives in procedural animation such as the paths of end-effectors or joint rotation can be described by using a parameterised formula. This is frequently used alongside inverse kinematic solvers [6–8]. Another approach is to us the direct application of torque and force on the figure to generate the animations in a physics based approach. The arthropod animation system described in [9] and the human movement system in [10] aim to combine physically and kinematic based techniques.

Examples of neural networks used to control animation include learning and optimisation strategies [11, 12], inverted pendulum models [13] and the commercial Euphoria Engine [14]. Active learning methods such as Bayesian optimisation [15] and the use of motion capture data to enhance procedural animation [16] are other examples of optimisation techniques.

Karl Sims [17] created a system in which an autonomous three-dimensional virtual creature can be generated and optimised without the need for heavy user interaction or specifications. This study was altered to accommodate a two-dimensional space a year later producing similar results [18]. Genetic algorithms (GA) are still being improved in terms of efficiency and effectiveness [19]. Similar methods have been used to solve problems outside of parameter optimisation [20] but the GA itself has the same structure and themes (selection, crossover, mutation). A GA's efficiency can be improved by combining a new crossover operator with a diversity operator as shown in [21].

A human-in-the-loop method for choosing appropriate optimised parameters is suggested to be the most effective method [22, 23]. A fundamental issue with

this method is that it requires the user to have a proficient knowledge of the underlying system that they are optimising. This is further confounded by the fact that complex animation systems often have parameters which coherently interact with each other, and modifying even a single parameter could produce unpredictable results.

2 Crowd Sourced Prototype

As part of a pilot study to test the validity and robustness of our technique, we developed a prototype system in the Unity game engine that uses crowd-sourcing to adjust the parameters of simple snake-like creatures.

There are two main aims of this prototype. Firstly, we aim to validate that it is possible to use crowd-sourcing to optimise a simple procedural animation system's parameters towards a set goal, from basic specifications such as "green snake with a long tail" and more abstract descriptions such as "a nervous, erratic snake". This includes ensuring that the system is robust, able to handle potentially hundreds of concurrent users, and capable of being directed towards a specified optimal parameter space.

Secondly, as procedural animation systems increase in complexity, tracking the result of optimisation will become increasingly difficult, as a motion might also be extremely complex and the result of many interdependent parameters and emergent behaviours. For example, while it is easy to validate that a snake creature is becoming faster or greener with each generation, perceiving the aggregate motion of a complex system will not be. Testing on a simple system allows us to ensure that there are no bugs or bias in the system that would be very difficult to detect when moulding more complex procedural animation systems.

2.1 Creature Model and Initialisation

The creature model for our prototype consists of a trail renderer attached to a rigid body sphere. The sphere resembles a snake's head, while the trail renderer resembles a tail (see Figure 1). The snakes move around a 2D plane environment by applying force and torque on the rigid body. MoveSpeed and TurnSpeed parameters are used to control the speed and turning rate respectively. Each snake steers towards an invisible target which changes position at a time set by a TargetMove parameter. TrailTime describes the length of the tail, while a parameter called TrailColour encodes the RGB colour.

At the start of the experiment, a script creates the first generation file consisting of 100 snakes with random parameters.

The colour parameters are in the range 0.0-1.0, while all other parameters are given suitable ranges. The full specification of the parameters can be seen in Table 1. Each parameter used to create the snake can be varied by 0.01 increments which means that across the seven parameters there are 2.4E16 possible snakes that can be generated. The 100 snakes in the first generation represent

Fig. 1: Single Snake

4.166E-15% of all permutations. The study uses 100 snakes in its original parameter file as it obtains approximately 100 ratings per generation, using anymore than 100 snakes per generation would result in too many snakes receiving no rating.

Table 1: Parameter list for prototype's snake creatures.

Parameter	Range	Description
Top Speed	5.0 - 20.0	The speed in which the snake moves around the environment.
Turn Speed	4.0 - 6.0	How sharply the snake can turn towards the target.
Trail Time	1.0 - 5.0	The time the trail renderer lasts for.
Target Move	1.0 - 5.0	How often the target changes position.
RGB	0.0 - 1.0	The RGB components of the trail renderer.

2.2   Rating System

We wish to use crowd sourcing as our method for optimising the parameter space, as we aim to gain the opinion of what constitutes a snake with a long purple tail from as wide a demographic as possible, ranging from computer illiterate to seasoned gamer. Getting as wide a range of opinions as possible means that the resulting snake will be tuned to the specification of the wider public as opposed to being specifically annealed by experts. For example a gamer could have very different or refined views on what they perceive to be a realistic procedurally animated creature when compared to a non-gamer. Their opinion might therefore be biased to a preferred game, behaviour or animation style whereas a non gamer will not necessarily have the same set of preconceived ideas. Using as wide a demographic as possible also allows us to test the robustness of our system more thoroughly.

The prototype system was built as a Unity OpenGL web app embedded into an HTML page. Upon loading the page, the user sees a screen similar to Figure 2.
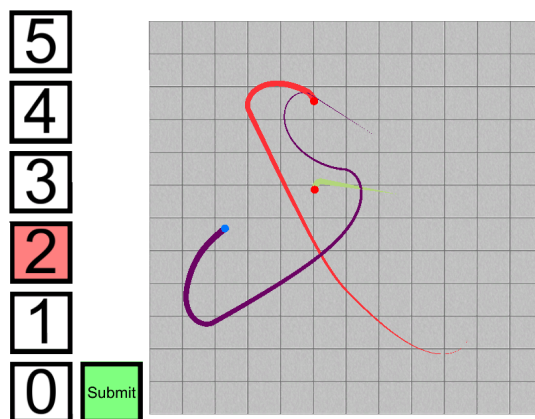
Fig. 2: Participant's view of the prototype snake study

The screenshot shows three snakes, one with a blue head and two with red heads. Once instantiated, the snakes will move around the plane indefinitely. The user is asked to rate the blue headed snake on how long and purple it is using a $0-5$ rating system. The webpage explains to the user that if the blue snake is inactive or appears broken, it would get a rating of 0, with progressively longer and more purple snakes given higher ratings. The user is then advised to keep on rating snakes for as long as they like, though we recommended that they rate at least 5 snakes so that they can see the general trends of the generation and start to form a more informed idea of the range of phenotypes as shown in Table.1. There is no quota for an individual session so the application will keep instantiating new snakes to rate indefinitely. Once the user feels they have contributed enough to the study they can simple close the page and their participation is terminated.

A server was set up to store and update the generated parameter files and associated ratings. Using a server allows for multiple users to run the program and rate snakes at the same time. When a snake is rated, the time-stamped parameters and associated rating are recorded. Snakes are selected randomly from the parameter file until there are sufficient ratings for a new generation. As the selection process is random, multiple users can rate the same snakes during a generation. As this study does not need any details from a participant there is no signing up or logging in process. Instead, they are automatically connected to the server and can start rating the creatures automatically. This also allows for users to leave and join the study with minimal overhead.

