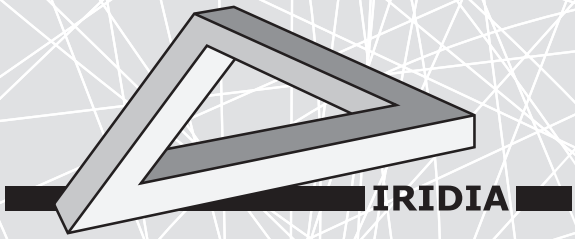


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

SLS-DS 2007: Doctoral Symposium on Engineering Stochastic Local Search Algorithms

Enda Ridge, Thomas Stützle,
Mauro Birattari, and Holger H. Hoos

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2007-014

September 2007

SLS-DS 2007

Doctoral Symposium on Engineering Stochastic Local Search Algorithms

Edited by:

Enda Ridge, The University of York, York, UK
Thomas Stützle, Université Libre de Bruxelles, Brussels, Belgium
Mauro Birattari, Université Libre de Bruxelles, Brussels, Belgium
Holger H. Hoos, University of British Columbia, Vancouver, Canada

7 September 2007, Brussels, Belgium

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2007-014

Preface

The inaugural **Doctoral Symposium on Engineering Stochastic Local Search Algorithms** (SLS-DS) was held at the Université Libre de Bruxelles, Belgium on 7 September 2007 jointly with the **SLS 2007 Workshop**. SLS-DS is a forum for doctoral students to present their work and obtain guidance from fellow researchers as well as to provide contact with other students at a similar stage in their careers. The symposium exposes students to helpful criticism before their thesis defence, and fosters discussions related to future career perspectives. The symposium consists of a series of short presentations followed by a poster session.

The papers in these proceedings were selected based on relevance, quality and clarity of presentation. They provide a useful guide to emerging research and new trends in the stochastic local search field. The topics covered include:

- Methodological developments for the implementation of SLS algorithms.
- Experimental studies of SLS algorithms (behaviour of SLS algorithms, comparison of SLS algorithms, ...), problem characteristics and their impact on algorithm performance.
- Case studies in the development of well designed SLS algorithms.
- Aspects that become relevant when moving from "classical" NP-hard problems to those including multiple objectives, stochastic information or dynamically changing data.

We would like to thank the people involved in the local organisation of the Engineering Stochastic Local Search Algorithms workshop for their time and effort assisting the running of the doctoral symposium.

Enda Ridge
Thomas Stützle
Mauro Birattari
Holger H. Hoos

Program Co-Chairs
SLS-DS 2007

Contents

Effective Stochastic Local Search Algorithms for the Genomic Median Problem

Renaud Lenne[1,3], Christine Solnon[2], Thomas Stützle[3],
Eric Tannier[4] and Mauro Birattari[3]
[1]Université Lyon 1, Lyon, France
[2]LIRIS, Université Lyon 1, Lyon, France
[3]IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
[4]INRIA Rhône-Alpes, LBBE, Université Lyon 1, Lyon, France

Abstract

The Genomic Median Problem is an optimization problem inspired by a biological issue: it aims at finding the genome organization of the common ancestor to multiple living species. It is formulated as the search for a genome that minimizes some distance measure among given genomes. Several attempts have been made at solving the problem. These range from simple heuristic methods to a stochastic local search (SLS) algorithm that is inspired by a well-known local search algorithm for the satisfiability problem in propositionnal logic, called `WalkSAT`. The objective of this study is to implement improved algorithmic techniques, particularly ones based on tabu search, in the quest for better quality solutions for large instances of the problem. We have engineered a new high-performing SLS algorithm, extensively tested the developed algorithm and found a new best solution for a real-world case.

1 Introduction

The objective of the Genomic Median Problem (GMP) is to find the probable organization of the genome of the common ancestor of two species, given a third more distant one as a comparison. The study and the solutions to this problem can lead to discover properties of common ancestors of existing species and help making better classifications.

The GMP is an optimization problem that can be formulated as follows. Given three genomes and a distance function that measures in some way the number of rearrangements needed to move from one genome to another one, find a fourth genome that minimizes the sum of the distances between this new one and the three given ones.

There have been various attempts at solving the problem algorithmically ranging from rather simple heuristics [8, 2] to more complex local search

algorithms [3, 6]. These existing algorithms produce solutions that are often of good quality but that are not necessarily optimal and for larger instances there may be significant gaps to optimal solutions. In addition, compared to the currently available local search techniques, the approaches are rather simple and therefore one can conjecture that there is room for improving their performance.

Motivated by these observations, we developed a new high performing stochastic local search (SLS) algorithm for the GMP. The goal of this new SLS algorithm is to improve upon the performance of the current state-of-the-art algorithms in terms of the run-time required to reach specific bounds on the solution quality and ideally to find also better quality solutions, thus, providing new state-of-the-art solutions that may be of biological relevance.

2 Model and methods

A *genome* is an unordered set of chromosomes; a *chromosome* is an ordered list of markers, where each *marker* is modelled by a different signed integer. The three genomes of a GMP instance share the same set of n markers. Hence, each genome is modelled by a signed permutation of the same n integers grouped by chromosomes. The number of chromosomes is not fixed a priori and it may vary from one genome to another.

Various methods for solving the *GMP* have been proposed. These either work on a simplified problem that considers only one chromosome and that involves finding the median of a signed permutation, like `GRAPPA` [8] or `AmGRP` [2]; or use rather simplistic search methods like `MGR-MEDIAN` [3], which uses a greedy constructive algorithm. The best performance results so far have been reported for `MedRByLS` [6], a local search algorithm that is inspired by `WalkSAT`, a well-known local search algorithm for the satisfiability problem in propositionnal logic.

We base our algorithm on the same data structures and neighborhood as used in `MedRByLS`:

- The distance between two genomes is approximated by the BreakPoint Graph distance described in [5, 7, 1], and later extended to multi-chromosomal genomes in [6]. This distance is defined with respect to the number of cycles and paths in an edge bicolored graph obtained from the two genomes by linking adjacent markers. This distance is computed in linear time with respect to the number of markers.

- A *move* consists in the change of two edges with the same colour by inverting one of their nodes. From the biological point of view, such a move corresponds to a transformation in one genome, while the distance between two genomes is the minimum number of such transformations needed to transform one genome into the other.

We have first re-implemented `MedRByLS`, thus allowing a direct comparison of our new algorithms to the original `MedRByLS` using a same implementation of the data structures. For this comparison, we verified that our re-implementation matches the performance of the original version.

As a next step, we enhanced the local search by a simple tabu search scheme. For the search diversification of the resulting tabu search algorithm, we integrated it into the iterated local search framework by adding appropriate perturbations and acceptance criteria. This resulted in an algorithm that we called `MedITaS` (for Median solver by Iterated Tabu Search). More into details, it consists of the following main algorithmic components.

1. A simple Tabu Search (TS) algorithm that forbids the reversal of the last t moves (where t is the tabu tenure), that is, the last changed nodes. We considered a *first-improvement* strategy for the choice of the move to perform because the neighbourhood is very large so that a best-improvement strategy (that implies a full scan of the neighborhood) would be too time-consuming.

2. An Iterated Local Search (ILS) algorithm that perturbates the solution when the search is stuck in plateau-moves or in a basin of attraction (that is, when too many already visited solutions are recalculated). The perturbation uses a rearrangement of k edges and then TS is rerun starting from the perturbed solution. Finally, an acceptance criterion decides whether either the solution before the perturbation or the one after is kept for the next iteration of ILS. The implemented acceptance criterion accepts a new solution if it is better than the previous one; otherwise, the previous solution has a user-defined probability of being kept (in our test, we used the default value of 0.2).

3. A reactive version of TS (that reactively adapts the tabu list length) and a reactive version of ILS (that reactively tunes the perturbation strength) have been implemented. This was done since initial experiments showed that both ILS and TS were very sensitive to parameter settings and that the optimal parameter settings were very different from one instance to one other.

3 Results

In order to evaluate our algorithms, we ran multiple comparisons. All runs were made on a same Dual-Core AMD Opteron2216 HE (2 processors at 2.4GHz) with 4GB of RAM; only one core is used for each execution since our algorithm is implemented as a fully sequential one.

The first set of data contains 22 randomly generated instances of different difficulties (with respect to the definition of the phase transition by [6]) but with the same size (500 markers). The set has 11 levels of hardness and

2 instances per level. On this set we run our `MedITaS` algorithm and our implementation of the basic local search algorithm `MedRByLS` from [6] for 20 independent trials on each instance and 40 seconds per trial. The comparison of the best solution qualities reached by both algorithms on each instances is given in Figure 1. From this figure, we can see that `MedITaS` always gives solution qualities that are at least as good as `MedRByLS` and that the gap between the two algorithms tends to increase as the instances become harder.

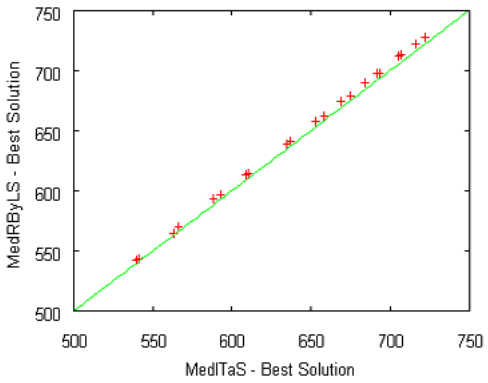


Figure 1: Solution Value Comparison

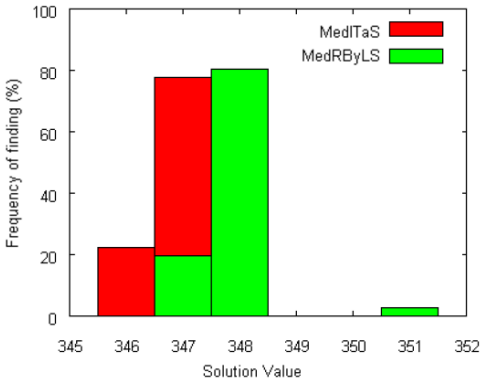


Figure 2: Frequency Comparison

In another experiment we used a real-world instance: the human-mouse-rat comparison, which was also used in [6]. This instance is made of 424 markers and the best median found so far had a value of 346. We ran each algorithm 35 times for a computation time limit of 60 seconds. From these runs, we generated Figure 2, which represents the histogram of the frequency of finding certain solution qualities with the two main algorithms (`MedITaS` and `MedRByLS`). It is clear from this graph that `MedITaS` finds solutions that are at least as good as those found by `MedRByLS` and always of a very good quality (of 347 or better); `MedRByLS` sometimes fails to find good solutions: on some runs it returned a solution of value 351). We should also notice that `MedRByLS` has a quite low probability (less than 20%) of finding a solution of 347 or better. Also, our algorithm found a new best solution for this instance with an evaluation function value of 345.

4 Discussion

Our implementation of the Iterated Tabu Search gave very promising results. First, we have seen that `MedITaS` always gives at least as good or better results, in the same computation time, than the former best algorithm (`MedRByLS`). We also found a new best solution for the human-mouse-rat common ancestor.

The developped algorithmic techniques perform significantly better than previously developped ones from a solution quality point of view. But from a biological point of view, the distance used here (as the one used in all preceding attempts at solving the problem) does not seem to reflect the biological reality of the evolution process (as it is explained in [4]). Thus, a research on a more appropriate distance measure has to be envisaged.

Also, we noted in our experiments that there were a lot of medians with exactly the same value. It could be a good idea to do some comparison between them trying to extract some valuable information on the most probable characteristics of the real ancestor.

Acknowledgements. The authors would like to thank to Yannet Interian for her kind help in any questions regarding the implementation of her algorithm. Thomas Stützle acknowledges support from the Belgian FNRS of which he is a Research Associate.

References

[1] V. Bafna and P. Pevzner. Genome rearrangements and sorting by reversal. *SIAM Journal on Computing*, 25:272–289, 1996.

[2] M. Bernt, D. Merkle, and M. Middendorf. Using median sets for inferring phylogenetic trees. *Bioinformatics - Oxford Univ Press*, Volume 23, Number 2:e129–e135, 2007.

[3] G. Bourque and P. Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Res.*, 12(1):26–36, 2002.

[4] N. Eriksen. Reversal and transposition medians. *Theoretical Computer Science*, 374(1-3), 2007.

[5] S. Hannenhalli and P. A. Pevzner. Polynomial algorithm for genomic distance problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, 1995.

[6] Y. Interian and R. Durrett. Computing genomic midpoints, 2007. Submitted.

[7] J. D. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13(1/2):180–210, 1995.

[8] B. Moret, S. Wyman, D. Bader, T. Warnow, and M. Yan. A new implementation and detailed study of breakpoint analysis, 2001. Proc. 6th Pacific Symp. on Biocomputing (PSB 2001), Hawaii, World Scientific Pub.

Composing Particle Swarm Optimization Algorithms

Marco A. Montes de Oca, Thomas Stützle,
Mauro Birattari and Marco Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

Abstract

Particle Swarm Optimization (PSO) is primarily used for solving continuous optimization problems. Over the years and as a result of the interest on PSO, many different variants of the original algorithm have been proposed and, more often than not, it is claimed that the new variant is superior in some way. It follows that if two variants differ only by some algorithmic components, then their difference in performance can be ascribed to differences in these components. A question then arises: Can we compose a high-performing PSO variant using components present in other variants? In this paper, we present an attempt to answer this question. The performance of the resulting composite variant suggests that the answer is positive. We believe that algorithm composition can help in devising effective optimization algorithms.

1 Introduction

Searching on the web for Particle Swam Optimization (PSO) algorithms[1], yields thousands of papers about it. In this body of literature there are hundreds of variations of the original algorithm. Anyone entering the field would soon realize that there is little agreement as to what is the state-of-the-art PSO algorithm. This is so because, in most cases, the newly introduced variant is reported to be superior to some reference algorithm in some way.

The differences between PSO variants range from added constants [1] to evolved particle-movement rules for specific problems [7]. In cases in which algorithms can be structurally decomposed into compatible algorithmic components (i.e., that perform the same function), differences in performance can be ascribed to differences in these components.

If some components are found to provide good performance on a class of problems, could a composite algorithm (that uses these components) exhibit better performance than the variants from which these components

[1] Try "Particle Swarm Optimization" in Google Scholar, for example.

were taken? We address this question by comparing some PSO variants on a set of common benchmark problems and then by building a composite PSO algorithm. This algorithm is, in turn, compared to the variants from which the components were taken. Our results suggest that the strengths of different variants can be combined into a single one. Algorithm composition is an interesting approach to devise effective optimization algorithms.

2 Comparing PSO Variants

A swarm is composed by a population of particles $\mathcal{P} = \{p_1, \ldots, p_n\}$. At time step t, a particle p_i has an associated position vector $\boldsymbol{x}_i^t$, a velocity vector $\boldsymbol{v}_i^t$, and a memory vector $\boldsymbol{pb}_i^t$ (called *personal best*) which stores the best position the particle has found until time step t. A particle p_i has a neighborhood $\mathcal{N}_i \subseteq \mathcal{P}$ of particles. Some commonly used neighborhoods are the whole swarm (fully connected topology), four neighbors (square topology), and two neighbors (ring topology). The best position in a particle's neighborhood is denoted by $\boldsymbol{lb}_i^t$ and is called *local best*.

