Jin-Kao Hao
Martin Middendorf (Eds.)

Evolutionary Computation in Combinatorial Optimization

LNCS 7245

12th European Conference, EvoCOP 2012
Málaga, Spain, April 2012
Proceedings

Springer

Lecture Notes in Computer Science 7245

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Jin-Kao Hao Martin Middendorf (Eds.)

Evolutionary Computation in Combinatorial Optimization

12th European Conference, EvoCOP 2012
Málaga, Spain, April 11-13, 2012
Proceedings

Springer

Volume Editors

Jin-Kao Hao
University of Angers, Faculty of Sciences
2, Boulevard Lavoisier, 49045 Angers Cedex 01, France
E-mail: jin-kao.hao@univ-angers.fr

Martin Middendorf
University of Leipzig, Department of Computer Science
Johannisgasse 26, 04103 Leipzig, Germany
E-mail: middendorf@informatik.uni-leipzig.de

Cover illustration:
"Chair No. 17" by The Painting Fool (www.thepaintingfool.com)

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-29123-4 e-ISBN 978-3-642-29124-1
DOI 10.1007/978-3-642-29124-1
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012933851

CR Subject Classification (1998): F.1, C.2, H.4, I.5, I.4, F.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Metaheuristics continue to demonstrate their effectiveness for an ever-broadening range of difficult combinatorial optimization problems appearing in a wide variety of industrial, economic, and scientific domains. Prominent examples of metaheuristics are evolutionary algorithms, tabu search, simulated annealing, scatter search, memetic algorithms, variable neighborhood search, iterated local search, greedy randomized adaptive search procedures, ant colony optimization and estimation of distribution algorithms. Problems solved successfully include scheduling, timetabling, network design, transportation and distribution, vehicle routing, the travelling salesman problem, packing and cutting, satisfiability and general mixed integer programming.

EvoCOP began in 2001 and has been held annually since then. It is the first event specifically dedicated to the application of evolutionary computation and related methods to combinatorial optimization problems. Originally held as a workshop, EvoCOP became a conference in 2004. The events gave researchers an excellent opportunity to present their latest research and to discuss current developments and applications. Following the general trend of hybrid metaheuristics and diminishing boundaries between the different classes of metaheuristics, EvoCOP has broadened its scope in recent years and invited submissions on any kind of metaheuristic for combinatorial optimization.

This volume contains the proceedings of EvoCOP 2012, the 12th European Conference on Evolutionary Computation in Combinatorial Optimization. It was held in Málaga, Spain, during April 11–13, 2012, jointly with EuroGP 2012, the 15th European Conference on Genetic Programming, EvoBIO 2012, the 10th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, EvoMUSART, the First International Conference and the 10th European Event on Evolutionary and Biologically Inspired Music, Sound, Art and Design, and EvoApplications 2012 (formerly EvoWorkshops), which consisted of the following 11 individual tracks: 9th European event on the Application of Nature-Inspired Techniques for Telecommunication Networks and Other Parallel and Distributed Systems (EvoCOMNET), Third European event on Evolutionary Algorithms and Complex Systems (EvoCOMPLEX), 6th European event on Evolutionary and Natural Computation in Finance and Economics (EvoFIN), 4th European event on Bio-inspired Algorithms in Games (EvoGAMES), 7th European event on Bio-inspired Heuristics for Design Automation (EvoHOT), 14th European event on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP), 5th European event on Bio-inspired Algorithms for Continuous Parameter Optimization (EvoNUM), First European event on Parallel Implementation of Evolutionary Algorithms (EvoPAR), First European event on Computational Intelligence for Risk Management, Security and Defence Applications (EvoRISK), 7th European event

on Nature-Inspired Techniques in Scheduling, Planning and Timetabling (EvoSTIM), and 9th European event on Evolutionary Algorithms in Stochastic and Dynamic Environments (EvoSTOC). Since 2007, all these events are grouped under the collective name EvoStar, and constitute Europe's premier co-located meetings on evolutionary computation.

Accepted papers of previous EvoCOP editions were published by Springer in the series *Lecture Notes in Computer Science* (LNCS – Volumes 2037, 2279, 2611, 3004, 3448, 3906, 4446, 4972, 5482, 6022, 6622). Below we report statistics for each conference:

EvoCOP	Submitted	Accepted	Acceptance ratio
2001	31	23	74.2%
2002	32	18	56.3%
2003	39	19	48.7%
2004	86	23	26.7%
2005	66	24	36.4%
2006	77	24	31.2%
2007	81	21	25.9%
2008	69	24	34.8%
2009	53	21	39.6%
2010	69	24	34.8%
2011	42	22	52.4%
2012	48	22	45.8%

The rigorous, double-blind reviewing process of EvoCOP 2012 resulted in the selection of 22 out of 48 submitted papers; the acceptance rate was 45.8%. The number of submissions was higher compared to the previous event. Each paper was reviewed by at least three members of the international Program Committee. All accepted papers were presented orally at the conference and are included in this proceedings volume. We would like to acknowledge the members of our Program Committee and external reviewers: we are very grateful for their thorough work. We also thank all the authors for submitting their work to this EvoCOP edition. EvoCOP 2012 contributions consist of novel algorithms together with important new insights into how well these algorithms can solve prominent test problems from the literature or real-world problems.

The success of the conference resulted from the input of many people to whom we would like to express our appreciation. First of all, we like to thank the local Chair of EvoStar 2012, Carlos Cotta from the University of Málaga. He and his team did an extraordinary job for which we are very grateful. We thank Marc Schoenauer from INRIA in France for his support with the MyReview conference management system. We thank Penousal Machado of the University of Coimbra for an excellent web site and publicity material. Thanks are also due to Jennifer Willies and the Institute for Informatics and Digital Innovation at Napier University in Edinburgh, UK, for administrative support and event coordination. We gratefully acknowledge the University of Málaga for its support of EvoStar, and in particular the School of Computer Science and the School of

Telecommunications and their respective directors, José M. Troya and Antonio Puerta. We also thank the Málaga Convention Bureau.

Last, but not least, we would like to thank Carlos Cotta, Peter Cowling, Jens Gottlieb, Jano van Hemert, Peter Merz, and Günther Raidl for their hard work and dedication in past editions of EvoCOP, which contributed to making this conference one of the reference events in evolutionary computation and metaheuristics.

April 2012

Jin-Kao Hao
Martin Middendorf

Organization

EvoCOP 2012 was organized jointly with EuroGP 2012, EvoBIO 2012, EvoMUSART 2012, and EvoApplications 2012.

Organizing Committee

PC Chairs

Jin-Kao Hao	University of Angers, France
Martin Middendorf	University of Leipzig, Germany

Local Chair

Carlos Cotta	University of Málaga, Spain

Publicity Chair

Penousal Machado	University of Coimbra, Portugal

EvoCOP Steering Committee

Carlos Cotta	Universidad de Málaga, Spain
Peter Cowling	University of Bradford, UK
Jens Gottlieb	SAP AG, Germany
Jin-Kao Hao	University of Angers, France
Jano van Hemert	University of Edinburgh, UK
Peter Merz	University of Applied Sciences and Arts, Hannover, Germany
Günther Raidl	Vienna University of Technology, Austria

Program Committee

Adnan Acan	Eastern Mediterranean University, Gazimagusa, Turkey
Hernán Aguirre	Shinshu University, Nagano, Japan
Enrique Alba	Universidad de Málaga, Spain
Mehmet Emin Aydin	University of Bedfordshire, UK
Ruibin Bai	University of Nottingham, UK
Thomas Bartz-Beielstein	Cologne University of Applied Sciences, Germany
Maria Blesa	Universitat Politècnica de Catalunya, Spain

Christian Blum	Universitat Politècnica de Catalunya, Spain
Rafael Caballero	University of Málaga, Spain
Alexandre Caminada	UTBM, France
Pedro Castillo	Universidad de Granada, Spain
Carlos Coello Coello	CINVESTAV-IPN, Mexico
Peter Cowling	University of Bradford, UK
Keshav Dahal	University of Bradford, UK
Karl Doerner	Universität Wien, Austria
Benjamin Doerr	Max-Planck-Institut für Informatik, Germany
Anton V. Eremeev	Omsk Branch of Sobolev Institute of Mathematics, Russia
Antonio J. Fernández	Universidad de Málaga, Spain
Francisco Fernández de Vega	University of Extremadura, Spain
Bernd Freisleben	University of Marburg, Germany
Philippe Galinier	Ecole Polytechnique de Montreal, Canada
Jens Gottlieb	SAP, Germany
Walter Gutjahr	University of Vienna, Austria
Jin-Kao Hao	University of Angers, France
Richard F. Hartl	University of Vienna, Austria
Geir Hasle	SINTEF Applied Mathematics, Norway
István Juhos	University of Szeged, Hungary
Graham Kendall	University of Nottingham, UK
Joshua Knowles	University of Manchester, UK
Mario Köppen	Kyushu Institute of Technology, Japan
Jozef Kratica	University of Belgrade, Serbia
Rhyd Lewis	Cardiff University, UK
Arnaud Liefooghe	Université des Sciences et Technologies de Lille, France
Arne Løkketangen	Molde College, Norway
José Antonio Lozano	University of the Basque Country, Spain
Zhipeng Lu	HUST, China
Penousal Machado	University of Coimbra, Portugal
Dirk C. Mattfeld	University of Braunschweig, Germany
Barry McCollum	Queen's University Belfast, UK
Juan Julián Merelo	University of Granada, Spain
Peter Merz	University of Applied Sciences and Arts, Hannover, Germany
Martin Middendorf	Universität Leipzig, Germany
Julian Molina	University of Málaga, Spain
Pablo Moscato	The University of Newcastle, Australia
Christine L. Mumford	Cardiff University, UK
Nysret Musliu	Vienna University of Technology, Austria
Yuichi Nagata	Tokyo Institute of Technology, Japan
Giuseppe Nicosia	University of Catania, Italy

Mario Pavone	University of Catania, Italy
Francisco J.B. Pereira	Universidade de Coimbra, Portugal
Daniel Cosmin Porumbel	University of Artois, France
Jakob Puchinger	Arsenal Research, Vienna, Austria
Günther Raidl	Vienna University of Technology, Austria
Marcus Randall	Bond University, Queensland, Australia
Marc Reimann	Warwick Business School, UK
Andrea Roli	Università degli Studi di Bologna, Italy
Eduardo Rodriguez-Tello	CINVESTAV-Tamaulipas, Mexico
Michael Sampels	Université Libre de Bruxelles, Belgium
Marc Schoenauer	INRIA, France
Patrick Siarry	Université Paris-Est Créteil Val-de-Marne, France
Jim Smith	University of the West of England, UK
Giovanni Squillero	Politecnico di Torino, Italy
Thomas Stützle	Université Libre de Bruxelles, Belgium
El-ghazali Talbi	Université des Sciences et Technologies de Lille, France
Kay Chen Tan	National University of Singapore, Singapore
Jorge Tavares	MIT, USA
Jano van Hemert	University of Edinburgh, UK
Sebastien Verel	Université de Nice Sophia Antipolis, France
Fatos Xhafa	Universitat Politecnica de Catalunya, Spain
Takeshi Yamada	NTT Communication Science Laboratories, Japan
Shengxiang Yang	Brunel University, UK

External Reviewers

Gerald Senarclens de Grancy	Graz University, Austria
Stephan Meisel	Technische Universität Braunschweig, Germany

Table of Contents

A Methodology for Comparing the Execution Time of Metaheuristics Running on Different Hardware

Julián Domínguez and Enrique Alba

Universidad de Málaga, ETSII,
Campus de Teatinos. 29071 Málaga, Spain
{julian,eat}@lcc.uma.es
http://neo.lcc.uma.es

Abstract. In optimization, search, and learning, it is very common to compare our new results with previous works but, sometimes, we can find some troubles: it is not easy to reproduce the results or to obtain an exact implementation of the original work, or we do not have access to the same processor where the original algorithm was tested for running our own algorithm. With the present work we try to provide the basis for a methodology to characterize the execution time of an algorithm in a processor, given its execution time in another one, so that we could fairly compare algorithms running in different processors. In this paper, we present a proposal for such a methodology, as well as an example of its use applied to two well-known algorithms (Genetic Algorithms and Simulated Annealing) and solving the MAXSAT problem.

Keywords: Comparisons, metaheuristics, performance, CPU, run time.

1 Introduction

It is fairly common in research to perform a comparison of algorithms running in different hardware (in fact, comparisons are a must in modern research). It is common to compare the results of our algorithms with previous state-of-the-art works, and it is not always easy to exactly implement or reproduce these previous works or to get access to the same hardware for running our experiments. As a consequence, several researchers purely dismiss this issue, while others try to get a quick idea on how many times their hardware is faster than the previous one and make a too simplistic (and, as we will show, wrong) factorial comparison of times. From this problem it arises the motivation of our work. Our aim in this article is to obtain a methodology for being capable of making a fair comparison between algorithms running in different hardware.

A first approach to the problem could be the use of a scientific benchmark program for obtaining a quantitative measure for the speed of the different processors involved and assume that this *speedup* can be translated into the execution time *speedup*. Although this is an intuitively good approach, in this work

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 1–12, 2012.

we will show that is not valid. Having a processor with good integer operations performance does not mean at all that it is a good processor in floating point operations, thus, as we will see, a better score running a benchmark could not be reflected in a better time executing a metaheuristic because of the specific operations used in each software.

In this paper we try to provide a different approach for developing a methodology for comparing algorithms running in different hardware, taking as a basis a scientific benchmark software. For our study we are going to use six different hardware configurations and two different algorithms. We are also going to analyze several different benchmark programs, widely used in the literature, being `Whetstone`, `Dhrystone`, `Livermore Loops`, `Linpack` and `CPUBenchmark`. Here we introduce the methodology followed:

- **Obtaining the data**: the first step is running the benchmarks and the algorithms on all the hardware platforms, and obtain their running times and scores (Section 4).
- **Choosing a benchmark**: Once obtained the data, we study the relationship between running times and benchmark scores, and choose the benchmark whose score best describes the behavior of the algorithms (Section 5).
- **Fitting the Data**: With this data we perform a regression analysis to obtain a set of equations for predicting the running time of the algorithms in different hardware configurations, depending on the score of the chosen benchmark and the running time in a baseline processor (Section 6)

The proposed methodology could be also useful for other different tasks, like approaching the time it will take to run an experiment - useful in a shared environment or with time restrictions - or choosing the best of our processors for running our algorithms.

This paper is organized as follows. The next section (Section 2) provides a brief review about benchmark programs and a first approach to our problem. Section 3 exposes the algorithms, problem and hardware used for our work, as well as describes the methodology we have followed. In Section 4 we show the data that we have obtained for the analysis. Section 5 contains the analysis of the benchmarks and their suitability for our task. How we have obtained the equations and how to use them is explained in Sections 6 and 7. Eventually, concluding remarks and future research lines are shown in Section 8.

2 Background

Let us suppose that we have developed a new algorithm and we want to compare our results with the ones obtained by another researcher some years ago. We might have to face some problems: perhaps we do not have access to the same hardware in which the past results were obtained or, although we get access to this platform, there is not enough information in the paper to exactly reproduce

the algorithm. What could we do to make a fair comparison between algorithms running on different CPUs? Can we characterize the speed of the CPU to help us with this task?

An important characteristic of computers used for scientific work is the speed of their CPU. From the '70s the scientific community has developed several tests for quantitatively compare the performance of different CPUs, what is usually know as benchmark programs. Such benchmark programs are designed to mimic a type of workload, usually integer and floating point mathematical operations.

As we have mentioned above, a common sense approach could be to use an inverse cross multiplication: if the algorithm A takes n seconds on processor P_1 which has a score of S_1 MFLOPS in a given benchmark program, in the processor P_2 with score S_2, the time will be $n * S_1/S_2$ seconds. Intuitively, we can think that this would give us a good approximation, but the problem is not so easy. The reason is that, when running a program, the access to memory, the input/output operations, the bus transferences of data and any specific feature of the run software (like the kind of loops typical in metaheuristics) can make a huge difference between two processors that look similar from their processor point of view. Also, in general, synthetic benchmarks are designed for running a set of operations that could not be representative of the operations performed in metaheuristics.

We have analyzed different benchmarks and concluded that this direct approach is not valid for none of the studied benchmarks (Section 5). For facing our problem it is necessary to define a more complex methodology, so we are going to try to provide the starting point of such a methodology, with a mathematic approach constructed with the basis of one of the studied benchmarks. In the next subsections we are going to introduce some of the most used scientific benchmark programs, which we will later analyze.

2.1 Classic Benchmarks

In this subsection we include two classic benchmarks, widely used in the scientific community, which were among the first programs that set standards of performance for computers. Although there exist some other classic benchmarks, we have chosen only two of them, which probably are the most popular in the literature: `Whetstone` and `Dhrystone`.

Whetstone. The `Whetstone` benchmark was written by Harold Curnow of CCTA [2], the British government computer procurement agency, based on work by Brian Wichmann of the National Physical Laboratory. An Algol version of the benchmark was released in 1972 and Fortran single and double precision versions were released in 1973. Fortran versions became the first general purpose benchmarks that set industry standards of performance; nowadays we can find "C" implementations. The benchmark produces a score in terms of Thousands of Whetstone Instructions Per Second (KWIPS).

Dhrystone. The `Dhrystone` "C" benchmark was released as an integer benchmark for UNIX systems, a sort of `Whetstone` benchmark without floating point operations. It became a standard benchmark, from 1984, with the growth of Unix systems. The first version was produced by Reinhold P. Weicker in ADA [9] and later translated to "C". Two versions of `Dhrystone` are available (versions 1.1 and 2.1). The 2.1 version was produced to avoid over-optimization problems encountered with version 1. Original versions of the benchmark gave performance ratings in terms of Dhrystones per second. This was later changed to VAX MIPS by dividing Dhrystones per second by 1757, the DEC VAX 11/780 result.

2.2 High Performance Benchmarks

In this subsection we present two classic benchmarks which were originally designed for assessing the performance of supercomputers, although they are nowadays also used for desktop computers. We have chosen the `Livermore Loops` benchmark and the `Linpack` benchmark because of their extended use.

Livermore Loops. This supercomputer numeric benchmark was introduced in 1970, initially comprising 14 kernels of numerical application, written in Fortran. The number of kernels was increased to 24 in the 1980's [8]. Performance measures are in terms of MFLOPS. The program also checks the results for computational accuracy. One main aim was to avoid producing single number performance comparisons, the 24 kernels being executed three times at different Do-loop spans to produce short, medium, and long vector performance measures.

Linpack. This benchmark was produced by Jack Dongarra [3] from the "LINPACK" package of linear algebra routines. It became the primary benchmark for scientific applications from the mid 1980's with a slant towards supercomputer performance. The original version was produced in Fortran but a "C" version appeared later. The standard "C" version operates on 100×100 matrices in double precision with rolled/unrolled and single/double precision options. Other versions are available with different sizes of matrices. Performance rating is in terms of MFLOPS.

2.3 Modern Benchmarks

In the last two decades, the rising of the PCs has fostered the development of non-scientific benchmarks. These benchmarks are focused in assessing the day-to-day use performance for desktop computers, so they include specific tests for tasks like compression, encryption and graphics. Among the vast amount of such benchmarking software, we have chosen `CPUBenchmark` as a popular one in this category because of its extended use and variety of aspects tested, as well as the existence of a huge and public database of results maintained by its developers.

CPUBenchmark. `CPUBenchmark` is a program provided by a software development group specialized in the development of performance benchmarking solutions. They also maintain one of the world largest CPU benchmark websites, `cpubenchmark.net`[1], which gives users access to `CPUBenchmark` results for over 350,000 systems covering more than 1,200 different models of CPUs. `CPUBenchmark` conducts eight different tests and averages its results to determine the CPU mark for a system. The tests cover different applications: integer math, compression, prime number calculation, encryption, floating point math, SSE/3D Now, image rotation and string sorting test. The software runs one simultaneous CPU test for every logical CPU (Hyper-threaded), physical CPU core (multi-core) or physical CPU package (multiple CPUs).

3 Experimental Context

In the present study we are going to focus in two representative algorithms of the two big families of metaheuristics: Evolutionary Algorithms (EAs) and Local Search Methods (LSMs). We have chosen a Genetic Algorithm and a Simulated Annealing as paradigmatic algorithms in search, optimization, and machine learning, because methods of both families are very similar to these algorithms and the results obtained could be easily extended to other algorithms. We are going to focus in combinatorial problems and binary representation, so the results might be not useful for other kind of problems or representation. Since we need to solve a problem, we have chosen for this first paper on benchmarking metaheuristics the well-known MAXSAT problem as it is a basic problem for combinatorial optimization. In the next subsections we are going to introduce the used algorithms, problem, and the used hardware, as well as describe the methodology followed for the study.

3.1 Algorithms and Problem

For our study we have used two representative and well-known algorithms, a Genetic Algotihm (GA) [5] and a Simulated Annealing (SA) [6].

GAs are one of the more popular EAs present in the literature. In Algorithm 1 we can see an outline of a panmictic GA. A GA starts by randomly generating an initial population $P(0)$, with each individual encoding a candidate solution for the problem and its associated fitness value. At each iteration, a new population $P'''(t)$ is generated using simple stochastic operators, leading the population towards regions with better fitness values.

SA has become one of the most popular LSMs. It is widely extended in the industry, and it could be presented as the representative trajectory method. SA is a stochastic algorithm which explores the search space using a hill-climbing process. A panmictic SA is outlined in Algorithm 2. SA starts with a randomly generated solution S. At each step, a new candidate solution S' is generated.

[1] http://www.cpubenchmark.net/

Algorithm 1. Standard Genetic Algorithm

```
Generation(P(0));
Evaluation(P(0));
t := 0;
while not stop_condition(P(t)) do
   P'(t) := Selection(P(t));
   P''(t) := Recombination(P'(t));
   P'''(t) := Mutation(P''(t));
   Evaluation(P'''(t));
   P(t + 1) := Replacement(P(t),P'''(t));
   t := t+1;
end while
```

If the fitness value of S' is better or equal than the old value, S' is accepted and replaces S. As the temperature T_k decreases, the probability of accepting a lower quality solution S' decays exponentially towards zero according to the Boltzmann probability distribution. The temperature is progressively reduced following an annealing schedule.

Algorithm 2. Standard Simulated Annealing

```
Generate(S);
Evaluate(S);
Initialize(T_0);
k := 0;
while not stop_condition(S) do
   S' := Generate(S,T_k);
   if Accept(S,S',T_k) then
      S := S';
   end if
   T_{k+1}:= Update(T_k);
   k := k+1;
end while
```

For our study, we have used a well-known problem, the maximum logical clauses satisfiability problem, MAXSAT [4]. MAXSAT is the most representative problem of combinatorial optimization, and using this problem we can reproduce the main characteristics of combinatorial problems.

Formally, the MAXSAT problem is formulated as follows. Being $U = \{u_1, ..., u_n\}$ a set of n logical variables, an assignment for u is a function $t : U \rightarrow \{true, false\}$. The literal u (or $\neg u$) is true conditioned by an assignment of values t if and only if $t(u) = true$ (or $t(\neg u) = false$). A clause is defined as a set C of literals that are connected by the disjunction. The set of clauses is called a formula. A formula f consists of the conjunction of its clauses (conjunctive normal form). An assignment t satisfies a formula f if and only if t satisfies all the clauses in f. Each clause C is satisfied if there exists, at least, a literal $u \in C$ which is true conditioned by

assignment t. The MAXSAT problem consists in finding an assignment t that maximizes the number of satisfied clauses of a formula f.

We have used three different instances of MAXSAT, consisting of 43 clauses with 10 variables (MS43), 430 clauses with 100 variables (MS430) and 4300 clauses with 1000 variables (MS4300) respectively. These are instances from the phase transition region [1] of the Random-3-SAT problem, the ones whose resolution in research is meaningful.

3.2 Hardware Platforms

We have obtained data from a wide range of different CPUs, including single and multicore, single and multithreaded, single and multiprocessor, 32 and 64 bits, and different families of processors. The used CPUs and their specifications are listed in Table 1.

Table 1. List of used CPUs and their main specifications

Alias	CPU Name	#Cores	Clock	Multithread	Year
CPU1	Intel Pentium 4 2.4GHz	1	2.40 GHz	No	2002
CPU2	Intel Pentium 4 2.8GHz	1	2.80 GHz	No	2002
CPU3	Intel Pentium D 2.8GHz	2	2.80 GHz	No	2005
CPU4	Intel Core2 Quad Q9400	4	2.66 GHz	No	2008
CPU5	2 × Intel Xeon E5405	2 × 4	2 × 2.00 GHz	No	2007
CPU6	Intel Core i7 920	4	2.67 GHz	Yes	2008

3.3 Methodology

The algorithms have been implemented in C++, and the stop condition is to achieve a fixed number of generations, being 10000 for GA and 20000 for SA. Due to the stochastic nature of the algorithms, the final running times are obtained averaging the running times of 30 independent runs. The classic and high performance benchmarks were written in C, and the source code can be obtained in Roy Longbottom's PC Benchmark Collection[2]. The score for `CPUBenchmark` was obtained from the public database. Both, algorithms and benchmarks, where compiled with `gcc` using '`-o3`' optimization. All the computers are managed by a GNU/Linux OS. All the computed results are shown in Section 4.

4 Obtaining the Data

The first phase in our work is to run the algorithms and benchmarks in every processor and obtain the run time and benchmark scores. In this section we show all the data we will use in the rest of the paper. Table 2 shows the execution time for each algorithm and problem size. In Table 3 we can see the scores obtained for each benchmark. As we mentioned above, the metrics used in the benchmarks are not always the same.

[2] http://www.roylongbottom.org.uk/

Table 2. Running time (ms) for each algorithm and problem size

Algorithm/Problem	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6
GA MS43	5212624	4524800	3788930	1817040	1187610	850194
GA MS430	22335980	20356130	21500000	12515498	11065430	10600700
GA MS4300	245730200	211729000	230456500	139558000	141758000	115942800
SA MS43	282868	230458	189655	90377	74296	42523
SA MS430	666934	548548	587401	336423	309566	199641
SA MS4300	4901484	4156282	4150450	2853240	3098920	2396406

Table 3. Scores for each benchmark and CPU

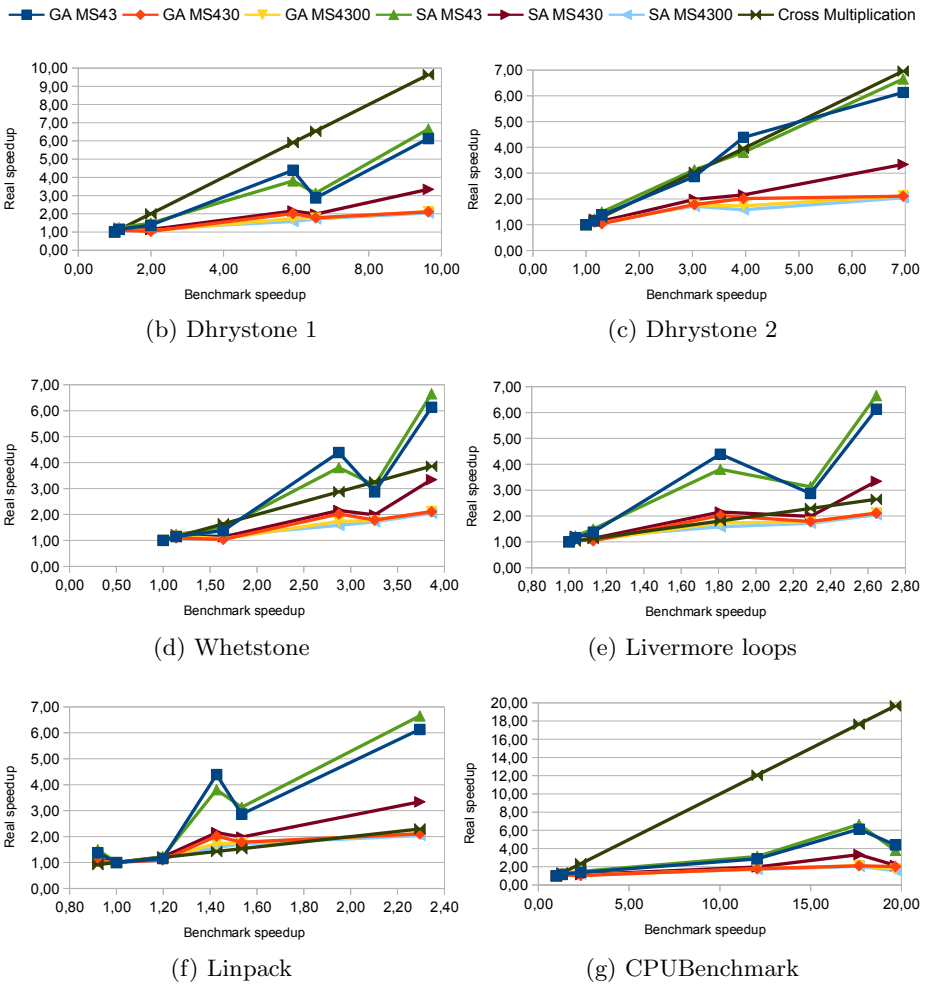
Benchmark	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6
Dhrystone 1 (VMIPS)	2139	2403	4271	13979	12642	20614
Dhrystone 2 (VMIPS)	1978	2286	2562	6006	7830	13764
Whetstone (MWIPS)	829	943	1360	2698	2382	3202
Livermore Loops (MFLOPS)	516	534	583	1184	935	1366
Linpack (MFLOPS)	864	1035	796	1327	1234	1984
CPUBenchmark score	315	415	736	3797	6195	5564

5 Choosing a Benchmark

For both running times and benchmark scores we have obtained the *speedup* with respect to the slowest processor - baseline processor - as a way of normalizing the obtained data. With this baseline results the comparison between runtime and benchmark and between different benchmarks can be easily carried out without caring about different problem instances or algorithms. We have presented the results in Table 4, and a graphical representation can be found in Figure 1.

Table 4. *Speedup* with respect to the baseline processor

Benchmark/Algorithm	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6
Dhrystone 1	1.00	1.12	2.00	6.54	5.91	9.64
Dhrystone 2	1.00	1.16	1.30	3.04	3.96	6.96
Whetstone	1.00	1.14	1.64	3.26	2.87	3.86
Livermore Loops	1.00	1.03	1.13	2.29	1.81	2.64
Linpack	1.00	1.20	0.92	1.53	1.43	2.30
CPUBenchmark	1.00	1.32	2.34	12.05	19.67	17.66
GA MS43	1.00	1.15	1.38	2.87	4.39	6.13
GA MS430	1.00	1.10	1.04	1.78	2.02	2.11
GA MS4300	1.00	1.16	1.07	1.76	1.73	2.12
SA MS43	1.00	1.23	1.49	3.13	3.81	6.65
SA MS430	1.00	1.22	1.14	1.98	2.15	3.34
SA MS4300	1.00	1.18	1.18	1.72	1.58	2.05



(b) Dhrystone 1 (c) Dhrystone 2

(d) Whetstone (e) Livermore loops

(f) Linpack (g) CPUBenchmark

Fig. 1. Benchmark baseline results vs. real time baseline results

One of the first facts we realize is that the cross multiplication approach is not valid for any of the studied benchmark and is not valid for any benchmark using a single score. For this approach to be valid, all the rows for GA and SA in Table 4 would be equal, and the lines in all the graphs of Figure 1 would be overlapped with the 'Cross Multiplication' line. However, the graphic representation clearly shows that the performance of a processor running a metaheuristics depends on the type of the algorithm as well as on the problem size, and not only in a performance index. Another fact is that only `Dhrystone` 2 was able to rank the processors in the same order than the running time ranking does. We can see how the score of the rest of the benchmarks do not reflect the behavior of the algorithms. At this point, we have to dismiss the rest of the benchmarks, so from now on we are going to use `Dhrystone` 2 as our base benchmark.

6 Fitting the Data

As we have noted before, the execution times depend on the type of algorithm, problem size, and CPU performance, and is not linearly related with `Dhrystone` score. Thereby, we have tried to fit the running times and the `Dhrystone` score into a 3D surface, and we have obtained an equation for each algorithm.

For the task of surface fitting we have used the Open Source project *Pythonequations*[3]. *Pythonequations* is a collection of Python equations that can fit themselves to both 2D and 3D data sets (curve fitting and surface fitting), output source code in several computing languages, and run a genetic algorithm for initial parameter estimation.

With *Pythonequations* we have obtained a ranking of more than 2500 equations fitting our data. Among the best fitting ones, we have chosen a Reciprocal Mixed Inhibition equation. We have made this election because it was the best fitting equation that was consistent with the expected behavior of the processors, with a reasonable number of coefficients. The equation is outlined in Equation 1. After the surface fitting for each algorithms, we obtained the coefficients for GA and SA that are shown in Table 5a. In Table 5b we can see the error statistics of the regression.

$$f(x,y) = \frac{1.0}{\frac{ax}{b\left(1+\frac{y}{c}\right)} + x\left(1+\frac{y}{d}\right)} \tag{1}$$

Table 5. Surface fitting coefficients and errors

(a) Coefficients

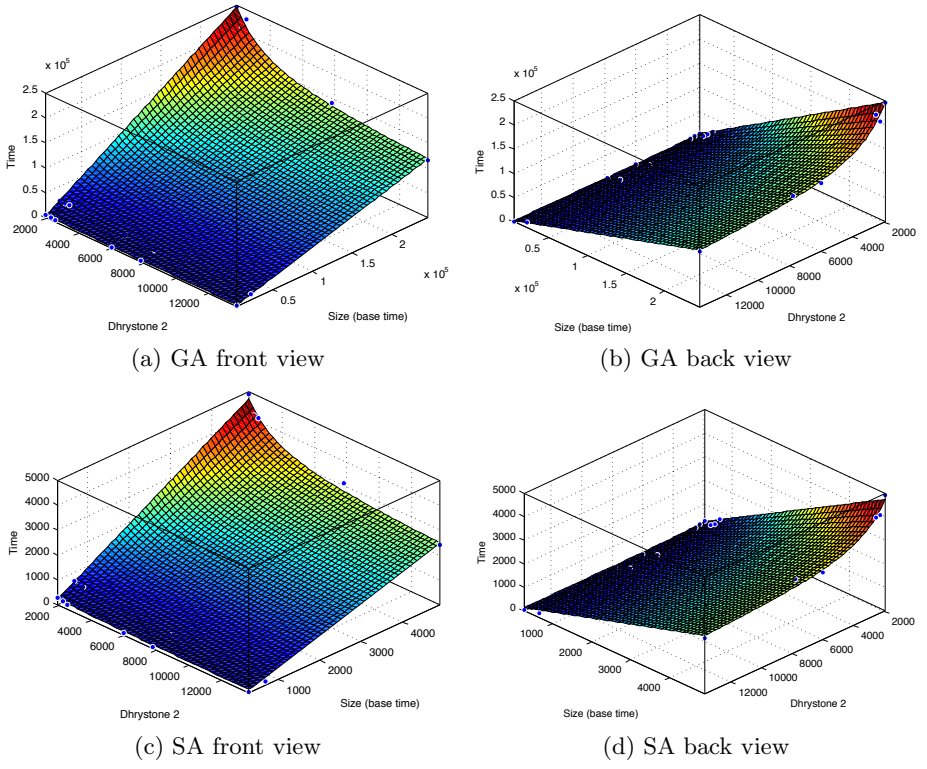
Coeff.	GA	SA
a	0.1401	12.54
b	-4632	-189100
c	-27.6	-14.26
d	18	0.1843

(b) Error statistics

Metric	GA	SA
Sum of squared errors	6.5424e+08	2.6444e+05
R^2	0.9951	0.9945
Adjusted R^2	0.9940	0.9934
Root mean squared error	6836.0	137.4346

In Figure 2 we can see the representation of the equations obtained in the fitting, as well as the real data obtained in our tests. With respect to the `Dhrystone` score (x) we can see how, as we could expect, the value for the time ($f(x,y)$) decreases when the score increases, and dramatically increases when x tends to 0. With respect to the size of the problem or base time, we can see how the time grows almost linearly with it. In the next section we are going to explain how to use the equations for the prediction of running time in a given hardware.