In order to get as many participants as possible, our experiment was advertised primarily through social media including Facebook, LinkedIn, Reddit and Twitter. We also recommended users to encourage others to participate in the studies.

2.3 Subsequent Generations

When a generation of snakes has been sufficiently rated, a script automatically generates the next generation. When creating a new generation, the current parameter and results are time-stamped and archived on a virtual machine. Through the genetic algorithm the ratings are then transferred into a 2D array and re-ordered back to the way the parameters are read by Unity. The parameters need to be re-ordered as the Unity build reads them in in a certain order where as the server saves them into alphabetical order. Our algorithm differs from others as we have included a heavy weighting on the crowd sourcing element. With our algorithm we also perform the selection prior to the mutation where as many GAs create a pool of new *children* perform a mutation chance and then proceed to use a selection method to choose which are included in the next generation

We decided that 100 ratings would be enough for each generation as it gives a reasonable chance that most snake variants have been rated at least once per generation. Once the snakes have been placed in order of rating, the top 25% are automatically selected for the new generation as the strongest candidates. All of the snakes' ratings are then given a fitness value as described in Algorithm 1. Using a roulette wheel selection method. two snakes are randomly chosen to be one of two parents which are used to generate two children for the next generation.

Using a single-point cross-over method, a random place within the parameters of the parents is selected as the crossover point. The children of these two snakes are then formed by combining the first portion of parent A and the second part of parent B. The opposite operation is performed to create the second child. These two children are then added to the new generation. The process is repeated until the next generations parameter file is full. Each individual parameter of a creature is given a 1% chance of mutating. If it is selected to mutate then the parameter value will increase or decrease within the upper and lower bounds of that parameter is created. After the new generation file is complete, it replaces the previous parameter file on the server. This process is automatic, seamless, and the end user will not notice any difference when rating. Users can start off rating one generation and finish rating a different generation.

3 Results

As previously mentioned, we asked participants to rate the snakes based on the description "purple snake with long tail". To judge how successful the system was, we only need to look at the trends in four of the parameters: TrialTime, and the RGB components of TrailTime. The other three parameters (TopSpeed, TurnSpeed & TargetMove), whilst still changeable, do not directly impact on the generated specification. Looking at the aforementioned four parameters, the model score is represented by the outer bounds of the parameters *model*:

$$TrailTime = 5, Red = 1, Green = 0, Blue = 1$$

Algorithm 1: Pseudo-code for the human-in-the-loop search algorithm to solve crowd sourced parameter optimisation

Input: Saved Ratings File
Output: Generation x Parameter File

1 $generation \leftarrow x$
2 download generation x parameter file and saved ratings to virtual machine
3 **while** *ratings open* **do**
4   re-order ratings file
5   add each re-ordered row into *newLines*

6 **while** *parameters open* **do**
7   add parameters to *params*

8 **for** *each ratingsRow* **do**
9   remove *rowId*

10 **if** *required ratings* **then**
11   **while** *true* **do**
12   average score for matching parameter row

13   sort *finalList* into score order
14   copy top 25% into *newGeneration*
15   **while** *true* **do**
16   determine parameter rows *fitness*

17   **while** *true* **do**
18   execute roulette wheel selection for *parent1*
19   execute roulette wheel selection for *parent2*
20   execute single-point crossover with *parent1* and *parent2*
21   execute mutation chance for *child1* and *child2*
22   add *child1* and *child2* to *newGeneration*

23   **while** *true* **do**
24   create new parameter file

25   update server parameter file

26 **else**
27   remove downloaded files

(a) Red Trends (b) Green Trends

(c) Blue Trends (d) TrailTime Trends
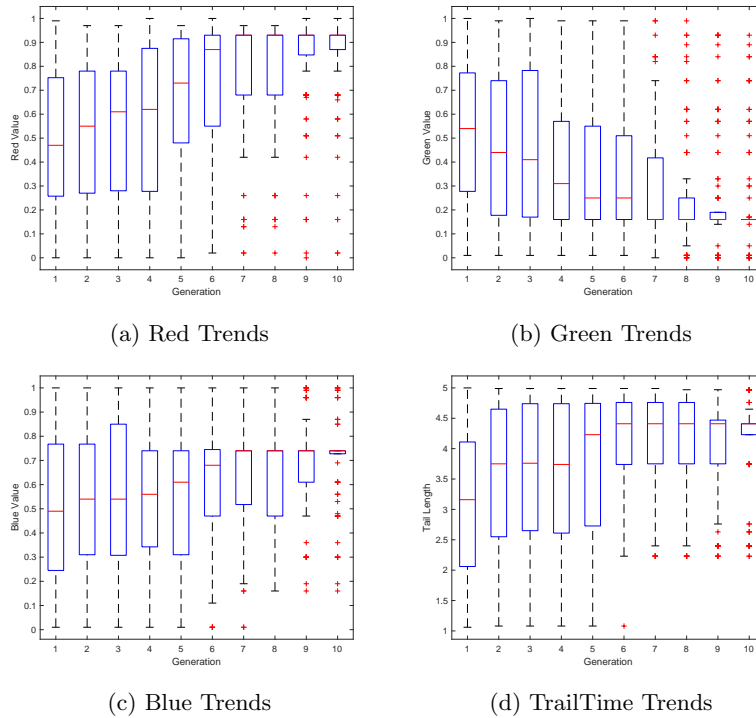
Fig. 3: RGB & TrailTime trends across generations

To quantify these results, we can observe the trends of these parameters over subsequent generations. As shown in Figure 3, the snake tail colour changes over the generations as expected with the Red & Blue elements increasing and the Green decreasing. The TrailTime also increases through the generations with an anomaly happening in generation four and a plateau occurring from generation six onwards. The anomaly in the TrailTime could be down to the participants placing more emphasis on the colour of the snake and therefore allowing for some shorter tailed snakes to gain higher ratings. Interestingly, throughout the generations the TopSpeed took a steady decrease (from 12.10 to 9.38) whereas the TurnSpeed and TargetMove parameters both stayed fairly stable to the original averages (4.97 to 5.19 and 2.04 to 1.74 respectively). So even though TopSpeed was not vital to the ratings description, it did slowly decrease over subsequent generations.

Another method of observing how successful the system was is to calculate the Manhattan and Square distances of each generation (Table. 2). The Manhattan distance shows the distance between two points on a grid based horizontal and vertical path, while the Square distance shows the distance between the two points in a diagonal or in a "*as the crow flies*" manner.
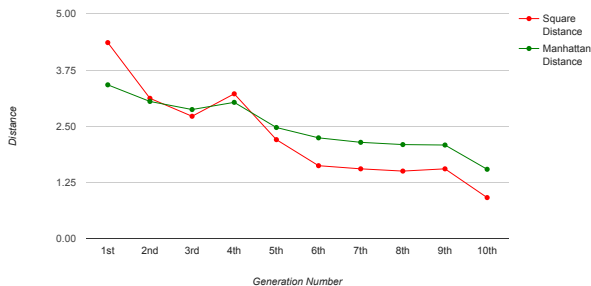
Fig. 4: Square and Manhattan distances for each Generation

As shown in Figure 4, as the study progressed both the Manhattan and Square distances gradually go closer to 0. This indicated that the snakes for each generation were trending towards to model answer for this study and therefore proving the system is capable of tuning parameters for a procedurally animated creature.