PSO algorithms iterate updating the particles' velocities and positions until a stopping criterion is met. The original velocity- and position-update rules are: $\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \varphi_1 \boldsymbol{U}_1^t(\boldsymbol{pb}_i^t - \boldsymbol{x}_i^t) + \varphi_2 \boldsymbol{U}_2^t(\boldsymbol{lb}_i^t - \boldsymbol{x}_i^t)$ and $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}$ where φ_1 and φ_2 are two parameters called the *cognitive* and *social* acceleration coefficients respectively, $\boldsymbol{U}_1^t$ and $\boldsymbol{U}_2^t$ are two diagonal matrices with random in-diagonal elements uniformly distributed (generated at every iteration) in the interval $[0, 1)$.

In this work, we use a high level decomposition of PSO algorithms. This decomposition is shown in Algorithm 1. The algorithmic variants that we included in our study are representatives of the most common approaches taken to date for updating various parameters that influence performance (second to last step in the algorithmic decomposition) and for updating a particle's velocity (last step).

Algorithm 1 Basic structure of a PSO algorithm

```
Set parameters and initialize particles
while termination condition is not met do
    Evaluate particles' position and update their memory
    Update dynamic/adaptive components, if any
    Update particles' velocity and position
end while
```

The variants that we studied are the constriction factor PSO [1], three variants of the time-varying inertia weight PSO [9, 2, 10], FIPS [4], HP-SOTVAC [8], and AHPSO [3]. The experimental setup and the results of a comparison of their relative performance on a common set of benchmark problems can be found in [5, 6].

The results of our experimental study suggest that stagnation tendencies (with respect to solution quality) of PSO algorithms can be alleviated by using large populations and/or low connected topologies. Some PSO algorithms use time-varying parameters and scheduling them in different ways affect their performance. Slow inertia weight schedules favor exploration in the decreasing inertia weight PSO while fast ones favor fast convergence rates. A variant that dynamically changes the population topology over time (AHPSO) outperforms other variants (when using low connected topologies) during the first iterations.

3 Variant Composition: Results and Discussion

After the analysis of the comparison results, a new PSO algorithm integrating three algorithmic components was designed. The components are (i) a dynamic topology – as in AHPSO, (ii) a mechanism for updating a particle's velocity that provides fast convergence – as in FIPS, and (iii) a decreasing inertia weight. This *Frankenstein's PSO* was compared with the variants from which its main components were taken. See Table 1 for the results. They were obtained using the best configuration (from a set of commonly used configurations) for each algorithm. For all tested run-lengths (except for that of 10^4 function evaluations) the best ranked algorithm is the composite PSO. This result suggests that indeed it is possible to get good performance by just combining existing algorithmic components into composite variants. Perhaps more importantly, this result highlights the importance of understanding the effects of different algorithmic components and their interactions on the algorithms' performance. We want to pursue this research direction in the future.

Table 1: Best overall configurations of different PSO variants for different termination criteria. Each group is sorted by the average standardized solution quality (i.e., the number of standard deviations a given score is from the group's mean) in ascending order, so the best overall configuration is listed first.

FES	Algorithm	Ackley	Griewank	Rastrigin	Salomon	Schwefel	Step	Rosenbrock	Sphere	Average
10^3	Frankenstein's PSO	**-2.024**	**-0.955**	-0.975	**-0.517**	1.378	**-1.315**	-0.302	**-1.108**	-0.727
	Increasing-IW	-0.013	-0.393	-0.950	-0.323	**-1.229**	-0.645	-0.367	-0.371	-0.536
	Decreasing-IW	-0.002	-0.386	**-1.067**	-0.316	-1.199	-0.359	-0.474	-0.425	-0.528
	FIPS	-0.765	-0.430	-0.080	-0.457	1.432	-0.932	0.206	-0.538	-0.195
	Canonical	0.476	-0.156	0.287	-0.276	-0.213	0.406	**-0.491**	-0.057	-0.003
	Stochastic-IW	0.656	0.124	0.652	-0.237	-0.046	0.693	-0.488	0.304	0.207
	AHPSO	0.476	-0.156	0.287	2.464	-0.213	0.406	**-0.491**	-0.057	0.340
	HPSOTVAC	1.198	2.353	1.847	-0.338	0.090	1.745	2.406	2.251	1.444
10^4	Increasing-IW	-0.129	-0.564	-0.593	-0.349	**-0.797**	-0.539	-0.348	-0.359	-0.460
	Canonical	-0.212	-0.616	-0.591	-0.373	-0.459	-0.539	-0.376	-0.359	-0.441
	Decreasing-IW	-0.065	-0.518	**-0.962**	-0.341	-0.754	-0.085	-0.370	-0.358	-0.431
	Frankenstein's PSO	**-1.061**	**-0.761**	0.056	**-0.386**	1.332	**-0.993**	**-0.414**	**-0.361**	-0.324
	Stochastic-IW	-0.131	0.443	-0.512	-0.361	-0.541	-0.085	-0.290	-0.359	-0.230
	FIPS	-1.056	-0.718	1.567	-0.378	1.760	-0.539	-0.364	**-0.361**	-0.011
	AHPSO	0.569	0.656	-0.512	2.474	-0.641	0.596	-0.312	-0.316	0.314
	HPSOTVAC	2.086	2.077	1.546	-0.287	0.101	2.185	2.473	2.475	1.582
10^5	Frankenstein's PSO	**-0.354**	**-0.883**	-1.192	**-0.359**	**-1.548**	-0.487	0.782	**-0.354**	-0.549
	Decreasing-IW	**-0.354**	0.631	-0.709	-0.355	-0.311	**-0.787**	-0.983	**-0.354**	-0.402
	Increasing-IW	**-0.354**	0.631	0.108	-0.355	-0.271	**-0.787**	-0.441	**-0.354**	-0.228
	Canonical	**-0.354**	**-0.883**	0.313	**-0.359**	0.729	-0.487	0.216	**-0.354**	-0.147
	Stochastic-IW	**-0.354**	0.631	1.130	**-0.359**	0.649	**-0.787**	-1.013	**-0.354**	-0.057
	FIPS	**-0.354**	**-0.883**	1.060	-0.355	1.372	0.712	1.008	**-0.354**	0.276
	AHPSO	**-0.354**	1.639	0.721	2.475	0.529	0.712	**-1.019**	**-0.354**	0.544
	HPSOTVAC	2.475	**-0.883**	**-1.431**	-0.334	-1.149	1.911	1.449	2.475	0.564
10^6	Frankenstein's PSO	**-0.354**	**-0.354**	-0.787	**-0.358**	-1.257	**-0.661**	-0.058	**-0.504**	-0.542
	Increasing-IW	**-0.354**	**-0.354**	0.002	-0.354	0.019	**-0.661**	0.039	**-0.504**	-0.271
	Decreasing-IW	**-0.354**	**-0.354**	0.472	-0.354	0.367	**-0.661**	**-0.778**	**-0.504**	-0.271
	FIPS	**-0.354**	**-0.354**	-0.546	-0.354	**-1.349**	0.661	0.685	**-0.504**	-0.264
	Stochastic-IW	**-0.354**	**-0.354**	0.415	**-0.358**	0.705	**-0.661**	-0.529	**-0.504**	-0.205
	Canonical	**-0.354**	**-0.354**	0.815	**-0.358**	1.072	**-0.661**	-0.717	**-0.504**	-0.132
	HPSOTVAC	2.475	**-0.354**	**-1.760**	-0.341	-0.705	0.661	2.129	2.184	0.536
	AHPSO	**-0.354**	2.475	1.388	2.475	1.149	1.984	-0.771	0.840	1.148

References

[1] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[2] R. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, pages 94–100, Piscataway, NJ, USA, 2001. IEEE Press.

[3] S. Janson and M. Middendorf. A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man and Cybernetics–Part B*, 35(6):1272–1282, 2005.

[4] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210, 2004.

[5] M. A. Montes de Oca, T. Stützle, M. Birattari, and M. Dorigo. Frankenstein's PSO: An Engineered Composite Particle Swarm Optimization Algorithm. Technical Report TR/IRIDIA/2007-006, IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium, 2007.

[6] M. A. Montes de Oca, T. Stützle, M. Birattari, and M. Dorigo. Frankenstein's PSO: Complete data, 2007. Supplementary information page at `http://iridia.ulb.ac.be/supp/IridiaSupp2007-002/`.

[7] R. Poli, C. D. Chio, and W. B. Langdon. Exploring extended particle swarms: A genetic programming approach. In *Proceedings of the 2005 conference on Genetic and Evolutionary Computation*, pages 169–176, New York, NY, USA, 2005. ACM Press.

[8] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8(3):240–255, 2004.

[9] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69–73, Piscataway, NJ, USA, 1998. IEEE Press.

[10] Y.-L. Zheng, L.-H. Ma, L.-Y. Zhang, and J.-X. Qian. Empirical study of particle swarm optimizer with an increasing inertia weight. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*, pages 221–226, Piscataway, NJ, USA, 2003. IEEE Press.

A Study of Stochastic Local Search Algorithms for the Quadratic Assignment Problem

Mohamed Saifullah Bin Hussin, Thomas Stützle and Mauro Birattari,
IRIDIA, CoDE, Université Libre de Bruxelles, Belgium

Abstract

In this article we present preliminary results on the comparison of stochastic local search algorithms for the Quadratic Assignment Problem. Our experimental comparison is based on re-implementations or adaptations of some high performing stochastic local search algorithms and it is done using default configurations and configurations tuned by F-race, a tool for the automatic tuning of parameters of stochastic algorithms.

1 Introduction

The Quadratic Assignment Problem (QAP) is an $\mathcal{NP}$-hard combinatorial optimisation problem. A QAP instance is defined by n items and n locations and two $n \times n$ matrices A and B, where a_{ij} is the distance from location i to location j, and b_{rs} is the flow between item r and item s. The objective of the QAP is to find an assignment of items to locations, such that each item is assigned to exactly one location, no location is assigned more than one item, and the sum of all products of the pairs of distances $\times$ flows is minimised.

The largest instance from QAPLIB, a widely used benchmark resource for research on the QAP, that has been solved by an exact algorithm is the $ste36a$ instance with $n = 36$; its solution took about 186 hours on a Pentium III 800 MHz CPU. The $ste36a$ instance, however, can be solved by state-of-the-art stochastic local search (SLS) algorithms in a few seconds on similar speed computers. Hence, given their clear advantage over exact algorithms, it is clear that there has been an enormous interest in SLS algorithms for the QAP. However, due to different experimental protocols, from the QAP literature the best performing algorithms for the various QAP instance classes cannot be fully reliably determined. Therefore, one main goal of our research is to investigate the relative performance of SLS algorithms for various instance classes of the QAP. In this extended abstract, we present some preliminary results of our research efforts where we compare SLS algorithms based on several of the most prominent SLS methods. For

each of these methods we use for our experiments one version with default parameter settings and one with the parameters tuned by F-race [2, 1].

2 SLS Methods

Each of the SLS methods we implemented is based on a local search that uses the 2-exchange neighbourhood, where two solutions are neighboured if they differ in the assignment of exactly two items.

Simulated Annealing (SA) is an SLS method that is inspired from the physical annealing process. We adopt the SA variant proposed by Connolly [3] in addition to implementing a rather basic SA variant using a standard cooling schedule. In the tuning both, Conolly's and the 'usual' SA variants were considered together.

Tabu Search (TS) uses the search history to escape from local minima and to implement an explorative search strategy. We reimplemented a TS variant of Taillard [7], the robust Tabu Search (RTS) algorithm. We tuned three parameters: the number of iterations before the secondary aspiration criterion that implements a search diversification is applied, and the interval from which the tabu tenure is chosen periodically at random; this interval is defined by a starting tabu list size, and the interval length.

Iterated Local Search (ILS) is a simple, yet effective metaheuristic that was derived from the idea of iteratively building a sequence of local minima to efficiently sample local optima. We implemented various possible operators for an ILS for the QAP. For the perturbation fixed perturbation size schemes and schemes from Variable Neighbourhood Search were considered; the possibilities for the local search range from first-improvement, best-improvement algorithms to short tabu search runs. For the acceptance criterion, we considered five different possibilities ranging from very greedy to very permisive ones. (Note that various of the ILS variants tested in [5] are not included in this present study.)

Ant Colony Optimisation (ACO) is a metaheuristic that is inspired by the pheromone trail laying and following behaviour of real ant colonies when foraging. We adopt the $\mathcal{MAX}-\mathcal{MIN}$ Ant System, ($\mathcal{MM}$AS), an improvement over the Ant System by Stützle and Hoos [6]. We tune three parameters for $\mathcal{MM}$AS; the local search option to be used (essentially the same options as available for the ILS), the number of ants, and the pheromone evaporation factor.

Memetic Algorithms (MAs) are a population-based search technique that can be seen as a skilled combination of evolutionary algorithms with problem-specific operators, most notably local search. Here, we use a re-implementation of a MA by Merz and Freisleben [4]. We tune five parameters which are the local search option to be used (the same options as for the ACO algorithm have been considered), the population size, the mutation factor, the child factor, and the mutation strength.

3 Experimental setup and results

We have generated a new set of benchmark instances for the QAP. These instances are composed of all combinations of three different possibilities for the distance matrix—random distance matrices (R), Manhattan distances from a grid (G), and Euclidean distances (S)—and two possibilities for the flow matrix—random flows (R) and structured flows (S). This results in six instance classes: GR, GS, RR, RS, SR, and SS. We use the F-race procedure using a sampling-based approach for the definition of the algorithm configurations to be considered to tune each of the above mentioned five SLS methods on each instance class [1]. On each instance, all of which are of size $n = 80$, the SLS algorithms have available 27 seconds on an AMD Opteron 244 1.75 GHz processor with 1 MB L2-Cache and 2GB RAM, running under the Rocks Cluster GNU/Linux. (The time corresponds roughly to a run of RTS of $1000 \cdot n$ iterations. For each instance class, 200 instances were available for tuning. After the F-Race has finished, 100 fresh instances from each instance class are run independently using default and tuned parameter settings for comparing the performance of the SLS algorithms.

We measure the performance of the different SLS algorithms running once each algorithm on each test instance. We then compute the average value of the returned solutions on each instance class when using different SLS algorithms. Here, we only present results at a high-level comparing the performance of the algorithms on the various instance classes. In Tables 1 and Table 2, we give the computational results for the default parameter settings and the winning parameter settings, respectively. In particular, for each algorithm and instance class (this is done independently for default and tuned parameter settings), we compute the average solution value. In the tables is given the percentage excess of each algorithm over the algorithm that obtained the lowest average solution quality. (Note that when comparing for each SLS algorithm and instance class pair the default and the tuned parameter settings, except for two such pairs, always a statistically significant difference in average solution quality was observed; that is, the tuned versions are in almost all cases statistically better than the defaults. In part, these differences were quite substantial.)