[3] http://code.google.com/p/pythonequations/



(a) GA front view (b) GA back view

(c) SA front view (d) SA back view

Fig. 2. GA and SA fitting (front and back views)

7 Predicting Running Times and Limitations

Let us suppose that a researcher has made a experiment in which he has run a GA algorithm A_1 in a processor P_1, and it has taken 30000ms. The first thing we need to know is the `Dhrystone` score for the processor P_1; we can find a table with results for `Dhrystone` 2 in [7]. Let us assume that the score for $P_1 = 1000$. Once known these two values and the kind of algorithm we can obtain the *baseline* value for our prediction: in Equation 6, we only have to substitute x by the `Dhrystone` value for P_1, and $f(x, y)$ by the value of the run time in P_1. Solving the resulting equation we obtain a value for y; this value will be the baseline for predicting the running time on another processor P_2, in this case $y = 18836$

For predicting the time in an arbitrary processor P_2 we only have to calculate $f(x, y)$ where x is the score for the new processor and y is fixed to the previously obtained baseline value. Fixing y value is like obatining a *slice* of the 3D graph, so that now, the running time only depends on the `Dhrystone` score. In our example, for a processor P_2 with a `Dhrystone` index of 3000, the value for the predicted running time is $f(3000, 18836) = 14983ms$. Now we can compare the obtained time with our algorithm A_2 running in P_2 without disposing the P_1 processor nor the implementation of the A_1 algorithm.

While the quality of the prediction is good for a given implementation and problem, this first approach is limited and further research is necessary to understand the effects of several factors as the language of the implementation or solution encoding. Also an analysis of the frequency of instructions for each family of algorithms (EAs and LSMs) could rely in a more general model, perhaps covering many algorithms of the same family.

8 Conclusions and Further Work

In this paper we have tried to provide a first approach for developing a methodology to fairly compare algorithms running in different hardware, using a scientific benchmark software as a basis for our model. We have analyzed several different options and build a method over the `Dhrystone` 2 benchmark. We have obtained an equation for each one of the studied problems, and have shown an example of their use for the prediction of the execution time.

As we noted before, the method needs to be enhanced, reducing its limitations. Further research is necessary to enhance the model. A next step for future work is the analysis of the frequency of the instructions on different metaheuristics, trying to generalize the method, as well as the study of the effects of the coding and language in the performance differences among different processors.

Acknowledgments. Authors acknowledge funds from the Spanish Ministry MICINN and FEDER under contracts TIN2011-28194 (roadME) and TIN2008-06491-C04-01 (M* `http://mstar.lcc.uma.es`) and CICE, Junta de Andalucía, under contract P07-TIC-03044 (DIRICOM `http://diricom.lcc.uma.es`).

References

1. Cheeseman, P., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In: Proceedings of the IJCAI 1991, pp. 331–337 (1991)
2. Curnow, H.J., Wichmann, B.A.: A synthetic benchmark. The Computer Journal 19(1), 43–49 (1976)
3. Dongarra, J.: Performance of Various Computers Using Standard Linear Algebra Software in a Fortran Environment, `http://netlib.org/benchmark/performance.pdf`
4. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
5. Holland, J.H.: Adaptation in Natural and Artificial Systems, 2nd edn. The MIT Press, Cambridge (1992)
6. Kirkpatrik, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science, 671–680 (1983)
7. Longbottom, R.: Dhrystone 2.1 Benchmark Results On PCs (2011), `http://www.roylongbottom.org.uk/dhrystone%20results.htm`
8. McMahon, F.H.: The Livermore Fortran Kernels: A Computer Test Of The Numerical Performance Range, Lawrence Livermore National Laboratory, Livermore, California, UCRL-53745 (December 1986)
9. Weicker, R.P.: Dhrystone: a synthetic systems programming benchmark. Commun. ACM 27, 10 (1984)

A Variable Neighborhood Search Approach for the Two-Echelon Location-Routing Problem

Martin Schwengerer[1], Sandro Pirkwieser[2], and Günther R. Raidl[1]

[1] Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
martin.schwengerer@chello.at, raidl@ads.tuwien.ac.at
[2] Destion – IT Consulting OG, Vienna, Austria
pirkwieser@destion.at

Abstract. We consider the two-echelon location-routing problem (2E-LRP), a well-known problem in freight distribution arising when establishing a two-level transport system with limited capacities. The problem is a generalization of the $\mathcal{NP}$-hard location routing problem (LRP), involving strategic (location), tactical (allocation) and operational (routing) decisions at the same time. We present a variable neighborhood search (VNS) based on a previous successful VNS for the LRP, accordingly adapted as well as extended. The proposed algorithm provides solutions of high quality in short time, making use of seven different basic neighborhood structures parameterized with different perturbation size leading to a total of 21 specific neighborhood structures. For intensification, two consecutive local search methods are applied, optimizing the transport costs efficiently by considering only recently changed solution parts. Experimental results clearly show that our method is at least competitive regarding runtime and solution quality to other leading approaches, also improving upon several best known solutions.

1 Introduction

We focus on a problem in the field of freight transportation, the Two-Echelon Location Routing Problem (2E-LRP)[1]. This problem is a generalization of the classical Location Routing Problem (LRP), dealing with a two-level distribution system to deliver goods from suppliers to customers as this is beneficial (and sometimes even mandatory) for many real world applications. Basically, the 2E-LRP includes components of classical $\mathcal{NP}$-hard problems like the Facility Location Problem (FLP) and the Vehicle Routing Problem (VRP), in combination with the requirements of an additional echelon. The 2E-LRP arises when selecting locations for two types of facilities (platforms and satellites) through which goods may be transported to final customers. This includes placing two types of facilities at given locations, assigning customers to satellites, satellites to platforms, and to serve these bipartite supply chain by two fleets of vehicles. Additionally, the vehicles as well as the facilities may impose capacity constraints representing e.g. load limits. Possible extensions of the problem may deal with

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 13–24, 2012.

direct routes from the platform to customers, either by a first-level vehicle or by locating vehicles of the second-level fleet at the platforms. In contrast to the FLP, where simple out-and-back routes are used, the 2E-LRP generally allows multiple stops per route. Similar to the LRP, efficiently solving the 2E-LRP usually requires strategic (platform and satellite location), tactical (allocation) as well as operational (routing) planning decisions at the same time. Considering these aspects in a subsequent way might seem beneficial in terms of complexity but unfortunately tends to result in suboptimal solutions compared to all-embracing, nested approaches [2].

From now on, we will refer to the echelon platform/satellite part of the problem as first level and to the satellite/customer part as second level, respectively.

The 2E-LRP can be defined on a complete, undirected, weighted graph $G = (V, E)$, with $V = V_C \cup V_S \cup V_P$ being the set of nodes consisting of n customers $V_C = \{0, \ldots, n-1\}$, m potential satellites $V_S = \{n, \ldots, n+m-1\}$ and o potential platforms $V_P = \{n+m, \ldots, n+m+o-1\}$, and $E = \{\{i, j\} \mid i, j \in V_C,\ i \neq j\} \cup \{\{i, j\} \mid i \in V_C,\ j \in V_S\} \cup \{\{i, j\} \mid i \in V_S,\ j \in V_P\}$ being the set of edges. For any pair of nodes $i, j \in V$ $c_{ij} \geq 0$ represents the given travel cost. Each facility $i \in V_D \cup V_P$ has an associated maximal capacity W_i and opening costs O_i. Furthermore, two homogeneous fleets F_1 and F_2 of K_1 (K_2) vehicles, each having capacity $Q_1 > 0$ ($Q_2 > 0$), are available at the platforms (satellites). The fixed cost of using a single vehicle of F_1 (F_2) is given by $f_1 > 0$ ($f_2 > 0$), and each vehicle is limited to perform one single route. Further, each customer $j \in V_C$ has defined a demand $d_j > 0$ which has to be satisfied. Additionally, we assume that the total capacity of the satellites as well as of the platforms can satisfy the whole demand.

The 2E-LRP then deals with finding a setting of platforms and satellites to be opened as well as determining efficient vehicle routes from the platform to the satellites and from satellites to customers on G such that:

- Each route starts and ends at the same opened facility,
- each customer j is visited exactly once by a vehicle of F_2, satisfying its demand d_j,
- each satellite is visited exactly once by a vehicle of F_1, satisfying its demand $d_{satellite}$ which is the sum of the demands of the assigned customers,
- the total number of vehicle routes of the first (second) level N_1 (N_2) is less than or equal to K_1 (K_2),
- the total load of each route transported by a vehicle of the first (second) fleet does not exceed the vehicle capacity limit Q_1 (Q_2),
- for each opened platform or satellite i the total load of each route assigned to it does not exceed the facility capacity limit W_i,
- and the total costs of opening platforms and satellites, fixed costs for used vehicles, and corresponding travel costs are minimized.

In this work we adapt and extend a a successful Variable Neighborhood Search (VNS) for the LRP [3] in order to account for the additional echelon in the 2E-LRP by modifying existing neighborhood structures, introducing new ones, adapting the local search procedures, and applying some additional improvements.

The remainder is organized as follows. Related work is presented in the next section, the VNS is the topic of Section 3 and experimental results are given in Section 4. Section 5 finishes the work with concluding remarks.

2 Related Work

The 2E-LRP is a special case of the Multi-Echelon Location-Routing Problem (MELRP) having only one level of satellites between platforms and customers. In [4] Gonzalez-Feliu defines a unified notation and a general model for the MELRP and gives a literature review on diverse variants, including the 2E-LRP. A literature research on the 2E-LRP itself only reveals a rather sparse number of publications, most of them appeared in recent years although the problem itself was already described in the 80s by Jacobsen and Madsen [1] in the context of newspaper delivery. Boccia et al. [5] introduce three mixed integer programming models for the 2E-LRP and present corresponding results. The same authors further proposed a tabu search using an *iterative-nested approach*, splitting the problem in two LRPs and solving them in a bottom-up manner [6]. A different heuristic has been proposed by Nguyen et al. [7], considering a slightly modified version of the problem using only a single platform. Their approach is based on a Greedy Randomized Adaptive Search (GRASP) using four different construction heuristics, a learning process and path relinking. In [8] the same authors suggest a multi-start iterated local search with a tabu list and path relinking, which, according to their evaluations, clearly outperforms the former approach. Among these algorithms, one of the currently most effective method has been proposed by Contardo et al. in [9]. It is an Adaptive Large Neighborhood Search (ALNS) applying eight destroy and four repair operations and an embedded local search. An interesting aspect is that none of these present-day heuristics is a population-based approach. To the best of our knowledge, only the contribution by Jin et al. [10] is based on evolutionary principles using a hybrid genetic algorithm with a tabu search, which, however, tackles a two-layer LRP having infinite platform capacities. Unfortunately the authors did not provide results on available or common instances making a meaningful comparison difficult.

As the 2E-LRP consists of locating and routing components, it is closely related to various other problems in freight distribution. For example, the Two-Echelon Vehicle Routing Problem (2E-VRP), which deals with a two-level routing of vehicles, can be seen as a special case of the 2E-LRP, containing no opening costs, always opened satellites and no facility capacities. For the 2E-VRP, Crainic et al. [11] proposed a multi-start heuristic by separately solving the platform-to-satellite and satellite-to-customer subproblems. A different approach by Hemmelmayr et al. uses an ALNS [12] with destroy and repair components and embedded local search methods. Another related problem is the Two-Echelon Single-sourced Capacitated Facility Location Problem, considering two-level facility location and allocation similar to the 2E-LRP, for which Tragantalerngsak et al. proposed six heuristics based on Lagrangian relaxation [13]. A further related problem is the Two-Echelon Uncapacitated Facility Location Problem, in

which satellites and customers can be supplied by multiple sources. Gao and Robinson [14] introduce a dual-based branch and bound algorithm for it. For more details about two-echelon freight transport optimization we refer to [15].

Finally, we remark that the classical LRP can be considered a special case of the 2E-LRP in which the first level is removed, e.g. by using only one platform and zero transport costs to the satellites; two recents works are reported in [3,16]. In fact, the current work is methodically based on the former approach [3], also a VNS, sharing some features with it. Therefore we omit to give details here but refer to the next section.

3 Variable Neighborhood Search for the 2E-LRP

Variable Neighborhood Search [17] is a well-known metaheuristic exploiting a set of (not necessary disjunct) neighborhood structures. Essentially, it applies random steps in neighborhoods of growing size as diversification mechanism, called shaking, each followed by an application of an embedded local search for intensification.

As many other problems in the field of freight distribution, the 2E-LRP is particularly challenging due to some strict constraints resulting in a rugged search space and limiting the amount of possible (legal) solutions for many neighborhoods. Hence to smooth the search space, our VNS relaxes the vehicle load as well as the facility load restrictions by allowing infeasible solutions in combination with penalties for excesses of these constraints. As the same (constant) weightings might be inefficient for the whole search process, we use an *adaptive* penalty function which gradually adjusts both values. Depending on whether the search is in a feasible or infeasible region regarding a constraint, it decreases or increases the corresponding penalty, respectively. We observed that useful weights and initial values often depend on the problem instance. After some experiments, we decided to use the average traveling costs c_{avg} of the second fleet as initial value with $c_{\text{avg}}/5$ ($c_{\text{avg}} \cdot 5$) as lower (upper) bound for the penalty.

Initially, our VNS creates a set of candidate solutions by the following construction heuristic, finally picking the best one for further improvement.

1. Sum up the customer demands to get a lower bound for the accumulated facility capacities of each echelon $W_{\text{LB}} = \sum_{j \in V_C} d_j$.
2. Select satellites to be opened at random one by one until their combined capacity is greater than or equal to W_{LB}.
3. Select platforms to be opened at random one by one until their combined capacity is greater than or equal to W_{LB} (as the satellite demand never exceeds the customer demand).
4. Assign each customer to be visited by a vehicle of its nearest opened satellite, thereby considering the penalties for satellite load violations.
5. Apply the well-known Clarke and Wright savings algorithm [18] until no more routes can be feasibly merged due to the vehicle load restriction.
6. If we end up with more than K_2 second-level routes, least customer routes are selected and customers contained therein are re-added in a greedy way, allowing (penalized) excess of vehicle load.

7. Assign each opened satellite with at least one route to be visited by a first-level vehicle of its nearest opened platform, considering penalties for platform load violations.
8. Apply step 5 in an analogous way on the first-level routes.
9. Similar to the second-level fleet, in case of ending up with more than K_1 first-level routes, least satellite routes are selected and satellites contained therein are re-added in a greedy way.

The reader may notice that our initialization method is rather simple in contrast to others like those suggested in [8,7]. However, a better initial solution does not necessarily lead to a better final result, in fact it might even be counterproductive for the subsequent heuristic search. For example also [9], one of the so far leading approaches, uses a comparable initialization procedure, and similar observations hold for problems like the LRP [3].

For shaking we define seven different types of neighborhood structures via possible moves on a current solution. These are used with several perturbation sizes, denoted by δ, providing a total of 21 shaking neighborhoods (i.e. $k_{\max} = 21$). Note however, that not all of them are always useful. Some of our benchmark instances deal with a single, fixed platform, which renders some neighborhoods rather useless. Our implementation detects such cases automatically and skips the corresponding neighborhoods. In the following these seven basic neighborhood structures are described.

Exchange-Segments. Exchange two random segments of variable length between two routes at the same facility (platform or satellite). This includes also a customer (or satellite) relocation, as one of the segments might be empty. The facility is selected at random with a probability directly (i.e. in a one to one fashion) according to the number of supplied customers or satellites. Since satellites usually serve more customers than platforms serve satellites, the former are selected more often, leading to the desired focus on second-level routes.

Exchange-Segments-Two-Facilities. Exchange two random segments of variable length where both segments are located at two distinct facilities in the same echelon. The selection criterion for the first facility, and hence the type of echelon, is as for *exchange-segments*, whereas the second facility of the same type is then determined using equal probabilities. In case of only one opened platform, always two satellites are selected (if possible) and if only a single satellite is opened, *exchange-segments* is applied. This neighborhood structure facilitates the (re-)assignment of customers (satellites) to the opened satellites (platforms).

Change-Two-Satellites. Opens a currently closed satellite and closes an opened satellite, ensuring that the actual lower bound regarding the satellite capacity W_{LB} is still satisfied. After opening the selected new satellite a relocation procedure is applied which tries to relocate routes in a cost saving manner by removing the old satellite of a route, connecting the first and last customer and placing the new satellite at minimum cost between two consecutive customers. In case no route is relocated, a more investigative method is used, trying to reduce costs by relocating single customers. Afterwards the routes of the satellite to be

Table 1. Order and parametrization of the shaking neighborhoods as used by the VNS

k	$\mathcal{N}_k$
1–5	*Exchange-segments* of maximal length $\delta = k$
6	*Exchange-segments* of maximal lengths bounded by corresponding route size
7–11	*Exchange-segments-two-facilities* of maximal length $\delta = k - 6$
12	*Exchange-segments-two-facilities* of maximal lengths bounded by corresponding route size
13–15	*Change-two-satellites* is applied up to $\delta = k - 12$ times
16–18	*Change-satellite* is applied up to $\delta = k - 15$ times
19–20	*Change-two-platforms* is applied up to $\delta = k - 18$ times
21	*Change-platform* is applied once

closed are relocated to an opened satellite one by one in the least expensive way, taking potential penalties into account. After such a neighborhood move the number of opened/closed satellites stays the same. In case all available satellites are already opened, the following *change-satellite* is applied instead.

Change-Satellite. This neighborhood deals with opening/closing a single satellite. A satellite is selected at random for changing its status (opened to closed or vice versa), thereby taking care of maintaining the required lower bound regarding the satellite capacity W_{LB}. As for *change-two-satellites*, it is tried to relocate customers to a newly opened satellite and shift routes from a closed satellite. This neighborhood move allows to alter the number of opened/closed satellites.

Change-Two-Platforms. Similar to *change-two-satellites*, in this move a closed platform is opened and subsequently a different opened platform is closed, ensuring that the actual lower bound regarding the platform capacity W_{LB} is still satisfied. Again, after opening a platform it is tried to minimize total costs by rerouting complete routes to it. In a subsequent step, the routes of the closed platform are relocated to open platforms one by one in a greedy way, allowing but penalizing excesses of platform capacities. After such a move, the number of opened/closed platforms stays the same. In case all available platforms are already opened, the following *change-platform* is applied instead.

Change-Platform. The platform-specific counterpart of *change-satellite* selects a platform at random and changes its status from opened to closed or vice versa. In case of opening, existing first-level routes are relocated to the newly opened platform in a cost-saving manner while in case of closing, the assigned routes are relocated to other opened platforms. Also here it is ensured that the sum of platform capacities is equal to or greater than W_{LB} by restricting the search space to valid moves, which also may result in infeasible neighboring solutions.

In this work we consider a fixed shaking neighborhood order, detailed in Table 1. A greater focus is laid on exchanging segments and selecting optimal satellites, as these two parts play a major role for obtaining good solutions, while rather drastic changes involving whole platforms occur less often.

For intensification, each newly derived solution is subject to the well-known 3-opt intra-route exchange method, considering only recently changed routes. As a second method, we propose a 2-opt* [19] inter-route exchange local search, which tests for each pair of routes if exchanging their end segments would lead to a better solution. Both methods are applied in a best improvement manner until a local optimum is encountered. Since 2-opt* is considerably more demanding than 3-opt it is primarily applied on new incumbent solutions, but additionally with a probability of 0.2 on newly derived solutions lying within 3% to the current incumbent.

Although this basic VNS already performs relatively well, it seems that besides always accepting better solutions as usual, also considering solutions having worse costs sometimes can boost efficiency. This is done in a systematic way using the Metropolis criterion as was previously done in [20,3], originating from simulated annealing [21]. Similar to the penalty weights we use an instance specific value for the initial temperature $T_0 = c_{\text{avg}}/10$ and apply a linear cooling scheme, decreasing the value after every 100 iterations by $T_0 \cdot 100/i$, with i iterations in total as a limit. In addition, we further enhance this method with a *reheating* procedure which resets the temperature to its initial value in case no improvements occurred over a longer period, taking $i/8$ iterations for it.

4 Experimental Results

Our algorithm has been implemented in C++, compiled with GCC 4.6 and performed on a single core of an Intel Xenon E5540 with 2.53 GHz. For the evaluation we took three benchmark data sets already used in previous work, comprising 147 instances in total. Two sets were proposed by Nguyen [8] and are provided by Prodhon [22]: the first set *Prodhon* is an extension of 30 instances originally created for the LRP by adding a single platform at coordinates $(0, 0)$. The instances are named *n-m-*#clusters[b][BIS], containing 20 to 200 customers n, either 5 or 10 satellites m, up to three customer clusters, and having different vehicle capacities ('b' denotes high) as well as separated clusters (denoted by 'BIS'). The second set *Nguyen*, with a naming schema *n-m-*$\{N, NM\}$), consists of 24 instances having also only one platform, between 25 and 200 customers and 5 or 10 satellites; 'N' denotes a normal distribution while 'NM' a multi-normal distribution. Finally, the third set *Sterle* was generated according to [6] and provided by Sterle [23]. It consists of three (sub-)sets of instances, called I1, I2 and I3, containing different distributions of the satellites. These instances have multiple platforms with restricted capacities and contain 8 to 200 customers, 3 to 20 satellites and 2 to 5 platforms; they are denoted as *I[1|2|3]-o-m-n*.

Our VNS is run for $i = 5 \cdot 10^6$ iterations for sets Prodhon and Nguyen, and for $i = 7.5 \cdot 10^6$ iterations for set Sterle; the difference is due to obtaining comparable runtimes to previous methods to perform a fairer comparison. Initially we create 10 candidate solutions and pick the best one as start solution of the VNS. Penalty weights are adapted by multiplying or dividing them by 1.000005 depending on whether or not the solution at hand is feasible with respect to the

Table 2. Results of the VNS on Prodhon's instances

Instance	BKS	best of 20 runs		average over 20 runs					overall best	
		costs	%-gap	costs	CV [%]	%-gap	t [s]	$t_{\min}$ [s]	costs	%-gap
20-5-1	89075	89075	0.00	89075.00	0.00	0.00	63.46	1.54	89075	0.00
20-5-1b	61863	61863	0.00	61863.00	0.00	0.00	82.84	0.27	61863	0.00
20-5-2	84478	84478	0.00	84489.50	0.06	0.01	62.22	11.33	84478	0.00
20-5-2b	60838	60838	0.00	61033.80	0.68	0.32	125.43	0.00	60838	0.00
50-5-1	130843	130843	0.00	130859.30	0.04	0.01	80.10	15.85	130843	0.00
50-5-1b	101530	101530	0.00	101548.80	0.06	0.02	127.87	34.87	101530	0.00
50-5-2	131825	131825	0.00	131825.00	0.00	0.00	96.71	11.28	131825	0.00
50-5-2b	110332	110332	0.00	110332.00	0.00	0.00	198.21	11.95	110332	0.00
50-5-2BIS	122599	122599	0.00	122599.00	0.00	0.00	111.58	90.66	122599	0.00
50-5-2bBIS	105696	105696	0.00	105935.50	0.14	0.23	197.73	155.29	105696	0.00
50-5-3	128379	128379	0.00	128436.00	0.10	0.04	79.75	9.03	128379	0.00
50-5-3b	104006	104006	0.00	104006.00	0.00	0.00	131.25	6.34	104006	0.00
100-5-1	319137	**318225**	-0.29	**318667.10**	0.10	-0.15	225.73	153.13	**318134**	-0.31
100-5-1b	257349	**256991**	-0.14	257436.35	0.11	0.03	301.11	219.65	**256878**	-0.18
100-5-2	231305	231305	0.00	231340.00	0.02	0.02	203.70	131.04	231305	0.00
100-5-2b	194729	194763	0.02	194812.70	0.02	0.04	240.39	202.27	194729	0.00
100-5-3	244194	244470	0.11	245334.90	0.12	0.47	174.10	123.70	**244071**	-0.05
100-5-3b	194110	195381	0.65	195586.20	0.11	0.76	180.41	111.35	195381	0.65
100-10-1	358068	**352694**	-1.50	**357381.40**	1.02	-0.19	233.37	166.94	**351243**	-1.91
100-10-1b	297167	298186	0.34	300239.15	0.43	1.03	299.03	194.47	297907	0.25
100-10-2	305402	**304507**	-0.29	**304931.20**	0.11	-0.15	247.66	193.78	**304438**	-0.32
100-10-2b	264389	**264092**	-0.11	264592.00	0.12	0.08	307.05	207.54	**263873**	-0.20
100-10-3	313249	**311447**	-0.58	**312701.25**	0.21	-0.17	226.64	141.31	**310312**	-0.94
100-10-3b	264096	**260516**	-1.36	**261577.90**	0.22	-0.95	302.85	217.89	**260328**	-1.43
200-10-1	552816	**548730**	-0.74	**552488.90**	0.45	-0.06	1009.49	748.05	**548703**	-0.74
200-10-1b	448236	**445791**	-0.55	**448095.45**	0.43	-0.03	634.59	575.87	**445301**	-0.65
200-10-2	498199	**497451**	-0.15	513673.40	3.34	3.11	1158.23	832.45	**497451**	-0.15
200-10-2b	423048	**422668**	-0.09	432487.00	2.12	2.23	730.12	695.92	**422668**	-0.09
200-10-3	533732	**527162**	-1.23	**529578.00**	0.30	-0.78	970.42	903.46	**527162**	-1.23
200-10-3b	404284	**402117**	-0.54	404431.25	0.30	0.04	591.90	556.91	**401672**	-0.65
Average			-0.21		0.35	0.20	313.13	224.14		-0.26

corresponding constraint, respectively. This factor might seem quite small, but we opted for a smooth adaptation, which was also confirmed to work well by preliminary test results.

The results for the different instance sets are shown in Tables 2–4. To obtain meaningful results we performed the VNS 20 times per instance. Column BKS states so far *best known solution* values from [9,8], For our VNS we list minimal costs (best of 20 runs), and following average values over 20 runs: costs, corresponding coefficients of variation (i.e. standard deviations divided by average values) in percent (CV [%]), average CPU-times in seconds (t [s]) and average times for obtaining the best solutions in the corresponding runs ($t_{\min}$ [s]). Due to limited space we, on the one hand, give in these tables also the costs of the overall best solutions we obtained, i.e. also including further, differently parameterized test runs, and, on the other hand, state for Sterle's instances in Table 4 only those having ≥ 50 customers or where a new BKS could be obtained; full details are provided via an online supplementary [1]. Results printed bold improve upon the BKS. Furthermore, best, minimum and average costs are also expressed as percentage gaps w.r.t. BKS.

[1] see at http://www.ads.tuwien.ac.at/w/Research/2E-LRP

Table 3. Results of the VNS on Nguyen's instances.

Instance	BKS	best of 20 runs		average over 20 runs					overall best	
		costs	%-gap	costs	CV [%]	%-gap	t [s]	$t_{\min}$ [s]	costs	%-gap
25-5N	80370	80370	0.00	80370.00	0.00	0.00	75.60	2.61	80370	0.00
25-5Nb	64562	64562	0.00	64562.00	0.00	0.00	90.51	0.10	64562	0.00
25-5MN	78947	78947	0.00	78947.00	0.00	0.00	61.15	0.96	78947	0.00
25-5MNb	64438	64438	0.00	64438.00	0.00	0.00	88.52	0.04	64438	0.00
50-5N	137815	137815	0.00	137815.00	0.00	0.00	116.30	35.07	137815	0.00
50-5Nb	110094	110094	0.00	110204.40	0.21	0.10	131.91	51.24	110094	0.00
50-5MN	123484	123484	0.00	123484.00	0.00	0.00	124.94	40.64	123484	0.00
50-5MNb	105401	105401	0.00	105687.00	0.38	0.27	201.51	47.57	105401	0.00
50-10N	115725	115725	0.00	115725.00	0.00	0.00	143.31	23.65	115725	0.00
50-10Nb	87315	87315	0.00	87345.40	0.10	0.03	175.86	66.35	87315	0.00
50-10MN	135519	135519	0.00	135519.00	0.00	0.00	144.06	10.33	135519	0.00
50-10MNb	110613	110613	0.00	110613.00	0.00	0.00	217.73	13.24	110613	0.00
100-5N	193228	193228	0.00	200685.05	3.18	3.86	168.12	101.32	193228	0.00
100-5Nb	158927	164156	3.29	164508.10	0.17	3.51	257.59	144.08	159429	0.32
100-5MN	204682	204682	0.00	206567.40	1.09	0.92	183.99	158.99	204682	0.00
100-5MNb	165744	165744	0.00	166357.35	0.25	0.37	314.87	246.88	165744	0.00
100-10N	212847	**212729**	-0.06	214585.60	0.62	0.82	222.99	167.17	**209952**	-1.36
100-10Nb	155489	155489	0.00	155790.60	0.17	0.19	352.13	251.36	155489	0.00
100-10MN	201275	201275	0.00	203798.05	0.88	1.25	229.30	162.90	201275	0.00
100-10MNb	170625	170625	0.00	170791.25	0.16	0.10	347.26	282.88	170625	0.00
200-10N	347395	**346181**	-0.35	349584.15	0.74	0.63	640.67	525.01	**345267**	-0.61
200-10Nb	256171	256759	0.23	264228.90	1.66	3.15	906.96	790.98	256759	0.23
200-10MN	326454	**325747**	-0.22	332207.50	1.04	1.76	453.32	440.96	**323801**	-0.81
200-10MNb	289742	**289239**	-0.17	292036.65	0.89	0.79	944.14	842.69	**287802**	-0.67
Average			0.11		0.48	0.74	274.70	183.63		-0.12

The results indicate that the VNS performs very well in general, reaching on small instances almost every time the optimal or best known solution (for a list of proven optimal solutions, we refer to [9]). In detail, our approach was able to reach the BKS in all 71 instances with less than 50 customers at least once and could even improve two of them. On the 76 larger instances with $n \geq 50$ customers, the VNS was able to find 35 of the previous BKS and even 30 new, improved solutions. Concerning the average results of our 20 runs per instance, the solution quality is only 0.45% worse than the previously known BKS with an average coefficient of variation of 0.40%. Since larger instances are harder to solve and contain more local optima in general, it is not surprising that solution values vary in these cases to a larger extent. However, it can be observed that the runtime of the VNS increases only moderately with increasing instance size, making the approach attractive for solving larger instances in relatively short time.

Table 5 displays a comparison of the VNS with other state-of-the-art approaches from literature: GRASP+PR [7], MS-ILS+PR [8], and ALNS [9]. As common reference, we now use the currently best known solutions, i.e. including those found by our method. Comparisons among runtimes must be done with care, especially Nguyen et al. used a notably slower machine (Intel Pentium 4 with 3.4 GHz), whereas those utilized by Contardo et al. is slightly faster than ours (Intel Xeon E5472 with 3.0 GHz). Note that for GRASP+PR and

Table 4. Results of the VNS on Sterles's instances; note that here we can only present results for a subset of all instances, but the averaged values are stated for the whole subsets each (lines "Average" and "Total Average").