Table 2: Square & Manhattan distance equations

$$dist(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2 \qquad\qquad dist(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Square distance   Manhattan distance

4   Conclusion

The process of tuning a procedurally animated systems is still very subjective and difficult to automate and there is no "perfect" behavioural profile. Our aim is to develop techniques that allow users to interactively rate a creature's behaviour, appearance, etc. and therefore guide the development of a complex procedural animation system. By altering what aspect the user is rating, the system can produce vastly different results. The system could also allow for the simultaneous development of multiple complex systems. As you can see in Figure 5, (where Generation 1 is represented by the red headed snake, Generation 5 the orange headed snake and Generation 10 by the green snake), this prototype experiment has proven that the genetic algorithm combined with the ratings system study has been successful and we can now apply it to more complex systems.
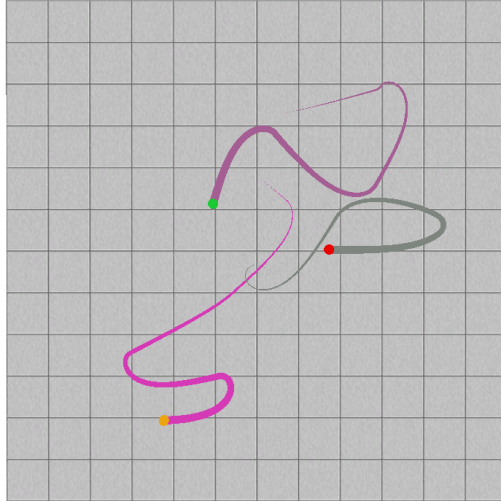
Fig. 5: The appearance of the average snakes for Generations 1 (red head), 5 (orange head) & 10 (green head).

4.1 Towards Crowd-Sourced Parameter Optimisation of 2D and VR Dolphins

The next step is to implement the same system within a much more complex procedurally animated system which we have been developing in tandem. This system consists of 37 adjustable parameters and is capable of producing life-like and diverse dolphin motion and behaviour. A screenshot of this system can be seen in Figure 6). The complexity and connectedness of the parameter space of this system vastly increases the variations of behaviour for each creature compared to our prototype. This larger parameter space will allow for us to thoroughly test if it is possible to crowd-source different types of animation behaviours by asking the user to create a creature based on adjectives such as 'playful', 'angry', 'calm' or 'friendly'. As these creature swim through a 3D virtual environment, they will be much harder to represent in image form, therefore using the same method as described above should allow us to ascertain the robustness of our system.

A further goal is to test whether a user's perception of the procedural creature is altered depending on whether it is viewed on a monitor (2D) or within Virtual Reality (VR). Using our crowd-sourcing method, we can see how the dolphins' parameter space alters through the generations giving developers a better idea as how to optimise a creature's parameters in the future. This experiment could potentially reveal if certain motion characteristics or even single parameters are more or less important across 2D and VR, and how best to develop and tune procedural animation systems for different mediums.
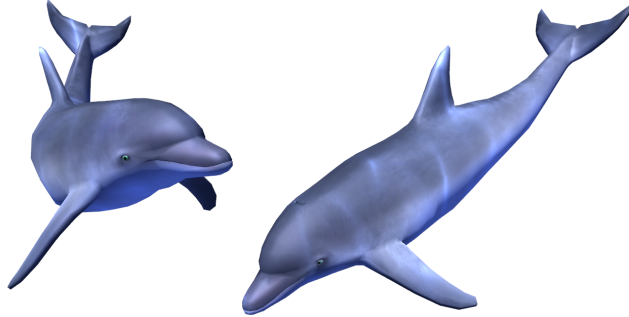
Fig. 6: Virtual dolphins generated by a procedural animation system we aim to optimise in future work.

References

1. Reynolds, Craig W. Flocks, Herds and Schools: A Distributed Behavioral Model. ACM SIGGRAPH '87 21.4: 25-34 (1987)
2. Fang, Anthony C and Pollard, Nancy S. Efficient Synthesis of Physically Valid Human Motion. ACM Transactions on Graphics (TOG) 22.3: 417–426 (2003)
3. Kovar, Lucas and Gleicher, Michael and Pighin, Frederic. Motion Graphs. ACM transactions on Graphics (TOG) 21.3: 472–482 (2002)
4. Lee, Yongjoon and Wampler, Kevin and Bernstein, Gilbert and Popovic, Jovan and Popovic, Zoran. Motion Fields for Interactive Character Locomotion. ACM Transactions on Graphs (TOG) 29.6: 138 (2010)
5. Lee, Jae Moon. Fast K-Nearest Neighbor Searching in Static Objects. Wireless Personal Communications: 93, 147–160 (2017)
6. Burrell, Thomas, et al. Feedback ControlBased Inverse Kinematics Solvers for a Nuclear Decommissioning Robot. IFAC-PapersOnLine 49.21: 177-184 (2016)
7. Meredith, Michael and Maddock, Steve. Real-Time Inverse Kinematics: The Return of the Jacobian. Technical Report No. CS-04-06, Department of Computer Science, University of Sheffield (2004)
8. van Basten, Ben JH and Stüvel, Sybren A and Egges, Arjan. A Hybrid Interpolation Scheme for Footprint-Driven Walking Synthesis. Proceedings of Graphics Interface 2011 9–16 (2011)
9. Cenydd, Llyr ap and Teahan, Bill. An Embodied Approach to Arthropod Animation. Computer Animation and Virtual Worlds 24.1: 65–83 (2013)
10. Bowman, Christopher, Hamido Fujita, and Gavin Perin. Towards a Knowledge Based Environment for the Cognitive Understanding and Creation of Immersive Visualization of Expressive Human Movement Data. International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer International Publishing (2016)
11. Yin, KangKang and Loken, Kevin and van de Panne, Michiel. Simbicon: Simple Biped Locomotion Control. ACM Transactions on Graphics (TOG) 26.3: 105 (2007)

12. Wang, Jack M and Fleet, David J and Hertzmann, Aaron. Optimizing Walking Controllers. ACM Transactions on Graphics (TOG) 28.5: 168 (2009)

13. Coros, Stelian and Beaudoin, Philippe and van de Panne, Michiel. Generalized Biped Walking Control. ACM Transactions on Graphics (TOG) 29.4: 130 (2010)