Considering default configurations, $\mathcal{MM}$AS and RTS showed the best

Table 1: Distance from the best solution by using default parameter settings

Instance Class	$\mathcal{MM}$AS	ILS	MA	RTS	SA
GR	**0.00**	*0.44*	0.09	0.03	0.14
GS	0.31	*2.90*	0.79	**0.00**	1.03
RR	0.41	*0.94*	0.62	**0.00**	0.84
RS	2.84	*6.11*	4.45	**0.00**	5.56
SR	**0.00**	0.47	0.09	*0.65*	0.11
SS	**0.00**	*1.75*	0.14	1.62	0.26

Table 2: Distance from the best solution by using winning parameter settings

Instance Class	$\mathcal{MM}$AS	ILS	MA	RTS	SA
GR	0.01	0.07	**0.00**	0.01	*0.16*
GS	0.10	0.33	0.17	**0.00**	*1.16*
RR	0.10	0.14	0.05	**0.00**	*0.65*
RS	0.27	0.79	0.14	**0.00**	*4.73*
SR	0.04	0.16	**0.00**	0.17	*0.18*
SS	0.16	*0.49*	**0.00**	0.38	0.29

performance in most of the cases. $\mathcal{MM}$AS achieves the best results for instance classes GR, SR, and SS, while RTS achieves the best results for instance classes GS, RR, and RS. In general, however, the dependence of the performance of RTS on the instance class seems to be higher than for $\mathcal{MM}$AS. While $\mathcal{MM}$AS is either the best or second best algorithm, RTS, performs very poorly for the instance classes SR and SS. ILS seems to be the lowest performing algorithm among the chosen ones. On most instance classes, except in case of instance class SR, it is the last ranked algorithm.

Considering the tuned versions, MA achieves the best performance for instance class GR, SR, and SS, while RTS achieves the best results for instance classes GS, RR, and RS. In almost all cases, SA achieves the worst performance compared to other SLS algorithms. The exception is for instance class SS where ILS is worst. Overall, MA achieves the best results compared to other SLS algorithms.

4 Conclusions

The best performing algorithms for the various instance classes were found to be the tuned version of either RTS or MA. Among the default settings, among the best ranked ones was the $\mathcal{MM}$AS, which indicates that it offers rather robust performance across a wide set of different types of instances. When considering the ranking of the different algorithm variants, apparently the MA was profiting most from the tuning in the sense that for most

instance classes it could improve its relative ranking when compared to the other algorithms. Finally, the performance of RTS was rather dependent on the instance class. While for several ones it was among the top two ranked algorithms, on two instance classes it was among the two worst ones.

References

[1] P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo. Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. In *4th International Workshop on Hybrid Metaheuristics, Proceedings, HM 2007*, LNCS. Springer Verlag, Berlin, 2007.

[2] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 11–18, 2002.

[3] D. T. Connolly. An improved annealing scheme for the QAP. *European Journal of Operations Research*, 46(1):93–100, 1990.

[4] P. Merz and B. Freisleben. Fitness landscapes, memetic algorithms and greedy operators for graph bi-partitioning. *Evolutionary Computation*, 8(1):61–91, 2000.

[5] T. Stützle. Iterated local search for the quadratic assignment problem. *European Journal of Operations Research*, 174(1):1519–1539, 2006.

[6] T. Stützle and H. H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[7] É. D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17(4–5):443–455, 1991.

Sampling Strategies and Local Search for Stochastic Combinatorial Optimization

Prasanna Balaprakash, Mauro Birattari,
Thomas Stützle and Marco Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

Abstract

In recent years, much attention has been devoted to the development of metaheuristics and local search algorithms for tackling stochastic combinatorial optimization problems. In this paper, we propose an effective local search algorithm that makes use of *empirical estimation* techniques for a class of stochastic combinatorial optimization problems. We illustrate our approach and assess its performance on the PROBABILISTIC TRAVELING SALESMAN PROBLEM. Experimental results show that our approach is very competitive.

1 Introduction

The PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) [4] is a paradigmatic example of a stochastic combinatorial optimization problem. It is similar to the TSP with the difference that each node has a probability of requiring a visit. The *a priori* optimization [1] approach for the PTSP consists in finding an *a priori* solution that visits all the nodes such that the expected cost of *a posteriori* solutions is minimized: The *a priori* solution must be found prior to knowing which nodes are to be visited; the associated *a posteriori* solution is computed *after* knowing which nodes need to be visited and it is obtained by skipping the nodes that do not require to be visited and visiting others in the order in which they appear in the *a priori* solution. This paper focuses on an iterative improvement algorithm, that is, an algorithm that starts from some initial solution and then iteratively moves to an improving neighboring one until a local optimum is found. Essential for designing and implementing an effective iterative improvement algorithm is that the cost differences among neighboring solutions are computed efficiently. Currently, the state-of-the-art iterative improvement algorithms for the PTSP, namely, `2-p-opt` and `1-shift` use for this task closed-form expressions based on heavy mathematical derivations [2]. Recently, we introduced a new algorithm called `2.5-opt-ACs` that also uses closed-form expressions and moreover adopts the classical TSP speedup techniques [3].

We showed that this algorithm is more effective than `2-p-opt` and `1-shift` with respect to both solution quality and computation time [3]. In this paper, we propose an effective iterative improvement algorithm that makes use of *empirical estimation* and variance reduction techniques.

2 Estimation-based iterative improvement algorithm for the PTSP

The PTSP is a stochastic combinatorial optimization problem that can be described as: Minimize $F(x) = E\big[f(x,\Omega)\big]$, subject to $x \in S$, where x is an *a priori* solution, S is the set of feasible solutions, the operator E denotes the mathematical expectation, and $f(x,\Omega)$ is the cost of the *a posteriori* solution that depends on a random variable Ω, which is an n-variate Bernoulli distribution; a realization ω of Ω prescribes which nodes need being visited. An unbiased estimator of $F(x)$ of a PTSP solution x can be computed on the basis of a sample of costs of *a posteriori* solutions obtained from M independent realizations of the random variable Ω.

In iterative improvement algorithms for the PTSP, we need to compare two neighboring solutions x and x' to select the one of lower cost. For x', an *unbiased* estimator of $F(x')$ can be estimated analogously to $F(x)$ by using a different set of M' independent realizations of Ω. However, in order to increase the accuracy of this estimator, the well-known variance-reduction technique *"common random numbers"* can be adopted. In the context of PTSP, this technique consists in using the same set of realizations of Ω for estimating the costs $F(x')$ and $F(x)$. Consequently, we have $M' = M$ and the estimator $\hat{F}_M(x') - \hat{F}_M(x)$ of the cost difference is given by: $\hat{F}_M(x') - \hat{F}_M(x) = \frac{1}{M}\sum_{r=1}^{M}\Big(f(x',\omega_r) - f(x,\omega_r)\Big)$. We implemented iterative improvement algorithms that use this way of estimating cost differences exploiting a neighborhood structure that uses the *node-insertion neighborhood* on top of the *2-exchange neighborhood* structure, that is, the well-known *2.5-exchange neighborhood*. To make the computation of the cost differences as efficient as possible, given two neighboring *a priori* solutions and a realization ω, the algorithm needs to identify the edges that are not common to the two *a posteriori* solutions. This is realized as follows: for every edge $\langle i,j\rangle$ that is deleted from x, one needs to find the corresponding edge $\langle i^*,j^*\rangle$ that is deleted in the *a posteriori* solution of x. We call this edge the *a posteriori edge* and it is obtained as follows: If node i requires visit, then $i^* = i$, otherwise, i^* is the first predecessor of i in x such that $\omega[i^*] = 1$, that is, the first predecessor for which the realization is one, indicating it requires visit. If node j requires visit, then $j^* = j$, otherwise, j^* is the first successor of j such that $\omega[j^*] = 1$. Recall that in a *2-exchange* move, the edges $\langle a,b\rangle$ and $\langle c,d\rangle$ are deleted from x and replaced by $\langle a,c\rangle$ and $\langle b,d\rangle$. For a given realization ω and the corresponding *a posteriori edges*,

$\langle a^*, b^* \rangle$, $\langle c^*, d^* \rangle$, the cost difference between the two *a posteriori* solutions is given by $c_{a^*,c^*} + c_{b^*,d^*} - c_{a^*,b^*} - c_{c^*,d^*}$, where $c_{i,j}$ is the cost of edge $\langle i, j \rangle$. The procedure described can be directly extended to *node-insertion* moves. Furthermore, the proposed algorithm adopts neighborhood reduction techniques such as *fixed-radius search*, *candidate lists* and *don't look bits*. This algorithm is called `2.5-opt-EEs`. For further reference, see [3].

Intuitively, the variance of the cost difference estimator depends on the probability associated with each node. The smaller the probability values, the higher the variance. In this case, the usage of a large number of realizations reduces the variance of the estimator. Nevertheless, using a large number of realizations for high probability values is simply a waste of time. In order to address this issue, we adopt an adaptive sampling procedure that saves computational time by selecting the most appropriate number of realizations with respect to the variance of the cost difference estimator. This procedure is realized using *Student's t-test* in the following way: Given two neighboring *a priori* solutions, the cost difference between their corresponding *a posteriori* solutions is sequentially computed on a number of realizations. As soon as the *t-test* rejects the null hypothesis that the cost difference estimator is equal to zero, the computation is stopped. If no statistical evidence is gathered, then the computation is continued until a maximum number of realizations—a parameter—has been considered. The sign of the estimator is determines the solution of lower cost.

In order to reduce the high variance of the cost difference estimator for low probability values, we use the variance reduction technique *"importance sampling"*. Given two neighboring *a priori* solutions, this technique, instead of using realizations from the given distribution Ω, considers realizations from another distribution Ω^*—the so-called biased distribution—that forces the nodes involved in the cost difference computation to occur more frequently. The resulting cost difference between two *a posteriori* solutions for each realization is corrected for the adoption of the biased distribution and the correction is given by the likelihood ratio of the original distribution with respect to the biased distribution. We denote the proposed algorithm `2.5-opt-EEais`.

Here we report some example results obtained on *clustered homogeneous* PTSP instances of 1000 nodes, which are arranged in a $10^6 \times 10^6$ square and where each node has a same probability p of appearing in a realization. We considered a probability range in $[0.050, 0.200]$ with a step size of 0.025; 100 instances were generated for each probability level. For the hardware setting and implementation specific details, we refer the reader to [3]. Each iterative improvement algorithm is run until it reaches a local optimum. In order to compare the cost of the *a priori* solutions reached by each algorithms, we used the closed-form expression that computes the exact cost [4]. The results, measured across the 100 instances, are shown in Table 1.

Regarding the time required to reach local optima, irrespective of the

Table 1: Mean and standard deviation (s.d.) of final solution cost and computation time in seconds.

	Algorithm	Solution Cost		Computation Time	
		mean	s.d.	mean	s.d.
$p = 0.050$	`2.5-opt-EEais`	3995478	366491	13.47	2.29
	`2.5-opt-EEs`	4012670	377854	41.95	6.41
	`2.5-opt-ACs`	3993213	372801	780.85	115.84
$p = 0.075$	`2.5-opt-EEais`	4576135	403363	6.90	0.98
	`2.5-opt-EEs`	4579572	381368	22.39	3.35
	`2.5-opt-ACs`	4579831	399972	581.56	77.68
$p = 0.100$	`2.5-opt-EEais`	5073047	414194	4.52	0.53
	`2.5-opt-EEs`	5078611	400207	14.57	1.94
	`2.5-opt-ACs`	5088197	400986	454.79	64.91
$p = 0.125$	`2.5-opt-EEais`	5524534	424238	3.39	0.40
	`2.5-opt-EEs`	5537658	427805	10.81	1.33
	`2.5-opt-ACs`	5555043	411029	367.22	45.81
$p = 0.150$	`2.5-opt-EEais`	5952696	432452	2.71	0.25
	`2.5-opt-EEs`	5963539	439965	8.51	1.00
	`2.5-opt-ACs`	5978640	431100	309.45	41.62
$p = 0.175$	`2.5-opt-EEais`	6349469	444421	2.23	0.21
	`2.5-opt-EEs`	6357512	443292	7.09	0.81
	`2.5-opt-ACs`	6380038	446660	258.70	36.76
$p = 0.200$	`2.5-opt-EEais`	6707241	476088	1.92	0.18
	`2.5-opt-EEs`	6715865	470162	6.01	0.64
	`2.5-opt-ACs`	6690302	454250	226.89	27.66

value of p, `2.5-opt-EEais` is approximately 1.5 to 2 orders of magnitude faster than `2.5-opt-ACs` and it is faster than `2.5-opt-EEs` by a factor of 3. The average cost of local optima obtained by `2.5-opt-EEais` is comparable to one of `2.5-opt-EEs` and `2.5-opt-ACs`: the paired Wilcoxon test (α=0.05) does not reject the null hypothesis that the algorithms produce equivalent results.

3 Conclusion and Future Work

The main novelty of our approach consists of using the *empirical estimation* techniques and variance reduction techniques in the *delta evaluation* procedure. The proposed approach is conceptually simple, easy to implement, scalable to large instance sizes and can be applied to problems in which the cost difference cannot be expressed in a closed-form. We will devote our further research to assess the behavior of the proposed approach when used as an embedded heuristic in metaheuristics such as iterated local search, ant colony optimization and genetic algorithms. From the application perspective, the *estimation-based* iterative improvement algorithms will be applied

to more complex problems such as stochastic vehicle routing, stochastic scheduling, and TSP with time windows and stochastic service time.

References

[1] D. Bertsimas, P. Jaillet, and A. Odoni. A priori optimization. *Operations Research*, 38(6):1019–1033, 1990.

[2] L. Bianchi. *Ant Colony Optimization and Local Search for the Probabilistic Traveling Salesman Problem: A Case Study in Stochastic Combinatorial Optimization.* PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2006.

[3] M. Birattari, P. Balaprakash, T. Stützle, and M. Dorigo. Estimation-based local search for stochastic combinatorial optimization. Technical Report TR/IRIDIA/2007-003, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2007. Submitted for journal publication.

[4] P. Jaillet. *Probabilistic Traveling Salesman Problems.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Two Phase Stochastic Local Search Algorithms for the Biobjective Traveling Salesman Problem

Thibaut Lust and Jacques Teghem
Laboratory of Mathematics & Operational Research
Polytechnic Faculty of Mons, Belgium

Abstract

In this work, we present two phase stochastic local search algorithms with the aim of finding a good approximation of the efficient solution set of the biobjective traveling salesman problem. In the first phase of the algorithms, a search for a good approximation of the supported efficient solution set is undertaken. After this first phase, the second phase is launched to generate non-supported efficient solutions. Three methods are presented and experimented for the second phase: Pareto local search, a memetic algorithm with a data perturbation technique and a path-relinking operator.

1 The mTSP

Given a set $\{v_1, v_2, \cdots, v_N\}$ of cities and K costs $c_k(v_i, v_j)$ (with $k = 1, \ldots, K$) between each pair of distinct cities $\{v_i, v_j\}$ (with $i \neq j$), the multiobjective traveling salesman problem (mTSP) consists of finding a solution, i.e. an order π of the cities, so as to minimize the following costs ($k = 1, \ldots, K$):

$$\text{“min”} z_k(\pi) = \sum_{i=1}^{N-1} c_k(v_{\pi(i)}, v_{\pi(i+1)}) + c_k(v_{\pi(N)}, v_{\pi(1)})$$

These K quantities z_k correspond to the values taken by the various objectives, for a tour realized by a traveling salesman who visits each city exactly once and then returns to the starting city. We are interested here only in the symmetric biobjective traveling salesman problem (bTSP), i.e. $c_k(v_i, v_j) = c_k(v_j, v_i)$ for $1 \leq i, j \leq N$ and $K = 2$.

Due to the contradictory features of the objectives, it does not exist a solution simultaneously minimizing each objective (and for this reason the notation “min” is used), but a set of solutions called *efficient solutions*. A solution π^* is efficient for the mTSP if there is no other solution π such that: $z_k(\pi) \leq z_k(\pi^*), k = 1, \ldots, K$ with at least one strict inequality.