Instance	BKS	best of 20 runs		average over 20 runs					overall best	
		costs	%-gap	costs	CV [%]	%-gap	t [s]	$t_{\min}$ [s]	costs	%-gap
I1-4-10-25	1607.94	**1559.36**	-3.02	**1565.92**	0.31	-2.61	139.52	12.43	**1559.36**	-3.02
I1-5-8-50	1162.44	1162.44	0.00	1175.67	0.95	1.14	168.54	47.88	1162.44	0.00
I1-5-10-50	1132.63	1132.63	0.00	1139.37	0.54	0.60	189.48	23.10	1132.63	0.00
I1-5-10-75	1540.23	1540.23	0.00	1554.63	0.41	0.94	237.73	135.45	1540.23	0.00
I1-5-15-75	1686.21	1686.21	0.00	1709.52	1.28	1.38	265.80	151.73	1686.21	0.00
I1-5-10-100	2124.09	**2123.44**	-0.03	2152.04	1.34	1.32	353.37	222.23	**2123.44**	-0.03
I1-5-20-100	1973.08	**1971.09**	-0.10	1981.38	0.59	0.42	491.68	287.53	**1971.09**	-0.10
I1-5-10-150	1883.44	1886.9	0.18	1908.02	0.76	1.31	1241.05	1020.53	**1882.6**	-0.04
I1-5-20-150	1869.53	**1841.51**	-1.50	1870.66	1.11	0.06	1358.06	930.26	**1835.30**	-1.83
I1-5-10-200	2443.8	2461.55	0.73	2488.20	0.68	1.82	1994.27	1706.85	2461.55	0.73
I1-5-20-200	2219.54	**2180.01**	-1.78	**2217.33**	0.96	-0.10	1932.42	1586.05	**2180.01**	-1.78
Average			-0.18		0.48	0.36	334.56	212.67		-0.20
I2-5-8-50	1121.13	1121.13	0.00	1122.13	0.27	0.09	175.83	83.10	1121.13	0.00
I2-5-10-50	1256.44	1256.44	0.00	1257.02	0.15	0.05	184.16	53.00	1256.44	0.00
I2-5-10-75	1691.15	1691.15	0.00	1702.61	0.45	0.68	240.27	83.72	1691.15	0.00
I2-5-15-75	1644.79	1743.46	6.00	1760.23	0.51	7.02	281.80	154.31	1743.46	6.00
I2-5-10-100	2231.21	2242.67	0.51	2281.94	0.57	2.27	332.43	232.81	2242.67	0.51
I2-5-20-100	1996.34	1999.61	0.16	2016.66	0.52	1.02	465.89	332.72	1999.61	0.16
I2-5-10-150	1728.05	1728.23	0.01	1746.83	0.90	1.09	951.50	774.49	1728.23	0.01
I2-5-20-150	1630.29	**1615.26**	-0.92	1636.21	0.95	0.36	1106.31	740.71	**1615.26**	-0.92
I2-5-10-200	2147.51	2155.78	0.39	2184.76	1.37	1.73	1522.63	1224.11	2155.78	0.39
I2-5-20-200	2049.01	**2039.67**	-0.46	2086.87	1.26	1.85	1807.87	1537.49	**2035.6**	-0.65
Average			0.18		0.40	0.66	304.07	179.94		-0.06
I3-2-8-25	951.59	**951.56**	0.00	**951.56**	0.00	0.00	120.65	23.55	**951.56**	0.00
I3-5-8-50	1162.44	1162.44	0.00	1174.18	1.03	1.01	163.75	47.81	1162.44	0.00
I3-5-10-50	1207.31	1207.31	0.00	1210.20	0.29	0.24	190.60	55.88	1207.31	0.00
I3-5-10-75	1721.47	1721.47	0.00	1727.94	0.31	0.38	245.00	117.26	1721.47	0.00
I3-5-15-75	1483.14	**1478.92**	-0.28	1486.39	0.62	0.22	256.56	152.54	**1478.92**	-0.28
I3-5-10-100	2178.35	**2177.86**	-0.02	2203.79	1.07	1.17	345.54	250.82	**2177.86**	-0.02
I3-5-20-100	2035.37	2022.55	-0.63	2037.09	0.67	0.08	480.41	336.27	**2022.55**	-0.63
I3-5-10-150	1274.44	1284	0.75	1293.43	0.64	1.49	904.36	611.32	1276.2	0.14
I3-5-20-150	1235.86	1240.77	0.40	1253.21	0.92	1.40	984.91	779.43	1235.86	0.00
I3-5-10-200	1766.46	1768.35	0.11	1788.41	0.79	1.24	1441.66	1213.78	**1765.33**	-0.06
I3-5-20-200	2531.21	**2515.8**	-0.61	2554.57	0.76	0.92	1944.09	1589.40	**2514.28**	-0.67
Average			-0.01		0.27	0.29	287.63	171.57		-0.05
Total Average			0.0		0.39	0.44	308.75	188.06		-0.11

MS-ILS+PR, the authors present only the best solutions, and results on Sterle's set are limited to instances having at least 50 customers (marked by an asterisk in the table). It is clearly shown that the VNS outperforms both, GRASP+PR and MS-ILS+PR, although the comparisons concerning only best solutions should be done with care. Compared to ALNS our VNS seems to be at least competitive with only marginal differences between the best and average values. Especially on Prodhon's instances the VNS provides better results on average while on the other instance sets ALNS achieves slightly better results. Also the last line of Table 5, stating how often the (new) BKS could be obtained by the corresponding method, emphasizes the good performance of the VNS.

Table 5. Comparison of leading approaches for the 2E-LRP; an asterisk marks the limited sets of Sterle (and corresponding results) considered by Nguyen et al.

Set	GRASP+PR		MS-ILS+PR		ALNS			VNS		
	%-gap$_{min}$	t_{avg}	%-gap$_{min}$	t_{avg}	%-gap$_{min}$	%-gap$_{avg}$	t_{avg}	%-gap$_{min}$	%-gap$_{avg}$	t_{avg}
Prodhon	1.79	14.20	0.94	178.30	0.32	0.83	465.82	0.08	0.50	313.13
Nguyen	1.52	19.70	0.71	112.20	0.15	0.48	191.98	0.26	0.89	274.70
I1	-	-	-	-	0.24	0.41	306.76	0.05	0.60	334.56
I2	-	-	-	-	0.25	0.45	331.01	0.25	0.72	304.07
I3	-	-	-	-	0.05	0.22	329.86	0.04	0.35	287.63
I1*	6.24	-	3.39	917.10	0.42	0.96	839.60	0.16	1.31	835.43
I2*	5.84	-	3.06	928.00	0.79	1.39	913.70	0.76	1.81	745.16
I3*	6.42	-	1.76	935.10	0.17	0.66	909.85	0.14	0.98	698.53
Average	-	-	-	-	0.20	0.48	330.03	0.14	0.61	302.82
Average*	3.36	-	1.60	426.73	0.37	0.86	538.26	0.28	1.10	461.64
No. BKS	8*		14*		111			136		

5 Conclusions

In this work, we extended and adapted a previous variable neighborhood search (VNS) initially proposed for the location-routing problem to tackle the two-echelon location-routing problem (2E-LRP) with capacitated vehicles and facilities. Two new types of neighborhood structures were introduced dealing with the additional echelon by opening and closing platforms including vehicle route relocations. Existing neighborhood structures were modified by also considering first-level routes from platforms to satellites. For smoothing the search space, violations of capacity constraints are allowed but penalized in the objective function; corresponding penalty weights are automatically adapted. The VNS sometimes also accepts worse solutions in a simulated annealing fashion, applying reheating if no improvement occurs over some time. Thorough experimental results on available benchmark sets reveal the very promising performance of our method, achieving results that are at least competitive to leading approaches for this problem, both in terms of solution quality and required runtime. Moreover, we were able to improve upon 32 of 147 (21.8%) of the best known solutions.

It might be promising to augment the presented method with further local search heuristics or even suitably combine it with exact, possibly mixed integer programming based approaches in order to achieve further improvements.

References

1. Jacobsen, S.K., Madsen, O.B.G.: A comparative study of heuristics for a two-level routing-location problem. European Journal of Operational Research 5(6), 378–387 (1980)
2. Salhi, S., Rand, G.K.: The effect of ignoring routes when locating depots. European Journal of Operational Research 39(2), 150–156 (1989)
3. Pirkwieser, S., Raidl, G.R.: Variable Neighborhood Search Coupled with ILP-Based Very Large Neighborhood Searches for the (Periodic) Location-Routing Problem. In: Blesa, M.J., Blum, C., Raidl, G., Roli, A., Sampels, M. (eds.) HM 2010. LNCS, vol. 6373, pp. 174–189. Springer, Heidelberg (2010)

4. Gonzalez-Feliu, J.: The n-echelon location routing problem: concepts and methods for tactical and operational planning. Working Papers halshs-00422492, HAL (2009)
5. Boccia, M., Crainic, T.G., Sforza, A., Sterle, C.: Location-routing models for designing a two-echelon freight distribution system. Technical Report CIRRELT-2011-06, University of Montreal (2011)
6. Boccia, M., Crainic, T., Sforza, A., Sterle, C.: A Metaheuristic for a Two Echelon Location-Routing Problem. In: Festa, P. (ed.) SEA 2010. LNCS, vol. 6049, pp. 288–301. Springer, Heidelberg (2010)
7. Nguyen, V.P., Prins, C., Prodhon, C.: Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. European Journal of Operational Research 216(1), 113–126 (2012)
8. Nguyen, V.P., Prins, C., Prodhon, C.: A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. Engineering Applications of Artificial Intelligence 25(1), 56–71 (2011)
9. Contardo, C., Hemmelmayr, V.C., Crainic, T.G.: Lower and upper bounds for the two-echelon capacitated location routing problem. Technical Report CIRRELT-2011-63, University of Montreal (2011)
10. Jin, L., Zhu, Y., Shen, H., Ku, T.: A hybrid genetic algorithm for two-layer location-routing problem. In: 2010 4th International Conference on New Trends in Information Science and Service Science (NISS), pp. 642–645 (2010)
11. Crainic, T.G., Mancini, S., Perboli, G., Tadei, R.: Multi-start heuristics for the two-echelon vehicle routing problem. Technical Report CIRRELT-2010-30, University of Montreal (2010)
12. Hemmelmayr, V.C., Cordeau, J.F., Crainic, T.G.: An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Technical Report CIRRELT-2011-42, University of Montreal (2011)
13. Tragantalerngsak, S., Holt, J., Rönnqvist, M.: Lagrangian heuristics for the two-echelon, single-source, capacitated facility location problem. European Journal of Operational Research 102(3), 611–625 (1997)
14. Gao, L.L., Robinson Jr., E.P.: A dual-based optimization procedure for the two-echelon uncapacitated facility location problem. Naval Research Logistics 39(2), 191–212 (1992)
15. Gonzalez-Feliu, J.: Two-echelon freight transport optimisation: unifying concepts via a systematic review. Post-Print halshs-00569980, HAL (2011)
16. Contardo, C., Cordeau, J.F., Gendron, B.: A grasp + ilp-based metaheuristic for the capacitated location-routing problem. Technical Report CIRRELT-2011-52, University of Montreal (2011)
17. Hansen, P., Mladenović, N., Brimberg, J., Moreno Pérez, J.A.: Variable neighborhood search. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, 2nd edn., pp. 61–86. Springer, Heidelberg (2010)
18. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12(4), 568–581 (1964)
19. Potvin, J.Y., Rousseau, J.M.: An exchange heuristic for routeing problems with time windows. Journal of the Operational Research Society 46, 1433–1446 (1995)
20. Hemmelmayr, V.C., Doerner, K.F., Hartl, R.F.: A variable neighborhood search heuristic for periodic routing problems. European Journal of Operational Research 195(3), 791–802 (2009)
21. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
22. Prodhon, C.: (December 2011), http://prodhonc.free.fr/
23. Sterle, C.: Private communication (2011)

An ILS-Based Metaheuristic for the Stacker Crane Problem

Thais Ávila[1], Ángel Corberán[1], Isaac Plana[2], and José M. Sanchis[3]

[1] Dept. de Estadística e Investigación Operativa, Universitat de València, Spain
{thais.avila,angel.corberan}@uv.es,
[2] Dept. Matemáticas para la Economía y la Empresa, Universitat de València, Spain
isaac.plana@uv.es,
[3] Dept. de Matemática Aplicada, Universidad Politécnica de Valencia, Spain
jmsanchis@mat.upv.es

Abstract. In this paper we propose a metaheuristic algorithm for the Stacker Crane Problem. This is an NP-hard arc routing problem whose name derives from the practical problem of operating a crane. Here we present a formulation and a lower bound for this problem and propose a metaheuristic algorithm based on the combination of a Multi-start and an Iterated Local Search procedures. Computational results on a large set of instances are presented.

Keywords: combinatorial optimization, metaheuristics, directed rural postman problem.

1 Introduction

The Stacker Crane Problem (SCP) basically consists of finding a tour that starts and ends at a given vertex and traverses a set of arcs of a mixed graph with minimum cost. Its name refers to the practical problem of operating a crane. The crane must start from an initial position, perform a set of movements, and return to the initial position. The objective is to schedule the movements of the crane so as to minimize the total tour cost. This problem can be considered an arc routing problem, particularly a special case of the Directed Rural Postman Problem, a Pickup and Delivery Problem, or a special case of the Asymmetric Traveling Salesman Problem (ATSP).

The SCP was proposed by Frederickson, Hecht, and Kim [7]. They defined the SCP as an arc routing problem on a mixed graph $G = (V, E, A)$, where V is the set of vertices, E the set of edges, and A the set of arcs. Vertex v_s represents the depot, i.e. the initial and final position of the crane. Each link (arc or edge) of the graph has an associated non-negative cost. The goal is to find a minimum cost tour, starting and finishing at v_s, which traverses all the arcs in A.

Arc routing problems are routing problems where the service that must be performed is located at the arcs or edges of the graph, unlike vehicle routing problems, where the service is located at the vertices. The most well-known arc routing problems with a single vehicle are the Chinese Postman Problem (CPP)

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 25–36, 2012.

and the Rural Postman Problem (RPP). The CPP was proposed by Guan [8] and its goal is to find a minimum cost tour traversing all the links of a given graph. This problem can be solved in polynomial time if G is a directed or undirected graph, while it is NP-hard if G is a mixed or windy graph. The RPP, proposed by Orloff [15], is a generalization of the CPP in which only a subset of links are required to be traversed. This is an NP-hard problem for any type of graph (Lenstra and Rinooy-Kan [12]). It is a very important problem because it has many real-life applications and it is the arc routing counterpart of the famous TSP.

The SCP can be considered a special case of the RPP defined on a mixed graph in which all the arcs must be traversed, while the traversal of the edges is optional. Frederickson et al. [7] showed that the SCP is NP-hard, by proving that any instance of the TSP can be transformed into an SCP one. They also proposed a heuristic algorithm with a worst-case performance ratio of 9/5 and $O(n^3)$ complexity. This procedure consists of two algorithms, "LARGEARCS", which works better if the cost of the arcs is large compared to the cost of the edges of an optimal solution, and "SMALLARCS", which works better in the opposite situation. The first one computes a minimum-cost matching and then a minimum-cost spanning tree. The second one is based on the transformation of the SCP instance into a TSP one and then using Christofides' algorithm ([4]) for the TSP. Both procedures need the costs of the instances to satisfy the triangle inequality.

Berbeglia et al. [3] presented the SCP as a pickup and delivery problem. This is an important class of vehicle routing problems in which commodities or people have to be transported from origins to destinations. They have been the object of study in recent years because of their many applications in logistics, ambulatory services, and robotics. Berbeglia et al. [3] defined the SCP as follows: single objects have to be transported from their origin to their destination using a unit capacity vehicle.

Other papers dealing with the SCP can be found in the literature. Eiselt et al. [6] presented a survey on the RPP and devote a section to the SCP. Zhang [18] proposed a simplification of Frederickson's algorithm with a worst-case ratio of 2 and $O(n^2)$ complexity. Zhang and Zheng [19] solved the SCP as an Asymmetric Traveling Problem (ATSP) using previously known algorithms for this problem. Hassin and Khuller [9] proposed a $\frac{1}{2}$ z-approximation algorithm for the directed TSP that can also be used for the SCP. Recently, Srour and van de Velde [17] have presented a statistical study comparing the difficulty of the resolution of SCP instances with that of general ATSP instances. The motivation for this study is that some authors claim that, although the SCP and the ATSP are known to be NP-hard, the SCP can be easily solved when transformed into an ATSP (Laporte [11]). Their conclusion is that SCPs are not necessarily easier than other ATSPs, but a special subset of SCPs, termed drayage problems, are more readily solved. Cirasella et al. [5] present a study about ATSP heuristics on eleven types of instances, one of which corresponds to the SCP.

The rest of the paper is organized as follows. In Section 2 we define the problem and introduce the notation used. A formulation and a lower bound for the SCP are proposed in Section 3. In Section 4 we present a metaheuristic algorithm based on the combination of a Multi-start and an Iterated Local Search algorithms. Computational results on a large set of instances from the Literature are reported in Section 5. Finally, some conclusions are drawn in Section 6.

2 SCP Definition and Notation

Let $G = (V, E, A)$ be a mixed graph, where V is the set of vertices, E is the set of edges, and A is the set of arcs. Associated with each arc and edge there is a nonnegative cost c_{ij}, and for every arc there is a parallel edge of no greater cost. The Stacker Crane Problem consists of finding a tour starting and finishing at the depot, vertex v_s, traversing each arc in A, and such that the cost of the tour is minimum.

The heuristic algorithms proposed by Frederickson et al. [7] require that each vertex is incident to at least one arc of A, and the edge costs satisfy the triangle inequality. As it was pointed out in [6], note that if G does not satisfy these two properties, it can be transformed into an equivalent graph G' that satisfies them. The problem is then solved on G' and the solution transformed into an equivalent solution in G. In what follows we will also assume that G satisfy these conditions.

We use the following notation:

- h_i and t_i represent the initial and final vertices of arc a_i.
- The vertices of the graph are denoted by v_i, while the arcs and edges are represented by $a_i =< t_i, h_i >$ and $e = (v_i, v_j)$, respectively.
- The cost of the shortest path from arc a_i to arc a_j is represented by $d(a_i, a_j)$ and is calculated as the shortest path from h_i to t_j. Note that $d(a_i, a_j)$ and $d(a_j, a_i)$ can be different.
- p denotes the number of connected components induced by the arcs. $V_1, \ldots, V_p$ are the corresponding vertex sets (R-sets), where $V_1 \cup ... \cup V_p = V$.
- Given a subset of vertices $S \subset V$, we define $x^+(S) = \sum_{v_i \in S, v_j \in V \setminus S} x_{ij}$ and $x^-(S) = \sum_{v_i \in S, v_j \in V \setminus S} x_{ji}$.
- For any vertex $v_i \in V$, $d^+(v_i)$ and $d^-(v_i)$ represent the number of arcs leaving and entering v_i, respectively.

A **tour for the SCP** is a closed walk starting and ending at v_s that traverses exactly once each arc in A. A **semitour for the SCP** is the subset of edges obtained after removing the arcs from any tour for the SCP. Figure 1 shows a feasible solution for the SCP, where black arrows represent the arcs in A.

3 A Lower Bound to the SCP

In this section we present a formulation for the Stacker Crane Problem. We associate two variables x_{ij} and x_{ji} with each edge $e = (v_i, v_j)$, representing the

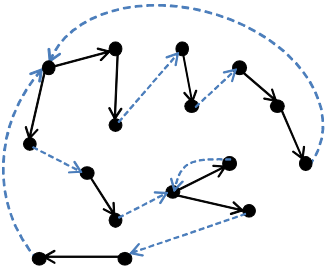


Fig. 1. A SCP solution

number of times edge $e = (v_i, v_j)$ is traversed from v_i to v_j and from v_j to v_i, respectively. Note that a tour for the SCP corresponds to a connected Eulerian graph. Then, the SCP can be formulated as follows:

$$\text{Minimize} \qquad \sum_{(v_i,v_j)\in E} c_{ij}(x_{ij} + x_{ji})$$

s.t.:

$$x^+(S) \geq 1, \quad \forall S = \bigcup_{i\in Q} V_i, \quad Q \subset \{1, 2, ..., p\} \tag{1}$$

$$x^+(v_i) + d^+(v_i) = x^-(v_i) + d^-(v_i), \quad \forall v_i \in V \tag{2}$$

$$x_{ij}, x_{ji} \geq 0, \quad \forall (v_i, v_j) \in E \tag{3}$$

$$x_{ij}, x_{ji} \quad integer, \quad \forall (v_i, v_j) \in E \tag{4}$$

Given that the R-sets are connected subgraphs, constraints (1), which guarantee the connectivity between them, force the solution to be connected. Constraints (2) are the symmetry conditions on the vertices and, together with (1), guarantee that the graph associated with any feasible solution will be strongly connected. Vectors $x \in \mathbb{R}^{2|E|}$ satisfying (1) to (4) correspond to the semitours for the SCP.

Note that the number of constraints of type (1) is exponential. Although they can be exactly separated in polynomial time, the separation algorithm requires max-flow computations. In order to obtain a lower bound easily, we have decided to relax the above formulation by including only one connectivity constraint associated with each R-set:

$$x^+(V_i) \geq 1, \quad \forall i \in \{1, \ldots, p\}.$$

The optimal integer solution to the above relaxation gives a lower bound for the SCP.

4 A Metaheuristic Algorithm for the SCP

In this section we present an algorithm for the SCP. The components of this procedure are a Multi-Start algorithm (MS), a Variable Neighborhood Descent

(VND), and an Iterated Local Search (ILS). It is based on the algorithms proposed by Belenguer et al. [1] for the Split Delivery Capacitated Arc Routing Problem and by Benavent et al. [2] for the Min-Max k-Vehicles Windy Rural Postman Problem. The basic structure of the procedure is as follows. The MS algorithm generates a feasible solution for the SCP and the VND improves it. The improved solution, if it is different from the previously generated solutions, will be considered as the initial solution for the ILS algorithm. This last algorithm disrupts a solution and then repairs it, generating a new solution that is improved by means of the VND. The best solution found is used as starting solution for the next iteration of the ILS. The output of the algorithm is the best solution found.

In what follows we briefly describe the main components of the proposed procedure.

4.1 Multi-start Algorithm

Basically, multi-start algorithms consist of the execution of a number of global iterations until some stopping criterium is satisfied. Each global iteration generates a solution with a constructive algorithm and then improves it with a local search method.

We have implemented a simple constructive algorithm to obtain SCP solutions whose details are shown in Algorithm 1. First an arc is randomly chosen. Let a be the last arc added to the partial tour. We randomly choose an arc from the set consisting of the *nclose* arcs closest to a, according to distance d defined in Section 2, which have not yet been inserted in the partial solution. The selected arc is inserted at the end of the partial solution. The algorithm ends when all the arcs have been inserted. After some preliminary computational testing we fixed *nclose* to five. From the ordering of the arcs obtained, we build an SCP tour by adding the edges in the shortest paths joining consecutive arcs (see Figure 2). Note that, since we are assuming that each vertex is incident with at least one arc, the constructed tour will visit the depot.

The random elements of this constructive algorithm allow us to generate different feasible solutions that will be used as starting solutions.

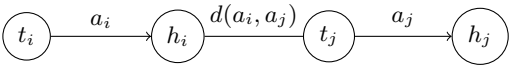


Fig. 2. Tour completion

4.2 Variable Neighborhood Descent Algorithm

The Variable Neighborhood Descent (VND), introduced by Mladenovic and Hansen [14], is an enhanced local improvement strategy based on a sequence $(N_1, ..., N_K)$ of K neighborhoods with growing cardinals. Starting from $k = 1$,

Algorithm 1. SCPSolGenerator

Input: A mixed graph $G = (V, E, A)$.
Output: A tour T for the SCP.
1: Choose a random arc a_0 in A.
2: $T := \{a_0\}$ and $A := A \setminus \{a_0\}$.
3: **while** $A \neq \emptyset$ **do**
4: Calculate $S(a_0)$, the set containing the *nclose* arcs closest to a_0.
5: Choose a random arc a_j in $S(a_0)$.
6: $T := T \cup \{a_j\}$ and $A := A \setminus \{a_j\}$.
7: $a_0 := a_j$.
8: **end while**
9: Complete T with the shortest paths between arcs.
10: **return** T

each VND iteration searches the neighborhood N_k. If one improving move is detected, it is executed and k is reset to 1, otherwise k is incremented. The method stops when $k = K$ and the exploration of N_K brings no improvement.

In our algorithm, we use $K = 3$ neighborhoods, associated with the three procedures described below.

Let T be a given sequence of arcs corresponding to a feasible solution that will be improved by the VND. The movements defining the neighborhoods of the VND are:

1-opt: For each arc in T, we try to insert it in a different position in T, so that the cost of the solution associated with the new sequence is improved. This procedure is applied until no further improvement is obtained.

2-opt: Given a couple of consecutive arcs in T, we try to insert it in a different position in T, so that the cost of the solution associated with the new sequence is improved. This procedure is applied until no further improvement is obtained.

3-opt: Same as 2-opt but with subsets of three consecutive arcs in T.

4.3 Iterated Local Search

ILS (Lourenço et al. [13]) is a metaheuristic that uses an initial solution, a local search and a perturbation procedure. Its structure is as follows. Given an initial solution, it is improved by local search. Then, the algorithm performs $niterILS$ iterations. At each iteration, a copy of the current best solution, called $BestSol$ is randomly modified using a perturbation procedure. The resulting solution is also improved by local search and the best solution is updated.

The same VND described above is used as local search procedure. The proposed perturbation algorithm works as follows. Again, let T be a given sequence of arcs corresponding to a feasible solution. First, *ndel* random arcs are removed from T. Each one of these arcs is reinserted in T in the best possible position making sure that the obtained solution is different from the previous one. Preliminary computational experiments with several values of *ndel* lead us to fix its value to five. A detailed description of this algorithm is shown in Algorithm 2. The pseudo-code of the whole metaheuristic algorithm is presented in Algorithm 3.

Algorithm 2. Perturbation

Input: $T = \{a_{i_1}, ..., a_{i_n}\}$.
Output: T', a different sequence of arcs.
1: Let $S \subset T$ be a set of *ndel* random arcs.
2: $T' := T \setminus S$.
3: **while** $S \neq \emptyset$ **do**
4: Choose $a_k \in S$ at random.
5: Choose $a_{i_j} = \underset{a_{i_j} \in T'}{argmin}\{d(a_{i_j}, a_k) + d(a_k, a_{i_{j+1}})\}$, so that a_k was not between a_{i_j} and $a_{i_{j+1}}$ in T.
6: Insert a_k in T' between a_{i_j} and $a_{i_{j+1}}$. $S := S \setminus \{a_k\}$.
7: **end while**
8: **return** T'

Algorithm 3. Metaheuristic

Input: A mixed graph $G = (V, E, A)$ where A is the set of required arcs.
Output: A tour T for the SCP.
1: **for** $iterMS = 1$ **to** *nsol* **do**
2: **SCPSolGenerator**(SCPsol)
3: **VND**(SCPsol)
4: *BestSol* :=SCPsol
5: **for** $iterILS = 1$ **to** $niterILS$ **do**
6: $ILSsol :=$ **Perturbation**($BestSol$)
7: **VND**($ILSsol$)
8: **UpdateBestSolution**($ILSsol$,$BestSol$)
9: **end for**
10: **UpdateBestSolution**($BestSol$,$GlobalBestSol$)
11: **end for**
12: **return** $GlobalBestSol$

5 Computational Results

In order to study the behavior of the proposed metaheuristic, it has been applied to several SCP instances generated from 92 instances for the Rural Postman Problem (RPP) proposed in Hertz et al. [10]. In these RPP instances, vertices are points in the Euclidean plane and edge costs are defined by the Euclidean distances. The set of edges E is defined according to three different methods. The first method is used in 20 instances (Hertzr class), where E is generated randomly in the plane. The second method is used in 36 instances (Hertzg class), where E defines a uniform grid. Finally, in the other 36 instances (Hertzd class), E defines a graph where all the vertices have degree 4. From these RPP instances, we have generated instances for the SCP. The original RPP instances have been modified in the following way. A random direction has been assigned to the required edges to define the set A of required arcs. Moreover, the instances have been transformed so that all its vertices are incident with required arcs. The transformation consists of removing all the vertices not incident with required arcs and adding edges corresponding to the shortest paths between the remaining

vertices. The obtained SCP instances correspond to mixed graphs where $5 \leqslant |V| \leqslant 100$, $3 \leqslant |A| \leqslant 116$, and $5 \leqslant |E| \leqslant 423$. The algorithm has been coded in C++ and run on an Intel Core 2 Duo computer at 2.4 GHz.

Tables 1 and 2 show the characteristics of the instances and the computational results, respectively. Columns $|V|$, $|E|$, and $|A|$ show, respectively, the average number of vertices, edges and arcs in each instance class. Column "# of inst." gives the number of instances in each class, while "p" shows the average number of R-sets induced by the required arcs.

The metaheuristic has been run with different values of parameters $nsol$, number of global iterations, and $niterILS$, number of ILS iterations. We have tried the combinations $\{100, 0\}$, $\{50, 15\}$, $\{25, 25\}$, and $\{10, 70\}$, as different values for parameters $\{nsol, niterILS\}$. The obtained results are shown in Table 2. In this table, "# of Best" shows the number of times a given combination of parameters gets the best solution found. "Gap(Best)" and "Gap(LB)" show the average gaps of the results obtained with any combination of the parameters with respect to the best solution found and to the lower bound, respectively. Finally, average computing times (in seconds) are given in "Time". The results have been grouped depending on the number of arcs, $|A|$, of the instance.

It can be seen from Table 2 that the best combinations for parameters $\{nsol, niterILS\}$ are $\{50, 15\}$ and $\{25, 25\}$ for the large size instances, while in the small and medium size instances, the behavior of all the variants is indistinguishable. The version with $\{50, 15\}$ is slightly better for the Hertzd class instances, while $\{25, 25\}$ seems to be the better one in the Hertzg class ones. We can observe that the deviations from the lower bound, except for the instances of the Hertzr class, where the costs of the arcs and edges are very large, are small, specially considering that these lower bounds may still be far from the optimal values.

In order to compare our algorithm with previously published methods, we have tested it on two sets of SCP instances presented in [17]. The first set of instances, named PK, were derived from a drayage problem from a Dutch Logistics Service Provider at the Port of Rotterdam. These instances were used in [16] and [17], where they were transformed into Asymmetric Traveling Salesman Problem instances and solved to optimality by means of specific exact algorithms.

Table 1. Characteristics of the instances

		# of inst.	$\|V\|$	$\|A\|$	$\|E\|$	p
Hertzd class	$\|A\| \leq 20$	13	14.9	10.8	36.7	5.1
	$20 < \|A\| \leq 40$	8	35.5	30.0	124.6	8,3
	$\|A\| > 40$	15	74.2	74.4	298.6	11.8
Hertzg class	$\|A\| \leq 20$	12	13.5	8.6	19.6	5.2
	$20 < \|A\| \leq 50$	15	44.0	35.2	82.2	10.6
	$\|A\| > 50$	9	82.4	84.0	160.1	8.4
Hertzr class	$\|A\| \leq 10$	10	10.7	6.6	28.1	4.1
	$\|A\| > 10$	10	22.9	15.7	106.7	7.6

Table 2. Computational results on Hertz instances

		Hertzd class			Hertzg class			Hertzr class	
	$\|A\|$	$[3, 20]$	$(20, 40]$	$(40, 121]$	$[3, 20]$	$(20, 50]$	$(50, 109]$	$[3, 10]$	$(10, 19]$
$\{100, 0\}$	# of Best	13	7	7	12	9	2	9	10
	Gap(Best)	0.00	0.01	0.44	0.00	0.00	2.39	0.00	0.00
	Gap(LB)	8.73	2.48	4.92	2.19	1.29	5.36	7.92	14.26
	Time	0.26	2.94	76.32	0.08	1.55	37.38	0.04	0.47
$\{50, 15\}$	# of Best	13	8	10	12	9	10	9	10
	Gap(Best)	0.00	0.00	0.25	0.00	0.00	0.37	0.00	0.00
	Gap(LB)	8.73	2.47	4.72	2.19	1.29	3.29	7.92	14.26
	Time	0.78	8.68	110.61	0.32	5.80	71.75	0.06	1.54
$\{25, 25\}$	# of Best	13	8	6	12	9	12	9	10
	Gap(Best)	0.00	0.00	0.63	0.00	0.00	0.28	0.00	0.00
	Gap(LB)	8.73	2.47	5.12	2.19	1.29	3.18	7.92	14.26
	Time	0.56	6.53	78.24	0.23	4.46	53.00	0.04	1.25
$\{10, 70\}$	# of Best	13	8	1	12	9	7	9	10
	Gap(Best)	0.00	0.00	0.90	0.00	0.00	1.12	0.00	0.00
	Gap(LB)	8.73	2.47	5.39	2.19	1.29	4.05	7.92	14.26
	Time	0.59	6.74	73.27	0.26	4.75	52.76	0.05	1.42

All these instances have 131 arcs, but their exact characteristics are unknown to us, since we have directly worked with the ATSP transformed ones. The second set of instances, named *Crane*, was used in [5] and [17]. The instances were randomly generated by using a generator with a single parameter $u \geq 1$ that constructs each source-destination pair as follows. The source is chosen randomly in a 10^6 by 10^6 square. Then two integers x and y are chosen in the interval $[-10^6/u, 10^6/u]$ at random and the destination is determined adding the vector (x, y) to the source. Ten instances with $|A| = 100$, named *crane*0 to *crane*9, and ten with $|A| = 316$, *crane*10 to *crane*19, were generated using parameter $u = 10$.

Tables 3 and 4 show the results obtained with our metaheuristic on *PK* and *Crane* instances respectively, using the combination of parameters $\{50, 15\}$ and a time limit of 900 seconds. Column *Heuristic* reports the cost of the solution provided by the proposed algorithm, while *Optimum* gives the optimal cost. The computing times, in seconds, of the metaheuristic procedure are given in column *Time*. Finally, column *Gap* presents the percentage deviation of the metaheuristic solution from the optimal one. We can observe that the behavior of our algorithm is good in the *PK* set of instances, finding nine out of 33 optimal solutions. The average gap for all the *PK* instances is 1.24%, with an average time of 240.2 seconds. On the other hand, the *Crane* instances, as previously stated in [17], are considerably more difficult than the *PK* ones. Moreover, there is a clear difference between those instances with 100 arcs and those with 316 in what refers to the behavior of our algorithm. While in the first subset the average gap and time are 4.19% and 231.47 seconds, respectively, for the second

Table 3. Computational results on PK instances

Name	Heuristic	Time	Optimum	Gap	Name	Heuristic	Time	Optimum	Gap
PK1	65005	239.30	63484	2.40	PK18	72336	234.76	69367	4.28
PK2	64786	229.13	63843	1.48	PK19	48981	264.48	48981	0.00
PK3	52015	233.78	50139	3.74	PK20	51145	245.79	51145	0.00
PK4	51712	236.84	51712	0.00	PK21	51477	243.64	50474	1.99
PK5	51092	217.43	50967	0.25	PK22	70582	245.93	68288	3.36
PK6	60349	237.90	57822	4.37	PK23	50282	266.71	49345	1.90
PK7	63222	256.30	62497	1.16	PK24	68105	248.22	67556	0,81
PK8	48602	214.23	48601	0.00	PK25	56265	259.36	56153	0.20
PK9	60329	229.47	59742	0.98	PK26	52480	237.82	52062	0.80
PK10	58983	249.42	58046	1.61	PK27	68756	239.32	68260	0.73
PK11	63007	227.57	62010	1.61	PK28	80588	251.09	80499	0.11
PK12	67247	232.86	67247	0.00	PK29	54009	244.56	52175	3.52
PK13	51027	230.19	51027	0.00	PK30	48807	215.40	48807	0.00
PK14	67180	250.64	66937	0.36	PK31	38164	263.73	37893	0.72
PK15	52720	253.11	51943	1.50	PK32	72517	244.03	72517	0.00
PK16	52798	222.19	51729	2.07	PK33	47331	223.82	47331	0.00
PK17	64580	237.43	64051	0.83					

Table 4. Computational results on Crane instances

Name	Heuristic	Time	Optimum	Gap	Name	Heuristic	Time	Optimum	Gap
*crane*0	8110760	234.39	7777997	4.28	*crane*10	15716000	900	13132907	19.67
*crane*1	7893750	242.15	7615069	3.66	*crane*11	14931500	900	13194492	13.16
*crane*2	8288450	237.46	8062054	2.81	*crane*12	14714300	900	12968098	13.47
*crane*3	7402210	242.53	7018782	5.46	*crane*13	14520000	900	12350138	17.57
*crane*4	8074990	242.47	7786309	3.71	*crane*14	14575400	900	12744226	14.37
*crane*5	8233300	235.59	7933180	3.78	*crane*15	15606200	900	13120938	18.94
*crane*6	7718400	241.78	7208062	7.08	*crane*16	15464800	900	12900450	19.88
*crane*7	7658470	239.29	7474720	2.46	*crane*17	15373200	900	13145193	16.95
*crane*8	8021430	232.68	7676198	4.50	*crane*18	15063100	900	13289851	13.34
*crane*9	7974160	237.83	7561872	5.45	*crane*19	15372400	900	13164800	16.77

subset the time limit is always reached and we obtain an average gap of 12.93%. Overall, we think the results obtained by our algorithm in medium-size instances are encouraging, while it seems that there is still room for improvement regarding its behavior in larger and more difficult instances.