14. Natural Motion. Dynamic Motion Synthesis, (2011). http://www.naturalmotion.com/middleware/euphoria/ (02/03/2017)

15. Brochu, Eric and Brochu, Tyson and de Freitas, Nando. A Bayesian Interactive Optimization Approach to Procedural Animation Design. Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation 103–112 (2010)

16. Liang, Chang-Hung and Li, Tsai-Yen. Enhancing Procedural Animations with Motion Capture Data (a156).

17. Sims, Karl. Evolving Virtual Creatures. Proceedings of the 21st annual conference on Computer graphics and interactive techniques. ACM (1994)

18. Drew, Kenrick E., Andrew S. Forsberg, and Paton J. Lewis. Evolving Virtual Creatures. (1995)

19. Leung, Frank Hung-Fat, et al. Tuning of the Structure and Parameters of a Neural Network Using an Improved Genetic Algorithm. IEEE Transactions on Neural networks 14.1: 79-88 (2003)

20. Deng, Yong, Yang Liu, and Deyun Zhou. An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP. Mathematical Problems in Engineering 2015 (2015)

21. Elsayed, Saber M., Ruhul A. Sarker, and Daryl L. Essam. A New Genetic Algorithm for Solving Optimization Problems. Engineering Applications of Artificial Intelligence 27: 57-69 (2014)

22. Yin, KangKang and Coros, Stelian and Beaudoin, Philippe and van de Panne, Michiel. Continuation Methods for Adapting Simulated Skills. ACM Transactions on Graphics (TOG) 27.3: 81 (2008)

23. Johansen, Rune Skovbo. Automated Semi-Procedural Animation for Character Locomotion. Aarhus University (2009)

Discussion of a More Practice-Aware Runtime Analysis for Evolutionary Algorithms

Eduardo Carvalho Pinto[1] and Carola Doerr[2]

No Institute Given

Abstract. Theory of evolutionary computation (EC) aims at providing mathematically founded statements about the performance of evolutionary algorithms (EAs). The predominant topic in this research domain is *runtime analysis*, which studies the time it takes a given EA to solve a given optimization problem. Runtime analysis has witnessed significant advances in the last couple of years, allowing us to compute precise runtime estimates for several EAs and several problems.

Runtime analysis is, however (and unfortunately!), often judged by practitioners to be of little relevance for real applications of EAs. Several reasons for this claim exist. We address two of them in this present work:

(1) EA implementations often differ from their vanilla pseudocode description, which, in turn, typically form the basis for runtime analysis. To reduce the resulting gap between empirically observed and theoretically derived performance estimates, we suggest to take this discrepancy into account in the mathematical analysis and to adjust, for example, the cost assigned to the evaluation of search points that equal one of their immediate parents (provided that this is easy to verify as is the case in almost all standard EAs).

(2) Most runtime analysis results make statements about the time to reach an optimal solution only. This performance measure, however, does not give any insight into how the function values evolve over time. We therefore suggest to extend runtime statements to *runtime profiles*, covering the expected time needed to reach search points of intermediate fitness values.

1 Introduction

Evolutionary algorithms (EAs) are bio-inspired black-box optimization heuristics that are successfully applied to a broad range of industrial and academic optimization problems. Their practical relevance has motivated theoreticians to analyze EAs by mathematical means, aiming at providing insights into the fundamental working principles of EAs. Unfortunately, we see today a rather big gap between theoretical and practice-driven research in evolutionary computation (EC). Unlike in classical computer science, where a fruitful interplay between fundamental and empirical research exist, theory of EAs is regularly considered "useless" in the more practically-oriented part of the EC research community, cf., e.g., Footnote 4 in [10], where it is stated that "a best a practitioner of EAs can do is to stay away from theoretical results". One critical reason

for such claims is the fact that EAs are particularly useful when the problem at hand is not analyzable by thorough mathematical means; while for problems that do admit such a mathematical approach, problem-tailored algorithms are typically much more powerful than heuristics. This gap is very difficult, if not impossible, to close. At the same time, however, we also see that some of the other reasons for practitioners not to follow too closely what theory of EC can offer could be easily addressed. We suggest in this work two steps towards a more practice-aware theory of EAs. Our hope is to trigger with this discussion a more constructive exchange between theoretical and empirical research streams in EC.

Scope and Disclaimer: In the following, we consider the maximization of single-objective pseudo-Boolean functions $f : \{0, 1\}^n \to \mathbb{R}$, but all our suggestion can be applied to other search and objective spaces. For reasons of space, we present in this extended abstract only the general ideas. Rigorous theoretical as well as a number of empirical results are available from the authors upon request.

2   Implementation-Aware Runtime Analysis

One seemingly negligible difference between the runtime studied in the theory of EAs and that studied in practice- or empirically-motivated work is the fact that in most theory works function evaluations are counted also for iterations in which the offspring equals its parent (or one of its parents in case of recombination). In many EAs this situation occurs frequently. For example, when using standard bit mutation (Algorithm 2 with ℓ sampled from a binomial distribution $\mathrm{Bin}(n, p)$) with mutation rate $p = 1/n$, the probability that the mutated offspring equals its parent is $(1 - p)^n \approx 1/e \approx 0.368$, showing that, for example, the $(1 + 1)$ EA (Algorithm 1), which only uses standard bit mutation as variation operator, uses a $1/e$ fraction of its function evaluations for offspring that are identical to the parent. In practical implementations one would of course avoid such evaluations, in particular when it is easy to check (as in this case) that the offspring equals its parent.

Algorithm 1: The well-known $(1 + 1)$ EA with mutation probability $p \in (0, 1)$ for the maximization of a pseudo-Boolean function $f : \{0, 1\}^n \to \mathbb{R}$

---

1 Initialization: Sample $x \in \{0, 1\}^n$ uniformly at random and compute $f(x)$;
2 Optimization: for $t = 1, 2, 3, \ldots$ **do**
3 $\quad$ Sample $\ell \sim \mathrm{Bin}(n, p)$;
4 $\quad$ $y \leftarrow \mathrm{mut}_\ell(x)$;
5 $\quad$ evaluate $f(y)$;
6 $\quad$ **if** $f(y) \geq f(x)$ **then** $x \leftarrow y$;

Algorithm 2: mut_ℓ chooses ℓ different positions and flips the entries in these positions.
---
1 Input: $x \in \{0,1\}^n$, $\ell \in \mathbb{N}$;
2 Select ℓ different positions $i_1, \ldots, i_\ell \in [n]$ u.a.r.;
3 $y \leftarrow x$;
4 for $j = 1, \ldots, \ell$ **do** $y_{i_j} \leftarrow 1 - x_{i_j}$;

A similar situation, in which the offspring is likely to equal one of its parents, occurs when two similar parents are recombined. This is quite frequent in typical $(\mu + \lambda)$ GAs. Furthermore, a biased crossover favoring entries from one of the parents as, for example, used in the $(1 + (\lambda, \lambda))$ GA proposed in [5], has a relatively high probability to reproduce one of the parent solutions even if the Hamming distance of the two parents is large. Since in these cases it is typically quite easy to determine if the recombined offspring equals one of its parents, the offspring would not be evaluated in typical implementations, while in existing theoretical runtime analysis statements the algorithm is charged one function evaluation for creating this offspring.