In this paper, only a minimal complete set will be sought, i.e. no equivalent efficient solution (two solutions π_1 and π_2 are equivalent if $z_k(\pi_1) = z_k(\pi_2)$, $k = 1, \ldots, K$) will be retained, and each solution found will correspond to a distinct non-dominated point in the objective space. We call this minimal complete set *Pareto set*.

2 Solution methods

Given the difficulty of the bTSP, we only try to find a good approximation of the Pareto set. Three different stochastic local search algorithms are experimented that are all based on the same two phases [8]:

1. Phase 1: Find a good approximation of the supported efficient solution set (solutions whose objective vectors lies in the border of the convex-hull of the Pareto set). These solutions can be obtain by resolution of single-objective problems obtained by applying a linear aggregation of the objectives: $\sum_{i=1}^{K} \lambda_k z_k(\pi)$ where λ is a weight set, i.e. a vector of dimension K, with $0 \leq \lambda_k \leq 1$ for $k = 1, \ldots, K$ and $\sum_{k=1}^{K} \lambda_k = 1$.

2. Phase 2: Find the non-supported efficient solutions (solutions not lying in the border of the convex-hull) located between the supported efficient solutions.

2.1 Approximation of the supported efficient solution set

We employ the method of Aneja and Nair [1], initially proposed for the resolution of a bicriteria transportation problem, that consists in generating all the weight sets which make it possible to obtain a minimal complete set of extremal supported efficient solutions (solutions whose objective vectors are located on the vertex set of the convex-hull) of a biobjective problem (non-extremal supported efficient solutions and equivalent solutions can however be generated). For each weight set generated, a linear aggregation of the objectives is carried out and the single-objective problem obtained is solved by an exact method. In this work, we do not use an exact method to solve the single-objective problem but the Lin-Kernighan (`LK`) heuristic implemented by Helsgaun [3]. This heuristic gives for the instances of 100 cities studied in this work very good solutions, close to the optimal solutions.

So, we have adapted the method of Aneja and Nair to take into account the fact that the `LK` heuristic is not exact, what implies that the solutions obtained are not necessarily efficient, nor supported efficient but that makes it possible to obtain a set of solutions very close to the minimal complete set of extremal supported efficient solutions, with a minimum number of resolution of single-objective problems resulting from linear aggregation.

2.2 Search for non-supported efficient solutions

Once a good approximation of the supported efficient solution set has been found, three methods are experimented with the aim of finding potentially non-supported efficient solutions. These three methods all use an archive containing the potentially efficient solutions found, which is updated as soon as a new potentially efficient solution is discovered, by adding the new solution in the archive and by removing the solutions of the archive which could be found dominated following the addition of the new solution. After the phase 1, the archive contains, for all the methods, an approximation of the supported efficient solution set.

2.2.1 Pareto local search

This method has been developed by Paquete et al. [6] and is based on the notion of *Pareto local optimum set* which is a generalization, in the multi-objective case, of the concept of local optimum. In this method, the neighborhood of each solution of the archive is explored, and each non-dominated neighbor is added to the archive. The algorithm stops when it is any more possible to find new non-dominated neighbors starting from a solution of the archive, that is to say, a Pareto local optimum set is found, which respect to the neighborhood used. In this work, we use the well-known 2-exchange neighborhood, also used by Paquete et al. However, they start their method from a randomly generated solution, whereas we use all the solutions of the archive generated in phase 1 as initial solutions. We call this method `PLS2`.

2.2.2 Memetic algorithm

We use `MEMOX` [5], scheme of resolution of multiobjective problems, based on a memetic algorithm. After the phase 1, a local search is applied from an offspring solution, generated by a crossover between a solution of the archive of minimal density and an another solution of the archive close, in the objective space, of the first solution. A dynamic hypergrid is used to compute the density of a potentially efficient solution. We use the `LK` heuristic as local search, by employing a linear aggregation of the objectives with a weight set fixed according to the first parent, i.e. the potentially efficient solution of minimal density. But, as the `LK` heuristic is very robust (very little influenced by the starting solution), few new solutions will be found by the local search based on a linear aggregation, since a search for the supported efficient solutions has already been applied during the phase 1. We thus use the *Data Perturbation* (`DP`) technique, originally proposed by Codenotti et al. for the single-objective TSP [2]. Instead of modifying the starting solution (as carried out, for example, in the Iterated Local Search method), `DP` suggests to modify input data. In this way, by application of the `LK` heuristic starting from the offspring with perturbed data, new solutions,

essentially potentially non-supported efficient, could be found since the data used for the linear aggregation are perturbed.

2.2.3 Path-relinking

Before applying the path-relinking (PR), the solutions of the archive generated in phase 1 are sorted according to the increasing order of the value taken by the first objective. Then, a path in the decision space between two consecutive solutions of the archive, called starting and guiding solutions, is created, with the goal of providing new solutions that reduce the distance with respect to the guiding solution, on the basis of the starting solution. A distance between two solutions is measured by the number of uncommon arcs in both solutions. The 2-exchange movement is used to create the path, and only movements that reduce the distance are considered. Among such movements, the one that generates the nearest solution in the objective space to the line which connects the starting and the guiding solution is selected. Every new non-dominated solution found during the path building process is added to the archive.

3 Results

First experimentations show that compared to the state-of-the-art algorithms (MOGLS [4], PLS [6], PD-TPLS [7]) the results obtained by these three methods on instances of 100 cities of the bTSP are of better qualities. Using PLS2 is very efficient and allows to obtain very good approximations in a reasonable time. The use of an initial archive of good quality (generated in phase 1) is clearly better than using as first archive only one randomly generated solution as done by Paquete et al. in [6]. Applying PR as phase 2 gives good results in little time, but of lower quality than using PLS2. Moreover, if we try to improve the results of PR by applying a Pareto local search on the archive obtained, the results are not better than PLS2. The disadvantages of PR and PLS2 are that their performance is limited, and more computational time will not give significant better results, being given that these two methods are limited by the quality of the 2-exchange neighborhood. On the other hand, although the MEMOX scheme with the LK heuristic as local search with perturbed data converges more slowly than the other methods, this method allows to obtain better results if the resolution time is increased, since thanks to the data perturbations, new solutions are constantly found what increases the quality of the solution set obtained.

4 Conclusion

Work still needs to be done to take advantage of each of the three methods used for the search of non-supported solutions, essentially by allowing perturbations in the `PLS` method to avoid being stuck in a Pareto local optimal set. The perturbations can be, for example, realized by the data perturbation technique, the path-relinking operator or by allowing to have dominated solutions in the archive.

References

[1] Y. P. Aneja and K. P. K. Nair. Bicriteria transportation problem. *Management Science*, 25:73–78, 1979.

[2] B. Codenotti, G. Manzini, L. Margara, and G. Resta. Perturbation: An efficient technique for the solution of very large instance of the euclidean tsp. *INFORMS Journal on Computing*, 8:125–133, 1996.

[3] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.

[4] A. Jaszkiewicz. Genetic Local Search for Multiple Objective Combinatorial Optimization. *European Journal of Operational Research*, 137(1):50–71, 2002.

[5] T. Lust and J. Teghem. MEMOX: A Memetic Algorithm Scheme for Multiobjective Optimization. In *Proceedings of the 7th International Conference devoted to Multi-Objective Programming and Goal Programming*, Tours, June 2006.

[6] L. Paquete, M. Chiarandini, and T. Stützle. Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study. *Metaheuristics for Multiobjective Optimisation*, pages 177–199, Berlin, 2004. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535.

[7] L. Paquete and T. Stützle. A Two-Phase Local Search for the Biobjective Traveling Salesman Problem. *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 479–493, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science Vol. 2632.

[8] E.L. Ulungu and J. Teghem. The two phases method: An efficient procedure to solve biobjective combinatorial optimization problems. *Foundation of Computing and Decision Science*, 20:149–156, 1995.

Parametric Models of Search Progression

Johan Oppen[1] and David L. Woodruff[2]
[1]Molde University College, Molde, Norway
[2]Graduate School of Management, University of California, Davis, USA

Abstract

This paper describes ongoing research and discusses parametric models for how stochastic local search algorithms progress over time. Preliminary computational results are reported.

1 Introduction

Algorithms that search for good solutions to optimization problems present a trace of best-found objective values over time. The progression of objective function values for best solutions found is reasonably modeled as stochastic because the algorithms are often stochastic, and even when they are not, the progression of the search varies with each instance in unpredictable ways. An example of the progression is illustrated in Figure 1, which shows the results of tabu search applied to an optimization problem.

This is in contrast to non-parametric models that estimate the probability that the search will achieve a value better than some given threshold in a given amount of time [1]. Both types of models have value, but parametric models offer the promise of greater predictive power, so in this research we explore them.

2 Models

The canonical optimization problem is to find

$$\min_x f(x) : x \in \mathcal{Q}$$

where x is a vector and $x \in \mathcal{Q}$ summarizes constraints whose form depends on the problem at hand. There are many algorithms for such problems and essentially all of them "visit" or evaluate the objective function value $f(\cdot)$ for a sequence of vectors $x^{(k)},\ k = 0, 1, \ldots, K$. We are interested in the subsequence of length N of *best so far* vectors, i.e., those for which $f(x^{(k')}) < f(x^{(k)}), k' = 1, \ldots, k-1$.

We are interested (not exclusively) in local search algorithms that are based on the notion of a neighborhood that is implied by *moves* that transform one solution vector into another. We refer to the set of neighbors of x as $\mathcal{N}(x)$. For example, suppose that $\mathcal{Q}$ includes a requirement that all elements in x must be binary; a possible move mechanism is to flip one of the elements in x; i.e. x_i takes the value $1 - x_i$. Local search algorithms proceed in general iterations from $x^{(k)}$ to a member of $\mathcal{N}(x^{(k)})$ where the selection mechanism is a defining aspect of the algorithm.

In order to produce models of search behavior that have some comparative power across computers, implementations, algorithms, instances and perhaps problems, methods of scaling computational effort and objective function values are needed. For computational effort, we advocate using the average CPU time required to explore one solution's neighborhood. In Section 3 we report on results obtained in a variety of settings using the smallest realistic neighborhood. A common mechanism for scaling the objective function is to use percent deviation from the best known solution for the instance [1]. Although it will not work in all settings, we find it works well for many problems. The scaled time sequence is t_i and the scaled objective value sequence is z_i, $i = 1, \ldots, N$. We refer to the resulting (t_i, z_i) pairs as *data points* in two dimensional space.

A key aspect of our models is the division of the time trace into two parts: the initial rapid improvement and the discovery phase. In some instances one of these parts might be missing; however, we have found that both are present for many algorithms, problems and instances. We divide the N data points into two contiguous clusters: cluster one is $(t_i, z_i), i = 1, \ldots, N_1$ and cluster two is $(t_i, z_i),\ i = N_1 + 1, \ldots, N$. As we will see in Section 3, a very effective clustering can be obtained by finding the division that results in the minimum *mean squared error* for a two-segment, piecewise linear regression.

For a particular division of the points into two sets, we compute a least squares regression line for each set:

$$z = \alpha_1 + \beta_1 t \ \text{ and } \ z = \alpha_2 + \beta_2 t,$$

which results in a mean squared error for the first group of

$$\frac{1}{N_1} \sum_{i=1}^{N1} (\alpha_1 + \beta_1 t_i - z_i)^2$$

with a similar expression for the second group. The division of points into two groups that results in the lowest total mean squared error is adopted. All values of N_1 can typically be tested very quickly so this particular optimization problem can reasonably be solved by enumeration.

Once we have the two sets of points, we are in a position to fit a model for the two parts of the time trace. Although in our research we are looking at other models, it turns out that the least squares regression lines are quite effective and we explore that here.

3 Computational Results

To test our model, we have run a Tabu Search (TS) and a Simulated Annealing (SA) algorithm on three different instances of the Livestock Collection Problem (LCP). Both the variant of TS used and the LCP, which is a rich Vehicle Routing Problem, are described in [3]. Our SA implementation is based on [2].

We will refer to the sequence of $(t_i, z_i), i = 1, \ldots, N$ generated by the execution of a particular algorithm on a particular instance of the LCP as a *run*. When we have a model for the algorithm running on an instance, we test the validity of the model by using a measure known as the *Root Mean Squared Forecast Error (RMSFE)*,

$$RMSFE \equiv \sqrt{\frac{\sum_{i=1}^{N} (\hat{z}_i - z_i)^2}{N}}$$

Specifically, for each (problem instance, algorithm) pair, we generate ten runs. Each of the ten subsets of nine runs are used to fit a model as desribed in Section 2 and $RMSFE$ is computed to find how well the tenth run was predicted. For comparison, we also compute the $RMSFE$ value for the tenth run based on a non-parameterized benchmark procedure from [1]. This procedure uses the median solution quality from the nine runs as the predicted value $\hat{z}_i$. For each point $(t_i, z_i), i = 1, \ldots, N$ from the run we are trying to predict, we find the corresponding solution quality obtained by each of the other runs at that time. Most of the time, the runs we are looking at did not find a new best exactly at time t_i. If t_i is earlier than the first feasible solution was found or later than the last improvement, we use the first or last point from the run, respectively. If improvements were found both before and after time t_i, we interpolate. Table 1 summarizes the results from these comparisons. As expected, the non-parametric estimates are better when applied to data from the specific instance and algorithm whence they came. However, we are encouraged by the relative quality of the parametric estimates.

Table 1: $RMSFE$ statistics, measured in % deviation from non-parametric estimate

Instance	Algorithm	Avg	Min	Max
n110_v5_pig	TS	26.58	-35.85	114.56
	SA	20.03	0.61	57.73
n162_v8_rw	TS	26.80	-8.87	109.51
	SA	22.60	-10.64	99.01
n196_v8_rw	TS	16.89	-23.76	47.20
	SA	20.44	-0.36	42.11

Figure 1 gives an example of a line fit, showing both prediction lines for the model and the actual data points for the run that is being predicted. Table 1 shows that the errors generated by the parametric model are less than 30% greater than the errors from a non-parametric model, which causes us to be optimistic that with further research the parametric models can be useful for predictions across instances. Figure 1 is typical in that the errors are small for both types of models.

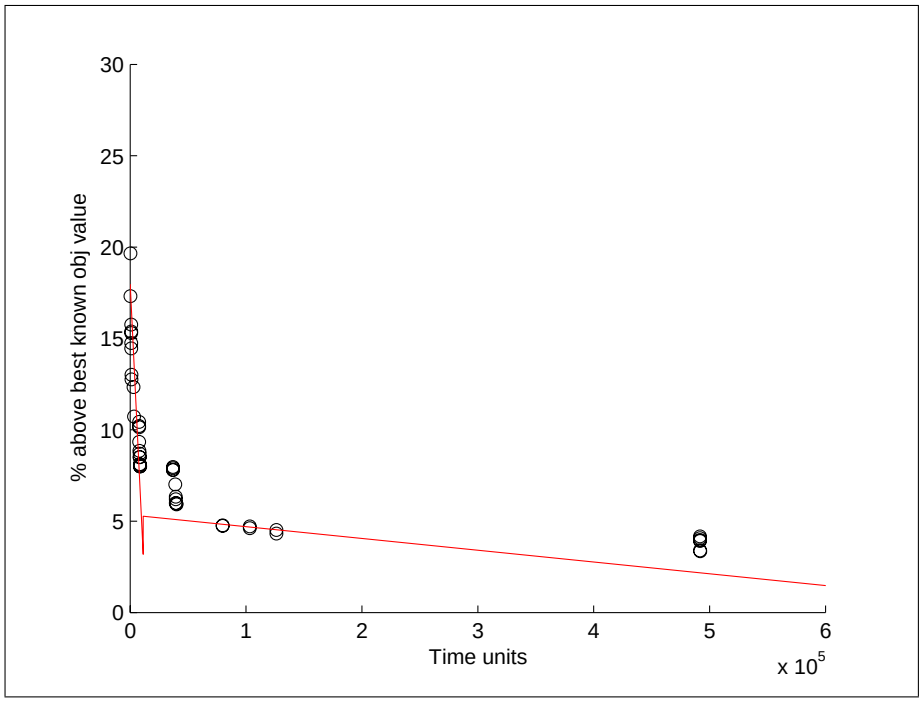


Figure 1: The data for the out-of-sample tabu search run of instance n196_v8_rw with prediction lines from the other 9 runs shown.