6 Conclusions

In this paper we have studied the Stacker Crane Problem, which is an NP-hard arc routing problem with practical applications. For this problem we have

proposed an Integer Linear Programming formulation from which a lower bound can be obtained. Moreover, we have presented a metaheuristic algorithm based on the combination of a Multi-start and an Iterated Local Search procedures. The computational results obtained on a large set of instances from the literature show that, although the behavior of this algorithm is promising in medium-size instances, there is still room for improvement regarding the resolution of larger and more difficult ones.

Acknowledgements. The authors wish to thank the Ministerio de Innovación y Ciencia of Spain (project MTM2010-19576-C02-02) and the Generalitat Valenciana (grant VALi+d for young researchers) for their support.

References

1. Belenguer, J.M., Benavent, E., Labadi, N., Prins, C., Reghioui, M.: Split Delivery Capacitated Arc Routing Problem: Lower Bound and Metaheuristic. Transportation Science 44, 206–220 (2010)
2. Benavent, E., Corberán, A., Sanchis, J.M.: A metaheuristic for the min−max windy rural postman problem with K vehicles. Computational Management Science 7, 269–287 (2010)
3. Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G.: Static pickup and delivery problems: a classification scheme and survey. TOP 15, 1–31 (2007)
4. Christofides, N.: Worst-case analysis of a new heuristic for the traveling salesman problem. Graduate School of Industrial Administration, Carnegie Mellon University (1976)
5. Cirasella, J., Johnson, D.S., McGeoch, L.A., Zhang, W.: The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests. In: Buchsbaum, A.L., Snoeyink, J. (eds.) ALENEX 2001. LNCS, vol. 2153, pp. 32–59. Springer, Heidelberg (2001)
6. Eiselt, H.A., Gendreau, M., Laporte, G.: Arc Routing Problems, Part II: The Rural Postman Problem. Operations Research 43, 399–414 (1995)
7. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Aproximation Algorithms for some routing problems. SIAM Journal on Computing 7, 178–193 (1978)
8. Guan, M.: Graphic programming using odd or even points. Chinese Mathematics 1, 237–277 (1962)
9. Hassin, R., Khuller, S.: z-Approximations. Journal of Algorithms 41, 429–442 (2001)
10. Hertz, A., Laporte, G., Nachen-Hugo, P.: Improvement procedures for the undirected rural postman problem. INFORMS Journal on Computing 11, 53–62 (1999)
11. Laporte, G.: Modeling and solving several classes of arc routing problems as traveling salesman problems. Computers & Operations Research 24, 1057–1061 (1997)
12. Lenstra, J.K., Rinnooy Kan, A.H.G.: On general routing problem. Networks 6, 273–280 (1976)
13. Lourenço, H.R., Martin, O., Stützle, T.: Iterated Local Search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 321–353 (2002)
14. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research 24, 1097–1100 (1997)
15. Orloff, C.S.: A fundamental problem in vehicle routing. Networks 4, 35–64 (1974)

16. Srour, F.J.: Dissecting drayage: an examination if structure, information, and control in drayage operations. ERIM Ph.D. Series reseaarch in management, Erasmus Research Institute in Management, 1786 (2010) ISBN978-90-5892-226-7
17. Srour, F.J., van de Velde, S.: Are stacker crane problems easy? A statistical study. Computers & Operations Research (2011) doi:10.1016/j.cor.2011.06.017
18. Zhang, L.: Simple Heuristics for some variants on the traveling salesman problem. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1175–1178 (1992)
19. Zhang, L., Zheng, W.: Genetic coding for solving both the stacker crane problem and its k-variant. In: IEEE International Conference on Systems, Man and Cybernetics 1995, pp. 1061–1066 (1995)

An NSGA-II Algorithm for the Green Vehicle Routing Problem

Jaber Jemai*, Manel Zekri, and Khaled Mellouli

[1] IS Department, College of Computer and Information Sciences,
Imam Mohammad Ibn Saud University. Riyadh, KSA
[2] Larodec Laboratory, Institut Supérieur de Gestion
University of Tunis. Tunis, Tunisia

Abstract. In this paper, we present and define the bi-objective Green Vehicle Routing Problem GVRP in the context of green logistics. The bi-objective GVRP states for the problem of finding routes for vehicles to serve a set of customers while minimizing the total traveled distance and the co_2 emissions. We review emission factors and techniques employed to estimate co_2 emissions and integrate them into the GVRP definition and model. We apply the NSGA-II evolutionary algorithm to solve GVRP benchmarks and perform statistical analysis to evaluate and validate the obtained results. The results show that the algorithm obtain good results and prove the explicit interest grant to emission minimization objective.

Keywords: Green vehicle routing, Multi-objective optimization, Evolutionary algorithms, NSGA-II.

1 Introduction

A supply chain is a network [1] of suppliers, manufacturers, warehouses and distribution channels organized to acquire materials, convert them into finished products and distribute them to clients. The Supply Chain Management (SCM) consists of finding best practices, policies and strategies to solve efficiently all encountered problems. That is by employing the available resources with respect to different constraints and while optimizing many different and generally conflicting objectives. One of the most important SCM phases is the logistics and transportation processes that allow the moving of different materials from and to different nodes in the supply chain network. Generally, the objective of the logistics process is to optimize transportation related costs like traveled distance, time, routes flexibility and reliability. Recently, the concept of greenness for sustainable development has emerged to represent a human concern for the undesirable effect of the industrial processes on the environment. This environmental awareness intend to show the effect of toxic emissions on the environment

* Corresponding author.

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 37–48, 2012.

and to call governments and industrials to seriously consider this concern. Several industries started enhancing their procedures to show an explicit interest to minimize the volumes of their missions. In transportation, the aim is to construct low cost routes for vehicles, trucks, planes and ships to transport goods. However, while moving theses engines generate huge quantities of co_2 that affect directly the quality of breathed air particularly in large cities. The major concern, for transportation firms, is the materiel benefit without reviewing vehicle emissions and their effect on the environment. Recently, and for many reasons, transportation companies start taking explicitly into account the emissions reduction objective in definition of their working plans. This trend was encouraged by governmental regulations and customer preference to consume environment friendly products. Then, the generated working plans must minimize costs and co_2 emissions. These two objective are not necessarily positively correlated and for some cases they are completely conflicting.

The basic transportation model generally used to represent the problem of finding routes for vehicles to serve a set of customers is the Vehicle Routing Problem (VRP) [27]. In the basic VRP and also in many other variants the objective to optimize is unique and it is to minimize the overall transportation costs in term of distance, time, number of vehicles, etc. Here, the literature is really huge where several single objective VRP was studied and solved efficiently. However, like other optimization problems, the objectives may be multiple and conflicting. Then, the multi-objective VRP was defined to represent a class of multi-objective optimization problem.

In this paper, the scope is the study and the definition of the bi-objective Green Vehicle Routing Problem (GVRP). The bi-objective GVRP asks for designing vehicle routes to serve set of customers while minimizing the total traveled distance and the total co_2 emissions with respect to classical routing constraints mainly capacity constraints. consequently, we will implement the NSGA-II evolutionary algorithm to solve the bi-objective GVRP model via solving some well known benchmarks. The NSGA-II is a non-dominating sorting genetic algorithm that solves non-convex and non-smooth multi-objective optimization problems. The objective of the paper is to show the effectiveness of explicitly considering emissions minimization as separate objective to optimize and to prove that short routes are not necessarily less pollutant.

The paper is organized as follow. In the next section, we present the concept of green logistics, enumerate all emission factors and how co_2 emissions could be estimated and then integrated into quantitative models. Section 3 is devoted to define the vehicle routing problem with emissions, review the corresponding literature and propose a mathematical model for the bi-objective GVRP. In the section 4, we present the evolutionary solving approach based on the NSGA-II algorithm. Section 5 will report the NSGA-II implementation details and computational results. Statistical analysis will be performed to measure the effectiveness of the model and the obtained results. Finally, we present the conclusions of this project and state some perspectives for future work.

2 Green Logistics

Traditional logistics ensure the movement of materials between all actors in the supply chain starting from raw materials locations to final customers via firms factories. These transportation tasks should be completed efficiently to report more benefit to the company. The efficiency is usually measured in terms of money, time and reliability. Recently, the concept of green logistics for sustainable development has soared due to governmental regulations and customers preference for green products. Consequently, transportation companies are reviewing their processes to take into account such concern. The revision consider all the steps in the production process including the choice of raw materials, factoring, packaging, alternative fuels, etc. In some cases transforming the traditional logistics systems to be environmentally friendly will give a cutting down in costs and then it will meet classic logistics objectives. However, in many other situations such review may cost more and come into conflict with traditional logistics.

For transportation companies, green logistics mean transporting goods with lower effect on the environment. Basically, the effect of transporting materials on the environment comes from gazes emissions generated from moving engines like trucks, planes and ships. Then, greener transportation yields to low co_2 emission routes. But, those routes are generally determined using analytical model that consider only saving money as primer objective. Then, the aim of considering the environmental effect will be transformed into a revision of the analytical tools used to generates routing policies and strategies. That, could be completed by determining emission factors and quantifying trucks emissions to integrate them into logistics systems.

2.1 Emission Factors

There are a number of factors that could affect vehicle fuel economy in real world:

1. Vehicle weight:a vehicle carrying more weight requires more energy to run, thus directly affect in fuel economy [4].
2. Vehicle speed and acceleration:fuel consumption and the rate of co_2 per mile traveled decrease as vehicle operating speed increase up to approximately 55 to 65 mph and then begin to increase again[1]. Moreover, the co_2 emission double on a per mile basis when speed drops from 30 mph to 12.5 mph or when speed drops from 12.5 to 5 mph [3]. The relationships between these factors and fuel economy are not simple. For example, the implication of vehicle operating speeds on fuel consumption is not linear and depends on vehicle type and size. It also varies on the model year and age of the vehicle. For instance, studies of vehicle fuel economy taken during the 1990s show less of a drop off in vehicle fuel economy above 55 miles per hours than similar studies of vehicles during the 1970s and 1980s, due to vehicle design changes and engine operating efficiency [14].

3. Weather conditions: weather condition affect vehicle fuel economy. For instance, head-winds reduce vehicle fuel economy as the vehicle needs additional power from the engine to combat the wind drag. Hot weather induces the use of air conditioning, which places accessory load require on the engine.
4. Congestion level: It is commonly known that as traffic congestion increases, co_2 emission (and in parallel fuel consumption) also increase. In general, co_2 emission and fuel consumption are very sensitive to the type of driving that occurs. In fact, traveling at a steady-state velocity will give much lower emissions and fuel consumption compared to a stop-and-go movement. Thus, by decreasing stop-and-go driving, co_2 emissions can be reduced [4].

2.2 Emission Estimation Techniques

To examine the environmental impact of the Vehicle Routing, it is necessary to weigh the environmental impacts of co_2 emission. It is difficult to do an exact estimation because of the uncertain effects of climate change and the setting of a price tag on human health. The DEFRA estimated in 2007 the cost of emitting a tone of co_2 at 25.5. Furthermore, the IPCC [15] published estimates range between 5 and 25. Emissions are estimated using average grams of co_2 per kilometer. The study of Mc Kinnon [25] shows that the load carried is an important parameter to estimate emissions. Thus, we can estimates co_2 from the distance traveled by vehicles and the quantity of goods carried. There are other methods to estimate co_2 emission for vehicle. We can cite for example the fuel-based approach and the distance-based method.

1. The fuel-based approach: In the fuel-based approach [11], the fuel consumption is multiplied by the co_2 emission factor for each fuel. The emission factor is developed based on the fuels heat content, the fraction of carbon in the fuel that is oxidized and the carbon content coefficient. The fraction of gasoline oxidized depends on the transportation equipments used. Therefore, this variability is minimal. In the US inventory, this fraction is assumed to be 99 percent. In the case of road transportation, companies and other entities have the option to override these defaults if they have appropriate data of fuel used. In most case, default emission factors will be used based on generic fuel type categories(e.g., unleaded gasoline, diesel, etc) The fuel-based approach requires essentially two main steps:

   (a) Gather fuel consumption data by fuel type: Fuel use data can be obtained from several different sources. We can cite for example fuel receipts, financial records on fuel expenditures or direct measurements of fuel use. When the amount of fuel is not known, it can be calculated based on distance traveled and an efficiency factor of fuel-per-distance.The distance traveled basically come in three forms:

      – distance(e.g., Kilometers)
      – passenger-distance(e.g.,passenger-kilometers)
      – freight-distance (e.g., ton-miles)

The fuel economy factors depend on the type, age and operating practice of the vehicle in question. Thus, we obtain the following equation:

$$fuel\_consumption = distance * fuel\_economy\_factor$$

(b) Convert fuel estimate to co_2 emissions by multiplying results from step 1 by fuel-specific factors; The recommended approach is to first convert fuel use data into an energy value using the heating value of the fuel. The next step is to multiply by the emission factor of the fuel.

The fuel-based approach is the same for the different mode of transportation. The following equation outlines the recommended approach to calculating co_2 emissions based on fuel use. Thus, we obtain the following equation:

$$co_2\_emissions = fuel\_used * heating\_value * emission\_factor$$

2. The distance-based method :The distance-based method [11] is another method to estimate the carbon dioxide emissions can be calculated by using distance-based emission factor. This method can be used when vehicle activity data is in the form of distance traveled but fuel economy data is not available. It is obvious to formulate our problem using a distance-based method for calculating co_2 emissions. Calculating emissions requires two main steps:

(a) Collect data on distance traveled by vehicle type and fuel type.

(b) Convert distance estimate to co2 emissions by multiplying results from step 1 by distance based emission factors. Thus, we obtain:

$$co_2\_emissions = traveled\_distance * emission\_factor$$

The estimation of emission factor is carried out following two main steps. The first one consists on estimate the fuel conversion factor ($2.61 kg.co_2$/ liter of diesel). The second step is to estimate the emission factor consists on finding a function taking into account data related to the average fuel consumption which depends on load factor.

3 The Vehicle Routing Problem with Emissions

3.1 Literature Review

In recent years, many research works about variants of the VRP in order to reduce the cost and the emission of co_2 was conducted. The Vehicle Routing and Scheduling Problem (VRSP) is an extension of the VRP. Its purpose is to determine the routes and schedules for a fleet of vehicle to satisfy the demand of a set of customers. Thus, it aims to minimize cost which is usually related to the number of vehicles and distance. The reduction in total distance will provide environmental benefits due to the reduction in fuel consumption.

The Time Dependent Vehicle Routing Problem (TDVRP) represents a method which should indirectly produce less pollution and achieve environmental benefits in congested area. The TDVRP is a variant of the VRP and has received less attention. It was originally formulated by Malandrakiand et al. Daskin [7] as mixed linear program. It consists of finding the solution that minimizes the

number of tours by considering traffic conditions. The TDVRP provide environmental benefits, but in an indirect way. Consequently, less pollution is created when vehicle are traveling at the best speeds and for shorter time. The Time Dependent Vehicle Routing and Scheduling Problem (TDVRSP) consists of finding the solution that minimizes the number of tours and the total traveling time. It is motivated by the fact that traffic conditions cannot be ignored, because at peak time, traffic congestion on popular routes will causes delays. The TDVRSP provides also environmental benefits in indirect way. There is an extensive literature related to vehicle emission.Turkay et al. [20] and Soylu et al. [18] demonstrated a collaborative supply chain management for mended business and for decreasing environmentally harmful chemicals, while satisfying local regulation and Kyoto protocol for greenhouse gas emissions. The study of Halicioglu [12] tried to empirically treat the dynamic causal relationship between carbon emissions, energy consumption, income and foreign trade in the case of Turkay [20]. Recently, Van Woensel et al. [21] considered a vehicle routing problem with dynamic travel time due to the traffic congestion. The approach developed introduced the traffic congestion component based on queuing theory in order to determine travel speed. A tabu search method was used to solve the model. Results showed that the total travel time can be improved significantly when explicitly taking into account congestion during the optimization phase. The study of Figliozzi et al. [8] proposed a new methodology for integrating real-world network status and travel date to TDVRP. It developed efficient algorithms TDVRP solution methods to actual road networks using historical traffic data with a limited increase in computational time and memory. The results shows the dramatic impacts of congestion on carriers fleet sizes and distance traveled.

Figliozzi [9] also created a new type of VRP which is denoted the Emission Vehicle Routing Problem(EVRP). The research presented a formulation and solutions approaches for the EVRP where the minimization of emission and fuel consumption is the primary objectives or is part of a generalized cost function. A heuristic is proposed to reduce the level of emission given a number of feasible routes for the TDVRP. Search results indicated that they may be significant emissions saving if commercial vehicles are routed taking emissions into consideration. Moreover, congestion impacts on emission levels are not uniform. Bauer et al. [2] identified and addressed some environmental consideration in the context of intermodal freight transportation and proposed ways to introduce environmental costs into planning model for transportation. They proposed a formulation for scheduled service network design problem with fleet management, it is an integer program in the form of a linear cost multi commodity and capacitated network design formulation that minimize the amount of green house gas emission for transportation activities. The formulation has been implemented on a real life intermodal rail network data.

3.2 The Bi-objective Green Vehicle Routing Problem

The green vehicle routing problem is an answer for the recent environmental awareness in the field of transportation and logistics. The objective is to find

routing and transportation policies that give the best compromise between traveling costs and co_2 emissions. The literature on transportation problems especially vehicle routing problems had considered this environmental interest. Later studies show and implicit interest to handle the objective of gazes minimization But, without viewing it as a major distinct objective like distance and time. We can cite the TDVRP, VRSP and the emissions VRP. In this paper, we consider the the emissions minimization as a separate major objective in addition to the distance minimization objective. Therefor, we define a bi-objective combinatorial optimization problem named the bi-objective green vehicle routing problem.

The bi-objective GVRP [28] could be defined as follow: Giving a set of N customers located in a transportation network and a distance matrix D_{ij} representing the costs of moving between customers i and j and a set of M vehicles hosted in a central depot. A solution of the bi-objective GVRP is composed by a set of routes with minimum traveled distance and the minimum volume of emitted co_2 while visiting each customer once and with respect to vehicles capacity constraints. It is clear that the bi-objective GVRP is an NP-hard problem due to the fact that it is an extension of the standard VRP which is NP-hard.

4 NSGA-II Algorithms for the Bi-objective GVRP

Genetic Algorithms (GA) are stochastic and evolutionary optimization algorithms based on mechanisms of natural selection and genetics. GAs attempt to solve hard non-convex single and multiobjective optimization problems. Multiobjective GAs are based on the concept of Pareto dominance, which emphasizes a research satisfying all objectives. They are well suited for the search of Pareto front through their implicit parallelism to reach optimal solutions more efficiently than an exhaustive method. Many multiobjective genetic algorithms can be cited [6].

The NSGA-II is more efficient than its previous version NSGA [5]. This algorithm tends to spread quickly and appropriately when a certain non dominated region is found. The main advantage is that the strategy of preserving of diversity used in NSGA-II requires no parameters to fix. For these reasons, we choose to resolve our problem using this approach. In NSGA-II, the child population $Q(t)$ is first created from the parent population $P(t)$(randomly filled). They are then met into a set $R(t) = P(t) \bigcup Q(t)$ that is sorted according to the principle of dominance: all non-dominated solutions of the population are assigned a fitness value 1 (first front), then they are removed from the population. All non-dominated solutions of the population are assigned a fitness value 2 (second front), then they are removed from the population. And so on. This process is iterated until all solutions whose fitness value is upon to evaluate is empty [6]. To select subsets that will be placed in the population, a measure of the density of solutions in the space of criteria called crowding distance is used.

5 Implementation and Computational Results

In order to evaluate the effectiveness of the proposed model and to prove the effect of considering explicitly the emissions minimization objective, the NSGA-II

algorithm was implemented to solve bi-objective GVRP instances. The proposed algorithm was implemented using the ParadisEO-MOEO library [26]. The performance of the metaheuristic has been tested on different instances taken from the VRPLIB [23]. These instances involve between 16 and 500 nodes. The number at the end of an instances name represents the number of vehicles while the number at the first is the number of customers. The stopping condition of all tests is based on the number of generation (100 generation). Computational runs were performed on an Intel Core 2 Duo CPU (2.00 GHz) machine with $2G\emptyset$ RAM. The results presented below are based on the following GA parameters:

- Chromosome encoding: a solution chromosome is represented by an integer string. A gene is a customer number, while a sequence of genes dictates a group of customers assigned to a vehicle. For instance, the chromosome (0,3,6,1,0,2,4,0,5,0) contains three routes (0 :: 3 :: 6 :: 1 :: 0), (0 :: 2 :: 4 :: 0) and (0 :: 5 :: 0). The population size is set to 100 chromosomes.
- Crossover: We utilized the standard crossover operator Partially-Mapped-Crossover (PMX). The first step is to Select a substring uniformly in two parents at random. The next step is to exchange these two substrings to produce proto-offspring. The third step is to determine the mapping relationship according to these two substrings. The last step is to legalize proto-offspring with the mapping relationship. The crossover probability is 0.25.
- Mutation: In the mutation stage, two customers are selected from different routes randomly. They are going to be swapped only if constraints are met after this operation. After swap, insertion is done in which we select randomly a customer from a route and try to insert rest of any one route if it satisfies all the constraints.The mutation probability is fixed to 0.35.

5.1 Computational Results

To demonstrate the efficiency of the metaheuristic implementation, measures related the computation time are computed and reported in Table.1. We can remark that the computation time of the implemented algorithm increases proportionally to the size of the instance due to algorithm complexity and especially the complexity of the computation of the crowding distance $O(MNlogN)$. It is important to observe that the cardinality of the pareto fronts is small. This fact can be explained by the correlation between our two objectives; for instance the emissions objectives was written as a function of the distance objective. From another side, we can see for four instances, that obtaining solutions with minimal distance does not imply minimal emissions.

5.2 Statistical Analysis

To evaluate the quality of the obtained solutions and measure the performance of the algorithm, metric measurements have been selected and calculated. We use three metrics: the first is the Generational Distance (GD) which measures how far from the Front Pareto is located a set of solutions, the second is the Spacing (S) metric which measures the distribution uniformity of points of the set of solution

Table 1. The obtained Pareto fronts and the needed computation time

Instance	Pareto front		CTime (s)
	Obj1(km)	Obj2(kg.co2)	
E101-08e			83.413
	1946	1411	
	1961	1398	
	1977	1349	
E301-28k			99.621
	2352	1598	
	2298	1643	
	2357	1597	
	2277	1683	
	2349	1602	
	2360	1592	
	2303	1631	
	2302	1641	
	2302	1596	
E421-41k			126.547
	4163	2982	
	4153	2998	
	4168	2971	
	4071	3094	
E484-19k			135.330
	2307	1365	
	2306	1361	
	2325	1348	

in the plan $(obj1, obj2)$, the third indicator of performance is the Entropy (E) metric that uses the concept of niche to evaluate the distribution of solutions on the front. The NSGA-II algorithm give different approximations for each execution. Thus, to empirically analyze the performance of our algorithms, we first run the same algorithm several times on the same instance of the problem. We get then a sample of approximation. We run the algorithm ten times for each instances. Table 2 presents averages of metrics GD, S and E over ten runs of the four instances.

Table 2. Averages of metrics GD, S and E for the algorithm NSGA-II

Instances	**GD**	**S**	**E**
E101-08e	3.931	4.632	0.227
E301-28k	4.063	6.878	0.422
E421-41k	3.009	6.515	0.095
E484-19k	5.396	2.570	0.371

The values obtained by the GD metric are small and vary between 0.6282 and 6.250 so are not large enough. We can then conclude that the set of solutions are near of the Pareto front. For the S metric, the results obtained for variants E101-08e and E421-41k and E484-19k are close to 0, so the points are well distributed in the set Parto front. For the instance E301-28k, the mean value is equal to 6.878, therefore the worse. By exploiting the solutions obtained by the Entropy metric, we note that the value found in the instance E421-41k is the closer to 1, thus the distribution of solutions for this instance on the front is better than the three other instances. To evaluate the metaheuristic rigorously and to estimate the confidence of the results to be scientifically valid, statistical tests are performed on the indicators of performance. Experiments are performed on the four instances E101-08e, E301-28k, E421-41k and E484-19k. The three algorithms are executed ten times for each instance and calculations of metrics GD, S and E are made. In order to determine whether the mean of the experiments are different or not at a statically significant level, an analysis of variance is done. By applying a Shapiro-test on the distribution, we found that this one follow a normal low. Consequently, we used a one factor analysis of variance (ANOVA) test which is based on the central assumption of normally data distribution to check whether a factor has a significant effect on the performance of the algorithm. In our case, the experiments are taken as factor and the metrics are taken as dependents variables. The hypothesis is:

$H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4$ Versus $H_1 : \mu_i \neq \mu_j$
with $i, j = 1, 2, 3, 4$ and $i \neq j$

Table 3 shows the ANOVA for metrics GD, S and E. The first ANOVA for metric GD don't found significant differences for the different experiments. Hence, the effect of the factor experiment does not influence the variables of measures of performance. The second ANOVA for metric S found significant differences. Consequently, there is an effect of the factor on the variable of measures of performance. The third ANOVA for metric E found also significant differences.

Table 3. ANOVA table for metrics GD, S and E

	Sum sq	DF	Mean sq	F-value	Prob > F
GD					
Factor	7.352	3	2.4508	0.7742	0.517
Residual	3.1728	36			
S					
Factor	137.09	3	45.696	10.977	2.9 $\exp^{-5}$
Residual	149.86	36	4.163		
E					
Factor	0.3981	3	0.1327	10.759	3.428 $\exp^{-5}$
Residual	0.4441	36	0.0123		

6 Conclusions

The green vehicle routing problem consists of designing a set of routes for a set of vehicles to serve customers over a transportation network. We model the GVRP as bi-objective optimization problem where the first objective is to minimize the overall traveled distance and the second objective is to minimize the volume of emitted co_2. Many solving approaches and algorithms are envisaged. In this paper, we choose evolutionary algorithms to find better pareto fronts for the GVRP. This choice is explained by the performance of evolutionary algorithms especially elitist algorithm like NSGA-II, SPEA-II and the IBEA algorithms for solving multi-objective combinatorial optimization problems. Hence, we implement the NSGA-II algorithm for solving GVRP benchmarks. The obtained results show and prove the effectiveness of considering the emissions minimization as a separate objective. Performed statistical tests confirm the quality of the generated pareto fronts and then the performance of the NSGA-II algorithm.

References

1. Aronsson, H., Brodin, M.H.: The environmental impact of changing logistics structures. Int. J. Logist. Manag. 17(3), 394–415 (2006)
2. Bauer, J., Bektas, T., Crainic, T.G.: Minimizing greenhouse gas emissions in intermodal freight transport: an application to rail service design. Journal of the Operational Research Society 61, 530–542 (2010), doi:10.1057/jors.2009.102
3. Bickel, P., Friedrich, R., Link, H., Stewart, L., Nash, C.: Introducing environmental externalities into transport pricing: Measurement and implications. Transp. Rev. 26(4), 389–415 (2006)
4. Boriboonsomsin, K., Vu, A., Barth, M.: CoEco-Driving: Pilot Evaluation of Driving Behavior Changes among U.S. Drivers. University of California, Riverside (August 2010)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 181–197 (2002)
6. Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007) ISBN 978-0-387-33254-3
7. Malandraki, C., Daskin, M.S.: Time-Dependent Vehicle-Routing Problems - Formulations, Properties And Heuristic Algorithms. Transportation Science 26(3), 185–200 (1992)
8. Figliozzi, M.A.: A Route Improvement Algorithm for the Vehicle Routing Problem with Time Dependent Travel Times. In: Proceeding of the 88th Transportation Research Board Annual Meeting CD rom (2009)
9. Figliozzi, M.A.: Vehicle Routing Problem for Emissions Minimization. Transportation Research Record 2197, 1–7 (2010)
10. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, pp. 416–423. University of Illinois at Urbana- Champaign, Morgan Kaufmann Publishers (1993)

11. The Greenhouse Gas Protocol Initiative: Calculating CO2 emissions from mobile sources. Guidance to calculation worksheets (2005), http://www.ghgprotocol.org/standard/mobile.doc
12. Halicioglu, F.: An econometric study of CO2 emissions, energy consumption, income and foreign trade in Turkey. Energy Policy 37, 1156–1164 (2009)
13. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Piscataway, New Jersey, vol. 1, pp. 82–87. IEEE Service Center (June 1994)
14. ICF Consulting: Assessment of Greenhouse Gas Analysis Techniques for Transportation Projects. 9300 Lee Highway, Fairfax, Virginia 22031 (May 2006)
15. http://www.ipcc.ch
16. Knowles, J.D., Corne, D.W.: Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation 8(2), 149–172 (2000)
17. David Scheffer, J., Grefenstette, J.J.: Multiobjective Learning via Genetic Algorithms. In: Proceedings of the 9th International joint Conference on Articial Intelligence(IJCAI 1985), Los Angeles, California, pp. 593–595. AAAI (1985)
18. Soylu, A., Oruc, C., Turkay, M., Fujita, K., Asakura, T.: Synergy Analysis of Collabo- rative Supply Chain Management in Energy Systems Using Multi-Period MILP. Eur. J. Oper. Res. 174(1), 387–403 (2006)
19. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), 221–248 (1994)
20. Turkay, M., Oruc, C., Fujita, K., Asakura, T.: Multi-Company Collaborative Supply Chain Management with Economical and Environmental Considerations. Comput. Chem. Eng. 28(6-7), 985–992 (2004)
21. Van Woensel, T., Kerbache, L., Peremans, H., Vandaele, N.: Vehicle routing with dynamic travel times: a queueing approach. European Journal of Operational Research 186, 990–1007 (2008)
22. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Giannakoglou, K., Tsahalis, D., Periaux, J., Papailou, P., Fogarty, T. (eds.) Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, EUROGEN 2001, Athens, Greece, pp. 95–100 (2001)
23. http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html
24. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, University of Michigan, USA (1975)
25. Mckinnon, A., Cullinan, S., Browne, M.: Green logistics: Improving the environmental sustainability of logistics. Kogan Page, limited (2010)
26. Liefooghe, A., Jourdan, L., Talbi, E.: A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research 209(2), 104–112 (2011)
27. Toth, P., Vigo, D.: The Vehicle Routing Problem. SIAM, Philadelphia (2001) ISBN 0898715792
28. Jozefowiez, N., Semet, F., Talbi, E.: Multi-objective vehicle routing problems. European Journal of Operational Research 189(2), 293–309 (2008)

Clustering Search Heuristic for Solving a Continuous Berth Allocation Problem

Rudinei Martins de Oliveira[1], Geraldo Regis Mauri[2], and Luiz Antonio Nogueira Lorena[3]

[1] National Institute for Space Research - INPE, São José dos Campos - SP, Brazil
[2] Federal University of Espírito Santo - UFES, Alegre - ES, Brazil
[3] National Institute for Space Research - INPE, São José dos Campos - SP, Brazil
rudmart@gmail.com, mauri@cca.ufes.br, lorena@lac.inpe.br

Abstract. Due to the increasing demand for ships carrying containers, the Berth Allocation Problem (BAP) can be considered as a major optimization problem in marine terminals. In this context, we propose a heuristic to solve a continuous case of the BAP. This heuristic is based on the application of the Clustering Search (CS) method with the Simulated Annealing (SA) metaheuristic. The results obtained by CS are compared to other methods found in the literature and its competitiveness is verified.

Keywords: Continuous Berth Allocation Problem, Clustering Search, Simulated Annealing, Metaheuristics.

1 Introduction

The marine transport of goods increased in the last years due to the increase of the trade and of the international economic growth [9]. In January of 2010, the world merchant fleet reached 1.276 million tons increasing by 84 million compared to the previous year [26]. Due to the growth of the international trade, the ports must be modernized to produce workmanship and technology capable to supply the increasing demand on the transport of goods. In that context, the need for good logistic planning for accommodating ships in berths induce the appearance of a problem known in the literature as Berth Allocation Problem (BAP).

The BAP consists in allocating ships to positions of mooring using the maximum space of the quay and minimizing the service time of the ships. The decisions to be made are concerning the position and time that the ship should moor [10].

The BAP can be modeled as discrete or continuous [12]. In the discrete case, the quay is divided into several berths and only one ship is serviced at a time in each berth, regardless of their size. In the continuous case, there is no division of the quay and thus the ships can moor at any position. Usually, the model of continuous berth allocation is used in large ports. Moreover, if we consider the arrival of ships, the problem can be treated as static or dynamic [10]. The static case assumes all ships already in port for the handling, while the dynamic case allows ships to arrive at any time (known in advance) [3].

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 49–62, 2012.

In this paper the BAP is considered as continuous and dynamic, adopting the model reported in [6] that divides the quay in sequences of fixed lengths, where a big ship can occupy more than one berth and a small ship can lend its unused space. In general, the BAP search for a better distribution of the quay space minimizing the service time of the ships in the port. In that context, this work presents an alternative to solve the continuous BAP. We proposed an application of a hybrid method known as Clustering Search - CS [22] using a Simulated Annealing (SA) algorithm to generate solutions. The remainder of the paper is organized as follows. Section 2 presents a brief literature review about the BAP. Section 3 describes our approach for the problem. The proposed CS are presented in Section 4 and the computational results are reported in Section 5. Finally, our conclusions are summarized in Section 6.

2 Literature Review

Initial works about the BAP appeared when Thurman [25] proposed an optimization model for planning berth in Norfolk Naval Station (USA).

Imai et al. [10] studied the discrete case of BAP considering the dynamic arrival of ships in the port. The authors presented a method based on a Lagrangian Relaxation for the original problem. In the same year, Nishimura et al. [21] developed a Genetic Algorithm to solve the dynamic and discrete BAP. Two years later, Imai et al. [11] upgraded their approach considering different service priorities between the ships. Furthermore, the authors proposed a Genetic Algorithm to solve the BAP. Imai et al. [13] considered physical restrictions for the port, represented by the diversity of the ships (length). The results were also obtained through a Genetic Algorithm. Cheong et al. [5] presented an application of the Multiobjective Evolutionary Algorithm (MOEA) to solve the BAP.