These discrepancies between the analyzed and the actually implemented EAs can result in significant gaps between empirically observed and mathematically derived performance estimates. To reduce this gap, we suggest to reflect these observations in the runtime bounds, by analyzing EA variants in which offspring are not evaluated if they equal one of their parents and this equality is easy to determine. In the case of standard bit mutation, two straightforward ways to implement this strategy exist:

1. The first one simply ignores such iterations, thus effectively resampling an offspring until it differs from its parent. This strategy can be efficiently implemented by sampling the number ℓ of bits to flip by the mutation operator from a conditional distribution $\mathrm{Bin}_{>0}(n, p)$, which assigns to each k a probability of $\mathbb{P}[\ell = k | \ell > 0] = \binom{n}{k} p^k (1-p)^{n-k} / (1 - (1-p)^n)$.
2. The second idea is to flip *one* random bit in case $\ell = 0$ is sampled (thus effectively shifting the $(1-p)^n$ probability mass to flip zero bits to the probability to flip exactly one bit).

In the case of crossover there are many ways to deal with situations in which the mutated offspring equals its parent. When crossover is followed by mutation, we suggest to use one of the just-mentioned two solutions to ensure flipping at least one bit in the mutation step. When crossover is not followed by mutation (as in the $(1+(\lambda, \lambda))$ GA presented in [5]), we suggest to simply omit the function evaluation of an offspring equaling one of its two or more parents.

We note that previous attempts to establish more meaningful complexity measures in EC exists, most notably in the work of Jansen and Zarges [12], who propose to profile how much time is spend in each of the steps of an EA relative to the cost of the function evaluation.

Illustration. Figure 1 reports the average optimization times of 100 independent runs of the $(1 + 1)$ EA, Randomized Local Search (RLS, which uses $\ell = 1$ in line 3 of Algorithm 1 throughout), the $(1 + 1)$ EA$_{>0}$ (sampling ℓ from Bin$_{>0}(n, 1/n)$ in line 3 of Algorithm 1), and the $(1 + 1)$ EA$_{0 \to 1}$ (using $\ell = 1$ when $\ell = 0$ is sampled in line 3 of Algorithm 1) with mutation rate $p = 1/n$ on ONEMAX, for n ranging from 100 to 4 000. We observe that the expected performance indeed differs substantially from that of the "vanilla $(1 + 1)$ EA". More precisely, for $n = 4,000$, the $(1 + 1)$ EA$_{>0}$ saves about 35.6% in running time in comparison to the $(1 + 1)$ EA. For the $(1 + 1)$ EA$_{0 \to 1}$ the relative expected savings is 47.1%. We also see that, non-surprisingly, the average performance of the $(1 + 1)$ EA$_{0 \to 1}$ is better than that of the $(1 + 1)$ EA$_{>0}$.
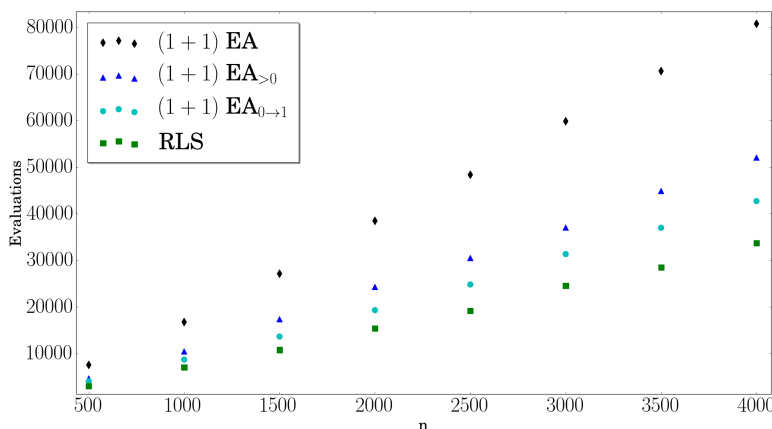
Fig. 1. Average runtimes for 100 independent runs of the respective algorithms (with mutation probability $p = 1/n$ for the $(1 + 1)$ EA and its variants) on ONEMAX for different problem sizes n.

3   Runtime Profiles

Our second suggestion concerns the problem that in most theoretical works on EAs for discrete optimization problems, only the expected times to hit an optimal solution for the first time are reported. In realistic environments, we do not know when this is the case, making it equally important to understand how the fitness values *evolve* over time. We therefore suggest the concept of *runtime profiles*, the expected time needed to hit intermediate target values. When canonical fitness levels exits, such as in the case of ONEMAX, LEADINGONES, royal road, and

several other functions, we suggest to use these. For other functions, such as linear functions or weighted combinatorial graph problems, the analysis of the expected optimization time often identifies useful to report target fitness levels. In the absence of these, a linear interpolation of the minimal and maximal fitness value could be used.

We emphasize the fact that runtime profiles have, very naturally, been reported in many empirical works on heuristic optimization. We are thus not suggesting a new concept here, but want this matter to the attention of researchers working on fundamental aspects of EC. Note that, in contrast to the previous section, our suggestion does not *change* any of the algorithms nor the runtime bounds. We merely suggest to report them in a different way.

As we can observe in Figure 2, it is quite interesting to observe how the first hitting times for the different targets evolve. Already for the simple LeadingOnes benchmark functions we see that the $(1+1)$ EA-variants proposed above are superior to RLS for all target fitness values i that are smaller than some relatively large threshold value v, while RLS reaches fitness levels $i \geq v$ faster than the $(1+1)$ EA-counterparts. Reporting only the expected optimization time therefore does not do justice to the better performance of the $(1+1)$ EA-variants in the earlier parts of the optimization process.

We are confident that the proposed runtime profiles will be very useful for the design and the analysis of non-static parameter and operator choices in EAs, such as adaptive mutation or crossover rates, selection pressure, etc. This topic, also studied under the notions of *hyper-heuristics*, *meta-heuristics*, etc., has very recently seen increased interest in the theory of EC community [3, 4, 6, 7]. It is a highly relevant topic in empirically-driven research in EC, cf. [1, 10, 11, 14] and references therein.

Our runtime profiles complement the *fixed-budget perspective* proposed in [13], where statements about the expected fitness values after a fixed number of fitness evaluations are sought. Runtime profiles and fixed-budget perspectives are orthogonal views on the performance of EAs, both aiming at providing more insight into the optimization behavior than what the single runtime measure can offer. Neither of these two measures is entirely new but rather summarize performance statements frequently reported in empirical works. The aim of [13] as well as our own work is to motivate researchers working in the theory of EC to include in their statements these more informative performance guarantees.