4 Conclusions and Directions for Future Research

This is ongoing research, and more work is needed to improve the model discussed here. We are finding ways to discover if one of the search phases described in section 2 is missing. Better models for the two phases should also be sought. For example, the initial improvement phase might be modeled by an exponential or Wiebull. To get a broader basis for our models, we have also looked at other classes of optimization problems and other local search based algorithms beyond those described here.

References

[1] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, USA, 2005.

[2] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations Research*, 37(6):865–892, November-December 1989.

[3] J. Oppen and A. Løkketangen. A tabu search approach for the livestock collection problem. *Computers and Operations Research*, 2007. Accepted for publication.

Communication Policies for a Parallel Multi-colony ACO Algorithm with Identical Colonies

Max Manfrin, Mauro Birattari, Thomas Stützle and Marco Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

Abstract

In this article, we present a study that analyzes the impact that communication policies have on the solution quality reached by a parallel multicolony ant colony optimization algorithm for the TRAVELING SALESMAN PROBLEM. We consider several factors of the parallel variants: the number of colonies, migration frequencies, communication strategies on different interconnection topologies, and the usage of local search (k-opt) or not. We adopt a full factorial design, empirically testing the parallel variants on a distributed-memory parallel architecture, and we analyze the results with a multi-factor analysis of variance. Conclusions are drawn about the performance of the different policies. It is shown that the parallel variant adopted has a large influence on the actual performance of the algorithm.

1 Introduction

Ant colony optimization (ACO) [1] is a metaheuristic for combinatorial optimization problems that is inspired by the pheromone trail laying and following behavior of some ants species. In ACO, artificial ants are a set of stochastic procedures that incrementally construct candidate solutions using artificial pheromone and possibly available heuristic information. The artificial pheromone is a parametrized probabilistic model that is modified at computation time, based on previously seen solutions [5].

In this paper, we present a study of various communication policies for a parallel homogeneous multi-colony $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [1], by means of a design of experiments approach. The idea of the multi-colony approach is to have several colonies running in parallel that exchange information to intensify the search for solutions toward promising regions of the search space. At the same time, the tendency of each colony to explore the search space differently must be preserved.

Unfortunately, previous studies on cooperation of multiple colonies often suffer from a lack of rigorous empirical analysis (i.e., empirical tests on very few instances of small size, lack of comparison with the *null policy*—parallel

independent runs (PIR)—or with a single colony algorithm that uses the same cumulative computational effort). Therefore, the question remained open on how to implement efficient parallel versions and what improvement in performance can be expected.

Here, we use factorial experiments based on a full factorial design, extensively testing the parallel variants on the TRAVELING SALESMAN PROBLEM (TSP) using message passing libraries and a distributed-memory parallel architecture; we analyze the impact of the factors studied on the solution quality with a multi-factor analysis of variance (ANOVA). The factors considered are: the number of colonies (2 levels), migration frequencies (2 levels), communication strategies on different interconnection topologies (4 levels), and usage or not of k-opt local searches (3 levels). All cited factors result in $2 \cdot 2 \cdot 4 \cdot 3 = 48$ different treatments.

2 Experimental settings

To evaluate the quality of a parallel multi-colony ACO algorithm, the *parallel independent runs* variant should be taken as a baseline for the comparison. In PIR, several runs of a sequential ACO algorithms are executed independently and in parallel. Some of our previous studies [2, 3] led us to conjecture that cooperation becomes less effective for increasing search length and more performing algorithms. In this study, we test this conjecture. Furthermore, we study the influence of the communication policy on the solution quality.

All the policies in this study share the following elements: each colony selects as the only *migrant* the best-so-far solution; a colony replaces its iteration-worst solution with a received solution, if and only if the latter has a lower cost than the colony's current best-so-far solution. For the *number of colonies* we consider two levels: 4 and 8 identical colonies of 25 ants each. We consider two types of *migration frequency*: fixed and increasing. In the fixed schema, colonies exchange solutions every 25 iterations after an initial period of independent exploration of 100 iterations. In the increasing schema, solutions are exchanged with increasing frequency but taking care that at least 25 iterations pass between two exchanges. The i^{th} exchange happens at iteration $\sum_{k=0}^{i-1} \lfloor 0.9^k \cdot 1000 \rfloor_{25}$, where $\lfloor x \rfloor_{25} = 25$ if $x < 25$, $\lfloor x \rfloor_{25} = \lfloor x \rfloor$ otherwise, where $\lfloor x \rfloor$ is the largest integer not exceeding x.

We examine the following communication strategies and interconnection topologies: *unidirectional ring* (R) over the *ring* topology, in which each colony has one predecessor and one successor; *hypercube* (HC) over the *hypercube* topology, in which each colony is located on a vertex of a hypercube and neighbor colonies are on adjacent vertices; *fully-connected* (FC) and *replace-worst* (RW) over the *fully-connected* topology, in which each colony is in the neighborhood of any other. In this way, we move from a localized communication, to a more global communication. We refer the reader to [2]

for a more detailed explanation of the adopted communication strategies. Each algorithm is tested with and without the k-opt local search component (both 2-opt and 3-opt are considered).

Experiments are performed on the same computational environment as in [2] on various instances from TSPLIB. For each instance, 30 runs of 10000 iterations each are performed in order to gain statistical evidence and provide conclusions that are meaningful. We refer the readers interested in the full data to [4].

3 Results

To identify the impact of problem size and algorithm features on the effectiveness of the communication policies, we analyze the full factorial experiments by means of the ANOVA technique. The response variable is the percentage error from the known optimal cost. We consider only first and second order interactions. To meet the assumptions of the ANOVA analysis, we need to separate the empirical results in three groups, according to the local search level. In each group, we divide the instances in three classes according to their sizes: large, medium, and small, and we consider the results after 1000, 3162, and 10000 iterations. We perform a separate ANOVA for each of the $3 \cdot 3 \cdot 3 = 27$ resulting groups. For full details of the various ANOVA we refer the readers to [4]. Globally, the two factors with the largest F-Ratio are the "instance size" and the "number of colonies". As expected, the larger the instances, the larger the average percentage error from the known optimal cost, and also the average results produced by 8 colonies are better than those produced by 4 colonies. The migration frequency (fixed vs. increasing schema) has the third largest F-Ratio. The use of the fixed schema produces better results, on average, for the less performing algorithms, while the use of the increasing schema produces better results, on average, for the more performing ones. In Figure 1, we show a high-level representation of which level of the "migration frequency" factor produces, on average, lower solution cost.

Once having identified the components for the communication policies that result in high performance, we need to assess the statistical significance of the differences in solution costs obtained by these policies with respect to the PIR approach. We perform a series of *Pairwise Wilcoxon rank sum tests* with *p-values* adjusted by Holm's method. We refer to [4] for the full set of data on the comparison and the boxplots of the normalized solution costs with respect to the percentage error from the known optimal cost.

The empirical results of the parallel variants under analysis support the initial conjecture. The beneficial effects of communication tend to reduce both when increasing the search length and when adopting better performing algorithms.

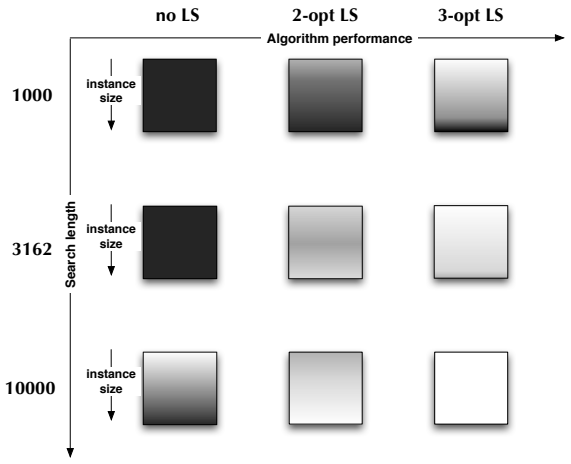


Figure 1: Overview of which level for the "migration frequency" factor in cooperation policies produces, on average, lower solution costs. Black stays for the fixed schema, white for the increasing schema. The level of grey is related to the advantage of the fixed schema over the increasing schema.

4 Conclusions

In this article, we have presented a study that analyzes the impact on solution quality of various communication policies for a parallel multicolony ACO algorithm for the TSP using message passing libraries. Previous studies on cooperation of multiple colonies did not fully answer the question on how to implement efficient parallel variants and what improvement in performance to expect from them. By means of a design of experiments approach we studied the impact of factors on the effectiveness of communication policies. Some of our previous studies [2, 3] led us to conjecture that communication becomes less effective for increasing search length and more performing algorithms. Empirical results of the algorithms under analysis support this conjecture. From the results, we observed that communication is more effective for increasing instance size, but the beneficial effects of communication tend to reduce both when increasing the search length and when adopting more performing algorithms. An extended empirical analysis is available in [4]. In future works we will analyze in more details the various trends that emerge from this study.

References

[1] M. Dorigo and T. Stützle. *Ant Colony Optimization.* The MIT Press, Cambridge, MA, USA, 2004.

[2] M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo. Parallel Ant Colony Optimization for the Traveling Salesman Problem. In *Ant Colony Optimization and Swarm Intelligence, ANTS 2006*, volume 4150 of *LNCS*, pages 224–234. Springer-Verlag, 2006.

[3] M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo. Parallel multicolony ACO algorithm with exchange of solutions. In *18th Belgium–Netherlands Conference on Artificial Intelligence, BNAIC 2006*, pages 409–410, 2006.

[4] M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo. Extendend Empirical Analysis: Communication Policies for a Parallel Multicolony ACO Algorithm with Identical Colonies. `http://iridia.ulb.ac.be/supp/IridiaSupp2007-006/`, 2007.

[5] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131:373–395, 2004.

On the Potential of Automatic Algorithm Configuration

Frank Hutter
University of British Columbia, Canada

Abstract

Design and implementation of efficient and robust algorithms are core topics of computer science and operations research, and the determination of appropriate values for free algorithm parameters is a challenging and tedious task in the design of effective algorithms for hard problems. Such parameters include categorical choices (e.g., neighborhood structure in local search or variable/value ordering heuristics in tree search), as well as numerical parameters (e.g., noise or restart timing). In practice, tuning of these parameters is largely carried out manually by applying rules of thumb and crude heuristics, while more principled approaches are only rarely used. In this paper, we study some tuning scenarios in more detail and demonstrate the large potential of even very simple automatic algorithm configuration approaches.

1 Introduction

The problem of setting an algorithm's free parameters for maximal performance on a class of problem instances is ubiquitous in the design and empirical analysis of algorithms. Examples of parameterised algorithms can be found in tree search [6] and local search [9]; commercial solvers, such as ILOG CPLEX[1], also offer a plethora of parameter settings. Considerable effort is often required to find a default parameter configuration that yields high and robust performance across all or at least most instances within a given set or distribution [3, 1].

The use of automated tools for finding performance-optimising parameter settings has the potential to liberate algorithm designers from the tedious task of manually searching the parameter space. Notice that the task of constructing an algorithm by combining various building blocks can be seen as a special case of algorithm configuration: Consider, for example, two tree search algorithms for SAT that only differ in their preprocessing and variable ordering heuristics – in fact, these can be seen as a single algorithm with two nominal parameters.

Algorithm configuration is commonly (either implicitly or explicitly) treated as an optimisation problem, where the objective function captures

[1] http://www.ilog.com/products/cplex/

performance on a fixed set of benchmark instances. Depending on the number and type of parameters, the methods used to solve this optimisation problem include exhaustive enumeration, beam search [15], experimental design [5, 2], genetic programming [16], the application of racing algorithms [4, 3], and combinations of fractional experimental design and local search [1].

Recently, we have introduced an iterated local search approach for algorithm configuration [12]. This approach has subsequently lead to enormous speed-ups of tree search algorithms for SAT for solving SAT-encoded software verification (speedups of a factor of 500) and bounded model-checking instances (speedups of a factor of 4.5) [10].

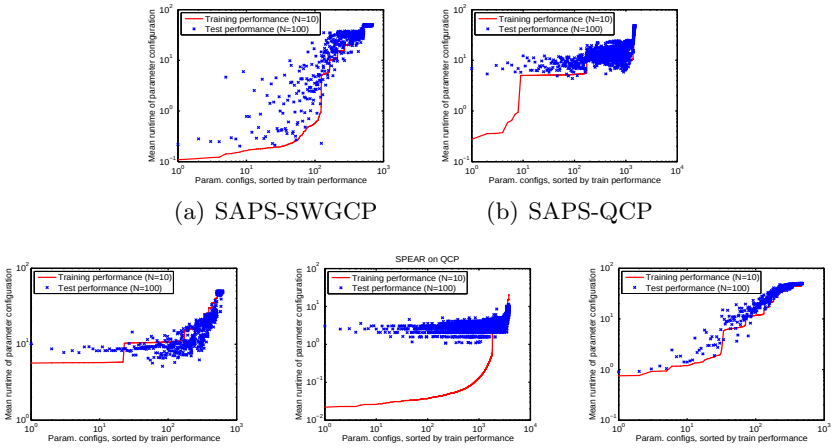
2 Tuning scenarios

In this paper, we study tuning scenarios including tree search and local search for propositional satisfiability (SAT) and mixed integer programming. In particular, we study the tree search algorithm SPEAR [10] with 26 mixed discrete/continuous parameters, the local search algorithm SAPS [13] with four continuous parameters, and the commercial software CPLEX for mixed integer programming (MIP)[2] with 80 mixed discrete/continuous parameters. All continous parameters are discretized to seemingly meaningful values spread around the algorithm defaults. As SAT domains, we employ a set of SAT-encoded quasi-group completion (QCP) problems [8] and a set of SAT-encoded graph colouring problems based on small-world graphs (SWGCP) [7]. For MIP, we employ a set of combinatorial auction (CATS) instances from the combinatorial auctions test set [14].

In order to get an idea about the potential of improvement for these tuning scenarios, and about the potential for overtuning to be expected, we sampled a number of parameter configurations uniformly at random, evaluating them on a small training set with $N = 10$ instances, and iterated this process up to a total CPU time usage of five hours. In Figure 1, we plot both performance on the small training set and on an independent test set with $M = 100$ instances; configurations are ordered with respect to training quality. Note that the cutoff time for each domain is five seconds, unsuccesful runs are counted as taking ten times this time, and we plot average performance over the training/test instances; thus, a performance of 50 is the absolute worst a parameter configuration can achieve, and zero is the best. We note that the differences between parameter configrations vary between tuning scenarios, as does the correlation between training and test performance.