Mauri et al. [18] dealt with the BAP with a hybrid method called PTA/LP, which uses an evolutionary algorithm with a linear programming model using the technique of column generation. Buhrkal et al. [3] treated the discrete case of BAP using a model based on the Vehicle Routing Problem with Time Windows and Multiple Depot (VRPTWMD), following the approach reported by Cordeau et al. [6].

Xu et al. [27] modeled the BAP as a Parallel-Machine Scheduling Problem in which the assignment of vessels to berths is limited by water depth and tidal condition. Tidal conditions were also considered in Barros et al. [1], which propose an integer linear programming model based on the transportation problem to represent the BAP in tidal bulk ports with stock level conditions.

Today, most studies address the discrete case of the BAP, but the continuous case has gained emphasis in recent years. In the paper of Lim [17], the continuous BAP was modeled as a restricted form of the packing problem in two dimensions and he showed that the problem is NP-Hard. In the same year, Li et al. [16] studied the problem of minimizing the makespan in the schedule of ships. The BAP can be also formulated as a Machine Scheduling Problem in which the ships are represented by jobs and processors by cranes [7].

In Park and Kim [23], a method with two phases was suggested to solve a mathematical model. The first stage determines the position of the berth and the service time of each ship. A subgradient technique was applied to solve the problem. In the second phase, the model uses the solution obtained in the first phase and the cranes are assigned to the ships by dynamic programming. Kim and Moon [14] formulated a mixed integer linear programming model to the berth scheduling problem and the results were compared with a Simulated Annealing algorithm. The similarity of the BAP with the cutting stock problem was demonstrated by Imai et al. [12]. In this paper, a heuristic algorithm was developed to solve it. The algorithm is composed of two stages: a solution is obtained; and a procedure reallocates ships that are overlapping or are separated by a space.

Cordeau and Laporte [6] proposed two formulations and two heuristics based on Tabu Search for solving the BAP. The authors presented tests for the port of Gioia Tauro (Italy). Another continuous approach was proposed by Mauri et al. [19], which used a Memetic Algorithm improving the solutions reported in [6]. Lee et al. [15] researched the continuous BAP and developed two versions of the Greedy Randomized Adaptive Search procedure, seeking to minimize the total flow time of the ships with a penalty. Finally, Seyedalizadeh Ganji et al. [24] applied a Genetic Algorithm to solve the model proposed by Imai et al. [12].

3 Problem Approach

The PAB can be classified as one of the main optimization problems in marine terminals. This is because of the growing demand for ships that reduce the cost of transportation and carry thousands of tons of goods. It consists in allocating ships to positions of mooring using the maximum space in the quay while minimizing the service time of the ships. The decisions to be made are: *where* and *when* the ships should moor [10].

Restrictions on the maximum water depth and the distance from the more favorable location along the quay must be considered to determine the position of mooring. The time needed to load or unload one ship is defined by the capacity of the berth where it will be allocated. In general, the handling time is determined by the number of cranes available, and the distance between the berth and the location of the containers on the yard [20], but a deterministic time can be also considered [2]. The service time is the waiting time plus handling time for each ship. In the dynamic case, the service time is usually accomplished within a time window.

In this paper, the continuous BAP (BAP-C) was solved seeking to minimize the sum of the service time, i.e., the time from arrival to departure of the ships in port. Figure 1 illustrates the time intervals used by each ship, where the ships are considered rectangles on a cartesian plane. The horizontal axis represents the physical space of the quay, and the vertical axis indicates the different types of time used by the ship in the port.

Considering N as the number of ships, $n = |N|$, and M as the set of berths, $m = |M|$, the variables used in Figure 1 are described in the following: a_i is the

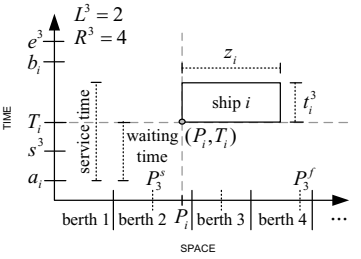


Fig. 1. Berths and time variables [19]

arrival time for the ship i; b_i is the end of time window for the ship i; t_i^k is the service time for the ship i in berth k; z_i is the length of the ship i; s^k is the opening hour of the berth k and e^k is the closing time of the berth k; P_i indicates the position for each ship i and P_k^s is related to the start position for the berth k; P_k^f is the final position for the berth k; R^k and L^k are the neighbors at right and left side for the berth k.

An important fact to be considered in the model presented in [6] are the discontinuities along the quay, represented in Figure 2 by the lines with an asterisk. This figure shows solutions of the discrete (a) and continuous (b) BAP for the first instance of the test set used in this paper (see Section 5). The solid lines are the separations of the berths and the dotted lines indicate the divisions of the berths on two sides (right and left). Note that there are berths that do not divide (berths 1, 7, 8, 9 and 13). The rectangles are the ships and the gray and black colors are for easy viewing.

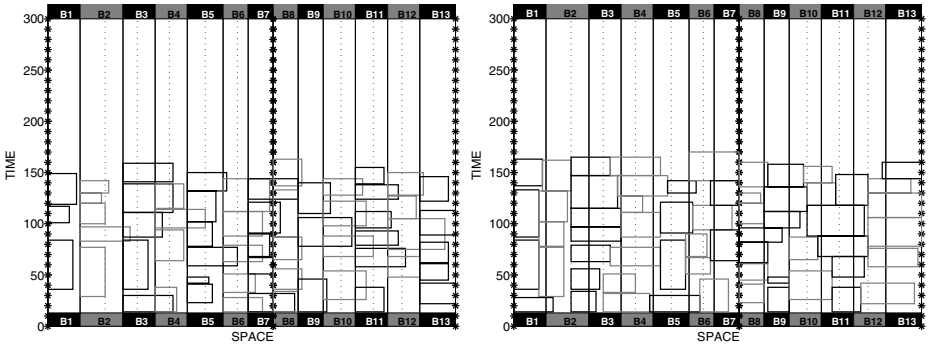


Fig. 2. Solutions for the discrete (a) and the continuous (b) case of the BAP [19]

The spatial dimensions are ignored for the case of discrete BAP (Figure 2a), causing overlap of ships. For the continuous case (Figure 2b), overlapping ships must be removed. Note that there is no overlap of ships (that is a feasible solution). As reported by Cordeau et al. [6], the discrete case is a relaxation of the continuous one.

4 Clustering Search (CS)

According to [4], the CS consists basically of three main components: a metaheuristic to generate the solutions; the clustering process; and a local search heuristic. At each iteration, a solution S is generated by the metaheuristic and sent to the clustering process. This solution is then grouped to the most similar cluster C_j and the center of this cluster c_j is updated with new information contained in the solution, making the center to move in the search space.

The volume v_j of the cluster j is then analyzed and, if this volume reaches a limit λ ($\nu_j \geq \lambda$), we realize that some solution pattern is predominantly generated by the metaheuristic. Therefore, this cluster may be a promising region of search. Finally, we analyze the inefficiency index r_j, and if the local search heuristic does not improve the solution for r_{max} ($r_j \geq r_{max}$) consecutive times, a random perturbation is then applied to the center c_j, aiming to escape of a possible local optimum. On the other hand, if $r_j < r_{max}$, the local search heuristic is applied at the center c_j analyzing the neighborhood of the cluster j. Ending this process, the metaheuristic will generate a new solution.

The stopping criterion of CS is generally defined by the chosen metaheuristic. Figure 3 shows the flowchart of CS. More details about this method are presented in [4] and [22].

Following the flowchart of CS (Figure 3), the initial clusters are created. For each cluster, a solution (cluster's center c_j) is created by using the distribution (Figure 4), scheduling (Figure 5) and update (Figure 6) heuristics presented

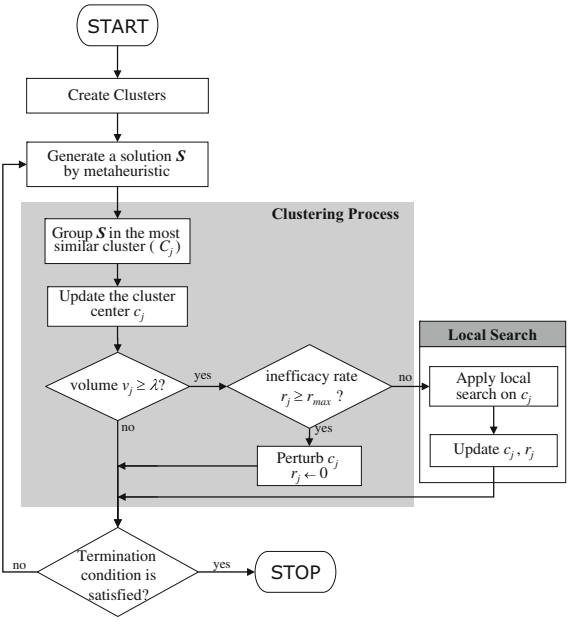


Fig. 3. Flowchart of CS [4]

in [18] and [19]. So, for some iterations, another solutions generated by the metaheuristic are sent to the clustering process and so on.

The Distribution Heuristic is a simple and balanced heuristic. Initially, the ships are organized by incoming order on port (a_i) and distributed to the berths in a random way. In this process, the selected berth must always be able to assist the selected ship. This procedure ensures that each ship will be assigned to a berth that must be able to attend it, i.e., the berth length is sufficient to receive the ship and the berth's equipments are suitable to operate the type of cargo into the ship. This procedure does not guarantee that the berthing time (T_i) and position (P_i) for the ship i present no overlapping on time and space dimensions.

```
1.  MAKE (m empty berths);
2.  MAKE (a list L with all ships);
3.  ORDER ( list L by time arrival of ships to port);
4.  FOR (each ship j in L, j = 1,2,...,n) DO
5.     CHOOSE ( a berth i, i = 1,2,...,m);
6.     IF (the berth i can not attend the ship j)
7.        BACK (to step 5);
8.     ELSE
9.        ASSIGN (the ship j the berth i);
10.    END-IF;
11. END-FOR;
```

Fig. 4. Distribution Heuristic [18]

After applying the Distribution Heuristic, we must define the berthing time and position for all of ships. Berthing time and position for all of ships assigned to a specific berth are set to initial values according to Figure 5. At this moment, the berthing times are set to equal the arrival times of the ships, if the berth is available. The berthing positions are set to equal the start position of the berth, so we have initial values for P_i and T_i.

In the Update Heuristic (Figure 6), the spatial distribution for the ships to the berths are updated and improved. This procedure updates the berthing times and positions considering a simple idea: if some overlap is detected for each ship, its berthing time is delayed until eliminate all of them. If a berth has no neighbor at the left side, all of ships assigned to it have the berthing position set to equal the start position of the berth, i.e., all of ships (rectangles) are aligned at left. If a

```
1. FOR (each berth k, k = 1,2,...,m) DO
2.     FOR (each ship i assigned to k) DO
3.         T_i = { max(a_i, s^k)          i = 1
                 { max(a_i, T_{i-1} + t^k_{i-1})   i > 1
4.         P_i = P^s_k
5.     END-FOR;
6. END-FOR;
```

Fig. 5. Programming Heuristic [18]

berth has no neighbor at the right side, all of ships assigned to it have the berthing position set to equal the final position of the berth minus the ship length, i.e., all of ships are aligned at right. Finally, if a berth has neighbors on both sides, we try to fit the ships among the ones assigned to the two neighbor berths.

To compute the objective function (Equation 1) for the solutions generated by the previous heuristics, we looked for minimizing the service times multiplied by an associated cost (ν_i), the violations of the time windows of the ships and the violations in the time windows of the berths. The omegas (ω_0,ω_1,ω_2) in each term of the expression are the penalties factors which guarantee the elimination of infeasible solutions. T_i^k, $\forall k \in M, i \in N$ is the time when the ship moored in the berth k; $T_{o(k)}^k$, $\forall k \in M$, is the time when the first ship moored at beth k; $T_{d(k)}^k$, $\forall k \in M$, is the time when the last ship leaves the berth k; $d(k)$ is the last

```
1.  INPUT: berth k,
2.  FOR (each ship i assigned to k) DO
3.      IF L^k=0 THEN
4.          P_i = P_k^s
5.          FOR (each ship j assigned to k+1) DO
6.              IF i overlaps j THEN
7.                  T_i = max(T_i, T_j + t_j^{k+1})
8.              END-IF;
9.          END-FOR;
10.     ELSE
11.         IF R^k=0 THEN
12.             P_i = P_k^j - z_i
13.             FOR (each ship j assigned to k-1) DO
14.                 IF i overlaps j THEN
15.                     T_i = max(T_i, T_j + t_j^{k-1})
16.                 END-IF;
17.             END-FOR;
18.         ELSE
19.             P_i = P_k^s
20.             WHILE ∃ j assigned to k-1 overlapping i DO
21.                 IF (P_j + z_j ≥ P_k^s) and (P_j + z_j + z_i ≤ P_k^f) THEN
22.                     P_i = P_j + z_j
23.                 ELSE
24.                     T_i = max(T_i, T_j + t_j^{k-1})
25.                     P_i = P_k^f - z_i
26.                 END-IF;
27.             END-WHILE;
28.             WHILE ∃ j assigned to k+1 overlapping i DO
29.                 T_i = max(T_i, T_j + t_j^{k-1})
30.                 WHILE ∃ l assigned to k-1 overlapping i DO
31.                     T_i = max(T_i, T_l + t_j^{k-1})
32.                 END-WHILE;
33.             END-WHILE;
34.         END-IF;
35.     END-IF;
36. END-FOR;
```

Fig. 6. Update Heuristic [19]

ship leaving the berth k and $o(k)$ is the first ship moored in the berth k; ν_i is the value (cost) of the service time of ship i; $x_{ij}^k \in \{0,1\}, \forall k \in M; i,j \in N, x_{ij}^k = 1$ if ship j is serviced by berth k after the ship i, and $x_{ij}^k = 0$ otherwise.

$$f(x) = \begin{cases} \omega_0 \sum\limits_{i \in N} \sum\limits_{k \in M} \nu_i \left(T_i^k - a_i + t_i^k \sum\limits_{j \in N \cup \{d(k)\}} x_{ij}^k \right) + \\ \omega_1 \left(\sum\limits_{i \in N} \sum\limits_{k \in M} \left[\max\left(0, a_i - T_i^k\right) + \max\left(0, T_i^k + t_i^k \sum\limits_{j \in N \cup \{d(k)\}} x_{ij}^k - b_i\right) \right] \right) + \\ \omega_2 \left(\sum\limits_{i \in N} \sum\limits_{k \in M} \left[\max\left(0, s^k - T_{o(k)}^k\right) + \max\left(0, T_{d(k)}^k - e^k\right) \right] \right) \end{cases} \quad (1)$$

The SA runs and at each temperature, the current solution (not the best) is sent to CS. Figure 7 presents a pseudo-code of the implemented SA. It may be noted that CS is called in line 20 of this algorithm, i.e., at each temperature. The neighborhood structure in SA (line 10) uses three different moves presented in [18]: Reorder Ships, Reallocate Ships and Swap Ships.

```
1.  GIVEN (α, SAmax, T0 e TC) DO
2.  GENERATE (a solution S through distribution heuristic);
3.  EVALUATE (a solution S through programming heuristic);
4.  S* ← S;                    { Best solution obtained so far }
5.  IterT ← 0;                 { Number of iterations in temperature T}
6.  T ← T0;                    { Current temperature }
7.  WHILE(T > TC) DO
8.    WHILE (IterT < SAmax) DO
9.      IterT ← IterT + 1;
10.     GENERATE (any neighbor S' through a exchange move);
11.     APPLY (programming heuristic on all berths in S');
12.     Δ ← f(S') – f(S);
13.     IF (Δ < 0)  S ← S';
14.       IF (f(S') < f(S*))  S* ← S';  END-IF;
15.     ELSE
16.       TAKE (x ∈ [0,1]);
17.       IF (x < e^(-Δ/T))  S ← S';  END-IF;
18.     END-IF;
19.   END-WHILE;
20.   EXECUTE-CS (current solution S);
21.   T ← α * T;   IterT ← 0;
22. END-WHILE;
23. S ← S*;
24. RETURN (S);
```

Fig. 7. Simulated Annealing algorithm used in CS [18]

After the execution of each movement, the programming and update heuristics are applied to eliminate overlaps and compute the value of the objective function for the new solution. Each solution in the SA neighborhood (line 10) is generated by one of these movements, and its choice is made at random enabling a good diversity among the generated intermediate solutions and a good exploration of the search

space. After executing the SA, i.e., the method CS-SA as a whole, the best solution is taken as the final solution to the problem. The flowchart of CS (Figure 3) after generating a solution by the SA, i.e., line 20 in Figure 7, is detailed in Figure 8.

```
1.  GIVEN (the solution S);
2.  FIND (the cluster center C_j more similar to S);
3.  v_j ← v_j + 1;
4.  UPDATE (the center C_j);
5.  IF v_j ≥ λ THEN
6.  v_j ← 1;
7.      IF r_j ≥ r_max THEN
8.          APPLY (perturbation in c_j)
9.          r_j ← 0;
10.     ELSE
11.         APPLY (local search – find c_j');
12.         IF f(c_j') < f(c_j) THEN
13.             c_j ← c_j';
14.             r_j ← 0;
15.         ELSE
16.             r_j ← r_j + 1;
17.         END-IF;
18.     END-IF;
19. END-IF;
```

Fig. 8. EXECUTE-CS algorithm

The smallest Hamming distance [8] is used to determine the most similar cluster (line 2 in Figure 8). A path-relinking between the solution S and c_j is applied to update the center of the cluster (line 4 in Figure 8). The idea of this algorithm is simple, and it consists in performing the necessary movements to transform the solution S in the solution c_j. From these movements, the best solution is taken as a new center of the cluster C_j.

The perturbation shown in line 8 (Figure 8) is given by a simple application of the movement Swap Ships. Finally, a simple local search (line 11 in Figure 8) is used to intensify the search in the promising clusters.

5 Computational Tests

We used 30 different instances with 60 ships and 13 berths. These instances are based on data from the port of Gioia Tauro (Italy) and they were randomly generated by Cordeau et al. [6]. All experiments were performed on a PC Intel core 2 duo with 1.66 GHz and 2 GB of RAM memory. The whole implementation was developed in C++.

5.1 Tuning Parameters

To tune the parameters of CS, we have used our previous experience and some computational tests. To set the main CS parameters (number of clusters, λ and

r_{max}), we have adopted several values for each parameter while the others were kept fixed. So, CS ran three times for each parameter setting, and the setting yielding the best average results was chosen. The number of clusters, λ and r_{max} were tested on the ranges $[5, 15]$, $[3, 10]$ and $[3, 5]$, respectively, with a step size of 1. The values used by the SA parameters was the same adopted in a previous work [19] ($T_0 = 15000$, $T_c = 0.01$, SAmax $= 1000$ and $\alpha = 0.975$). The omegas $(\omega_0, \omega_1, \omega_2) = (1, 10, 10)$ was also defined according to [19].

5.2 Results

This section presents the computational results for the proposed CS compared to a Tabu Search - TS [6] and a Memetic Algorithm - MA [19] reported in previous works. A statistical analysis based on the average, deviation, differences and improvements among all the methods are reported in the next tables.

Table 1 shows a comparison between TS, MA and CS. The first three columns show the instances, the optimum value for the BAP-D (OPT DISC) and the best known solution (BKS) between the three methods (TS, MA and CS). The results of each method are presented in the columns (4-17), where: BST FO and AVG FO are the best and the average solutions; AVG BKS is the average time to find the best known solution reported in [6]; and AVG BST is the average time to find the best solution (BST FO). The running times are reported in seconds.

The columns DEV and DEV BKS are related to the deviations between the average and best solutions. DEV is computed between the average (AVG FO) solution over the best solution (BST FO) for each method (DEV = 100×(AVG FO - BST FO)/BST FO)) and DEV BKS is computed between the the best solution (BST FO) over the best known solution for all the methods (DEV BKS = 100×(BST FO - BKS*)/BKS*)).

In general, the values of the solutions in Table 1 (BAP-C) are greater than those of BAP-D (column OPT DISC), which is correct because the discrete case is a relaxation of the continuous one [6].

Comparing all the methods, we can observe that CS achieved the best results for all the instances. Moreover, CS was faster than MA to find the best solutions (AVG BST), and it provides the lowest values for the deviations DEV = 1.17% and DEV BKS = 0.00%.

Table 2 shows the crossing over all methods showing the improvements over the solutions achieved for all the instances. We can note that both MA and CS obtained improvements over TS, and the CS reports average improvements of 1.7% and 6.8% when compared to MA (CSxMA) and TS (CSxTS), respectively. Table 2 also shows the differences between the BAP-D and the BAP-C, and CS got the lowest levels of difference in all the cases, with an average of 8%. It is interesting to note that the solutions obtained by our CS were close to the optimal values for the discrete case, indicating good upper bounds for the continuous BAP.

Table 1. Comparison among the results of TS, MA and CS

Inst.	OPT DISC	BKS	TS		MA						CS					
			BST FO	DEV BKS	BST FO	AVG FO	AVG BKS	AVG BST	DEV (%)	DEV BKS	BST FO	AVG FO	AVG BKS	AVG BST	DEV (%)	DEV BKS
i01	1409	1583	1706	7.77	1613	1666.50	37.49	85.41	3.32	1.90	1583	1599.10	24.96	67.76	1.02	0.00
i02	1261	1315	1355	3.04	1326	1347.80	51.91	75.85	1.64	0.84	1315	1321.80	26.90	68.92	0.52	0.00
i03	1129	1207	1286	6.55	1234	1261.90	20.58	79.99	2.26	2.24	1207	1226.50	25.78	86.28	1.62	0.00
i04	1302	1380	1440	4.35	1392	1421.00	36.36	84.54	2.08	0.87	1380	1388.80	26.40	65.16	0.64	0.00
i05	1207	1262	1352	7.13	1285	1302.00	8.55	90.02	1.32	1.82	1262	1274.90	26.02	86.51	1.02	0.00
i06	1261	1413	1565	10.76	1461	1492.10	5.84	85.67	2.13	3.40	1413	1468.80	24.58	71.96	3.95	0.00
i07	1279	1324	1389	4.91	1333	1352.00	12.38	92.50	1.43	0.68	1324	1333.00	24.58	71.96	0.68	0.00
i08	1299	1381	1519	9.99	1425	1466.40	12.88	91.94	2.91	3.19	1381	1418.00	26.09	64.62	2.68	0.00
i09	1444	1633	1713	4.90	1651	1698.20	40.06	89.57	2.86	1.10	1633	1665.80	25.19	80.43	2.01	0.00
i10	1213	1333	1411	5.85	1371	1393.00	38.93	72.64	1.60	2.85	1333	1347.70	35.25	80.21	1.10	0.00
i11	1368	1527	1696	11.07	1557	1602.10	5.70	72.93	2.90	1.96	1527	1551.20	23.99	76.27	1.58	0.00
i12	1325	1505	1629	8.24	1537	1565.80	7.64	88.01	1.87	2.13	1505	1521.00	24.74	81.50	1.06	0.00
i13	1360	1424	1519	6.67	1449	1482.50	23.66	83.02	2.31	1.76	1424	1442.50	25.61	52.14	1.30	0.00
i14	1233	1273	1369	7.54	1287	1306.90	6.57	80.19	1.55	1.10	1273	1284.60	24.90	59.46	0.91	0.00
i15	1295	1345	1455	8.18	1362	1394.20	14.44	75.85	2.36	1.26	1345	1356.60	24.96	67.75	0.86	0.00
i16	1364	1491	1715	15.02	1508	1581.10	4.28	87.78	4.85	1.14	1491	1512.50	23.79	72.49	1.44	0.00
i17	1283	1302	1322	1.54	1318	1335.30	68.71	83.41	1.31	1.23	1302	1305.30	27.96	59.88	0.25	0.00
i18	1345	1518	1594	5.01	1519	1552.10	21.68	89.47	2.18	0.07	1518	1531.30	25.88	85.80	0.88	0.00
i19	1367	1517	1673	10.28	1573	1628.60	26.00	84.78	3.53	3.69	1517	1546.80	24.24	92.88	1.96	0.00
i20	1328	1406	1450	3.13	1428	1469.30	67.47	91.01	2.89	1.56	1406	1414.90	27.21	87.27	0.63	0.00
i21	1341	1461	1565	7.12	1481	1510.40	7.64	80.47	1.99	1.37	1461	1464.40	24.22	76.96	0.23	0.00
i22	1326	1437	1618	12.60	1484	1521.00	12.57	86.19	2.49	3.27	1437	1449.10	24.34	72.13	0.84	0.00
i23	1266	1395	1539	10.32	1425	1456.10	3.56	84.86	2.18	2.15	1395	1413.60	24.45	89.60	1.33	0.00
i24	1260	1351	1425	5.48	1359	1383.40	14.40	87.65	1.80	0.59	1351	1362.00	26.10	69.03	0.81	0.00
i25	1376	1513	1590	5.09	1546	1604.50	38.04	79.12	3.78	2.18	1513	1545.10	26.32	67.80	2.12	0.00
i26	1318	1448	1567	8.22	1475	1520.10	22.25	80.51	3.06	1.86	1448	1458.40	24.40	67.92	0.72	0.00
i27	1261	1349	1458	8.08	1356	1382.30	6.59	70.23	1.94	0.52	1349	1354.70	25.22	89.33	0.42	0.00
i28	1359	1461	1550	6.09	1486	1553.90	53.80	81.15	4.57	1.71	1461	1471.50	25.62	63.48	0.72	0.00
i29	1280	1323	1415	6.95	1338	1360.80	6.76	74.49	1.70	1.13	1323	1329.70	25.17	75.00	0.51	0.00
i30	1344	1487	1621	9.01	1512	1555.60	7.43	84.63	2.88	1.68	1487	1504.10	25.29	66.10	1.15	0.00
AVG	1306.77	1412.13	1516.87	7.36	1436.37	1472.23	22.81	83.13	2.46	1.71	1412.13	1428.79	25.67	73.89	1.17	0.00

Table 2. Improvements and comparison with the discrete case

	IMPROVEMENTS			CONT × DISC		
Inst.	MAxTS	CSxMA	CSxTS	TS	MA	CS
i01	5.5	1.9	7.2	21.08	14.48	12.35
i02	2.1	0.8	3.0	7.45	5.15	4.28
i03	4.0	2.2	6.1	13.91	9.30	6.91
i04	3.3	0.9	4.2	10.60	6.91	5.99
i05	5.0	1.8	6.7	12.01	6.46	4.56
i06	6.6	3.4	9.7	24.11	15.86	12.05
i07	4.0	0.7	4.7	8.60	4.22	3.52
i08	6.2	3.2	9.1	16.94	9.70	6.31
i09	3.6	1.1	4.7	18.63	14.34	13.09
i10	2.8	2.9	5.5	16.32	13.03	9.89
i11	8.2	2.0	10.0	23.98	13.82	11.62
i12	5.6	2.1	7.6	22.94	16.00	13.58
i13	4.6	1.8	6.3	11.69	6.54	4.71
i14	6.0	1.1	7.0	11.03	4.38	3.24
i15	6.4	1.3	7.6	12.36	5.17	3.86
i16	12.1	1.1	13.1	25.73	10.56	9.31
i17	0.3	1.2	1.5	3.04	2.73	1.48
i18	4.7	0.1	4.8	18.51	12.94	12.86
i19	6.0	3.7	9.3	22.38	15.07	10.97
i20	1.5	1.6	3.0	9.19	7.53	5.87
i21	5.4	1.4	6.6	16.70	10.44	8.95
i22	8.3	3.3	11.2	22.02	11.92	8.37
i23	7.4	2.2	9.4	21.56	12.56	10.19
i24	4.6	0.6	5.2	13.10	7.86	7.22
i25	2.8	2.2	4.8	15.55	12.35	9.96
i26	5.9	1.9	7.6	18.89	11.91	9.86
i27	7.0	0.5	7.5	15.62	7.53	6.98
i28	4.1	1.7	5.7	14.05	9.35	7.51
i29	5.4	1.1	6.5	10.55	4.53	3.36
i30	6.7	1.7	8.3	20.61	12.50	10.64
AVG	5.2	1.7	6.8	16.0	9.8	8.0

6 Conclusions

This work aimed to study the continuous BAP. Within this context, we sought to contribute to the improvement of logistics in the distribution of the quay space by minimizing the total service time of ships.

To solve the continuous BAP, we have proposed an application of the hybrid method known as Clustering Search (CS) with the Simulated Annealing metaheuristic. The CS shown to be effective and appropriate to locate promising regions in the search space by the clusters exploration. Thus, we believe that CS is useful as an alternative to find good solutions for the continuous BAP. This fact becomes evident when the results are compared directly with the TS reported in [6] and the MA presented in [19].

Overall, the results showed that CS was able to generate good quality solutions for all the instances in low computational times. These results were also

compared against other recent approaches in the literature and the solutions were more favorable in all the cases.

Acknowledgments. The authors thank National Council for Scientific and Technological Development - CNPq (processes 300692/2009-9, 300747/2010-1 and 477148/2011-5) and CAPES for their financial support. Thanks are due to the anonymous referees for their valuable suggestions.

References

1. Barros, V.H., Costa, T.S., Oliveira, A.C.M., Lorena, L.A.N.: Model and heuristic for berth allocation in tidal bulk ports with stock level constraints. Computers & Industrial Engineering 60, 606–613 (2011)
2. Bierwirth, C., Meisel, F.: A survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research 202(3), 615–627 (2010)
3. Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J., Lusby, R.: Models for the discrete berth allocation problem: a computational comparison. Transportation Research Part E: Logistics and Transportation Review 47, 461–473 (2011)
4. Chaves, A.A., Lorena, L.A.N.: Hybrid evolutionary algorithm for the capacitated centered clustering problem. Expert Systems with Applications 38(5), 5013–5018 (2010)
5. Cheong, C.Y., Tan, K.C., Liu, D.K., Lin, C.J.: Multi-objective and prioritized berth allocation in container ports. Annals of Operations Research 180(1), 63–103 (2010)
6. Cordeau, J.F., Laporte, G., Legato, P., Moccia, L.: Models and tabu search heuristics for the berth allocation problem. Transportation Science 39, 526–538 (2005)
7. Guan, Y., Xiao, W.Q., Cheung, R., Li, C.L.: A multiprocessor task scheduling model for berth allocation: heuristic and worst-case analysis. Operations Research Letters 30, 343–350 (2002)
8. Hamming, R.W.: Error detecting and error correcting codes. Bell System Technical Journal 26(2), 147–160 (1950)
9. Hansen, P., Oguz, C., Mladenovic, N.: Variable neighborhood search for minimum cost berth allocation. European Journal of Operational Research 191, 636–649 (2008)
10. Imai, A., Nishimura, E., Papadimitriou, S.: The dynamic berth allocation problem for a container port. Transportation Research Part B: Methodological 35, 401–417 (2001)
11. Imai, A., Nishimura, E., Papadimitriou, S.: Berth allocation with service priority. Transportation Research Part B: Methodological 37, 437–457 (2003)
12. Imai, A., Sun, X., Nishimura, E., Papadimitriou, S.: Berth allocation in a container port: using a continuous location space approach. Transportation Research Part B 39(3), 199–221 (2005)
13. Imai, A., Chen, H.C., Nishimura, E., Papadimitriou, S.: The simultaneous berth and quay crane allocation problem. Transportation Research Part E 44, 900–920 (2008)
14. Kim, K., Moon, K.: Berth scheduling by simulated annealing. Transportation Research B 37, 541–560 (2003)

15. Lee, D.H., Chen, J.H., Cao, J.X.: The continuous berth allocation problem: a greedy randomized adaptive search solution. Transportation Research Part E 46, 1017–1029 (2010)
16. Li, C.L., Cai, X., Lee, C.Y.: Scheduling with multiple-job-on-one-processor pattern. IIE Transactions 30, 433–445 (1998)
17. Lim, A.: The berth scheduling problem. Operations Research Letters 22, 105–110 (1998)
18. Mauri, G.R., Oliveira, A.C.M., Lorena, L.A.N.: A Hybrid Column Generation Approach for the Berth Allocation Problem. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 110–122. Springer, Heidelberg (2008)
19. Mauri, G.R., Andrade, L.N., Lorena, L.A.N.: A memetic algorithm for a continuous case of the berth allocation problem. In: ECTA 2011 - International Conference on Evolutionary Computation Theory and Applications, Paris, France (2011)
20. Monaco, M.F., Sammarra, M.: The berth allocation problem: a strong formulation solved by a lagrangean approach. Transportation Research Part B 41(2), 265–280 (2007)
21. Nishimura, E., Imai, A., Papadimitriou, S.: Berth allocation planning in the public berth system by genetic algorithms. European Journal of Operational Research 131, 282–292 (2001)
22. Oliveira, A.C.M., Lorena, L.A.N.: Hybrid evolutionary algorithms and clustering search. SCI, vol. 75, pp. 77–99 (2007)
23. Park, Y.M., Kim, K.H.: A scheduling method for Berth and Quay cranes. OR Spectrum 25, 1–23 (2003)
24. Seyedalizadeh Ganji, S.R., Babazadeh, A., Arabshahi, N.: Analysis of the continuous berth allocation problem in container ports using a genetic algorithm. Journal of Marine Science and Technology 15, 408–416 (2010)
25. Thurman, K.P.: Optimal ship berthing plans. Dissertation (Masters of Science in Operations Research) - Naval Postgraduate School, Monterey, California - EUA (1989)
26. UNCTAD: Review of maritime transport. United Nations conference on trade and development (2010)
27. Xu, D., Li, C.L., Leung, J.Y.: Berth allocation with time-dependent physical limitations on vessels. European Journal of Operational Research 216(1), 47–56 (2012)

Combining Heuristic and Exact Methods to Solve the Vehicle Routing Problem with Pickups, Deliveries and Time Windows

Penny L. Holborn, Jonathan M. Thompson, and Rhyd Lewis

School of Mathematics, Cardiff University, Cardiff, Wales, UK
{holbornpl}@Cardiff.ac.uk

Abstract. The vehicle routing problem with pickups, deliveries and time windows (PDPTW) is an important member in the class of vehicle routing problems. In this paper a general heuristic to construct an initial feasible solution is proposed and compared with other construction methods. New route reconstruction heuristics are then shown to improve on this. These reconstruction heuristics look to reorder individual routes and recombine multiple routes to decrease the number of vehicles used in the solution. A tabu search scheme where the attribute to be recorded has been specifically adapted to the PDPTW is proposed. A new method based on branch and bound optimisation attempts to optimise the final ordering of requests in routes to further improve the solutions. Results are analysed for a standard set of benchmark instances and are shown to be competitive with the state of the art.