In Figure 2 we plot runtime profiles of the $(1+1)$ EA and its above-proposed variants, as well as for Randomized Local Search. These bounds have been computed by adjusting the proofs in [2]. We observe that the $(1+1)$ EA$_{>0}$ is the best of all four algorithms for intermediate fitness levels $\leq 7\,980$. The $(1+1)$ EA$_{0 \to 1}$ has the smallest expected runtime to reach the intermediate fitness values between $7\,980$ to $8\,998$, while RLS is the best of the four algorithms only for fitness levels $i > 8\,999$. RLS is faster than the $(1+1)$ EA$_{>0}$ for intermediate fitness values $i > 8\,566$. As mentioned above, such insights are very important for the design of parameter/operator selection schemes.
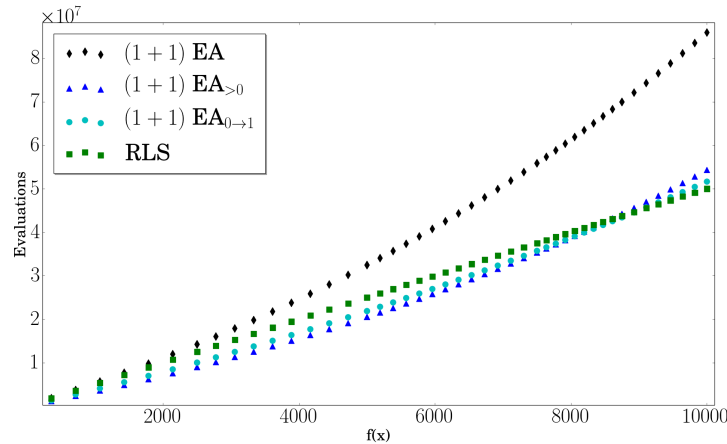
Fig. 2. Runtime profile of the respective algorithms (with mutation rate $p = 1/n$ for the (1+1) EA and its variants) on LeadingOnes for problem size $n = 10\,000$ according to our theoretical result.

4 Discussion and Future Works

We hope to trigger with this work an extended discussion on how to make theoretical results in the domain of evolutionary computation more relevant and interpretable for practitioners. We have suggested two different steps into this direction, (1) do not charge an algorithm for function evaluations when the offspring equals one of its parents (in case this is easy to detect), and (2) report first hitting times not only for the optimum but also for intermediate fitness levels. Naturally, our work can only be a pointer to a more practice-aware theory, and we are aware that there are many more steps that have to be taken. In particular, we believe that the following observations need to be discussed in more detail.

- Many runtime statements report only expected optimization times. However, it is often interesting to understand the *probability distribution of the optimization time*, in particular for problems where the variance can be large. Runtime analysis has recently seen an increased interest in these runtime distributions, cf., for example, [8,15] and follow-up works.
- Similar to the previous point, problems exist where the expected optimization time can be very large even if the probability to hit an optimal solution within a small number of iterations is small. In [9] a so-called p-Monte Carlo complexity measure has been introduced, measuring the expected time to *hit an optimal solution with probability at least* $1 - p$. Similar suggestions can be found in [16].

303

- The runtime profiles suggested in Section 3 complements the *fixed-budget view* advocated by Jansen and Zarges [13]. We feel that there is a need for combined measures that are capable of describing the *anytime behavior of an EA*. Regret-based measure as used in machine learning could be a key here, but we haven't been able so far to identify a fully satisfying measure.

From a theoretical point of view, our suggested changes are easily implementable. Our work has nevertheless unveiled a quite remarkable result, the superiority of the Greedy $(2+1)$ GA_{mod} over any unary unbiased black-box algorithm. We are confident that our performance measure will yield similar results for other problems and algorithms, with the potential of changing our view on fundamental questions like the benefits of crossover over mutation, (dis-)advantages of elitism vs. non-elitism, etc.

References

1. A. Aleti and I. Moser. A systematic literature review of adaptive parameter control methods for evolutionary algorithms. *ACM Comput. Surv.*, 49:56:1–56:35, 2016.
2. S. Böttcher, B. Doerr, and F. Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Proc. of Parallel Problem Solving from Nature (PPSN'10)*, volume 6238 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2010.
3. D. Dang and P. K. Lehre. Self-adaptation of mutation rates in non-elitist populations. In *Proc. of Parallel Problem Solving from Nature (PPSN'16)*, volume 9921 of *Lecture Notes in Computer Science*, pages 803–813. Springer, 2016.
4. B. Doerr and C. Doerr. Optimal parameter choices through self-adjustment: Applying the 1/5-th rule in discrete settings. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15)*, pages 1335–1342. ACM, 2015.
5. B. Doerr, C. Doerr, and F. Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87 – 104, 2015.
6. B. Doerr, C. Doerr, and T. Kötzing. Provably optimal self-adjusting step sizes for multi-valued decision variables. In *Proc. of Parallel Problem Solving from Nature (PPSN'16)*, volume 9921 of *Lecture Notes in Computer Science*, pages 782–791. Springer, 2016.

7. B. Doerr, C. Doerr, and J. Yang. k-bit mutation with self-adjusting k outperforms standard bit mutation. In *Proc. of Parallel Problem Solving from Nature (PPSN'16)*, volume 9921 of *Lecture Notes in Computer Science*, pages 824–834. Springer, 2016.

8. B. Doerr and L. A. Goldberg. Adaptive drift analysis. *Algorithmica*, 65:224–250, 2013.

9. C. Doerr and J. Lengler. Elitist black-box models: Analyzing the impact of elitist selection on the performance of evolutionary algorithms. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15)*, pages 839–846. ACM, 2015.

10. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3:124–141, 1999.

11. A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 19–46. Springer, 2007.

12. T. Jansen and C. Zarges. Analysis of evolutionary algorithms: from computational complexity analysis to algorithm engineering. In *Proc. of Foundations of Genetic Algorithms (FOGA'11)*, pages 1–14. ACM, 2011.

13. T. Jansen and C. Zarges. Performance analysis of randomised search heuristics operating with a fixed budget. *Theoretical Computer Science*, 545:39–58, 2014.

14. G. Karafotias, M. Hoogendoorn, and A. Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19:167–187, 2015.

15. T. Kötzing. Concentration of first hitting times under additive drift. *Algorithmica*, 75:490–506, 2016.

16. D. Zhou, D. Luo, R. Lu, and Z. Han. The use of tail inequalities on the probable computational time of randomized search heuristics. *Theoretical Computer Science*, 436:106–117, 2012.

SIDE EVENT:
ART & SCIENCE
IN EVOLUTIONARY COMPUTATION

Art&Science
in Evolutionary Computation

a side event of the EA2017 conference with

Galerie
LOUCHARD

`http://www.galerielouchard.paris`

For the second time[4], an artistic event is associated to the **Artificial Evolution** conference. Selected artwork obtained through an **evolutionary** paradigm (evolution, artificial life, swarm...) will be exposed in **Galerie Louchard** (Paris, 19th) from October 24th to November 12th. The corresponding papers are published in the newborn journal: Art and Science (ISTE OpenScience) (https://www.openscience.fr/Arts-et-sciences).