Figure 2 takes a closer look at test performance of a number of selected parameter configurations, namely the default parameter configuration of the algorithms, and five of the randomly sampled configurations: the best and worst in terms of training performance, as well as the 25%, 50%, and 75%

[2] http://www.ilog.com/products/cplex/



(a) SAPS-SWGCP (b) SAPS-QCP

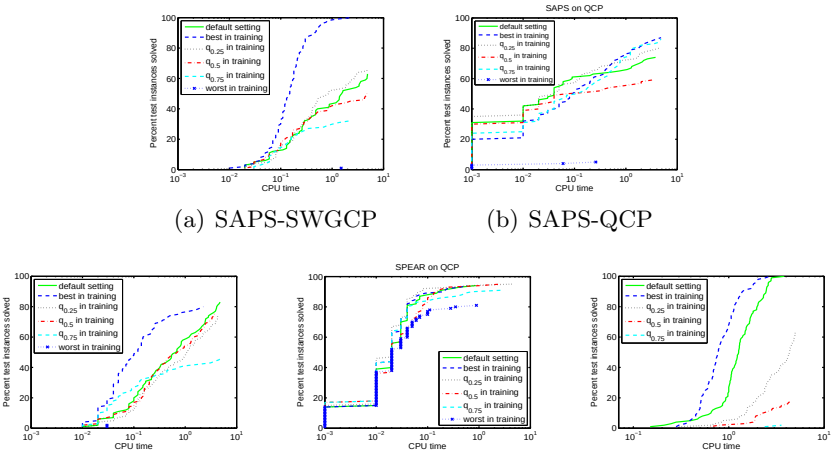
(c) SPEAR-SWGCP (d) SPEAR-QCP (e) CPLEX-CATS

Figure 1: Performance of randomly sampled parameter configurations on a small training set of $N = 10$ instances and a test set of 100 instances. For details see text.

quantiles. For each of these parameter configurations, we plot the cumulative distribution of the probability of solving the instances in the test set. We note that in each single scenario, the best parameter configuration based on the small training set of ten instances already performs better than the default parameter setting. Of course, we do not anticipate this to generalize to arbitrary tuning scenarios, but it at least speaks for the potential of automatic tuning.

3 Experiments with iterated local search

In this section, we study the effectiveness of our iterated local search from [12] in the above tuning scenarios. We compare test performances of pure random sampling based on a training set of 100 instances, BasicILS based on the same training set, and FocusedILS (which uses a different number of instances to evaluate each parameter configuration). Compared to [12], we implemented one important improvement for all approaches, namely a pruning technique that stops evaluations of parameter configurations when they are already proovably worse than a previously seen parameter configurations. This technique improves random sampling the most, followed by BasicILS, and only improves FocusedILS marginally. Without this pruning technique BasicILS and FocusedILS outperformed random sampling in our experiments for [12] and FocusedILS outperformed BasicILS. In Table 1 we see that the picture becomes less clear when pruning is used: while FocusedILS performs statistically significantly better than the other approaches on two domains, the other two approaches reach similar or better (albeit not statistically significantly better) performance in the remaining three scenarios.



(a) SAPS-SWGCP (b) SAPS-QCP

(c) SPEAR-SWGCP (d) SPEAR-QCP (e) CPLEX-regions100

Figure 2: Test performance for a number of selected parameter configurations; for details, see text.

Based on our above analysis of correlations between training and test set, we expected overtuning effects to be strongest for the tuning scenarios involving the QCP domain. This expectation is confirmed by the results in Table 1. Finally, note that the automatically found parameter configurations always clearly outperform the default configuration, much more so than the random parameter configurations, sampled without pruning.

4 Conclusions

Automatic algorithm configuration can greatly improve performance for all tuning scenarios we studied here, and even a simple random sampling of parameter configurations shows very good performance when combined with a simple pruning technique that stops algorithm evaluations once they are proovably worse (on the training set) than the incumbent solution.

In future work, we plan to study model-based approaches in order to speed up the search for good parameter configurations. We also plan to integrate computationally cheap instance features into this model and use it to perform per-instance algorithm configuration, where we automatically choose an appropriate parameter configuration for each given instance [11].

References

[1] B. Adenso-Diaz and M. Laguna. Fine-tuning of algorithms using fractional experimental design and local search. *Operations Research*, 54(1):99–114, Jan–Feb 2006.

[2] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation*. Springer Verlag, 2006.

[3] M. Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[4] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proc. of GECCO-02*, pages 11–18, 2002.

Scenario	Default	Random(100)	BasicILS(100)	FocusedILS
SAPS-SWGCP	45.41	0.21 ± 0.03	0.21 ± 0.03	0.35 ± 0.05
		0.32 ± 0.05	0.32 ± 0.06	0.32 ± 0.05
SPEAR-SWGCP	9.74	6.71 ± 1.2	6.65 ± 1.48	8.26 ± 0.73
		7.97 ± 1.14	8.05 ± 0.9	8.3 ± 1.06
SAPS-QCP	15.80	3.4 ± 1.53	$\mathbf{2.78 \pm 1.28}$	3.95 ± 0.27
		5.92 ± 0.44	5.5 ± 0.53	$\mathbf{5.21 \pm 0.39}$
SPEAR-QCP	2.65	0.45 ± 0.51	0.36 ± 0.41	1.08 ± 0.18
		1.2 ± 0.18	1.39 ± 0.33	1.29 ± 0.2
CPLEX-regions100	1.61	0.7 ± 0.12	$\mathbf{0.39 \pm 0.12}$	0.35 ± 0.04
		0.71 ± 0.12	0.4 ± 0.11	$\mathbf{0.35 \pm 0.04}$

Table 1: Performance for our tuning scenarios; the top row for each scenario gives training performance, the bottom row test performance, mean±stddev over 25 executions of the tuning approach. The training performance of Random(100) and BasicILS(100) is directly comparable since they use the same training set: bold face indicates statistically better performance for BasicILS, but due to limited representativeness of the training set this does not transfer to statistically better performance on the test set. For FocusedILS, bold face indicates statistically better performance than the other approaches on the test set; the cases where Random performs best are not statistically significant.

[5] S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, 2001.

[6] N. Eén and N. Sörensson. An extensible SAT solver. In *Proc. of SAT-03*, pages 502–518, 2003.

[7] I. P. Gent, H. H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. In *Proc. of AAAI-99*, pages 654–660, Orlando, Florida, 1999.

[8] C. P. Gomes and B. Selman. Problem structure in the presence of perturbations. In *Proc. of AAAI-97*, 1997.

[9] H. H. Hoos and T. Stützle. *Stochastic Local Search – Foundations & Applications*. Morgan Kaufmann, 2005.

[10] F. Hutter, D. Babić, H. H. Hoos, and A. J.Hu. Boosting verification by automatic tuning of decision procedures. In *Formal Methods in Computer Aided Design (FMCAD'07)*, 2007. To appear.

[11] F. Hutter, Y. Hamadi, H. H. Hoos, and K. Leyton-Brown. Performance prediction and automated tuning of randomized and parametric algorithms. In *Proc. of CP-06*, pages 213–228, 2006.

[12] F. Hutter, H. H. Hoos, and T. Stützle. Automated algorithm configuration based on local search. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, 2007. To appear.

[13] F. Hutter, D. A. D. Tompkins, and H. H. Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In *Proc. of CP-02*, pages 233–248, 2002.

[14] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *ACM Conference on Electronic Commerce (EC-00)*, 2000.

[15] S. Minton. Automatically configuring constraint satisfaction programs: A case study. *Constraints*, 1(1):1–40, 1996.

[16] M. Oltean. Evolving evolutionary algorithms using linear genetic programming. *Evolutionary Computation*, 13(3):387–410, 2005.

Development of Algorithms for Knowledge Discovery. Swarm Intelligence and Rough Set Theory as Tools.

Yudel Gómez[1], Rafael Bello[1] and Ann Nowe[2]
[1]CS Department, Universidad Central de Las Villas, Cuba
[2]Como Lab, Vrije Universiteit Brussel, Belgium

Abstract

This research deals with the development of algorithms for Knowledge Discovery, specifically feature selection and rules induction. Two models to feature selection based on Ant Colony Optimization and Rough Set Theory and Particle Swarm Optimization and Rough Set Theory are presented. Variants of these algorithms in two stages are developed. I use Learning Rules using Rough Set (LRRS) algorithm from Rough Set Theory to rule induction. A new model to learning rules taking into account sharing meta information from several dataset should be developed.

1 Introduction

New papers continuously appear related to Feature Selection Problem (FSP) and Rule Induction [15] [13]. Classification rule induction [12] is an area of machine learning where formal rules are extracted from observations. The extracted rules may represent a full scientific model of the data.

Swarm intelligence [5] can be defined as the collective intelligence that emerges from a group of simple entities; these agents enter into interactions, sense and change their environment locally. There are two popular swarm inspired methods in computational areas: Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). ACO is inspired by the behavior of ants and has many applications in discrete optimization problems. The approach relies on a metaheuristic which is used to guide other heuristics in order to obtain better solutions than those that are generated by local optimization methods. This computational model was introduced by Marco Dorigo; for an overview see [6]. PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy [9, 7], inspired by social behavior of bird flocking. This is a metaheuristic for the optimization of continuous functions; but has also been extended to a discrete particle swarm optimization algorithm [10].

Rough Set Theory (RST) offers the heuristic function to measure the quality of one feature subset. RST was proposed by Z. Pawlak [14]. An important issue in the RST is about feature reduction based on the reduct concept. A reduct is a minimal set of attributes that preserves the partitioning of universe and hence the ability to perform classifications [11].

2 Problem

The problem faced is to solve a typical classification problem. For a new object is necessary to know which class it belongs to. Usually there is a dataset with objects that are described by many attributes, and its class. Those attributes can be nominal, integer or real (continuous). So, the problem is to determine the class (or classes) for a new object with same attributes.

There are many algorithms that have been developed in order to solve this problem [16] [13]. Even though, the results can be improved and also, their run time and complexity can be decreased.

This problem can be solved by using a Rule Base System. This is the simplest expert system, but it is very difficult to find the rules with high global performance. It involves several tasks; one of them is feature selection, that is, to choose from the original ones the subset of relevant attributes that provides the best results for the classification. This is a combinatorial problem with exponential complexity and, hence, heuristics methods are useful.

The feature selection problem (FSP) can be viewed as a particular case of a more general subset selection problem in which the goal is to find a subset maximizing some adopted criterion. Feature selection methods search through the subsets of features and try to find the best subset among the competing 2N-1 candidate subsets according to some evaluation measure, where N denotes the total number of features.

Swarm intelligence techniques have been used as the search algorithm in the FSP. Methods which combine ACO and RST to find reducts with good results have been proposed [8]. An algorithm to find rough set reducts by using PSO was introduced in [18, 19]. In this case, a binary representation of the particles was used. The experimental results developed in that work showed that PSO is efficient for rough set-based feature selection. The application of this approach for rule learning was presented in [17].

The effect of introducing local search in the PSO and ACO meta-heuristics to feature selection is been studied. For instance, ACO algorithms perform best when coupled with local search algorithms because these locally optimize the ant's solutions, the coupling can therefore greatly improve the quality of the solutions generated by ants [6]. In this work, the local search algorithm is used to delete some features from the subsets found by the swarm intelligence methods.

3 Tasks

T1. Improve the performance of the ACO+RST model to feature selection problem. Methods which combine ACO and Rough Set Theory (RST) to find reducts with promising results were proposed. They are based on the reduct concept. Because the setting of parameters is crucial for the performance of the ant algorithms [6], we have developed a study about the parameters of this algorithm in the problem of feature selection. We have studied three variants of ant algorithms and the influence of the parameters on the performance both in terms of quality of the results and the number of reducts found [4, 2, 3].

T2. Improve the performance of the PSO+RST model to feature selection problem by the new Two Step PSO+RST. A new model of PSO called Two-Step PSO will be presented [1], the basic idea is to split the heuristic search performed by particles into two stages. The algorithm divides the search process made by the particles in two stages, so that in the first stage preliminary results are reached which are used to build the initial swarm for the second stage. In FSP, this means that subsets of features which are potential reducts are generated in the first stage; these subsets are used to modify the swarm resulting from the last cycle in the first stage, the modified swarm is used as initial population of the second stage.

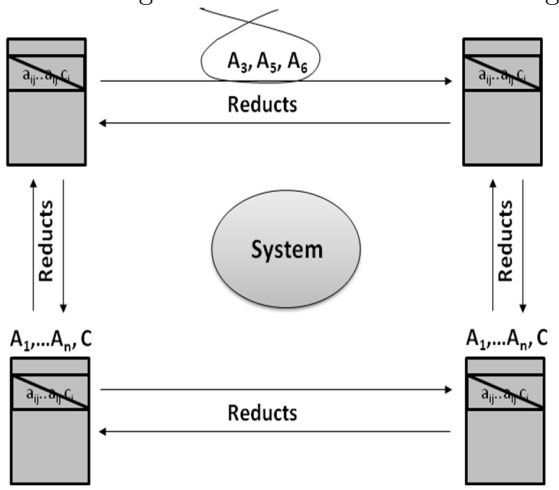
An important issue in the feature selection problem is the length of reducts, that is, the quantity of features included in the reduct. Shorter reducts are preferable. In order to decrease the length of reducts a post processing step can be applied. I studied the effect of introducing local search in the PSO and ACO meta-heuristics to FSP by using local search algorithm to delete some features from the subsets found by the swarm intelligence methods.

T3. Rule induction. I will use classical Learning Rules using Rough Set (LRRS) algorithm from Rough Set Theory to rule induction but taking into account reducts calculated with proposed algorithm. A comparison between algorithms should be established.

T4. Develop a new computational model to learning for the following problem. Suppose there are some entities (see Figure 1) with their dataset and need a system to classify new object based on dataset and attributes values from the new object. Those entities decide to collaborate to have the system. It could be solved in some ways:

1. each entity gets its rules, these are mixed and the system work with all rules together. It fails because different conditions/situation could appear in different places.

Figure 1: Sharing "meta" information to learning rules.



2. all data are put together and the system obtains the rules. It fails if entities do not want to give their data.

3. So, what happen if factories interchange meta information? These could share just the relevant attributes, and each entity gets its rules.

References

[1] P. R. Bello, Y. Gómez, et al. Two step particle swarm optimization to solve the feature selection problem. In *To appear in Proceedings of ISDA 200707*. IEEE Computer Society, 2007.

[2] P. R. Bello, A. Nowe, Y. Caballero, Y. Gómez, and P. Vrancx. A model based on ant colony system and rough set theory to feature selection. In *Proceedings of GECCO 2005*, pages 275–276, New York, NY, USA, 2005. ACM Press.

[3] P. R. Bello, A. Nowe, Y. Caballero, Y. Gómez, and P. Vrancx. Using ant colony system meta-heuristic and rough set theory to feature selection. In *Proceedings of MIC 2005, August 22-26, Vienna, Austria. 2005*, 2005.

[4] P. R. Bello, A. Nowe, P. Vrancx, Y. Caballero, and H. Gómez. Using aco and rough set theory to feature selection. *WSEAS Transactions on Information Science and Applications*, 2(5):512–517, May 2005.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, 1999.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, USA, 2004.

[7] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[8] R. Jensen and S. Q. Finding rough set reducts with ant colony optimization. In *Proceedings of UK Workshop on Computational Intelligence*, pages 15–22, 2003.