Keywords: Vehicle Routing, pickup and delivery and tabu search.

1 Introduction

The Vehicle Routing Problem (VRP) plays a central role in distribution management. It can be described as the problem of designing a set of routes that start at a depot and visit a set of geographically scattered customer locations, subject to a variety of side constraints. The VRP is known to be *NP*-hard due to it being an extension of the well known Travelling Salesman Problem (TSP), which is itself *NP*-hard. A helpful survey paper on the VRP is that of Laporte & Osman [1].

The Pickup and Delivery Problem with Time Windows (PDPTW) was first formulated by Savelsbergh and Sol [2]. The constraints are as follows: (1) each vehicle must start at the depot and return to the depot before the end of its operating interval; (2) a request's pickup must be scheduled before its corresponding delivery; (3) loads present within a vehicle at any one time must not exceed the maximum capacity of that vehicle; and (4) requests' pickup and delivery time windows must be adhered to. Note that a vehicle may wait at a location if some waiting time is expected at the vehicle's next destination and the problem is one where all requests are known in advance and no uncertainty exists.

Early research surrounding the PDPTW was concerned with the transportation of people instead of goods and is sometimes known as the dial a ride problem

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 63–74, 2012.

(DARP) [3,4,5]. The first metaheuristic proposed to solve the PDPTW was the reactive tabu search approach of Nanry and Barnes [6]. A two phase method proposed by Lau and Liang [7] developed this, where a construction heuristic was followed by tabu search. Another approach is that of Li and Lim [8] who have applied a tabu-embedded simulating annealing algorithm to solve the problem. They also produced benchmark instances for the PDPTW which are generated from Solomon's 56 benchmark instances [9]. These have since been used as the main basis for comparison of algorithms to solve the problem. Alternatively a two-stage hybrid algorithm has been presented by Bent [10] where the first stage uses a simple simulated annealing algorithm to decrease the number of routes, while the second stage uses large neighbourhood search to decrease the total travel cost. An adaptive large neighbourhood search heuristic has also been proposed by Ropke [11]. Another approach to this problem is by Pankratz [12], who use a grouping genetic algorithm (GGA) and this is extended to a multi-strategy grouping genetic algorithm (MSGGA) by Ding et al. [13]. In addition Dergis and Döhmer [14] show that the approach of indirect local search with greedy decoding gives results which are competitive with both [8] and [12]. Metaheuristics that apply learning mechanisms have been proposed by Lim et al. [15], specifically a squeaky wheel optimisation and more recently ant colony System was applied by Carabetti et al. [16].

In the design of our algorithm we first examine methods previously discussed in the literature, such as initial construction heuristics, neighbourhood search operators and tabu search. We will examine reconstruction heuristics previously applied to the TSP and VRP and look to adapt and evolve these to the PDPTW. Finally we augment our algorithm with a branch and bound method in order to improve the results.

The rest of the paper is organised as follows. The problem is formulated in Section 2. Section 3 provides details on the operators used in our algorithm including the construction of initial feasible solutions, route reconstructions and the branch and bound method. Section 4 provides information on the tabu search heuristic and our algorithmic framework. Finally Section 5 gives computational results and Section 6 provides a conclusion and directions for future research.

2 Problem Formulation

To define the PDPTW, let $V = \{v_0, v_1, \ldots v_n\}$ be a set of geographically dispersed locations where v_0 denotes the depot and n is even. The set $N = V \setminus \{v_0\}$ defines the set of pickup and delivery requests and is partitioned into two subsets of equal size. The subset N^+ denotes the set of pickup locations and N^- the set of delivery locations. Therefore, $N^+ \cup N^- = N$, $N^+ \cap N^- = \emptyset$ and $|N^+| = |N^-| = \frac{n}{2}$ = number of pickup and delivery requests. In this problem each location $v_i \in V$ has an associated demand q_i, $(q_0 = 0)$, a service time s_i, $(s_0 = 0)$ and a service time window $[e_i, l_i]$, $(e_0 = l_0 = 0)$, where e_i, is the earliest time that service at location i can begin and l_i, the latest time that service at location i can begin. With regards to the demand, $q_i > 0$ for $v_i \in N^+$ and $q_i < 0$

for $v_i \in N^-$. For each pair of nodes (v_i, v_j) $(0 \leq i \neq j \leq n)$ a non-negative distance d_{ij} is known, $d_{ij} = d_{ji}$, where distance is equal to time. If a vehicle reaches node v_i before time e_i, it needs to wait until e_i before the service can take place. Let A_i be the arrival time, D_i be the departure time and W_i the waiting time at location i. Then $D_i = \max\{A_i, e_i\} + s_i$. If $A_i < e_i$, then the vehicle has to wait at location i and $W_i = e_i - A_i$. Let M be the number of vehicles, C be the maximum length of operating interval and Q the maximum capacity of each vehicle.

To formulate the PDPTW, two variables are introduced:

$$x_{ij}^k = \begin{cases} 1, \text{ if vehicle } k \text{ goes from node } i \text{ to node } j \\ 0, \text{ otherwise.} \end{cases}$$

$$y_j = \text{load of the vehicle at node } j, \text{ after service at } j$$

and a constant:

$$z_{ij} = \begin{cases} 1, \text{if node } i \text{ and node } j \text{ are the corresponding pickup and} \\ \text{delivery nodes of a single request} \\ 0, \text{otherwise.} \end{cases}$$

The constraints are as follows:

$$\sum_{k=1}^{M} \sum_{j=1}^{n} x_{ij}^k = 1, \forall i \in V \tag{1}$$

$$\sum_{i=1}^{n} x_{i0}^k = 1, k \in [M] \tag{2}$$

$$\sum_{j=1}^{n} x_{0j}^k = 1, k \in [M] \tag{3}$$

$$\sum_{i=1}^{n} x_{ih}^k - \sum_{j=1}^{n} x_{hj}^k = 0, \forall h \in V, k \in [M] \tag{4}$$

$$z_{ij} = 1 \Rightarrow \sum_{l=1}^{n} x_{li}^k - \sum_{p=1}^{n} x_{pj}^k = 0, \forall i, j \in V, k \in [M] \tag{5}$$

$$y_j \leq Q, \forall j \in V \tag{6}$$

$$x_{ij}^k = 1 \Rightarrow y_i + q_i = y_j, \forall i, j \in V, k \in [M] \tag{7}$$

$$x_{ij}^k = 1 \Rightarrow D_i + d_{i,j} \leq A_j \Rightarrow A_j \leq D_j \Rightarrow D_i \leq D_j,$$

$$D_0 = 0, \forall i, j \in V, k \in [M] \tag{8}$$

$$z_{ij} = 1 \Rightarrow A_i \leq A_j, \forall i, j \in V \tag{9}$$

$$A_0 \leq C \tag{10}$$

In the above, constraint 1 ensures that each location is visited exactly once, while constraints 2 and 3 ensure that each vehicle departs from and arrives at the depot. Constraint 4 ensures that if a vehicle arrives at a location then it must also depart from that location. Constraint 5 ensures that the pickup and delivery of a request is carried out by exactly one vehicle. Constraints 6 and 7 together form the capacity constraints. Finally, the time window and precedence constraints are ensured by 2 and 9 and the constraint on the maximum operating interval is ensured by 10. The objective function is:

$$\text{Minimise} \sum_{k=1}^{M} \sum_{i,j \in N: i \neq j} d_{ij} x_{ij}^{k} \tag{11}$$

3 Algorithm Operators

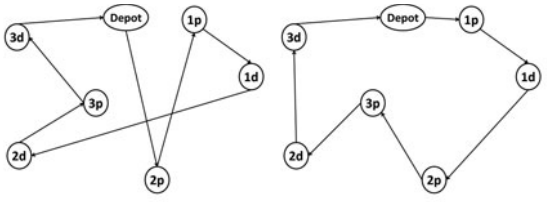
3.1 Construction Methods

To construct an initial feasible solution a combination of random and greedy heuristics are applied. The algorithm builds a feasible solution by inserting, at each iteration, a random un-routed request into a current partial route or into a new route using a greedy method. All feasible insertions of both the pickup and delivery request are examined. The insertion which provides the minimal increase in cost to the solution is accepted. This includes the option of inserting the request into a new route. A similar method is also used in [12] and [14], where it is shown that adding an element of randomness generates varied initial solutions which are beneficial when applying neighbourhood operators as a larger search space is examined.

Preliminary results have shown that this method outperforms the simple greedy heuristic of Nanry and Barnes [6], which at each iteration inserts the request from *all* remaining requests that involves the lowest additional cost to the objective function. It also outperforms the method used by Li and Lim [8], which first initialises a route with a request using criteria based on the maximum increase to the objective function with routes then being completed using a greedy method.

3.2 Route Reconstruction Heuristics

To attempt to improve on the initial solutions constructed we first examine 2 neighbourhood operators of Li and Lim [8]. The first of these is a *shift* operator. This denotes a reassignment of a request from one route to another. Secondly an *exchange* operator swaps a request from one route with a request of another. In both cases infeasible exchanges are forbidden and the operators attempt to insert a request into a route without making any change to the current ordering of that route. Naturally a higher proportion of neighbourhood moves will be



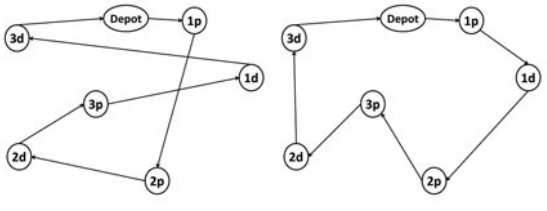
(a) Before reconstruction (b) After reconstruction

Fig. 1. Example of applying the single move within a route reconstruction

seen to retain feasibility if the existing ordering in a route can also be changed, though of course this will also bring additional overheads. To achieve this we suggest three different reconstruction heuristics.

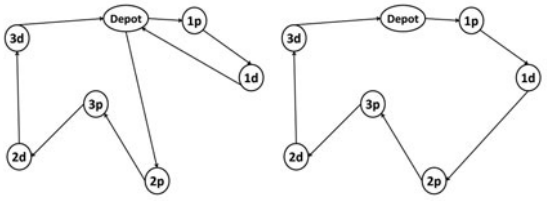
The **single move within a route** heuristic randomly selects a request and removes its pickup and delivery location from its route. It then attempts to insert the pickup and delivery locations in all other feasible positions within that route. If one exists, the insertion position which amounts to the largest reduction in distance is accepted. An example is shown in Figure 1. This method is based on that of *Or*-opt exchanges, see Or [17] but is adapted to the PDPTW.

The **single route reconstruction** attempts to reorder an entire route by first removing all requests from that route and re-inserting them based on three different methods. These are as follows: (1) by allocating at each iteration the location at which the next service can begin first, (this is the maximum of the time the vehicle can arrive at a location and the opening of the time window at that location); (2) by allocating the first pickup location to the route at random and each of the remaining pickup or delivery locations greedily; and (3) by allocating the first pickup whose location is the maximum distance from the depot first and then each of the remaining pickup or delivery locations greedily. Each location (pickup or delivery) is inserted separately. An example of this is shown in Figure 2. The single route reconstruction also attempts to reform a route whilst inserting a request from another. It attempts to find a feasible solution that includes the insertion of this request whilst also minimsing the overall total distanced travelled over all routes.



(a) Before reconstruction (b) After reconstruction

Fig. 2. Example of applying the single route reconstruction to a route



(a) Before reconstruction (b) After reconstruction

Fig. 3. Example of applying the multiple route reconstruction to two routes

Finally the **multiple route reconstruction** attempts to form multiple routes simultaneously. This is carried out for two routes with the aim of reducing to one or three routes with the aim of reducing to two. All requests are removed from the routes and the first route is initialised with the pickup whose location is the maximum distance from the depot. For the case of the second route, if one is used, the pickup which is maximum distance from the first is chosen. The routes are then reconstructed simultaneously using a greedy heuristic. For the second case this is only applied on a combination of routes, if at least one of the routes is an outlier with regards to the number of requests present. An example of this is shown in Figure 3.

3.3 Branch and Bound Method

To further improve our algorithm a method based on the large neighbourhood search (LNS) of [10] for the PDPTW is incorporated. The main idea is to take a part of a solution (in this case a single route or subset of that route) and find the optimal solution for this sub-problem via a branch and bound routine.

The process starts with a set of currently adjacent locations. According to the constraints of the problem, partly constructed solutions can be discarded: (a) if the delivery location of a request is inserted before the corresponding pickup; (b) if there remains a location still to be inserted that can no longer be feasibly serviced within its time window; (c) if a location cannot be feasibly serviced within its time window when placed after another location; (d) if the current total distance travelled exceeds the minimum recorded so far; and (e) if the minimum distance still to travel plus the current distance exceeds the minimum found. The limit of the initial bound is the total distance travelled of the route before the locations are removed. Branches are searched in the order of location where service can begin first. The search terminates once a complete exploration has taken place and the best solution is returned.

As this method is an exact approach it can be computationally expensive. Our results suggest it can be applied to routes with up to 14 locations. In cases where there are more than this, our approach is to apply branch and bound to successive overlapping sub-sections. This ensures locations located closely to one another are optimised in the same sub-section. In cases where $n > 14$ locations, the route is split into $2\left\lceil \frac{n}{14} \right\rceil - 1$ sub-sections. For example, if a route consists of 28 locations it is split into 3 sub-sections containing locations 1-14, 7-21 and 14-28 respectively.

4 Overall Algorithm

To further improve the algorithm a tabu search heuristic is to be added to the *shift* operator defined in Section 3.2. Within the the *shift* operator the request to be reassigned is selected at random and all feasible insertion positions of both the pickup and delivery are examined. If one is found, the insertion which provides the largest reduction in total travel distance is accepted. It is found that adding the *exchange* operator, both increased computational times and did not provide an improvement to the results when used in conjunction with the reconstruction heuristics.

From the literature, a tabu length and cycle length proportional to the number of requests to be serviced generally yields the most positive results, see [6] who present a reactive tabu search approach to solve the PDPTW. Our tabu search heuristic follows the general guidelines provided in [18] and a tabu tenure equal to the number of requests, $|N|$, and a cycle length equal to the number of locations to be visited, $|2N|$ are found to be most promising. The stopping condition is based on achieving a given number of iterations without improvement to the objective function or there being no more feasible moves.

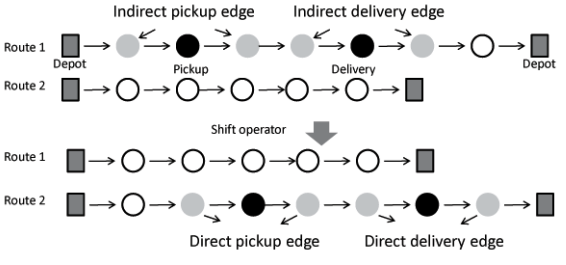
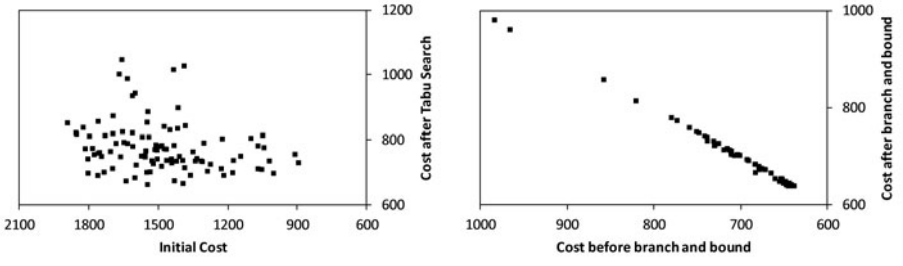


Fig. 4. Example of the tabu attributes added to the tabu list during a move by the Shift operator

The attribute to be stored within the tabu list follows the approach of [19] where edges removed and inserted within a solution are recorded for the VRP with simultaneous pick-up and delivery. In our case this is adapted to the PDPTW where the edges inserted into a solution are classed as the direct edges. These are the locations either side of the new insertions i.e. the locations before and after the insertion of the pickup location and the delivery location of a request. The edges removed from the solution are the indirect edges, i.e. the the locations before and after the pickup and delivery location that has been removed. The attribute to be recorded in the tabu list consists of both the inserted and removed pickup and delivery edges. If the arrangement of locations in a route after the insertion results in both the pickup and delivery edge being either directly or indirectly tabu then the move is disallowed. An example of the attribute to be stored in this tabu list is shown in Figure 4.

Preliminary results suggest that applying the branch and bound method as a final phase to the algorithm and not within the improvement phase yields

promising results. The branch and bound method optimises both routes and sub-sections of routes therefore if intertwined within the improvement phase it limits the number of successful tabu moves and reconstructions as routes and sub-sections of routes are at a local minima. Applying the tabu search heuristic and the branch and bound method are computationally expensive. Investigations are performed to determine if each method may be performed on a subset of the most promising solutions and still achieve competitive results. Figure 5 contains a plot of the cost after the initial construction phase compared to the cost after applying the tabu search heuristic, and a plot of the the cost after the tabu search phase compared to the cost after applying the branch and bound method. Both for 100 random trials with the instance lc204. There is no correlation between achieving a lower cost for an initial solution and achieving a lower cost for the final solution after the tabu search heuristic. Therefore it will not be possible to select a proportion of initial solutions with a certain initial cost to apply the tabu search heuristic to achieve the most promising results. However there is a direct correlation between achieving a lower cost after the tabu search heuristic and achieving a lower cost after the branch and bound method. Therefore it seems reasonable to select a small subset of of low cost solutions after application of the tabu search heuristic which can then be improved via our branch and bound method.



(a) Tabu Search heuristic (b) Branch and bound method

Fig. 5. Scatter plot showing the before and after costs of each heuristic for 100 random trials with instance lc204

Our overall algorithm consists of first constructing an initial feasible solution which is passed to the improvement phase consisting of the tabu search and reconstruction heuristics outlined in Section 3.2 and above. These methods are carried out until no further improvement can be made. This process is repeated for a given number of iterations and the best 10% of solutions are passed to the branch and bound procedure outlined in Section 3.3. This method differs from others in the literature as a single run involves multiple restarts and only a portion of the best found solutions are passed to the final improvement phase. Preliminary results suggest that applying 300 iterations of the initial construction and improvement phase before passing 30 of the best found solutions to the branch and bound method yields run times that improve on [12] and are similar to that of [13]. This comparison is made using the average run time provided

and multiplying by the 30 trials that were required to achieve their best found solutions. Only computational times for 10 runs of the algorithm of [14] are provided in the literature and are a significant improvement on the others stated, however applying this figure to the case of 30 runs, these would also be comparable to our algorithm. The computational times of [8] will not be compared as their description does not indicate whether these are average values. Caution is needed when making a direct comparison between these computational times due to differences in computer specification.

5 Experimental Results

For our experiments we used instances derived by [8], which are based on Solomon's 56 VRPTW 100-customer instances [9]. Each has 100-106 nodes, (i.e. 50-53 requests) and they are organised into 6 classes; lc1 and lc2 are clustered instances; lrc1 and lrc2 are those where requests are partially clustered and partially random; and lr1 and lr2 have randomly distributed requests. Instances ending in a 1 have a short scheduling horizon and those with a 2 have a longer scheduling horizon. Table 1 compares the results of our algorithm with those of Li and Lim [8], Pankratz [12], Dergis & Döhmer [14] and Ding et al. [13].

Caution is needed when making a direct comparisons with these results as our objective function is to minimise the total travel distance which is comparable to that of [12]. Li and Lim [8] however use a prioritised objective function with the order being: (1) minimise number of vehicles; (2) minimise total travel distance; (3) minimise total schedule duration; and (4) minimise total waiting time. The objective of Ding et al. [13] is similar to this although it does not include minimising the total schedule duration. The objective of [14] is to minimise the number of vehicles followed by minimising the total travel distance. For the approach of [8] the overall number of independent runs per instance is not reported and the average solution quality is not discussed. The best results of [12] and [13] are reported after 30 runs of their algorithm and for [14] best results are reported after 10 runs. Our algorithms are implemented using C++ and executed on a PC under Windows XP with a 3.10GHz processor.

Considering the results in Table 1 our algorithm achieves the best known solutions for 51 of the problem instances and with a total travel distance of 57662.02 are competitive with the state of the art. For [8] 40 of the best known solutions are achieved with a total travel distance of 58184.91, [14] achieve 42 with a total travel distance of 57678.4 and [13] achieve 51 with a total travel distance of 57652.05. The minimal total travel distance of 57638.48 is achieved by [12], however with only 42 of the best known solutions found.

The initial improvement phase of our algorithm achieves an average total travel distance for the 56 instances of 61162.92. This is a 40% improvement from 101883.30 for the construction phase alone. This is reduced by the branch and bound method to 58302.08, a further decrease of 4.7%. For the branch and bound method the average decrease in cost for the instances with a longer scheduling horizon is 6.8 % with a 10.0% decrease in the lr2 instances. For the instances with a longer scheduling horizon, the number of vehicles is dramatically

Table 1. Summary of results

	Holborn et al.					Li & Lim [8]	Pankratz [12]	Dergis & Döhmer [14]	Ding et al. [13]
Instance	**TD**[1]	**NV**[2]	**TDI**[3]	**TDB**[4]	**CT**[5]	**TD**[1]	**TD**[1]	**TD**[1]	**TD**[1]
lc101	828.94	10	828.94	828.94	92	828.94	828.94	828.94	828.94
lc102	828.94	10	828.94	828.94	367	828.94	828.94	828.94	828.94
lc103	827.86	10	830.58	827.87	679	827.86	827.86	827.86	827.86
lc104	818.60	9	828.17	818.60	1135	861.95	818.60	860.01	860.01
lc105	828.94	10	828.94	828.94	131	828.94	828.94	828.94	828.94
lc106	828.94	10	829.76	828.94	102	828.94	828.94	828.94	828.94
lc107	828.94	10	828.94	828.94	99	828.94	828.94	828.94	828.94
lc108	826.44	10	827.05	826.44	232	826.44	826.44	826.44	826.44
lc109	827.82	10	837.66	827.82	714	827.82	827.82	827.82	827.82
Total lc1	**7445.41**	**89**	**7468.96**	**7445.41**	**3550**	**7488.77**	**7445.42**	**7486.83**	**7486.83**
lr101	1650.80	20	1658.69	1650.80	44	1650.78	1650.80	1650.80	1650.80
lr102	1487.57	19	1515.57	1487.57	459	1487.57	1487.57	1487.57	1487.57
lr103	1292.68	14	1322.47	1292.68	391	1292.68	1292.68	1292.68	1292.68
lr104	1013.39	12	1095.19	1026.02	867	1013.39	1013.99	1013.99	1013.39
lr105	1377.11	16	1383.94	1377.11	142	1377.11	1377.11	1377.11	1377.11
lr106	1252.62	13	1261.18	1252.62	215	1252.62	1252.62	1252.62	1252.62
lr107	1111.31	12	1146.91	1111.31	475	1111.31	1111.31	1111.31	1111.31
lr108	968.97	11	989.88	968.97	438	968.97	968.97	968.97	968.97
lr109	1208.96	15	1271.37	1208.96	317	1239.96	1208.96	1208.96	1208.96
lr110	1165.83	14	1206.88	1166.90	696	1159.35	1165.83	1159.35	1159.35
lr111	1108.90	14	1144.68	1108.90	640	1108.90	1108.90	1108.90	1108.90
lr112	1003.77	12	1059.20	1021.80	929	1003.77	1003.77	1003.77	1003.77
Total lr1	**14641.91**	**172**	**15055.96**	**14673.64**	**5614**	**14666.41**	**14642.51**	**14636.03**	**14635.43**
lrc101	1703.21	14	1731.58	1703.21	98	1708.80	1703.21	1708.80	1708.80
lrc102	1558.07	12	1612.43	1558.07	300	1563.55	1558.07	1558.07	1558.07
lrc103	1258.74	11	1294.23	1258.74	417	1258.74	1258.74	1258.74	1258.74
lrc104	1128.40	10	1141.07	1128.40	481	1128.40	1128.40	1128.40	1128.40
lrc105	1637.62	13	1670.72	1640.15	290	1637.62	1637.62	1637.62	1637.62
lrc106	1424.73	11	1510.12	1439.66	242	1425.53	1424.73	1424.73	1424.73
lrc107	1230.14	11	1265.43	1230.14	357	1230.15	1230.14	1230.14	1230.15
lrc108	1147.43	10	1203.62	1147.43	369	1147.97	1147.43	1147.96	1147.43
Total lrc1	**11088.34**	**92**	**11429.20**	**11105.80**	**2555**	**11100.76**	**11088.34**	**11094.46**	**11093.94**
lc201	591.56	3	591.56	591.56	284	591.56	591.56	591.56	591.56
lc202	591.56	3	598.88	591.56	1416	591.56	591.56	591.56	591.56
lc203	591.17	3	625.56	591.17	1776	585.56	591.17	591.17	591.17
lc204	590.60	3	664.14	635.01	2835	591.17	590.60	590.60	590.60
lc205	588.88	3	603.78	588.88	1127	588.88	588.88	588.88	588.88
lc206	588.49	3	618.24	602.06	2316	588.49	588.49	588.49	588.49
lc207	588.29	3	598.08	588.29	1125	588.29	588.29	588.29	588.29
lc208	588.32	3	606.66	588.32	1237	588.32	588.32	588.32	588.32
Total lc2	**4718.87**	**24**	**4906.89**	**4776.84**	**12116**	**4713.83**	**4718.87**	**4718.87**	**4718.87**
lr201	1253.23	4	1297.60	1254.29	1756	1263.84	1253.23	1253.23	1253.23
lr202	1197.67	3	1355.43	1261.68	2602	1197.67	1197.67	1197.67	1197.67
lr203	949.40	3	1128.34	970.32	3809	949.40	952.29	949.40	949.40
lr204	849.05	2	1042.35	904.49	6153	849.05	849.05	849.05	849.05
lr205	1054.02	3	1168.50	1073.66	3327	1054.02	1054.02	1054.02	1054.02
lr206	931.63	3	1083.69	935.75	3800	931.63	931.63	931.63	931.63
lr207	905.45	2	1094.37	972.03	5471	903.06	903.60	903.06	903.05
lr208	734.85	2	882.29	752.46	7266	734.85	736.00	734.85	734.85
lr209	930.59	3	1049.56	947.17	3657	937.05	932.43	930.59	930.59
lr210	964.22	3	1108.68	988.93	3650	964.22	964.22	964.22	964.22
lr211	884.29	3	978.92	905.68	4976	927.80	888.15	896.76	884.29
Total lr2	**10654.39**	**31**	**12189.72**	**10966.46**	**46467**	**10712.59**	**10662.29**	**10664.48**	**10652.00**
lrc201	1406.94	4	1491.96	1436.85	1705	1468.96	1407.21	1406.94	1406.94
lrc202	1374.27	3	1508.18	1403.33	2600	1374.27	1385.25	1374.27	1374.27
lrc203	1089.07	3	1231.43	1127.13	3502	1089.07	1093.89	1089.07	1089.07
lrc204	818.66	3	920.82	826.70	4415	827.78	818.66	818.66	818.66
lrc205	1302.20	4	1423.97	1326.12	1989	1302.20	1302.20	1302.20	1302.20
lrc206	1159.03	3	1253.95	1185.97	3000	1162.91	1159.03	1159.03	1159.03
lrc207	1062.05	3	1241.62	1091.31	3311	1424.60	1062.05	1062.05	1062.05
lrc208	900.89	3	1040.26	936.53	4274	852.76	852.76	865.51	852.76
Total lrc2	**9113.12**	**26**	**10112.19**	**9333.94**	**24797**	**9502.55**	**9081.05**	**9077.73**	**9064.98**
Total	**57662.02**	**434**	**61162.92**	**58302.08**	**95099**	**58184.91**	**57638.48**	**57678.40**	**57652.05**

[1] Total travel distance of best found solution.

[2] Number of vehicles required for best found solution.

[3] Average total distance travelled for 300 runs of the initial improvement phase.

[4] Average total distance travelled for 30 runs of the branch and bound method using the best found solutions from the improvement phase.

[5] Total computational time in seconds for one run of the algorithm including 300 runs of the initial improvement phase and 30 runs of the branch and bound method on the 30 best found solutions.

reduced compared to the instances with a shorter scheduling horizon, resulting in significantly more requests allocated to each vehicle. Therefore the problem now becomes one of finding the best ordering of requests to a route rather than the allocation of requests to routes. In the case of the lr2 instances this becomes increasingly difficult as locations are randomly dispersed, hence the success of a method which specifically focuses on optimising large portions of locations in routes such as our branch and bound method. Due to this our algorithm performs consistently well across the varying instance types whereas [8] and [12] struggle with the instances of a longer scheduling horizon, in particular lr2 and lrc2. The average coefficient of variation across each of the 6 classes of instances ranges from 1% to 7% for the results after the initial improvement phase and is reduced to less than 2% for all results after the branch and bound method.

For the instances lc104 and lrc101, a solution has been found by both [14] and [13] (and by [8] for the case of lrc101), that uses one less vehicle by increasing the total travel distance in the solution. This is the best found solution when the objective is to first minimise the number of vehicles, however these are the only two cases which do not share an identical best found solution. This shows the robustness of our algorithm to changes in the objective function as it also achieved the two solutions stated above but they were disregarded due to the increase in distance. Finally it should be noted that for [8] and [12] Euclidean distances calculated directly from the instances were rounded to 2 decimal places and this could account for some small discrepancies when comparing the total distance travelled.

6 Conclusions

We have shown that the methods applied in this paper generate results which are competitive with the state of the art results found in the literature. Our results obtain the best known solutions in 51 out of a possible 56 instances with the algorithm appearing to perform consistently well over all types of instance. One of the main advantages of our approach is the speed of individual constructions. In this case it has allowed us to produce large samples of solutions in times that are consistent with other approaches and by reducing the number of runs of the initial improvement phase still achieves promising results. This advantage can be exploited when applying these methods to the dynamic PDPTW (DPDPTW) where our algorithm will be repeatedly restarted over a rolling horizon framework to incorporate the arrival of new requests and decisions will need to be made in real time. The DPDPTW has received much less interest in the literature, hence this will be the area for future research. For a recent survey on dynamic pickup and delivery problems see [20].

References

1. Laporte, G., Osman, I.: Routing problems: A bibliography. Annals of Operations Research 61, 227–262 (1995)
2. Savelsbergh, M.W.P., Sol, M.: The general pickup and delivery problem. Transportation Science 29(1), 17–29 (1995)

3. Psaraftis, H.: An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. Transportation Science 17(3), 351–357 (1983)
4. Jaw, J., Odoni, A., Psaraftis, H., Wilson, N.: A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research Part B: Methodological 20, 243–257 (1986)
5. Cordeau, J.F., Laporte, G.: The dial-a-ride problem: models and algorithms. Annals of Operations Research 153(1), 29–46 (2007)
6. Nanry, W., Barnes, J.: Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research Part B: Methodological 34(2), 107–121 (2000)
7. Lau, H., Liang, Z.: Pickup and delivery with time windows: algorithms and test case generation. In: The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2001, Dallas, USA, pp. 333–340 (2001)
8. Li, H., Lim, A.: A metaheuristic for the pickup and delivery problem with time windows. In: The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2001, Dallas, USA, pp. 160–167 (2001)
9. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35(2), 254–265 (1987)
10. Bent, R., Van Hentenryck, P.: A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Computers & Operations Research 33(4), 875–893 (2006)
11. Ropke, R., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40(4), 455–472 (2006)
12. Pankratz, G.: A grouping genetic algorithm for the pickup and delivery problem with time windows. OR Spectrum 27, 21–41 (2005)
13. Ding, G., Li, L., Ju, Y.: Multi-strategy grouping genetic algorithm for the pickup and delivery problem with time windows. In: GEC 2009: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp. 97–104. ACM, New York (2009)
14. Dergis, U., Dohmer, T.: Indirect search for the vehicle routing problem with pickup and delivery and time windows. OR Spectrum 30, 149–165 (2008)
15. Lim, H., Lim, A., Rodrigues, B.: Solving the pickup and delivery problem with time windows using "squeaky wheel" optimization with local search. In: Proceedings of AMCIS 2002, pp. 2335–2344 (2002)
16. Carabetti, E., de Souza, S., Fraga, M.: An application of the ant colony system metaheuristic to the vehicle routing problem with pickup and delivery and time windows. In: Eleventh Brazilian Symposium on Neural Networks 2010, pp. 176–181 (2010)
17. Or, I.: Traveling salesman-type combinatorial problems and their relation the logistics of regional blood banking. PhD thesis, Northwestern University, Evanston, IL (1976)
18. Glover, F.: Tabu search part 1. ORSA Journal on Computing 1(3), 190–206 (1989)
19. Montané, F.A.T., Galvão, R.D.: A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Computers and Operations Research 33(3), 595–619 (2006)
20. Berbeglia, G., Cordeau, J.F., Laporte, G.: Dynamic pickup and delivery problems. European Journal of Operational Research 202(1), 8–15 (2010)

D^2MOPSO: Multi-Objective Particle Swarm Optimizer Based on Decomposition and Dominance

Noura Al Moubayed, Andrei Petrovski, and John McCall

Robert Gordon University
School of Computing
Aberdeen, Uk
n.al-moubayed@rgu.ac.uk

Abstract. D^2MOPSO is a multi-objective particle swarm optimizer that incorporates the dominance concept with the decomposition approach. Whilst decomposition simplifies the multi-objective problem (MOP) by rewriting it as a set of aggregation problems, solving these problems simultaneously, within the PSO framework, might lead to premature convergence because of the leader selection process which uses the aggregation value as a criterion. Dominance plays a major role in building the leader's archive allowing the selected leaders to cover less dense regions avoiding local optima and resulting in a more diverse approximated Pareto front. Results from 10 standard MOPs show D^2MOPSO outperforms two state-of-the-art decomposition based evolutionary methods.

1 Introduction

Real-life optimisation problems may have several conflicting objectives. This leads to an irregular multi-objective space where the optimisation method must be able to find trade-off solutions among these objectives.

MOEA/D [1,2] introduced a novel approach to discover Pareto optimal solutions. The original MOP is rewritten as an aggregation of single objective problems. These problems are then solved using Genetic Algorithms (GA). The advantages of this approach in terms of mathematical soundness, algorithmic structure and computational cost are explained in [2]. MOEA/D has been applied successfully on several real-life MOPs [3].