The vernissage is part of the EA2017 conference on

26 October 2017,19:30

Scientific & artistic committee

Emmanuel Cayla,
Evelyne Lutton,
Marie-Christine Maurel,
Nicolas Monmarché.

---

[4]The catalog of the first event is available at: `http://ea2013.inria.fr//EA2013-catalogue_side-event_A3.pdf`

ART EXPOSITION ABSTRACTS

THE HORIZON PROJECT: "Emotion in lines"

Patricia Hernández Rondán[1], Francisco Fernández de Vega[2], Cayetano Cruz[2], Vicente Albarrán[2], Mario García[3], Lilian Navarro[2], Tania Gallego[4], Itzel Andrea García[5]

[1] *Department of Drawing at the Faculty of Fine Arts, Seville, Spain,*
[2] *University of Extremadura, Spain,*
[3] *Tijuana Institute of Technology, Mexico,*
[4] *Autonomous Community of Extremadura, Spain,*
[5] *CETYS University, Baja California campus, Tijuana, Mexico.*

Abstract: This artistic project originated as an experiment undertaken to analyse the mental imagery the brain uses for the expression of emotions on a formal, conceptual and iconographic level. In the interest of understanding the nature of creativity in human beings as a tool that favours innovation and the ability to rediscover ourselves, it is considered essential to dive into its waters, in order to generate new approaches, and to pave new paths, transcending what has been experienced in the world of visual arts until now. After two prior experiments, we decided to approach this knowledge applying new premises. As in previous projects, we contemplated the Artificial Intelligence screen, more specifically, pondering Evolutionary Algorithms (EAs) because these deal with a technique that is linked more directly to biological processes than traditional computing techniques, to draw out everything possible on the subject. In this area we have adopted as constructive cornerstones towards a new form of creation:

- Connected art
- Shared authorship
- The community of artists
- Interaction with the work

Topics: Evolutionary
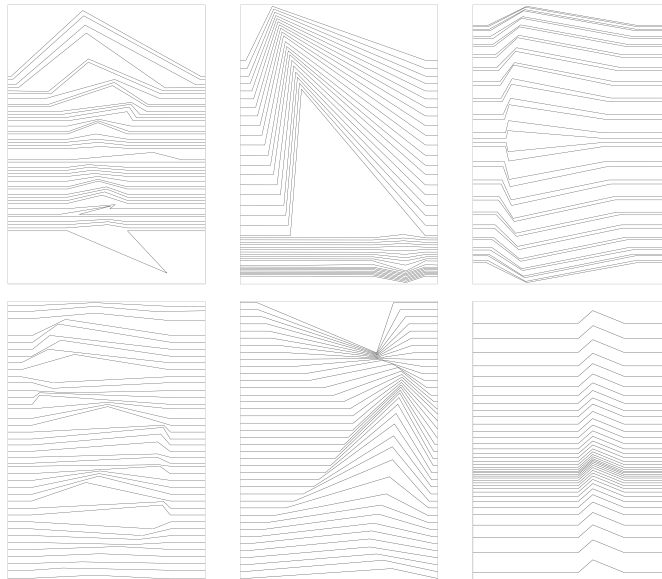
Sleep Paintings

Carlos M. Fernandes

LARSyS: Laboratory for Robotics and Systems in Engineering and Science University of Lisbon, Portugal

Abstract: Sleep Paintings is a swarm art project that consists of a series of coloured drawings generated by an ant-based clustering algorithm operating with sleep data sets extracted from electroencephalogram (EEG) signals.
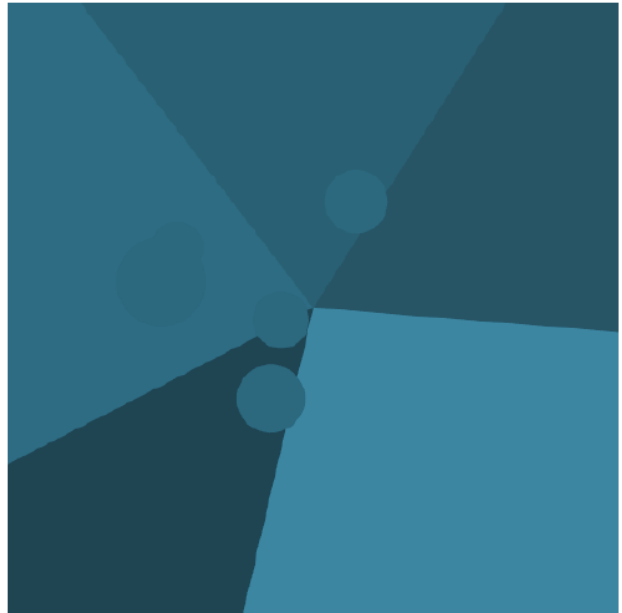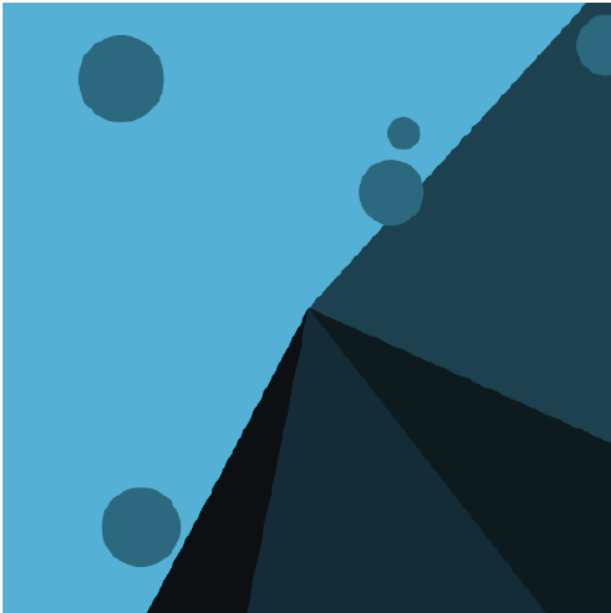
Topics: Swarm art
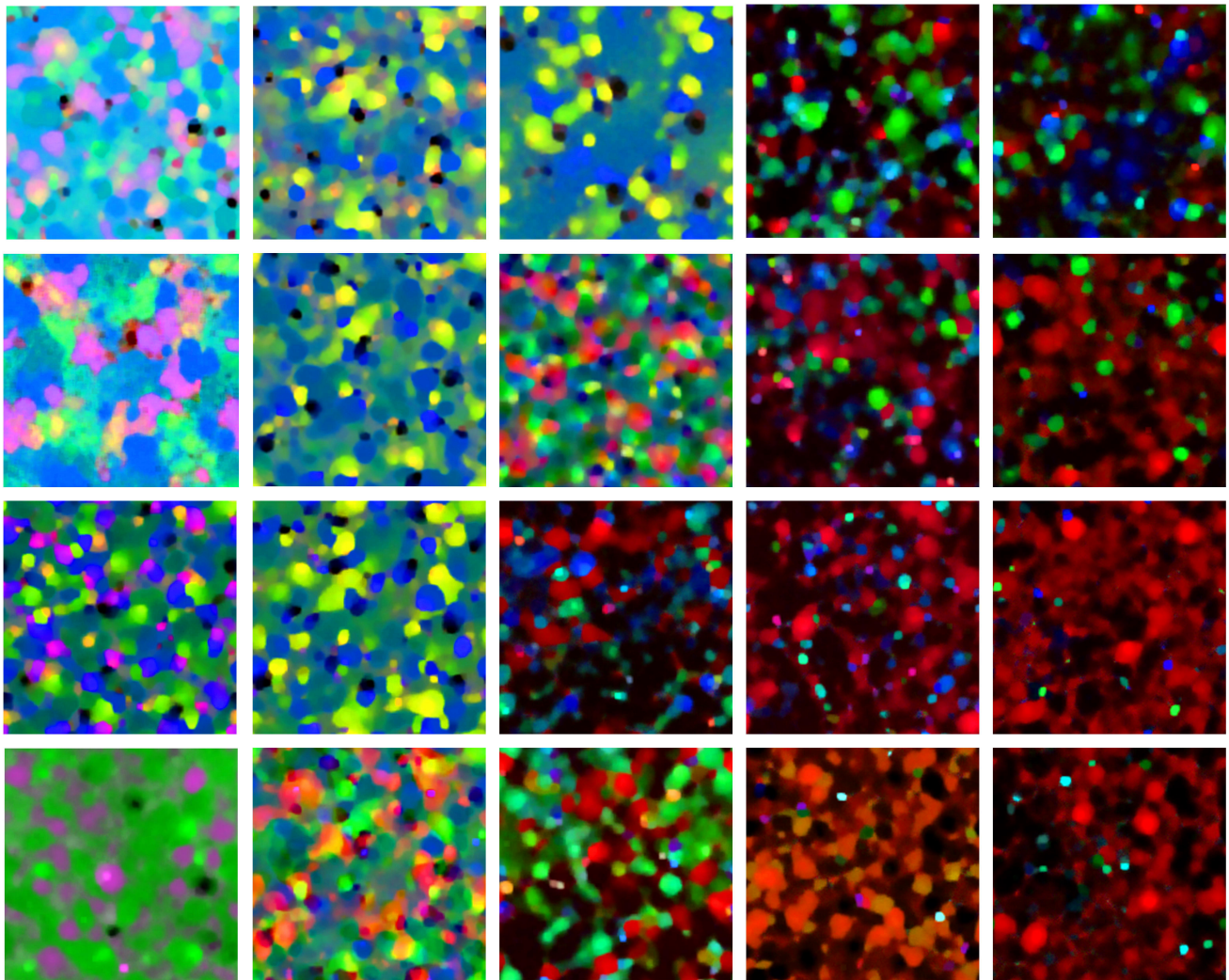
EMERGILIENCE