[9] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[10] J. Kennedy and R. Eberhart. A discrete binary version of the particle swarm optimization algorithm. In *Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia*, pages 4104–4108, 1997.

[11] J. Komorowski, L. Polkowski, and A. Skowron. Rough sets: a tutorial, 1998.

[12] T. M. Mitchell. Does machine learning really work? *AI Magazine*, 18(3):11–20, 1997.

[13] R. Parpinelli, H. Lopes, and A. Freitas. Data Mining with an Ant Colony Optimization Algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4):321–332, August 2002.

[14] Z. Pawlak. Rough sets. *International Journal of Information & Computer Sciences*, 11:341–356, 1982.

[15] R. C. Prati and P. A. Flach. ROCCER: An algorithm for rule learning based on ROC analysis. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 823–828. Professional Book Center, 2005.

[16] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[17] X. Wang, J. Yang, R. Jensen, and X. Liu. Rough set feature selection and rule induction for prediction of malignancy degree in brain glioma. *Computer Methods and Programs in Biomedicine*, 83(2):147–156, 2006.

[18] X. Wang, J. Yang, N. Peng, and X. Teng. Finding minimal rough set reducts with particle swarm optimization. In D. Slezak et al., editors, *Proceedings of RSFDGrC 2005*, volume 3641 of *LNCS*, pages 451–460. Springer, 2005.

[19] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen. Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters*, 28(4):459–471, 2007.

Novel Genetic Algorithm Crossover Approaches for Time-Series Problems

Paul M. Godley, Julie Cowie and David E. Cairns
Department of Computing Science and Mathematics
University of Stirling, Scotland

Abstract

Genetic Algorithms (GAs) are a commonly used stochastic search heuristic which have been applied to a plethora of problem domains. GAs work on a population of chromosomes (an encoding of a solution to the problem at hand) and breed solutions from fit parents to hopefully produce fitter children through a process of crossover and mutation. This work discusses two novel crossover approaches for GAs when applied to the optimisation of time-series problems, with particular application to bio-control schedules.

1 Introduction

In optimization of intervention schedules for models of dynamic systems, Genetic Algorithms (GAs) commonly use Uniform Crossover (UC) as a method of achieving recombination [8]. Recent work [4] has produced alternative crossover approaches which work on variable length chromosomes that have been shown to outperform UC when applied to dynamic problems, with specific application to the area of bio-control scheduling. Although alternative variable length crossover approaches such as Messy GAs (mGA) exist [6], these have considerable complexity and a two-phase evolutionary approach [1]. This work discusses a simplified approach, which is specifically designed for time-series problems.

2 Problem Domain

This work focuses on the optimisation of intervention schedules and has been tested initially in scheduling bio-control agents to combat sciarid flies. In mushroom farming, the presence of sciarid flies can drastically affect the quality of crop produced. Sciarid fly larvae feed on the mycelium in the casing layer of mushrooms which cause degradation of the crop. The nematode worm *Steinernema feltiae* has proven effective as a bio-control agent to

combat this pest. A set of differential equations which represents the life-cycle of the sciarid flies and potential infection from nematode worms has been produced [3]. These equations have been utilised as a fitness function for testing the novel crossover approaches developed in this work, described in [4] and [5].

3 Crossover Approaches

The novel crossover approaches detailed in [4] were designed to investigate if incorporating the number of interventions (application of bio-control agent) used by good solutions could be used to effectively drive the crossover process. These approaches, CalEB (Calculated Expanding Bin) and TInSSel (Targeted Intervention with Stochastic Selection) both provide mechanisms for crossover of variable and fixed length chromosomes, where each chromosome represents an intervention schedule. CalEB and TInSSel both use the number of interventions present in the parents to calculate the number required in the children, with CalEB utilising a "binning" approach to select the genetic material from the parents, whereas TInSSel contains an element of stochastic selection.

4 Experiments

Previous experiments reviewed CalEB, TInSSel and UC across varying initial intervention samples [4],[5]. These experiments were undertaken for initial population samples from min intervention to min intervention (i.e. 1 to 1, where each member of the initial population has 1 intervention) to min intervention to max intervention (i.e. 1 to 50, where each member of the initial population has between 1 and 50 interventions). These differing variances in possible initial interventions enabled evaluation of how initial spread affects each of the crossover approaches in finding a solution. In addition it demonstrates how the initial variance in population affects the robustness of each crossover approach. Current work reviews the quality of solutions produced when all experiments have min intervention to max intervention (1 to 50), which represents the decision maker being unsure of the sample population to use. The aim of this work is to evaluate the search ability of UC, CalEB and TInSSel across varying limits of fitness functions evaluations to ascertain if there is any difference in search ability between these crossover types. The run parameters used for both these and previous experiments are shown in Table 1. Each run was undertaken 500 times and averaged for each fitness function limit. Tournament selection was used to select parents for breeding as it has been shown to provide better or equivalent convergence and computational properties when compared to alternative approaches [2]. The average scores for these experiments along

Table 1: GA Run Parameters

Parameter	Value	Parameter	Value
Population size	50	Crossover probability	1
Number of parents	2	Mutation probability	0.05
Number of children	2	Days in nematode schedule	50
Fitness Evaluations	50 - 500	Nematodes / intervention	1000

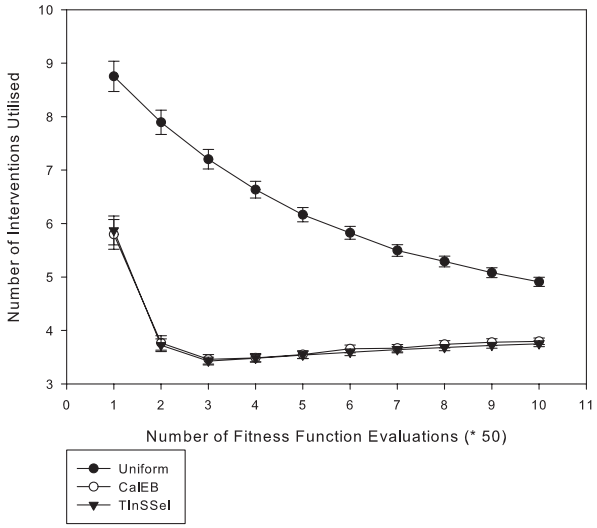


Figure 1: Intervention Utilisation for Solutions

with the associated 95% confidence intervals are depicted in Figures 1 and 2. Figure 1 shows that regardless of the number of fitness functions available, UC solutions require more interventions than solutions returned by CalEB and TInSSel. Figure 2 shows a clear difference in fitness score between TinSSel, CalEB and UC for most experiments, with the exception being those experiments where the number of fitness functions are very large (as all approaches have sufficient time to find a solution). This experiment shows that both CalEB and TInSSel outperform UC in terms of intervention usage and quality of solution found over a varying number of fitness function limits. This was also true when experiments were undertaken over varying initial population intervention numbers [4].

5 Future Work

In order to better understand the dynamics of the novel approaches, application to varying problem domains is required. Future work will focus on

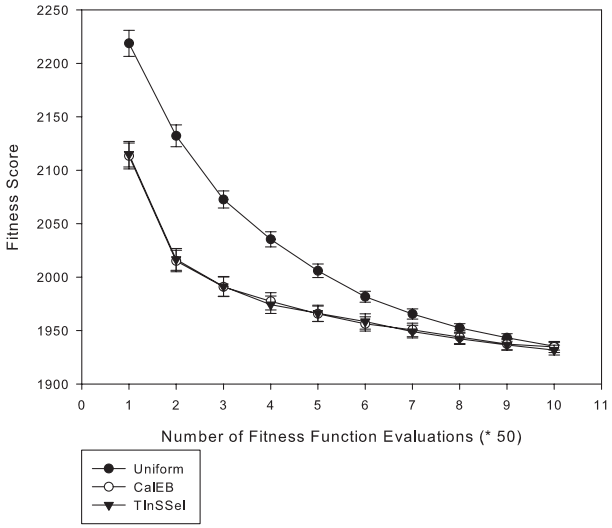


Figure 2: Fitness Scores for Solutions

deriving optimal treatment schedules for cancer chemotherapy [9], using the single drug model detailed in [7].

References

[1] D. Dasgupta and D. McGregor. Sga: A structured genetic algorithm, 1992.

[2] K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[3] A. Fenton, R. L. Gwynn, A. Gupta, R. Norman, J. P. Fairbairn, and P. J. Hudson. Optimal application strategies for entomopathogenic nematodes: integrating theoretical and empirical approaches. *Journal of Applied Ecology*, 39(3):481–492, 2002.

[4] P. M. Godley, D. E. Cairns, and J. Cowie. Directed intervention crossover applied to bio-control scheduling. In *IEEE CEC 2007: Proceedings of the IEEE Congress On Evolutionary Computation*, 2007.

[5] P. M. Godley, D. E. Cairns, and J. Cowie. Maximising the efficiency of bio-control application utilising genetic algorithms. In *EFITA / WCCA 2007: Proceedings of the 6th Biennial Conference of European Federation of IT in Agriculture*, Glasgow, Scotland, UK, 2007. Glasgow Caledonian University.

[6] D. Goldberg, K. Deb, and B. Korb. *Messy genetic algorithms : motivation, analysis, and first results.* Clearinghouse for Genetic Algorithms, Dept. of Mechanical Engineering, University of Alabama, 1989.

[7] A. Petrovski. *An Application of Genetic Algorithms to Chemotherapy Treatments.* PhD thesis, 1999.

[8] A. Petrovski, A. Brownlee, and J. McCall. Statistical optimisation and tuning of ga factors. In *Congress on Evolutionary Computation*, pages 758–764, 2005.

[9] A. Petrovski, J. McCall, and E. Forrest. An application of genetic algorithms to optimisation of cancer chemotherapy. *Int. J. of Mathematical Education in Science and Technology*, 29:377–388, 1998.

Parameterized Random Greedy Algorithms for the Heterogeneous VRP with Time Windows

Zhi Yuan and Armin Fügenschuh
Department of Mathematics, Darmstadt University of Technology, Germany

Abstract

We apply the new meta-heuristic framework of parameterized greedy algorithms (PGreedy) to four different real-world heterogeneous vehicle routing problems with time windows. We extend PGreedy by including more than one scoring function and by a further randomization of the selection within the construction step. Numerical results indicate that our heuristic algorithms are able to achieve large savings in the various application contexts.

1 Introduction

Greedy algorithms are in general simple heuristics to construct feasible solutions for combinatorial optimization problems from scratch in short time. In some cases, they even find proven optimal solutions (for example, Prim's algorithm for spanning trees). For hard optimization problem however they only give sub-optimal solutions. Research interest in the past two decades has thus turned to more sophisticated meta-heuristics (such as genetic algorithms, tabu search, ant colony optimization, GRASP, or simulated annealing). Our approach differs from that. It is well known that the scoring function plays an essential role in the success of a greedy algorithm. We notice in many hard combinatorial optimization problems, it is in general difficult to identify an immediate best local step during the construction process. Thus a simple scoring function statically depending on single criterion is unreliable and even misleading. To overcome this structural problem of greedy algorithms, the parameterized greedy algorithm (PGreedy, for short) was developed by Fügenschuh in [1], [2], [3] as a new type of meta-heuristic. Here more than one criterion is introduced and parameterized with individual weights into a linear scoring function. A parameter tuning technique is performed to find the best parameter weight setting. To this end, methods form Global Optimization, such as improving hit-and-run, can be applied [9]. PGreedy can be further hybridized with existing local search heuristics,

or can be included in a GRASP framework [4], or to generate a starting population for a genetic algorithm.

In this work, we adapt the PGreedy framework for certain vehicle routing problems to be described in the next sections. It turns out that the general framework introduced by Fügenschuh could be further improved by including a higher degree of randomization. For instance, not only the best local step that was identified by the parameterized scoring function is selected immediately. Instead, we select randomly from the list of all scores, where the random function is biased by the score itself, so that lower scores have a higher probability for being selected.

2 Heterogeneous VRP with Time Windows

The heterogeneous vehicle routing problem (VRP) with time windows is stated as follows. Given is a heterogeneous fleet of vehicles. These vehicles differ in home depot location, size, power, loading capacity, costs, and maximal or average speed, for instance. Given is also a set of customers, who expect exactly one of the vehicles to arrive at a certain time or within a certain time window. If the vehicle arrives early then waiting is permitted, but late arrival is strictly forbidden. The objective of the problem is to find a minimum cost assignment of vehicles to customers. That means, it is desired to serve all customers with a minimum size fleet, and on a subsidiary level the total length of all driving distances has to be minimal. A mathematical description of the constraints and objective as a mixed-integer programming model can be found in the book of Toth and Vigo [7]. The adaption of PGreedy for finding good feasible solutions of this VRP problem can be done as follows. We make use of three individual scoring functions.

The first scoring function evaluates each possible assignment of a "fresh" vehicle from the depot. More detailed, for each vehicle type, this parameter-dependent scoring function takes into account the fixed cost for using a vehicle of this type. As a second criterion, it takes into account the capability of the vehicle, that is, how many customers can potentially be served by it.

The second scoring function uses three parameters for evaluating which customer to serve first with the previously selected "fresh" vehicle. To this end, it takes into account the driving cost from the depot to each customer, the earliest possible starting time for this customer, and the number of possible further customers after this one has been served.

The third scoring function is the most critical one for the quality of the produced solution. It also makes use of three parameters. The first parameter takes the driving cost between two customers into account. The second parameter evaluates the idle time the vehicle has to accept when possibly arriving early at the next customer's site. The third parameter evaluates the compatibility of the time windows of the two customers.

These three scoring functions are called iteratively. By the first, we find the most promising vehicle. By the second, we find the most promising first customer for it. Then we apply the third scoring function to generate its route until no further customer can be added, and the vehicle has to be sent back to the depot. Then we start again with the first scoring function, and so on, until all customers are served. At that point in the algorithm, we have a feasible solution to the problem, which is now undergoing a local search phase for further improvement.

The local search consists of two different steps which are carried out iteratively, until no further improvement is found. First, we do a local tour optimization, where we apply classical node insertion and node exchange steps between two routes. Second, we do a local exchange on the vehicle type, where we assign the cheapest possible type to each existing route. Finally, the generated solution is returned.

3 Applications and Results

We applied our general PGreedy strategy to four different real-world applications of heterogeneous VRP with time windows. Besides the general VRP structure, each of the four problem has individual constraints depending on the type of application. The main implementation effort was to adapt PGreedy to each individual situation.

3.1 Locomotive Scheduling

In scheduling locomotives for Deutsche Bahn, several new aspects have to be taken into account, such as cyclic departures of the trains, network-load dependent travel times, and wagon transfers between trains. For further details we refer to [5] and [6]. A complex starting time propagation procedure is developed to keep the multiple and coupled time window feasible during each solution construction. Experiment results show that the heuristic usually generates a good solution (in average around 10% from optimality) in several minutes, and in combination with the MILP solver, where the heuristic solution serves as a start value, it has also significantly speeded up the process of solving the medium instances to optimality.

There has been another locomotive scheduling project cooperated with Siemens, where depots are geographically dispersed and coupling locomotives for pulling a train is allowed. A carefully tuned PGreedy algorithm yields a solution with less than 5% gap from optimality where available, and shows at least 20% potential saving. More details can be found in Yuan [8].