Particle Swarm Optimisation (PSO) is an efficient optimisation method that is capable of providing competitive solutions in many application domains [4,5,6]. Multi-Objective variants of PSO (MOPSO) have recently been developed [7,8]. The authors in [9] argued that although PSO and GA on average yield the same effectiveness (solution quality), PSO is more computationally efficient and uses fewer evaluations. This claim was supported by two statistical tests, which confirmed similar effectiveness of the methods but superior efficiency of PSO over GA. In addition, PSO requires less subjective tuning making it much easier to implement [9].

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 75–86, 2012.

SDMOPSO [10] uses the decomposition approach proposed in MOEA/D. The global best set of each particle is defined by all the solutions located within a certain neighborhood. In SDMOPSO the particle uses the aggregation value as a leader selection criterion; in addition, the updated position is only adopted if its aggregation value is enhanced. As a result, when the particles are unable to find better positions, the swarm falls into a local optima.

dMOPSO [11] uses decomposition to update the leaders' archive and select the leader corresponding to each particle. To maintain the diversity of the swarm, dMOPSO uses memory re-initialization process based on a Gaussian distribution. One drawback of dMOPSO is that the particles re-initialize their memory when they exceed a pre-defined age losing all the experience gained throughout the process and adding more complexity to the algorithm. Besides, it uses decomposition as a way to substitute dominance. With the absence of dominance, the decomposition is left to lead the swarm into a limited number of destinations equal to the swarm size. With complicated Pareto fronts (i.e. disconnected) and due to the limited size of the swarm, dMOPSO might fail to cover the entire PF.

D^2MOPSO utilizes the dominance concept [12] along with decomposition. It uses bounded crowding leaders' archive to store the non-dominated particles. The leader selection is then applied to the archive using the aggregation value as the selection criterion. The particle personal movement trajectory is updated using decomposition. All objectives are normalized in order to give them equal priorities when decomposition is applied. Towards the end of the optimisation process the size of the leaders' archive is substantially reduced to contain only non-dominated particles with the lowest crowding distance aiming at increasing the diversity and covering low dense regions.

2 Multi-objective Particle Swarm Optimisation

The difficulty of multi-objective optimisation is that an improvement in one objective often happens at the expense of deteriorating the performance with respect to other objectives. The optimisation challenge therefore is to find the entire set of trade-off solutions that satisfy the conflicting objectives. The objectives are represented as a vector F in a solution space $\Omega \subset R^n$.

$$F(x) = \{f_1(x), f_2(x), \ldots, f_m(x)\} \tag{1}$$

where $x \in \Omega$, and m is the number of objectives.

When minimizing $F(x)$, a domination relationship is defined between the solutions as follows: let $x, y \in \Omega$, $x \succ y$ if and only if $f_i(x) \leq f_i(y)$ for all $i = \{1, 2, ..., m\}$, and there is at least one j for which $f_j(x) < f_j(y)$. X is a Pareto optimal solution if there is no other solution $y \in \Omega$ such that $y \succ x$.

Pareto optimality of a solution guarantees that any enhancement of one objective would results in worsening of at least one other objective. The image of the Pareto optimal set in the objective space (i.e. $F(x^*)$) is called the Pareto Front (PF)[7]. MOPSO can be used to find the Pareto optimal solutions or to approximate the PF. Each particle in the swarm represents a potential solution in the

solution space and exchanges positional information with the global leader(s) or best local neighbour(s), as well as consulting its own personal memory. These information are then used to move the particle around the search space [7,13]. A particle is characterized by its position and velocity. The position is the location in the solution space, whereas the velocity is a vector representing the positional change. The particle uses the position of the selected leader and its personal movement trajectory to update the velocity and position values.

3 D^2MOPSO Approach

Decomposition transforms the MOP into a set of distinct aggregation problems. The transformed problem then is to solve the aggregated problems. Each particle solves the corresponding problem by assigning a priority to each objective according to a weight vector (λ). This assists the optimisation process to find potential solutions that are evenly distributed along the PF. By associating each particle with a distinct aggregation problem (i.e. λ value), the exploration activity of each particle is focused on a specific region in the objective space and aimed at reducing the distance to the reference point.

Substituting entirely the dominance approach in MOPSO with decomposition (i.e. using the aggregation value instead of dominance as the leaders' selection criterion) might lead to premature convergence as each particle is strictly directed to one destination [2]. At some point during optimisation, the particles would be unable to update their positions and personal best memory as the global best and neighborhood information are not changing. In addition to this, solving a MOP with complicated PF raises a serious challenge as some λ vectors direct the related particles to unpromising areas. In this case, part of the swarm is wasting a large number of evaluations investigating undesirable regions. Another drawback of decomposition is that while solving MOP with high dimensional objective space, it fails to produce a sufficient number of non-dominated solutions that cover the entire PF as the space required to be covered by the swarm using λ vectors grows exponentially with the number of dimensions. To cope with this growth, decomposition based approaches need to use a large swarm to be able to offer a good PF coverage, which increases the number of necessary function evaluations. The number of evaluations any evolutionary method needs to cover the PF is an important features as large number of evaluations counts as a big disadvantage for any EA and especially in real-life problem where evaluation can be very expensive.

To overcome these drawbacks, D^2MOPSO integrates both dominance and decomposition approaches. The bounded crowding leaders' archive [12], where the leaders of the swarm are selected from, is based on the dominance approach where only non-dominated particles are stored. When the archive is full, the none-dominated particles are only added at the low dense regions replacing those ones at the high dense regions. The personal best values are updated and the leaders are selected based on the decomposition's aggregation function.

Decomposition requires an aggregation function to decompose the MOP into several aggregation problems. Many functions have been proposed in the literature (e.g., wighted sum, Tchebycheff, weighted Tchebycheff and Penalty based boundary intersection (PBI)). Recently the weighted PBI method is reported to be of interest [11] and is used in this paper. PBI is originally proposed in [2] by modifying Normal Boundary Intersections (NBI). PBI uses a weighted vector λ and a penalty value θ for minimizing the distance to the utopia vector (i.e. a hypothetical vector between the reference point ($z^* = \min\{f_i(x)|x \in \Omega\}$) and the center of the PF) d_1 and the direction error to the weighted vector d_2 from the solution $F(x)$ in the objective space. PBI is then defined in [1]:

$$minimize \quad g(x|\lambda, z^*) = d_1 + \theta d_2 \tag{2}$$

where

$$d_1 = \parallel (F(x) - z^*)^T \lambda \parallel / \parallel \lambda \parallel, d_2 = \parallel (F(x) - z^*) - d_1 \lambda / \parallel \lambda \parallel \parallel \tag{3}$$

D^2MOPSO uses the PBI approach to decompose the optimisation objective defined by Eq. 1 into N scalar optimisation problems, where N is the swarm's size. By changing the weights and using the reference point defined above, any Pareto optimal solution could be reached [1].

In addition to combining dominance and decomposition, D^2MOPSO normalizes the MOP objectives. As the ranges of the objectives' values can differ considerably and are rarely known a priori for the majority of real life problems, objectives needed to be normalized before aggregation. This ensures equal priorities for all objectives, thereby preventing one objective from dominating the others when the aggregation is applied. The objective values are normalized using a sigmoid limiting transformation function defined in Eq.4. The Sigmoid limiting transformation is chosen as it does not need any prior knowledge of the objectives' ranges.

$$S(f_i(x)) = 1/(1 + e^{-f_i(x)}) \tag{4}$$

In Eq. 3 the normalized value of each objective is used instead of the objective values: $S(F(x)) = (S(f_1(x), f_2(x), \ldots, f_m(x))$ instead of $F(x)$.

Towards the end of the optimisation process the swarm is likely to be converged but the particles are still evaluated till the termination of the algorithm. At the last $\alpha\%$ of the iterations, D^2MOPSO reduces the leaders archive size to $\beta\%$ of its original size. $100 - \beta\%$ of the particles in the original archive are removed according to their crowding distance leaving the particles with the lowest crowding distance in the archive (i.e. particles at the low dense regions in the objective space). The particles then select their leaders from the new archive. This directs all the particles at the end of the optimisation towards the low dense regions enhancing thereby the coverage and diversity. α and β are set based on the convergence of the leaders' archive, i.e. the leaders' archive is no more updated with new none-dominated particles. The following steps summarize D^2MOPSO:

– ***Initialization***:
D^2MOPSO starts by initializing the swarm with N particles and initializing N vectors: $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$, where m is the number of objectives and N is the swarm size. λ vectors are uniformly distributed in $[0,1]^m$ subject to $\sum_{i=1}^{m} \lambda_i = 1$. Every particle is assigned a unique λ vector. A λ vector is selected so that it gives the best aggregated fitness value for the initialized particle. For example, in the case of minimization problems the particle is assigned to the λ vector that minimizes the aggregated fitness, taking into account that each λ is unique and is assigned to only one particle in the swarm. The initial value of the particle's memory *pbest* is its own information as it lacks any experience at the beginning of the process. The initial velocity of the particle is set to zero. The leaders crowding archive is set to a fixed size equal to the swarm size (N), and is initialized using the non-dominated particles in the swarm. The reference point z^* is the vector in the objective space with the best objective values found so far.
– ***Evolution***:
In this phase D^2MOPSO goes through a pre-defined number of iterations. In each iteration each particle defines a local view in the objective space. The particle determines the next move by finding the new velocity and new position using Eq. 5. The new velocity is calculated using *pbest* and the information of one global leader selected from the leaders' archive.

$$\begin{aligned} V_{i_t} &= w * V_{i_{t-1}} + C_1 * r_1 * (x_{pbest_i} - x_{i_{t-1}}) + C_2 * r_2 * (x_{gbest_i} - x_{i_{t-1}}) \\ x_{i_t} &= x_{i_{t-1}} + V_{i_t} \end{aligned} \tag{5}$$

where $pbest_i$ is the personal best performance of $particle_i$, *gbest* is the global best position of the leader selected from the archive, $r_1, r_2 \in [0,1]$ are random values, $w \in [0.1, 0.5]$ is the inertia weight, and $C_1, C_2 \in [1.5, 2.0]$ are the learning factors that take uniformly distributed random values in their predefined ranges. The process of leader selection uses a uniformly distributed random variable $r \in [0,1]$ to decide whether to select the leaders randomly or using their aggregation values (i.e. each particle selects the leader that gives the best aggregation value using the particle's λ) depending on a 0.5 threshold. As discussed before this is done to avoid premature convergence. The aggregation is calculated after normalizing the objectives' values using Eq. 2. After the particle updates its position and velocity, it has to update its *pbest* as well. *pbest* is replaced with the new position only if the new position aggregation value is better than the aggregation value of the current *pbest*. The leaders crowding archive is then updated with new non-dominated particles, if found, subject to the crowding restriction. The reference point is updated if needed. Finally the external archive is updated to contain the new none-dominated particles.
– ***Finalization***:
When the swarm starts to converge, at the last $\alpha\%$ of the iterations, D^2MOPSO reduces the size of the leaders' archive to $\beta\%$. The unwanted

particles are removed from the high dense regions of the archive (i.e. they have the high crowding distance). The leaders are then selected randomly, using a uniformly distributed random variable, from the new archive. The goal is to improve the PF coverage and increase the diversity of the none-dominated particles found. *pbest* positions, new leaders crowding archive, and the external archive are updated as in the previous step. When the algorithm terminates, the content of the external archive is the approximated PF. The pseudo-code of D^2MOPSO is listed in Algorithm 1.

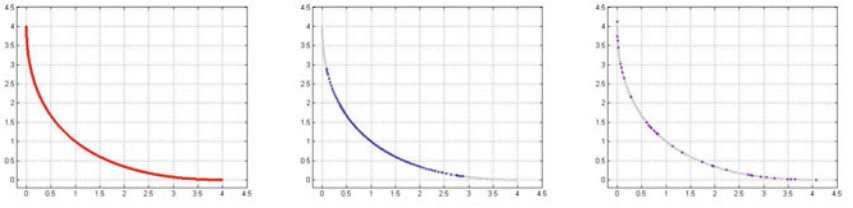
Algorithm 1. D^2MOPSO

```
Initialize the swarm with N particles and N λ vectors
for i = 1 to N do
    assign the particle i to the λ vector that produces the best aggregation
    initialize velocities V = {v_1, ..., v_N} and pbest_i
    Initialize leaders' archive, external archive and z*
end for
Crowding(leader archive)
for i = 1 to MaxIteration do
    if i == 0.9 * MaxIteration then
        reduce leaders archive size
    end if
    for j = 1 to N do
        update Velocity, v_j(t + 1)
        update position, x_j(t + 1)
        evaluate the new position
        normalize objectives and calculate aggregate function for j
        update pbest_j, leaders archive, external archive, and z*
    end for
end for
Return the final result in the external archive
```

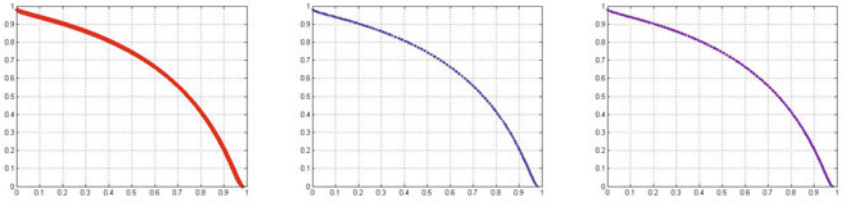
4 Experiments and Results

4.1 Experimental Setup

D^2MOPSO is tested on several standard problems defined in the test suite [14]. 10 problems were chosen (Schaffer, Fonseca, Kursawe, Poloni, Viennet2-3, DTLZ4-6 and DTLZ7) which cover diverse MOPs with convex, concave, connected and disconnected PFs. The method is then compared to MOEA/D [2] and dMOPSO [11]. Each algorithm is run 30 times for each test problem. For the bi-objective problems 100 iterations per run, and 100 particles per generation are used for all methods. For the three-objective problems, 300 iterations and 300 individuals were used. All compared algorithms adopt real encoding, perform the same number of objective evaluations and use the same aggregation function (NBI) with $\theta = 5$. MOEA/D uses differential evolution crossover (DE)

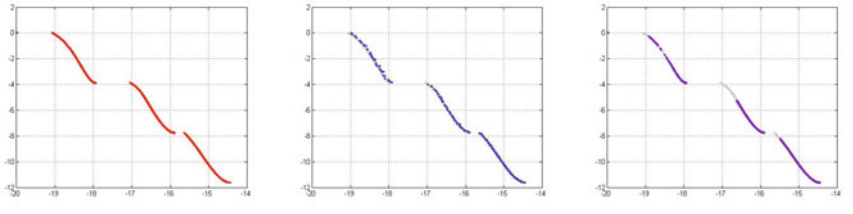


(a) Schaffer: D^2MOPSO (b) dMOPSO (c) MOEA/D

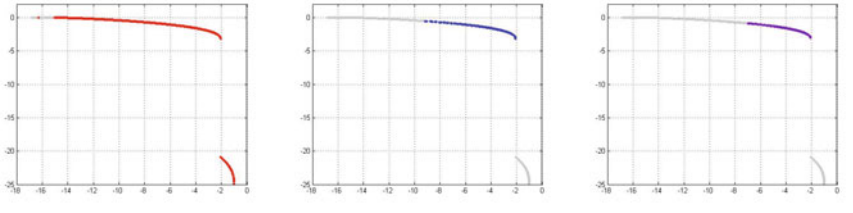


(d) Fonseca: D^2MOPSO (e) dMOPSO (f) MOEA/D

Fig. 1. The $PF_{approximation}$ of Schaffer and Fonseca using the three methods. Grey represents PF_{true}.

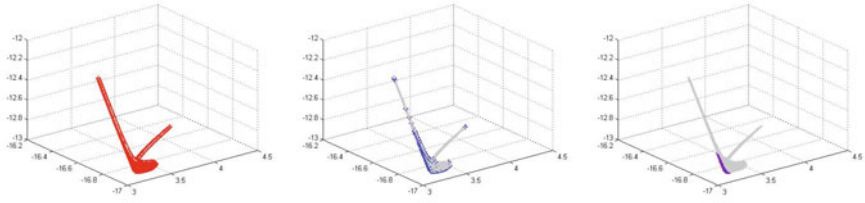


(a) Kursawe: D^2MOPSO (b) dMOPSO (c) MOEA/D

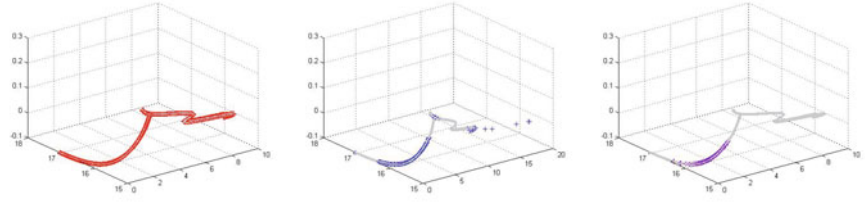


(d) Poloni: D^2MOPSO (e) dMOPSO (f) MOEA/D

Fig. 2. The $PF_{approximation}$ of Kursawe and Poloni using the three methods



(a) Viennet2: D^2MOPSO (b) dMOPSO (c) MOEA/D



(d) Viennet3: D^2MOPSO (e) dMOPSO (f) MOEA/D

Fig. 3. The $PF_{approximation}$ of Viennet2 and Viennet3 using the three methods. In (e) dMOPSO generates outliers affecting the scale of the figure.

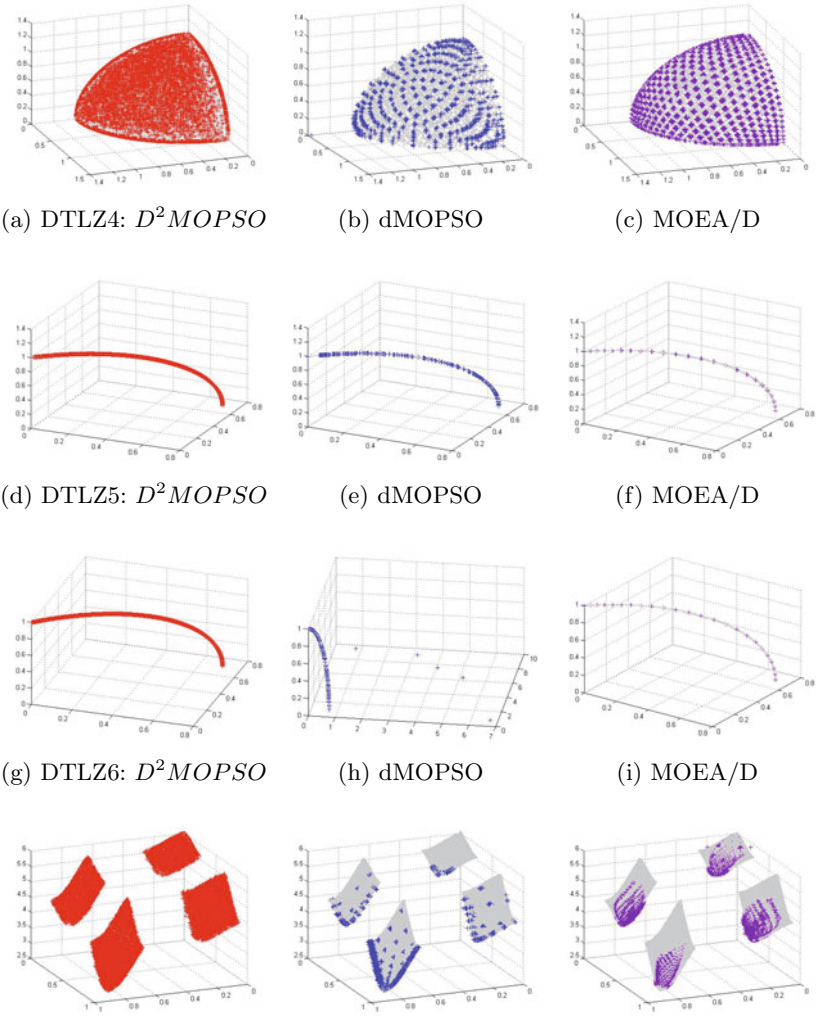
(probability = 1.0 and differential weight = 0.5) polynomial mutation (probability = 1/number of decision variables), the mutation distribution index is equal to 20, and the neighbourhood size is set to 30. dMOPSO sets the age threshold to 2, $C_1, C_2, w \geq 0$, and r_1, r_2 are set to a random value in $[0, 1]$. These values where chosen according to recommendations by the authors of MOEA/D and dMOPSO. D^2MOPSO uses the parameters explained in the previous section with $\alpha = 10\%$ and $\beta = 10\%$.

4.2 Quality Measures

To validate our approach, four indicators [15] which estimate the convergence and diversity of the solutions are used. Hypervolume indicator ($I_{Hypervolume}$) measures the volume of the objective space that is weakly dominated by a PF approximation (A). $I_{Hypervolume}$ uses a reference point v^* which denotes an upper bound over all objectives. v^* is defined as the worst objective values found in A (i.e. v^* is dominated by all solutions in A). Using the Lebesgue measure (Λ), $I_{Hypervolume}$ is defined as:

$$I_{Hypervolume}(A) = \Lambda\Big(\bigcup_{a \in A} \{x \,|a \prec x \prec v^*|\}\Big). \tag{6}$$

Inverted generational distance (I_{IGD}) measures the uniformity of distribution of the obtained solutions in terms of dispersion and extension. The average distance



(a) DTLZ4: D^2MOPSO (b) dMOPSO (c) MOEA/D

(d) DTLZ5: D^2MOPSO (e) dMOPSO (f) MOEA/D

(g) DTLZ6: D^2MOPSO (h) dMOPSO (i) MOEA/D

(j) DTLZ7: D^2MOPSO (k) dMOPSO (l) MOEA/D

Fig. 4. The $PF_{approximation}$ of DTLZ4-7 using the three methods. In (h), dMOPSO generates outliers affecting the scale of the figure.

Table 1. The Results of Applying $I_{Hypervolume}$ and I_{IGD} Measures

	$I_{Hypervolume}$			I_{IGD}		
	D^2MOPSO	dMOPSO	MOEA/D	D^2MOPSO	dMOPSO	MOEA/D
Fonseca	**0.314240**	0.310809	0.312262	**0.000212**	0.000498	0.000430
Schaffer	**0.833229**	0.822686	0.810658	**0.000014**	0.006258	0.002140
Kursawe	**0.404477**	0.398544	0.399500	**0.000024**	0.000141	0.001232
Poloni	**0.913954**	0.899036	0.891183	**0.000181**	0.017408	0.018043
Viennet2	**0.931879**	0.926400	0.846134	**0.000017**	0.000590	0.002226
Viennet3	**0.841510**	0.830819	0.817380	**0.000012**	0.001402	0.004968
DTLZ4	**0.456455**	0.437521	0.453703	**0.000283**	0.000492	0.000382
DTLZ5	**0.095629**	0.093547	0.084997	**0.000004**	0.000070	0.000250
DTLZ6	**0.095855**	0.093257	0.084517	**0.000000**	0.000089	0.000257
DTLZ7	**0.331488**	0.270847	0.312294	**0.000055**	0.000816	0.000698
p-value		**0.0368**	**0.0256**		0.1513	0.1169

is calculated for each point of the actual PF (PF_{True}) A and the nearest point of the approximation PF (PF_{approx}) B.

$$I_{IGD(A,B)} = (\sum_{a \in A}(\min_{b \in B} \parallel F(a) - F(b) \parallel^2))^{1/2}/|A| \tag{7}$$

ϵ-Indicator (I_ϵ) measures the minimum distance which a PF approximation (A) has to be translated in the objective space to weakly dominate the actual PF B. ϵ-Indicator is defined as:

$$I_\epsilon(A, B) = \min_{\epsilon \in \mathbb{R}}\{\forall b \in B, \exists b'_i - \epsilon \leq b_i, \forall 1 \leq i \leq n\} \tag{8}$$

Cardinality measure ($I_{cardinality}(A, B)$), calculates the percentage of solutions in A that belongs to B, where A is an approximation of the PF and B is a reference

Table 2. The Results of Applying I_ϵ and $I_{cardinality}$ Measures

	I_ϵ			$I_{cardinality}$		
	D^2MOPSO	dMOPSO	MOEA/D	D^2MOPSO	dMOPSO	MOEA/D
Fonseca	**0.00079**	0.00517	0.00384	**60%**	6%	34%
Schaffer	**0.00120**	0.09028	0.22934	**92%**	7%	1%
Kursawe	**0.01858**	0.08431	0.28084	**52%**	2%	46%
Poloni	**0.03026**	1.06964	1.07510	**80%**	2%	18%
Viennet2	**0.00212**	0.01119	0.05978	**93%**	3%	4%
Viennet3	**0.00105**	0.06098	0.10490	**86%**	3%	11%
DTLZ4	**0.03068**	0.06889	0.04765	34%	11%	**55%**
DTLZ5	**0.00044**	0.00983	0.02307	**72%**	5.6%	22.4%
DTLZ6	**0.00005**	0.03202	0.02324	**98.6%**	1.2%	0.2%
DTLZ7	**0.01370**	0.16680	0.07199	**65%**	5%	30%
p-value		0.1672	0.1019		**4.67E-005**	**0.0116**

set. For fair comparison, B is a bounded set of the none-dominated solutions produced by all algorithms under comparison. For bi-objective problems the size of B is 100 and 10,000 for 3-objective problems where all solutions are equally distributed along the PF.

$$I_{cardinality}(A, B) = |A \cap B|/|B| \tag{9}$$

Table 1 shows the results of applying $I_{Hypervolume}$ and I_{IGD}. Table 2 shows the results obtained using I_ϵ and $I_{cardinality}$. The last row presents the p-value of two tailed paired t-test between the D^2MOPSO and the other two methods where bold font indicates a statistically significant difference. Fig.1, Fig.2, Fig.3 and Fig.4 depict PF_{true} and $PF_{approximated}$ for the three algorithms under investigation.

5 Discussion and Conclusions

In this paper, a multi-objective particle swarm optimisation method combining decomposition and dominance is presented. The method works by dividing the MOP into scalar aggregation problems which are solved simultaneously using PSO. In order to maintain the diversity of the final solutions, D^2MOPSO uses a crowding archive that assures the low density regions are covered as much as the higher density ones. Unlike other decomposition based evolutionary methods, D^2MOPSO does not use any genetic operators or parametric probability density function. D^2MOPSO takes advantages of both decomposition and dominance while maintaining the simplicity and efficiency of MOPSO.

The results presented in this paper show that D^2MOPSO outperforms dMOPSO, MOEA/D on all test problems.Paired t-test results (p-value in Table1 and Table2) show D^2MOPSO performing significantly better in terms of hypervolume and cardinality indicators suggesting better spread and more diverse coverage of PF.

Sigmoid limiting transformation might not be necessary in all standard MOPs, but it is essential for real life data where different objectives' ranges can vary considerably and hence once objective might overshadow the other objectives in the aggregation function. Some of the used problems do not have normalized objectives by definition (e.g. Kursawe, Viennet2, and Viennet3). The results obtained demonstrates the advantage of normalizing the objectives. The finalizing step, which shrinks the leaders' archive, is an efficient and effective step to maintain the diversity of the approximated PF. This is a general approach that can be easily used in other methods.

Decomposition based evolutionary algorithms have advantage in problems where a linear structure exists in the objective space as the λ values are uniformly distributed. When, however, a non-linear structure exists or is confounded by the interplay of several competing objectives, dominance helps to direct the particles to better exploration of the objective space. This can explain the advantage of the D^2MOPSO method as shown in the results of our experiments.

References

1. Zhang, Q., Li, H.: MOEA/D: A multi-objective evolutionary algorithm based on decomposition. IEEE Trans. on Evolutionary Computation 11(6), 712–731 (2007)
2. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-ii. IEEE Trans. on Evolutionary Computation 13(2), 284–302 (2009)
3. Awwad Shiekh Hasan, B., Gan, J.Q., Zhang, Q.: Multi-objective evolutionary methods for channel selection in brain-computer interfaces: some preliminary experimental results. In: WCCI. IEEE (2010)
4. Wang, Z., Durst, G.L., Eberhart, R.C., Boyd, D.B., Ben Miled, Z.: Particle swarm optimization and neural network application for qsar. In: Parallel and Distributed Processing Symposium, International, vol. 10, p. 194 (2004)
5. Jaishia, B., Ren, W.: Finite element model updating based on eigenvalue and strain. Mechanical Systems and Signal Processing 21(5), 2295–2317 (2007)
6. Al Moubayed, N., Petrovski, A., McCall, J.: Multi-objective optimisation of cancer chemotherapy using smart pso with decomposition. In: 3rd IEEE Sym. Comp. Intel. IEEE (2011)
7. Reyes-Sierra, M., Coello, C.A.C.: Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)
8. Baltar, A.M., Fontane, D.G.: A generalized multiobjective particle swarm optimization solver for spreadsheet models: application to water quality. In: The Twenty Sixth Annual American Geophysical Union Hydrology Days (2006)
9. Hassan, R., Cohanim, B., de Weck, O., Venter, G.: A comparison of particle swarm optimization and the genetic algorithm. In: Structural Dynamics and Materials, Texas, USA (2005)
10. Al Moubayed, N., Petrovski, A., McCall, J.: A novel smart multi-objective particle swarm optimisation using decomposition. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part II. LNCS, vol. 6239, pp. 1–10. Springer, Heidelberg (2010)
11. Martínez, S.Z., Coello, C.A.C.: A multi-objective particle swarm optimizer based on decomposition. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011. ACM (2011)
12. Sierra, M.R., Coello, C.A.C.: Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005)
13. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm intelligence. Morgan Kaufmann, San Francisco (2001)
14. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Genetic and Evolutionary Computation. Springer, New York (2007)
15. El-Ghazali, T.: Metaheuristics: from design to implementation. John Wiley & Sons (2009)

Domain Reduction Using GRASP Construction Phase for Transmission Expansion Planning Problem

Mohsen Rahmani*, Ruben A. Romero, Marcos J. Rider, and Miguel Paredes

Universidade Estadual Paulista, Faculdade de Engenharia de Ilha Solteira,
Ilha Solteira - SP, Brasil
rahmani@ieee.org,
{ruben,mjrider}@dee.feis.unesp.br, miguel.paredes.q@gmail.com

Abstract. This paper proposes a new strategy to reduce the combinatorial search space of a mixed integer linear programming (MILP) problem. The construction phase of greedy randomized adaptive search procedure (GRASP-CP) is employed to reduce the domain of the integer variables of the transportation model of the transmission expansion planning (TM-TEP) problem. This problem is a MILP and very difficult to solve specially for large scale systems. The branch and bound (BB) algorithm is used to solve the problem in both full and the reduced search space. The proposed method might be useful to reduce the search space of those kinds of MILP problems that a fast heuristic algorithm is available for finding local optimal solutions. The obtained results using some real test systems show the efficiency of the proposed method.

Keywords: GRASP-CP, MILP, TM-TEP.

1 Introduction

The mathematical model for the transmission expansion planning (TEP) problem is a mixed integer, non-linear, non-convex optimization problem, which is very complex and computationally demanding [1], [2]. This problem presents a large number of local optimal solutions and when the size of system becomes large, the number of solutions grows exponentially. Various approaches have been proposed to obtain a high quality solution for this problem, some examples are; classical methods [3], [4] heuristics algorithm [5], [6], meta-heuristic strategies [7], [8], relaxed models [9], [10] and hybrid methods [11]. A comprehensive review of these strategies is given in [12], [13].

Significant progress has been made in the direction of exact methods for combinatorial optimization, such as branch and bound, branch and cut, and dynamic programming. However, these methods suffer from the curse of dimensionality,

* This work was supported in part by CNPq under process 500485/2009 and FAPESP under processes 2009/14816-7 and 2010/04608-5.

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 87–98, 2012.

i.e. they tend to break down as the size of the problem, specially the integer variables, increases [14], [15]. Therefore the reduction of the search space becomes necessary. This paper proposes a methodology to reduce combinatorial search space of a MILP problem. In order to present the performance of the proposed methodology for large scale systems we use transportation model of the TEP problem [10] which is a MILP problem, however the proposed method can be also applied to reduce the search space of other TEP models. The construction phase of greedy randomized adoptive search optimization (GRASP) [16] is used in the first phase to reduce the search space of the problem, and in the second phase the branch and bound algorithm is employed to find the optimum solution on the reduced search space. GRASP [7], [14] and [16] is a multi-start procedures which apply local search to a set of starting solutions generated with a greedy randomized algorithm or semi-greedy method. The best local optimum found over the iterations is returned as the heuristic solution. This paper only benefits from GRASP construction phase to reduce the search space of the integer variables. The local search of the GRASP is not used in the paper and will not be discussed. This work is organized as follows. Section 2 presents the mathematical model of the TM-TEP and it proposes a new sensitivity index to be used in GRASP-CP, the novel strategy of reducing search space using GRASP-CP is provided in section 3. The results of some tests and real systems are analyzed in section 4. And finally, in section 5, conclusions are drawn.

2 Transportation Model of the Transmission Expansion Planning Problem

The main objective of transmission expansion planning (TEP) problem, in an electric power system (EPS), is to define where, how many and when new transmission lines must be added to the EPS in order to provide forecasted power demand and to make its operation viable for a pre-defined planning horizon at minimum cost [1], [4], [5], [6], [10].

There are several models for TEP problem, the AC model [6], DC model, Hybrid model and transportation model [10]. The proposed domain reduction method can be used to reduce the search space of all models. This paper uses the transportation model of the TEP problem. The TM was originally suggested by Garver [17]. This model is presented in the following equations (1)-(6).

$$\min\ v = \sum_{km \in \Omega} c_{km} n_{km} \tag{1}$$

s.t.

$$\sum_{mk \in \Omega} f_{mk} - \sum_{km \in \Omega} f_{km} = d_k - g_k \qquad \forall k \in \beta \tag{2}$$

$$|f_{km}| \leq (n^0_{km} + n_{km}) \bar{f}_{km} \qquad \forall km \in \Omega \tag{3}$$

$$0 \leq n_{km} \leq \bar{n}_{km} \qquad \forall km \in \Omega \tag{4}$$

$$n_{km} \ \text{ integer} \qquad \forall km \in \Omega \tag{5}$$

where:

c_{km} cost of a circuit that can be added to branch k-m,(constant), (US$),
n_{km} number of circuits added to branch k-m, (variable),
f_{km} power flows in branch k-m, (variable), (MW),
d_k demand in bus k, (constant), (MW),
g_k power generation at k-th bus, (variable), (MW),
n^0_{km} number of circuits in the base case in branch k-m, (constant),
$\bar{f}_{km}$ maximum power flow in branch k-m, (constant), (MW),
$\bar{n}_{km}$ maximum number of the candidate lines in branch k-m, (constant),
Ω set of all transmission lines,
β set of all buses.