Sophie Lavaud

*Visual artist and researcher at Institut ACTE (Arts, Créations, Théories et Esthétiques),
Université Paris 1/CNRS, Paris, France*

Abstract: This work explores shape emergence due to self-organised phenomena.

Topics: Self organised shapes

ART BEINGS

Alain Lioret

Arts et Technologies de l'Image, Université Paris 8

Abstract: The series of Art Being has been initiated in 2002, based on the simple principle of a population of artificial being, as the organic matter of a numeric artwork. The series consists in creating evolving ecosystems, based on evolutionary computation.

Topics: Genetic evolutionary art

Fractal Beings. © Alain Lioret . 2017

Fly4Arts: Evolutionary Digital Art with the Fly Algorithm

Zainab Ali Abbood and Franck P. Vidal

School of Computer Science, Bangor University, UK

Abstract: The aim of this study is to generate artistic images, such as digital mosaics, as an optimisation problem without the introduction of any a priori knowledge or constraint other than an input image. The usual practice to produce digital mosaic images heavily relies on Centroidal Voronoi diagrams. We demonstrate here that it can be modelled as an optimisation problem solved using a cooperative co-evolution strategy based on the Parisian evolution approach, the Fly algorithm. An individual is called a fly. Its aim of the algorithm is to optimise the position of innitely small 3-D points (the flies). The Fly algorithm has been initially used in real-time stereo vision for robotics. It has also demonstrated promising results in image reconstruction for tomography. In this new application, a much more complex representation has been studied. A fly is a tile. It has its own position, size, colour, and rotation angle. Our method takes advantage of graphics processing units (GPUs) to generate the images using the modern OpenGL Shading Language (GLSL) and Open Computing Language (OpenCL) to compute the difference between the input image and simulated image. Different types of tiles are implemented, some with transparency, to generate different visual effects, such as digital mosaic and spray paint. An online study with 41 participants has been conducted to compare some of our results with those generated using an open-source software for image manipulation. It demonstrates that our method leads to more visually appealing images.

Topics: Digital mosaic, Evolutionary art, Fly algorithm, Parisian evolution, cooperative co-evolution
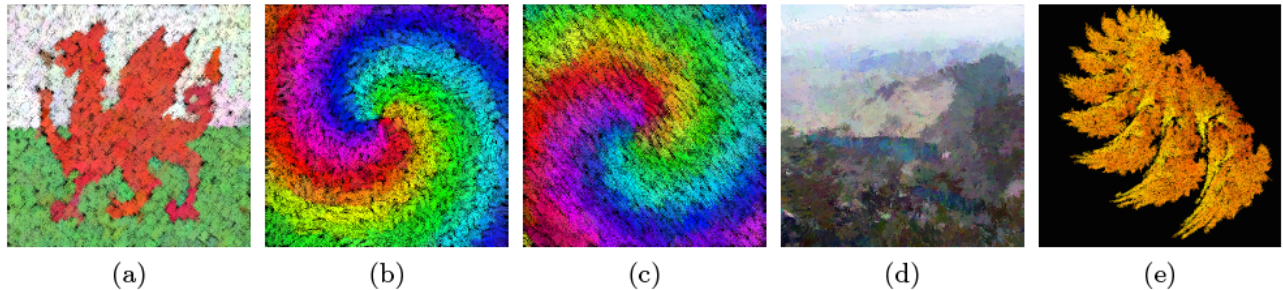

Fig. 1: Evolutionary art using the Fly algorithm: (a) Welsh flag, (b) a multicolour spiral, (c) another multicolour spiral, (d) view from Garnedd Ugain of Y Lliwedd in Snowdonia (Wales), and (e) from an image generated using ArtiE-Fract (courtesy of Dr Évelyne Lutton).

Index of authors