3.2 Home Health Care Services

In a project in cooperation with a local home health care service provider, we focus on the specific field of nurses visiting and providing medical services to clients at home. The task is to plan a weekly schedule, to assign each patient visit to a nurse with competent qualification on an appropriate day at a time within a patient-specified time window. Many other practical restrictions from the problem nature need to be considered, for example, patients need several visits per week, and a minimum inter-visit day difference between some pair of visits should be retained, e.g. some injections must be given twice a week with at least 3 days' break in between; patients can indicate on which day(s) they are expecting a visit, while each nurse has their preferred working days; under the legal framework and personal preference, a maximum daily resp. weekly working time is imposed on each nurse; each patient should, if possible, be visited by the same nurse. The goal is to minimize the operating cost, especially the number of staff members required, while improve the route quality including balancing the workload among staff. The scheduling tool should also be robust, if any changes happen, the reallocation of a new schedule should be made in a short time with as few modifications to the original schedule as possible.

To this end the local search procedure is crucial for this project. After building a set of routes with our PGreedy construction search, a two-phase local search is performed, applying a neighborhood of node insertion followed by a node exchange in each phase. In the first phase we iteratively insert nodes (patient visits) from the nurses with less workload to those with heavier workload, in the hope of reducing the number of nurses. After the number of nurses cannot be reduced anymore, in the second phase conversely, we iteratively insert nodes from nurses with heavier workload to those with less workload, to balance among nurses. In order to ensure each step of local search to be feasible and efficient, a linear constraint propagation is performed, including intra-route propagation for time windows and maximum working time, and inter-route propagation for inter-visit day difference. Current experiment result on a real-world instance shows saving of one nurse out of nine, to serve in total 130 patients with weekly 460 visits.

3.3 School Taxi Routing for Handicapped Pupils

This project differs from school bus scheduling for normal pupils (see Fügenschuh [2]), in the sense that handicapped pupils are usually picked up directly at home by taxis, and more capacity restrictions come to the scene, such as vehicle capacity with respect to different types of wheelchairs. Also the pupils usually have to attend a special school far away from home, and a maximum driving time from home to school is specified for each pupil and must be complied with. There is no explicit time window for each pupil,

however pupils should arrive at school within a given time window, coupled with the maximum driving time, the time interval in which they should be picked up is implied. This problem is a heterogeneous VRP with pickup and delivery (VRPPD) and coupled time windows.

We currently tackle the problem using our PGreedy heuristic. Our first implementation is to pick up pupils always from the same school into one car, and make sure they arrive at school within their driving time tolerance. Thereafter we implement it in a way that pupils from different schools can also be on one car simultaneously, which is more subtle, since during each route construction step one has to check all permutations of to be reached schools to determine a candidate node's feasibility. But experiment confirms that allowing pupils from different schools on board results in a noticeable route efficiency enhancement, in other words, reduced fleet size and cost. A current real-world instance with 11 vehicle types, 700 pupils and 50 schools is solved, and the heuristic solution can potentially save 27 vehicles comparing to the current schedule with 130 vehicles in use.

4 Outlook and Conclusions

In this article we applied the meta-heuristic framework of PGreedy to four different real-world heterogeneous vehicle routing problems with time windows. We adapted PGreedy and extended the general framework with further randomization ideas. The latter were crucial for generating high quality solutions for the given instances of the four problems. It turns out that our solutions improve the status quo to a large extent. In cases where lower bounds were available, we are able to show that our solutions are reasonably close to global optimality.

We think this type of PGreedy algorithm is further applicable in a wide variety of other hard combinatorial optimization problems, such as job shop, flow shop, and open shop scheduling problems, among others.

References

[1] A. Fügenschuh. Parametrized greedy heuristics in theory and practice. *Lecture Notes in Computer Science, Vol. 3636: "Hybrid Metaheuristics, Second International Workshop, HM 2005, Barcelona, Spain, August 29-30, 2005"*, pages 21 – 31, 2005.

[2] A. Fügenschuh. *The Integrated Optimization of School Starting Times and Public Bus Services*. Logos Verlag Berlin, ISBN 3-8325-1037-0, 2006.

[3] A. Fügenschuh. The vehicle routing problem with coupled time windows. *Central European Journal of Operations Research*, 14(2):157 – 176, 2006.

[4] A. Fügenschuh and B. Höfler. Parametrized grasp heuristics for three-index assignment. *Lecture Notes in Computer Science, Vol. 3906: "Evolutionary Computation in Combinatorial Optimization: 6th European Conference, EvoCOP 2006, Budapest, Hungary, April 10-12, 2006"*, pages 61 – 72, 2006.

[5] A. Fügenschuh, H. Homfeld, A. Huck, and A. Martin. Locomotive and wagon scheduling in freight transport. *Proceedings of the ATMOS06*, 2006.

[6] A. Fügenschuh, H. Homfeld, A. Huck, A. Martin, and Z. Yuan. Locomotive and wagon scheduling in freight transport. *submitted*.

[7] P. Toth and D. Vigo. *The Vehicle Routing Problem*. SIAM, Philadelphia, 2002.

[8] Z. Yuan. *Multi-Depot Locomotive Scheduling Problem with Coupling Trips and Time Windows*. Bachelor thesis in Darmstadt University of Technology, also available on request: ericyuanzhi@gmail.com, 2007.

[9] Z. Zabinsky, R. Smith, J. McDonald, H. Romeijn, and D. Kaufman. Improving hit-and-run for global optimization. *Journal of Global Optimization*, 3:171 – 192, 1993.

A Study of Ant Colony Optimization Algorithms for a Biobjective Permutation Flowshop Problem

Trung Truc Huynh, Thomas Stützle,
Mauro Birattari and Yves De Smet
Université Libre de Bruxelles, Brussels, Belgium

Abstract

In this article, we present a study that compares variants of two ACO algorithms designed to tackle a biobjective permutation flowshop scheduling problem where the makespan and the total tardiness are the objectives considered. These two algorithms use respectively one and two pheromone matrices. The analysis of the results gives indications on the choices to adopt when designing an ACO approach for biobjective flowshop scheduling.

1 Introduction

Ant Colony Optimization (ACO) is a population-based SLS method inspired by the foraging behaviour of some ants species [1]. The main idea in ACO algorithms is to mimic the pheromone trails used by real ants. In ACO algorithms, artificial pheromone trails serve as a distributed, numerical information that the ants use to probabilistically construct solutions to the problem being tackled and to adapt these pheromone at run-time to reflect their search experience.

The permutation flowshop scheduling problem (PFSP) requires scheduling n jobs with given processing times on each of m machines such that the job sequence on all machines is identical. Typical further assumptions are that each job can be processed on only one machine at a time, operations are not preemptable, jobs are available for processing at time zero and setup times are independent. Given its industrial relevance, various variants of the PFSP have been considered, including variants that consider various objective functions to be optimized simultaneously. In multiobjective optimization, the goal is to identify all efficient alternatives, that is, the set of Pareto optimal solutions that comprises all solutions that are non-dominated solutions w.r.t. Pareto dominance, which is defined as follows: if all objectives are minimization, an objective vector $V(x)$ is said to dominate a vector $V(x')$ if and only if $\forall i : v_i(x) \leq v_i(x') \wedge \exists i.v_i(x) < v_i(x')$. In this work, we consider

a biobjective version of the PFSP and the two objectives, f_1 and f_2, considered are to minimize the makespan ($f_1 = C_{\max} = \max\{C_1, C_2, \ldots, C_n\}$) and the total tardiness ($f_2 = T = \sum_{i=1}^{n} T_i$, where $T_i = \max\{0, C_i - d_i\}$, where d_i is the due date and C_i the completion time of of the job i).

2 ACO algorithms for biobjective PFSP

The two proposed approaches uses multiple runs of an ACO algorithm and the idea is to force each run to search in different regions of the space. The first method (1phero) consists in aggregating the two objective functions into one single objective function $F = \lambda_1 f_1 + \lambda_2 f_2$, $\lambda_1 + \lambda_2 = 1$. For approximating different areas of the Pareto front, dynamically the search directions are modified by modifying the weights λ_1 and λ_2: if k weight vectors are used, λ_1 and λ_2 change by $\pm 1/k$ when moving from one weight vector to the next one, that is, the weights are uniformly spaced and the minimum amount of change is done when moving from one weight vector to the next one. The second approach (2phero) associates to each objective one pheromone matrix and ants will construct their solutions based on an aggregation of the two pheromone matrices, where a pheromone matrix entry at position i, j is defined by $\tau_{ij} = \lambda_1 \tau^1_{ij} + \lambda_2 \tau^2_{ij}$; again the values of λ_1 and λ_2 are modified to attain different areas of the Pareto front. The use of two pheromone matrices has already been proposed in [2].

Each resulting ACO algorithm is combined with an iterative improvement algorithm in the insert neighbourhood and with a *component-wise* local search that looks for non dominated solutions in the insert neighbourhood and adds these to the archive. For each of the 1phero and 2phero algorithms, two further variants have been studied. In the scratch approach, the colonies for each weight vector λ work independently of each other. In the 2phase approach, the best solution found for the previous weight, is used for the initialisation for the current weight vector. Hence, for each λ vector, a colony starts with a solution that was good for the previous weight. Hence, the solutions in the 2phase approach are treated like a chain. For the two pheromone matrices approach (2phero), two further sub-variants have been studied. These two configurations, 2pheroG and 2pheroL differ in the update of the pheromones. In 2pheroG, only the best solutions found for each objective function across all already considered weight vectors are allowed to update the pheromone matrices τ^1_{ij} and τ^2_{ij}, respectively. For 2pheroL, the update is done by the best solutions found for the current aggregation weight. In addition to these variants, we also have studied the influence of the number of weights used and the influence of the direction in which the weight vector changes (initial weight one for either f_1 or f_2).

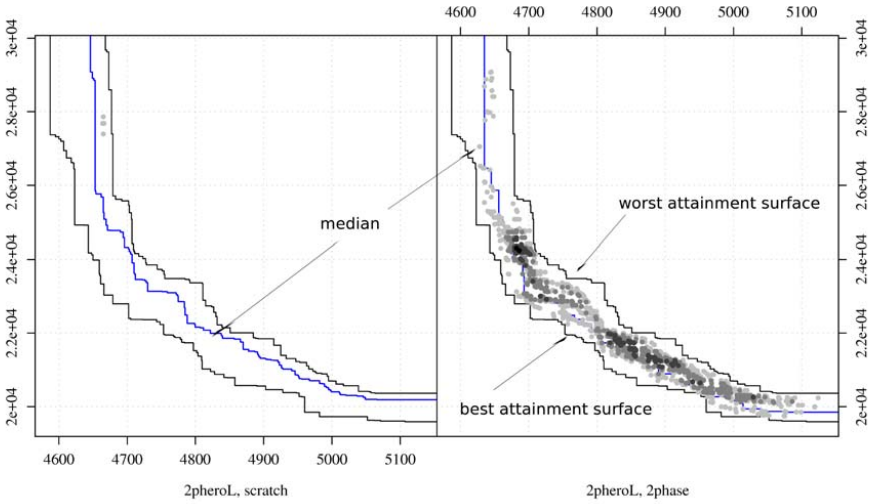


Figure 1: Difference of EAFs, comparison scratch-2phase approach

3 Experimental results

Each of the variants defined above was experimentally tested in a number of benchmark instances using ten independent trials. The computation times were chosen the same for all algorithmic variants. The analysis of the results is based on pairwise comparisons between the outcomes of algorithms. To avoid the known short-comings of performance measures for multi-objective optimizers [4], we first examine whether for one algorithm the outperformance relation holds and, if this is not the case, we compute attainment functions, apply statistical tests on the equality of two attainment functions and, if the test is rejected, we use the visualization of the difference of two attainment surfaces to detect these [3]. In the plots of the attainment surfaces (see Figures 1 and 2), we draw three lines where the rightmost connects the set of points attained by any run of the two configurations (worst case performance) and the leftmost connects the best set of point attained (best case performance). The line between the two gives the median. The different shades of gray indicate how large are the differences at specific points of the attainment function (only differences above 0.2 are given); the darker the point, the larger is the difference. On the left is given the advantage of algorithm A over B, while on the right side, the advantage of B over A is given, where A and B are the variants indicated below the x-axis of each plot.

The two examples of the visualization of the differences between two algorithms indicate the following facts. Figure 1 indicates that 2pheroL,2phase performs better than 2pheroL,scratch in most regions of the objective space.

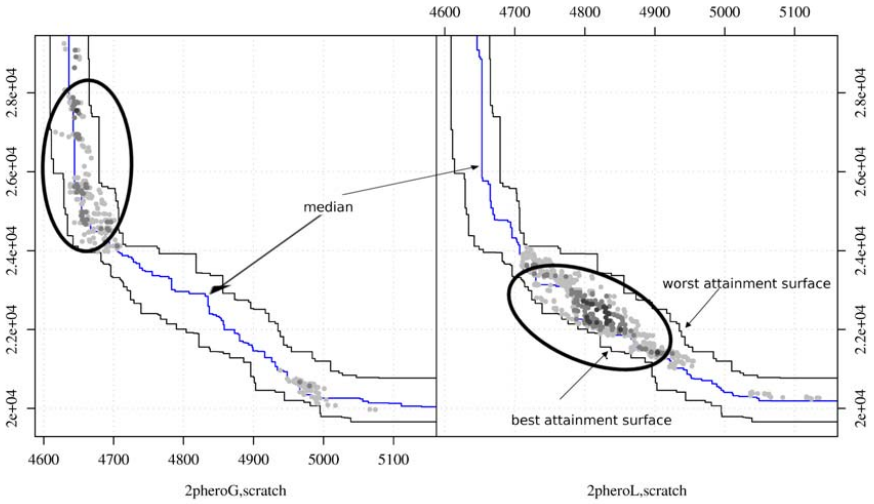


Figure 2: Difference of EAFs, comparison global-local strategy

This means that for the tackled instance, it is advantageous to keep information from previous colonies. In Figure 2, we can observe that the two strategies, 2pheroL,scratch and 2pheroG,scratch have advantages in two distinct regions: 2pheroL,scratch is clearly better in the center, while 2pheroG,scratch is better in the upper left corner.

The main observations of these and further comparisons can be summarized as follows.

– The use of the component-wise local search often improves significantly the quality of the approximation obtained.

– The 2phase approach in most cases leads to an improved performance over the scratch approach.

– A minimum number of weight vectors seems to be necessary to obtain good approximation sets.

– When comparing 2pheroG and 2pheroL, typically the former is less performing in the middle of the front but it can give advantages towards the extremes, where one objective receives a very high weight.

– The direction of the changes of the weight vector can have an influence on the performance in the 2phase approach.

– In general, the performance of the different variants depends strongly on the instances tackled and, apparently, strongly on the number of machines.

Future work can focus on different directions. A first would certainly be to extend further the empirical basis of our findings by enlarging the experimental study by more instances or a more systematic modification of instance characteristics. Other directions are to explore further some of the

observations made here. One is to study with our or similar algorithms the influence of different directions taken in the changes to the weight vectors. Another would be to provide combinations between the 2pheroG and 2pheroL search strategies, given that each has its own advantages in different areas of the approximation to the Pareto front. Finally, comparisons with other multiobjective optimizers will also be necessary.

References

[1] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, 2004.

[2] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. In E. Zitzler et al., editors, *Proceedings of EMO'01*, volume 1993 of *LNCS*, pages 359–372. Springer Verlag, 2001.

[3] M. López-Ibáñez, L. Paquete, and T. Stützle. Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, 5(1):111–137, 2006.

[4] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.