In transportation model, (1) stands for investment in transmission lines; (2) represents the power flow balance constraint; (3) present the power flow limitation for each transmission lines and (4) denotes the limits of each transmission lines in a branch.

In the proposed domain reduction methodology, which will be explained in section 3, several solutions are needed to be constructed based on the sensitivity index. There are various sensitivity indexes (SI) for an integer decision variable in a MILP problem. SI's are supposed to assess the impact of any integer variable that affects the performance of the problem. There are a couple of sensitivity index in the TM-TEP problem. The first SI is proposed by Garver [17] in which it directly calculated from integer variables. Although this index is considered to be the best in terms of finding the closest solution to the optimum [5], it is not very suitable to be used alone for the GRASP-CP, since this index is too greedy and it assess the best lines, and considers the other lines to have zero value in the construction process, while this is in contradiction with the fact that every line may affect the performance of the optimum solution after addition. There are other indexes proposed in [18] that calculated from the dual value of the LP problem at the optimum solution. One of these indexes is based on the maximum flow limit and the other in circuit susceptance. However it is not possible to use the second index since in TM-TEP problem the susceptance of the lines are not considered. Although the index based on maximum power flow has some limitations in disconnected networks, we use it together with Garver index to overcome this problem. This paper uses the following indices:

1 Garver sensitivity index (SIG): this index is obtained after solving the LP problem of the TM-TEP when the integrality of the integer variables is relaxed. The SIG for each transmission line is calculated in (6);

$$SIG_{km} = n_{km}\bar{f}_{km} \qquad \forall km \in \Omega \tag{6}$$

2 SI with respect to maximum flow limit (SIF): This index is obtained using dual variables related to equation 3, at the optimum solution of the LP problem and is calculated as:

$$SIF_{km} = \frac{\partial v}{\partial f_{km}} = \pi_{f_{km}} \qquad \forall km \in \Omega \tag{7}$$

where v is the objective function of the TM-TEP and $\pi_{f_{km}}$ is the dual variables associated to constraints (3). In conclusion the sensitivity index to allocate the lines in an iterative process for each line is calculated in (8), where, The SIG and SIF are normalized in order to have equal effects in the total sensitivity index (SI).

$$SI_{km} = \frac{SIG_{km}}{\max(SIG_{ij\in\Omega})} + \frac{SIF_{km}}{\max(SIF_{ij\in\Omega})} \qquad \forall km \in \Omega \tag{8}$$

3 Domain Reduction Using GRASP Construction Phase

GRASP is a heuristic iterative sampling technique composed by two phases, a construction phase and a local search phase. A complete description of GRASP can be found in [16]. However, in this paper, only the construction phase of the GRASP is used. The generic pseudo code of the GRASP-CP procedure for creating a feasible solution for a MILP problem is shown in Procedure 1. Several calls to the GRASP-CP will create several feasible high-quality solutions. It should be noted that there are some MILP problems that even finding a feasible solution is not easy. Therefore the proposed method may not be useful for this kind of problems.

Procedure 1. GRASP-CP for creating high quality integer solution for a MILP problem

```
solution = ∅ ;
repeat
    Solve relaxed problem of the MILP model;
    Evaluate the Greedy Function;
    Build the RCL;
    η= Element selected randomly from RCL;
    solution =∪ η ;
until a feasible solution is obtained
return with feasible solution
```

In GRASP-CP each solution is obtained through an iterative algorithm in which an LP is solved and the restricted candidate list (RCL) is constructed from candidate elements with a greedy function value above a specified threshold. The greedy function value of an element is evaluated by measuring the local benefit of including that element in the constructed solution. The next element to be included in the solution is selected at random from the RCL. Its inclusion in the solution alters the greedy functions and the set of candidate elements used to determine the next RCL [19]. The construction procedure terminates when a feasible solution is obtained. Eq. (9) gives the RCL set for TM-TEP.

$$RCL = \{n_{km\in\Omega} | SI_{km\in\Omega} \geq SI_{\min} + \alpha(SI_{\max} - SI_{\min}) \tag{9}$$

where SI_{km} is the sensitivity index for each candidate variable and obtained by solving an LP problem, as explained in section 2. The $SI_{\max}$ and $SI_{\min}$are the maximum and minimum value of the sensitivity index ($SI_{km\in\Omega}$) over all candidate variables. α is an experimental parameter in the range 0 to 1 and defines the greediness or randomness of the RCL set. If $\alpha = 1$, then the semi-greedy construction phase reduces to a greedy algorithm, and if $\alpha = 0$, it changes to a random algorithm.

The pseudo code of domain reduction of an MILP problem using the GRASP-CP is proposed in Procedure 2, such that the best solution over all GRASP-CP iterations is considered to be the incumbent solution of branch and bound algorithm, and the maximum number of lines over all GRASP-CP iterations in each branch is considered as the candidate line limit in that branch. As a result, the search space is reduced significantly. To ensure that the reduced search space contains the optimum solution, or at least very high-quality solutions, the randomness of GRAP-CP must be set to high, i.e., a small value for α and/or enough iterations of the GRASP-CP must be implemented. We can also add some noise to the cost of transmission lines [20] to obtain a more diverse search space, bearing in mind that the computational time increases.

Procedure 2. Domain reduction of a MILP problem using GRASP-CP. This code defines an upper bound ($\overline{x}$) for integer variables and an incumbent solution (x^*) for the BB algorithm

Input: Number of iterations i_{max} ;
  X = $\emptyset$;
  for $i = 1, \ldots, i_{max}$ **do**
    x ← GRASP-CP;
    X = X ∪ x;
    if $f(x) \leq f^*$ **then**
      $f^* \leftarrow f(x)$
      $x^* \leftarrow x$
    end if
  end for
  return $\overline{x} = max(X)$ and x^*

4 Tests and Results

In this section three test systems have been analyzed to show the efficiency of the domain reduction using GRASP-CP. The Garver system, the Southern Brazilian system and the Brazilian North-Northeast system are considered for tests. The solutions of Garver system and Southern Brazilian system is reported in various papers [10], [21]. We also make tests on these systems for performance comparison of the proposed method. The optimum solution for Brazilian North-North system is still unknown and it is a benchmark for transmission expansion planning problem. This paper proposes the best solutions for North-Northeast

Brazilian system. The AMPL/CPLEX 12.1 [22], [23] solver is used to solve the TM-TEP problem. All the tests are carried out using a 2.93 GHZ, 8 cores CPU with 4 GB memory RAM and 300 GB hard memory. The CPLEX is able to utilize all the processors in parallel for solving the problems.

In all tests systems firstly we try to obtain the optimum solution in full space of the problem using BB. Following some tests using various GRASP-CP parameters are carried to reduce the search space and BB again is used to obtain the solution in the reduced space. α and i_{max} are two important parameters in domain reduction using GRASP-CP. As it mentioned earlier these parameters are empirical. In TM-TEP problems the size of the problem is very important to define these parameters. For small size problems small number of iteration and high value of α may lead us to optimum solution while in large scale problems a large number of iteration and a low value of α are selected. However in this paper different amounts for these parameters are selected in order give a deeper sight of the method.

4.1 Garver System

For the illustrative examples, the Garver 6-bus system (Fig. 1) is used. This system has 6 buses and 15 candidate lines with a total demand of 760 MW and a maximum of 5 lines can be added to each branch.

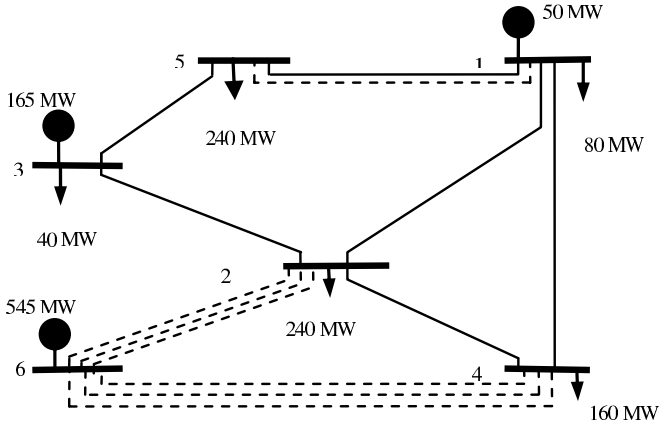


Fig. 1. Garver system containing base line and optimum solution, the solid lines are existence and the dashed ones are new lines. The ball shaped objects are representing generators and the arrow ones are for demands.

The Garver system data is given in [10]. The tests can be carried out for both planning with generation rescheduling and without generation rescheduling. However, since our main goal in this test is to show the process of the domain reduction we only show the results of planning without generation rescheduling and with base topology proposed by Garver. The optimum solution without

generation rescheduling is reported in [10], with US$ 200 investment on seven new transmission lines with following topology: $n_{1-5} = 1$, $n_{2-6} = 3$ and $n_{4-6} = 3$.

The GRASP-CP is employed to reduce the domain of integer variables. Five GRASP-CP iteration with $\alpha = 0.6$ are implemented. The black bars in Fig. 2 show the reduced space of the problem. The CPLEX branch and bound solver is then applied to obtain the best solution over reduced space. The gray bars in Fig. 2 show this solution. This figure shows the domain of integer variables is reduced without losing the optimum solution. As it mentioned the maximum number of lines in each branch in the Garver system is 5, multiplying this number with the number of candidate lines, gives 75 as the total bounds of integer variables, while using GRASP-CP this number is 16, showing that the upper bounds of integer variables is reduced by 78 %.

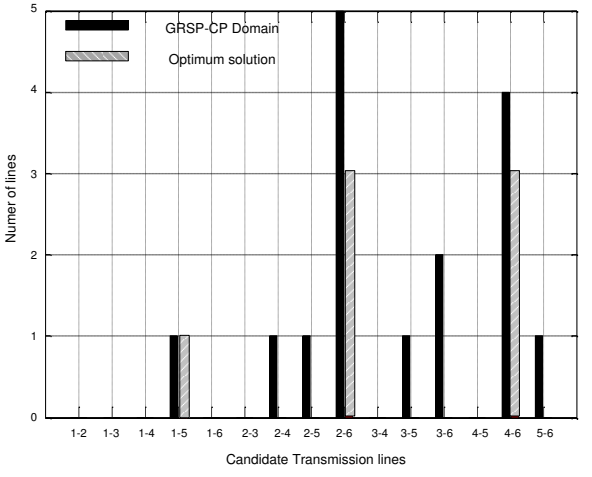


Fig. 2. GRASP-CP for reducing the domain of integer variables in Garver system. The integer variables are limited by black bars while the gray ones are their value in optimum solution.

4.2 Southern Brazilian System

This system has 46 buses, 79 circuits and 6880 MW of demand and has been studied in many references [6], [10], [20], [21]. It is possible to execute the planning with and without generation rescheduling and also with and without base topology. Three tests are performed for each planning to evaluate the behavior of the algorithm.

Planning with Generation Rescheduling. Table 1 shows the results of the tests in which tests 1-3 are provided for planning with base topology and tests 4-7 for planning without base topology. In tests 1 and 4, the branch and bound algorithm is applied on the full space of the problems to obtain the optimum solution, while in tests 2, 3, 5 and 6, with different parameters, GRASP-CP along with branch and bound is applied to solve the problem. The number of LP solved,

Table 1. Southern Brazilian system with generation rescheduling

	With Base Topology			Without Base Topology		
	Test-1	Test-2	Test-3	Test-4	Test-5	Test-6
	BB	GRASP-CP&BB	GRASP-CP&BB	BB	GRASP-CP&BB	GRASP-CP&BB
α	—	0.8	0.6	—	0.8	0.6
GCI[1]	—	10	10	—	20	20
No. LP	16	0	11	11298	1558	3491
SUBINV[2]	237	15	23	237	89	102
Time (sec)	0.2	0.02	0.02	2.65	0.61	1.17
RSP[3] (%)	—	93	90	—	62	56
gap (%)	0	6.43	0	0	0.87	0
Cost (US$)	53334*	57005	53334*	402748*	406288	402748*
Installed Lines in Optimum Solution	$n_{33-34}=1$, $n_{20-21}=2$, $n_{42-43}=1$, $n_{5-11}=2$, $n_{46-11}=1$			$n_{5-8}=1$, $n_{4-5}=2$, $n_{2-5}=2$, $n_{12-14}=2$, $n_{13-20}=1$, $n_{19-21}=1$, $n_{14-22}=1$, $n_{22-26}=1$, $n_{20-23}=1$, $n_{23-24}=1$, $n_{26-27}=1$, $n_{24-34}=1$, $n_{33-34}=1$, $n_{27-36}=1$, $n_{34-35}=1$, $n_{37-40}=1$, $n_{39-42}=3$, $n_{38-42}=1$, $n_{32-43}=1$, $n_{42-44}=1$, $n_{44-45}=1$, $n_{46-16}=1$, $n_{20-21}=3$, $n_{42-43}=3$, $n_{46-6}=1$, $n_{25-32}=1$, $n_{31-32}=1$, $n_{28-31}=1$, $n_{24-25}=1$, $n_{5-6}=2$		

[1] GCI: GRASP-CP Iteration [2] SUBINV: Summation of upper bound of integer variables
[3] Reduction in search space * Optimum solution

Table 2. Southern Brazilian system without generation rescheduling

	With Base Topology			Without Base Topology		
	Test-7	Test-8	Test-9	Test-10	Test-11	Test-12
	BB	GRASP-CP&BB	GRASP-CP&BB	BB	GRASP-CP&BB	GRASP-CP&BB
α	—	0.8	0.6	—	0.8	0.6
GCI[1]	—	50	50	—	50	50
No. LP	455	404	155	96466	7052	15026
SUBINV[2]	237	41	51	237	96	113
Time (sec)	0.33	0.05	0.06	9.17	2.13	5.56
RSP[3] (%)	—	82	78	—	59	52
gap (%)	0	2.8	0	0	0	0
Cost (US$)	127272*	130943	127272*	473246*	473246*	473246*
Installed Lines in Optimum Solution	$n_{14-22}=1$, $n_{20-21}=2$, $n_{42-43}=2$, $n_{5-11}=2$, $n_{25-32}=1$, $n_{31-32}=1$, $n_{28-31}=1$, $n_{46-11}=1$, $n_{24-25}=2$			$n_{5-8}=1$, $n_{4-5}=2$, $n_{2-5}=2$, $n_{12-14}=2$, $n_{13-20}=1$, $n_{19-21}=1$, $n_{16-17}=1$, $n_{17-19}=1$, $n_{14-26}=1$, $n_{14-22}=1$, $n_{22-26}=1$, $n_{20-23}=1$, $n_{23-24}=1$, $n_{26-27}=1$, $n_{24-34}=1$, $n_{24-33}=1$, $n_{27-36}=1$, $n_{27-38}=1$, $n_{34-35}=2$, $n_{35-38}=1$, $n_{37-39}=1$, $n_{37-40}=1$, $n_{39-42}=1$, $n_{38-42}=1$, $n_{42-44}=1$, $n_{44-45}=1$, $n_{46-16}=1$, $n_{20-21}=3$, $n_{42-43}=3$, $n_{14-15}=1$, $n_{46-6}=1$, $n_{19-25}=1$, $n_{31-32}=1$, $n_{28-31}=1$, $n_{31-41}=1$, $n_{41-43}=1$, $n_{15-16}=1$, $n_{24-25}=2$, $n_{5-6}=2$		

[1] GCI: GRASP-CP Iteration [2] SUBINV: Summation of upper bound of integer variables
[3] Reduction in search space * Optimum solution

the upper bound of integer variables, the processing time and the investment cost are given for all in addition the degree of randomness or greediness (α), the reduction in search space in percent and the number of GRASP-CP iteration are also

provided for tests 2, 3, 5 and 6. In all the tests the gap between current solutions with respect to the optimum solution is also provided. In test 2 and 5 the optimum solution is not achieved since the GRASP-CP is considered too greedy ($\alpha = 0.8$). In tests 3 and 6 the optimum solutions are obtained with much less effort than BB algorithm when applied in full space. The summation of upper bound of variables in tests 3 and 6 are decreased by 90% and 56% respectively without losing the optimum solution of the problem. The installed transmission lines in optimum solution for both planning scheme are also provide in this table.

Planning without Generation Rescheduling. Similar to the planning with generation rescheduling several tests have been implemented for both planning with and without base topology. Table 2 shows the results of the tests in which the domain of search space, the processing time and number of iterations in tests 8, 9 11 and 12 are reduced significantly. In all tests except test-8, which is too greedy, the optimum solutions are obtained with little effort which can be confirmed from processing time and number of LP solved for each test.

Table 3. North-Northeast Brazilian system without generation rescheduling Plan 2002

	With Base Topology			Without Base Topology		
	Test-13 BB	Test-14 GRASP-CP&BB	Test-15 GRASP-CP&BB	Test-16 BB	Test-17 GRASP-CP&BB	Test-18 GRASP-CP&BB
α	—	0.6	0.1	—	0.6	0.1
GCI[1]	—	100	100	—	100	100
No. LP	179156	28619	74173	1.51×10^{9}	7.61×10^{7}	1.2×10^{8}
SUBINV[2]	2700	203	360	2700	331	432
Time (sec)	23.57	8.46	14.06	365400	16694	31585
RSP[3] (%)	—	99.14	86.66	—	87.74	84.00
gap (%)	0	3.58	0	0.73	0.53	0.49
Cost (US$)	1194561*	1238944	*	2344023◇	2356727	2355601
Installed Lines in Optimum Solution	$n_{2-60}=2$, $n_{5-58}=2$, $n_{5-60}=2$, $n_{5-68}=1$, $n_{8-17}=1$, $n_{8-62}=2$, $n_{9-10}=1$, $n_{10-11}=1$, $n_{11-17}=1$, $n_{13-15}=2$, $n_{14-59}=1$, $n_{15-16}=2$, $n_{16-44}=3$, $n_{17-18}=2$, $n_{18-50}=6$, $n_{20-21}=1$, $n_{20-38}=1$, $n_{24-43}=1$, $n_{25-55}=1$, $n_{30-63}=1$, $n_{35-51}=1$, $n_{40-45}=1$, $n_{41-64}=3$, $n_{42-44}=2$, $n_{42-85}=1$, $n_{43-55}=1$, $n_{43-58}=1$, $n_{48-49}=3$, $n_{54-58}=1$, $n_{54-63}=1$, $n_{62-67}=2$, $n_{63-64}=1$, $n_{67-69}=1$, $n_{69-87}=1$			$n_{1-2}=2$, $n_{2-4}=1$, $n_{2-60}=2$, $n_{4-5}=1$, $n_{4-69}=1$, $n_{5-56}=1$, $n_{5-58}=3$, $n_{5-60}=2$, $n_{5-68}=1$, $n_{7-8}=1$, $n_{7-62}=1$, $n_{8-17}=2$, $n_{8-53}=1$, $n_{8-62}=1$, $n_{9-10}=1$, $n_{10-11}=2$, $n_{11-15}=2$, $n_{11-17}=1$, $n_{12-17}=3$, $n_{12-35}=2$, $n_{13-15}=3$, $n_{13-45}=1$, $n_{14-59}=1$, $n_{15-16}=4$, $n_{15-46}=1$, $n_{16-44}=7$, $n_{16-61}=1$, $n_{17-18}=6$, $n_{18-50}=11$, $n_{19-20}=1$, $n_{20-21}=2$, $n_{20-21}=1$, $n_{20-56}=2$, $n_{22-23}=1$, $n_{22-37}=1$, $n_{22-58}=1$, $n_{24-43}=1$, $n_{25-26}=1$, $n_{25-55}=3$, $n_{26-29}=1$, $n_{27-28}=2$, $n_{27-53}=1$, $n_{30-3}=1$, $n_{30-63}=1$, $n_{34-39}=1$, $n_{34-41}=1$, $n_{35-51}=3$, $n_{36-39}=1$, $n_{36-46}=2$, $n_{39-42}=1$, $n_{40-45}=2$, $n_{40-46}=1$, $n_{41-64}=3$, $n_{42-44}=1$, $n_{42-85}=1$, $n_{43-55}=2$, $n_{43-58}=2$, $n_{48-49}=1$, $n_{48-50}=3$, $n_{49-50}=4$, $n_{51-52}=1$, $n_{52-59}=1$, $n_{54-58}=1$, $n_{54-63}=1$, $n_{61-85}=2$, $n_{62-67}=2$, $n_{63-64}=1$, $n_{67-69}=1$		

[1] GCI: GRASP-CP Iteration [2] SUBINV: Summation of upper bound of integer variables
[3] Reduction in search space * Optimum solution ◇ Best solution

4.3 North-Northeast Brazilian System

The North-Northeast Brazilian System is used as another case study. This system consists of 87 buses and 183 circuits. The system data is available in [1] and from the authors. This system represents a benchmark in the transmission planning problem, due to its high complexity and the unknown global optimal solution. There are two levels of demand, one considered for 2002 (P1) with level of 20316 MW and the other for 2008 (P2) with level of 29748 MW. In this section several test of domain reduction, for both plans P1 and P2, with different parameters have been implemented. The tests are carried out with fixed generator levels, i.e. without generation rescheduling but with and without base topology. Tables 3 and 4 show the result of tests.

When the plans are carried out with the base topology, the tests 13-15 for P1 and 19-21 for P2, the optimum solutions or near optimal solutions are obtained.

Table 4. North-Northeast Brazilian system without generation rescheduling Plan 2008

	With Base Topology			Without Base Topology		
	Test-19 BB	Test-20 GRASP-CP&BB	Test-21 GRASP-CP&BB	Test-22 BB	Test-23 GRASP-CP&BB	Test-24 GRASP-CP&BB
α	—	0.6	0.1	—	0.6	0.1
GCI[1]	—	100	100	—	100	100
No. LP	1.42×10^{8}	1.32×10^{6}	3.26×10^{6}	1.24×10^{9}	1.01×10^{6}	7.10×10^{7}
SUBINV[2]	2700	305	518	2700	436	596
Time (sec)	1299	193	419	249794	11238	20750
RSP[3] (%)	0	88.73	80.81	0	83.85	77.92
gap (%)	0	1.14	0.74	1.73	0.04	1.03
Cost (US$)	2370680 *	2398025	2388359	3534832	3536290	3533929 ◇
Installed Lines in Optimum Solution	$n_{1-2}=1$, $n_{2-60}=1$, $n_{4-5}=2$, $n_{4-6}=1$, $n_{4-68}=1$,$n_{4-81}=3$, $n_{5-58}=3$, $n_{5-60}=1$, $n_{13-15}=4$, $n_{14-45}=1$, $n_{15-16}=4$, $n_{16-44}=6$, $n_{16-61}=1$, $n_{18-50}=11$, $n_{18-74}=6$, $n_{20-21}=2$, $n_{20-38}=2$, $n_{22-23}=1$, $n_{22-58}=2$, $n_{24-43}=1$, $n_{25-55}=3$, $n_{26-29}=2$, $n_{29-30}=2$, $n_{39-86}=4$, $n_{40-45}=2$, $n_{41-64}=2$, $n_{42-44}=1$, $n_{43-55}=2$, $n_{43-58}=2$, $n_{48-49}=2$, $n_{49-50}=3$, $n_{52-59}=1$, $n_{53-86}=1$, $n_{61-64}=1$, $n_{61-85}=2$, $n_{67-68}=1$, $n_{67-69}=1$, $n_{67-71}=3$, $n_{71-72}=1$, $n_{72-73}=1$, $n_{73-74}=2$, $n_{73-75}=1$, $n_{75-81}=1$			$n_{1-2}=3$, $n_{2-60}=2$, $n_{4-6}=2$, $n_{4-60}=1$, $n_{4-68}=1$, $n_{4-81}=3$, $n_{5-56}=1$, $n_{5-58}=3$, $n_{5-60}=2$, $n_{5-68}=2$, $n_{6-70}=2$, $n_{7-53}=1$, $n_{7-62}=1$, $n_{8-9}=1$, $n_{8-62}=1$, $n_{9-10}=2$, $n_{10-11}=2$, $n_{11-12}=1$, $n_{11-15}=1$, $n_{11-17}=2$, $n_{12-15}=2$, $n_{12-17}=3$, $n_{12-35}=2$, $n_{13-15}=3$, $n_{13-45}=1$, $n_{13-59}=1$, $n_{14-45}=1$, $n_{15-16}=5$, $n_{15-46}=2$, $n_{16-44}=10$, $n_{17-1}=5$, $n_{18-50}=16$, $n_{18-74}=3$, $n_{19-20}=1$, $n_{19-22}=1$, $n_{20-21}=1$, $n_{20-66}=2$, $n_{21-57}=2$, $n_{22-23}=1$, $n_{22-37}=1$, $n_{22-58}=2$, $n_{24-43}=1$, $n_{25-55}=4$, $n_{26-27}=2$, $n_{26-2}=2$, $n_{27-28}=1$, $n_{27-53}=2$, $n_{28-35}=1$, $n_{29-30}=2$, $n_{30-31}=1$, $n_{30-63}=1$, $n_{31-34}=1$, $n_{34-39}=1$, $n_{35-51}=2$, $n_{36-39}=1$, $n_{36-46}=3$, $n_{39-86}=1$, $n_{40-45}=3$, $n_{41-64}=3$, $n_{42-44}=3$, $n_{43-55}=3$, $n_{43-58}=3$, $n_{44-46}=2$, $n_{48-50}=3$, $n_{49-50}=7$, $n_{51-52}=1$, $n_{52-59}=2$, $n_{53-70}=1$, $n_{53-86}=1$, $n_{54-63}=1$, $n_{54-70}=1$, $n_{56-57}=1$, $n_{60-66}=1$, $n_{61-85}=3$, $n_{61-86}=1$, $n_{62-67}=2$, $n_{63-64}=1$,$n_{67-69}=1$, $n_{68-69}=1$, $n_{73-74}=1$, $n_{73-75}=1$, $n_{75-81}=1$		

[1] GCI: GRASP-CP Iteration [2] SUBINV: Summation of upper bound of integer variables
[3] Reduction in search space * Optimum solution ◇ Best solution

In tests 13 and 19 the BB solver is applied in the full space and optimum solution for P1 and the best solution for P2 are obtained. In tests 14-15 and 20-21 the GRASP-CP is used for domain reduction, where, over 80% of the domain has been decreased resulting in less processing time and less number of LP. When the plans are made without base lines, tests 16-18 for P1 and 22-24 for P2, we have to deal with a highly complex problem, in which after several days of processing time of the BB solver on the full space of the problem, the optimum solutions are not obtained and the process stops due to the lack of memory. However the given solutions are the best proposed so far, and are very near to the optimum solution since they have very little gap with respect to the best relaxed node of the branch and bound three. The results of the domain reduction using GRASP-CP in tests 17-18 and 23-24 shows very high quality solutions obtained with respect to the best solution. These solutions are achieved in a few hours instead of several days when full space is considered.

The performance of the method is revealed in test-24 when the GRASP-CP along with BB finds better solution than the BB applied in the full space of the problem. The former solution is obtained in about 6 hours while the later one obtained in about 70 hours.

5 Conclusions

In this paper the GRASP construction phase is used to reduce the search space of integer variables in a challenging MILP problem. The transportation model of the transmission expansion planning problem as a MILP problem is considered to show the performance of the proposed methodology. The branch and bound method is employed to obtain the optimum solution of the problem in the full space as well as on the reduced space of the problem. The proposed method can be used for domain reduction of any optimization problem with integer variables. Several real test sys-tems have been considered to show the performance of the problem. The best solu-tions for the most challenging problem of the TEP problem are proposed in this paper.

References

1. Escobar, A.H., Gallego, R.A., Romero, R.: Multistage and coordinated planning of the expansion of transmission systems. IEEE Trans. Power Syst. 19(2), 735–744 (2004)
2. Verma, A., Panigrahi, B.K., Bijwe, P.R.: Harmony search algorithm for transmission network expansion planning. Generation, Transmission & Distribution, IET 4(6), 663–673 (2010)
3. Binato, S., Pereira, M.V.F., Granville, S.: A new Benders decomposition approach to solve power transmission network design problems. IEEE Transactions on Power Systems 16(2), 235–240 (2001)
4. Rider, M.J., Garcia, A.V., Romero, R.: Transmission system expansion planning by a branch-and-bound algorithm. Generation, Transmission & Distribution, IET 2(1), 90–99 (2008)

5. Romero, R., Asada, E.N., Carreno, E., Rocha, C.: Constructive heuristic algorithm in branch-and-bound structure applied to transmission network expansion planning. Generation, Transmission & Distribution, IET 1(2), 318–323 (2007)
6. Rahmani, M., Rshidienjad, M., Carreno, E.M., Romero, R.: Efficient method for AC transmission network expansion planning. Electric Power Systems Research 80(9), 1056–1064 (2010)
7. Faria Jr., H., Binato, S., Resende, M.G.C., Falcao, D.M.: Power transmission network design by greedy randomized adaptive path relinking. IEEE Transactions on Power Systems 20(1), 43–49 (2005)
8. Leite da Silva, A.M., Rezende, L.S., Honório, L.M., Manso, L.A.F.: Performance comparison of metaheuristics to solve the multi-stage transmission expansion planning problem. Generation, Transmission & Distribution, IET 5(3), 360–367 (2011)
9. Taylor, J.A., Hover, F.S.: Linear Relaxations for Transmission System Planning. IEEE Transactions on Power Systems 26(4), 2533–2538 (2011)
10. Romero, R., Monticelli, A., Garcia, A., Haffner, S.: Test systems and mathematical models for transmission network expansion planning. IEE Proc., Gener., Transm. Distrib. 149(1), 27–36 (2002)
11. Chung, T.S., Li, K.K., Chen, G.J., Xie, J.D., Tang, G.Q.: Multiobjective transmission network planning by a hybrid GA approach with fuzzy decision analysis. Int. J. Electr. Power Energy Syst. 25, 187–192 (2003)
12. Lee, C.W., Ng, S.K.K., Zhong, J., Wu, F.F.: Transmission Expansion Planning From Past to Future. In: Power Systems Conference and Exposition, PSCE 2006, pp. 257–265. IEEE PES (2006)
13. Latorre, G., Cruz, R.D., Areiza, J.M., Villegas, A.: Classification of publications and models on transmission expansion planning. IEEE Transactions on Power Systems 18(2), 938–946 (2003)
14. Festa, P., Resende, M.G.C.: Hybridizations of GRASP with path-relinking. AT&T Labs Research, 1–19 (2011)
15. Schrijver, A.: Theory of linear and integer programming. John Wiley & Sons, Ltd., West Sussex (1996)
16. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8, 67–71 (1989)
17. Garver, L.L.: Transmission network estimation using linear programming. IEEE Trans. Power Appl. Syst. 89(7), 1688–1697 (1970)
18. Pereira, M.V.F., Pinto, L.M.V.G.: Application Of Sensitivity Analysis Of Load Supplying Capability To Interactive Transmission Expansion Planning. IEEE Transactions on Power Apparatus and Systems 104(2), 381–389 (1985)
19. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 219–249. Kluwer Academic Publishers (2003)
20. Rahmani, M., Rashidinejad, M., Carreno, E.M., Romero, R.A.: Evolutionary multimove path-relinking for transmission network expansion planning. In: 2010 IEEE Power and Energy Society General Meeting, pp. 1–6 (2010)
21. Romero, R., Rider, M.J., de Silva, I.J.: A Metaheuristic to Solve the Transmission Expansion Planning. IEEE Trans. Power Systems 22(4), 2289–2291 (2007)
22. Fourer, R.D., Gay, M., Kernighan, B.W.: AMPL: A Modeling Language for Mathematical Programming, 2nd edn. Brooks/Cole-Thomson Learning, Pacific Grove (2003)
23. CPLEX Optimization Subroutine Library Guide and Reference, Version 12.1.0. CPLEX Division, ILOG, Inc., Incline Village, NV (2009)

Electrical Load Management in Smart Homes Using Evolutionary Algorithms

Florian Allerding[1], Marc Premm[2],
Pradyumn Kumar Shukla[1], and Hartmut Schmeck[1]

[1] Karlsruhe Institute of Technology – Institute AIFB
76128 Karlsruhe, Germany
{florian.allerding,pradyumn.shukla,hartmut.schmeck}@kit.edu
[2] Universität Hohenheim – Wirtschaftsinformatik 2
70599 Stuttgart-Hohenheim, Germany
marc.premm@uni-hohenheim.de

Abstract. In this paper, we focus on a real world scenario of energy management of a smart home. External variable signals, reflecting the low voltage grid's state, are used to address the challenge of balancing energy demand and supply. The problem is formulated as a nonlinear integer programming problem and a load management system, based on a customized evolutionary algorithm with local search, is proposed to control intelligent appliances, decentralized power plants and electrical storages in an optimized way with respect to the given external signals. The nonlinearities present in the integer programming problem makes it difficult for exact solvers. The results of this paper show the efficacy of evolutionary algorithms for solving such combinatorial problems.

Keywords: Nonlinear Integer Program, Energy Management, Evolutionary Algorithms.

1 Introduction

The climate protection targets of the German government aim to cover 35 percent of the electricity consumption from renewable sources by 2020. This growing share of renewable energy sources will cause a higher demand for flexible power suppliers and consumers. So here we present an energy management system that has been developed in response to the challenge of balancing supply and demand in the electric grid in spite of volatile, widely uncontrollable power production. This approach has been investigated using a real smart home built with about 60 m^2 of living area. This smart home is equipped with decentralized power plants, intelligent appliances, and an electrical car. External signals, reflecting the state of the low voltage grid, are sent to the smart home, which is able to adapt its energy demand and production automatically without constraining the end consumer. Complying with constraints specified by the household appliance itself or by the user, the load of appliances is shifted within the specified degrees of freedom.

This paper is divided into seven sections of which this is the first. In Sect. 2 some technical aspects of the real smart home setup are shortly introduced,

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 99–110, 2012.

whereby Sect. 3 focuses on the description of the given optimization problem. The modeling of this problem is pointed out in Sect. 4. A solver for the specific problem based on an evolutionary algorithm is presented in Sect. 5. After some experimental results (including comparison with an exact solver) in Sect. 6 the paper concludes with a summary and a discussion of future work in Sect. 7.

2 In-House Energy Management

The complexity of technical systems is constantly increasing. Breakdowns and fatal errors are occurring quite often, respectively. To achieve the goals of designing and controlling complex systems, adequate methods, techniques, and system architectures are needed. Therefore, a regulatory feedback mechanism is proposed, the so called generic observer/controller (o/c) architecture [10], which constitutes one way to achieve controlled self-organization in technical systems. The o/c uses a set of sensors and actuators to measure system variables and to influence the system. Together with the system under observation and control (SuOC), the o/c forms the so called organic system. An o/c loop activates adequate reactions to control the emerging global behavior resulting from interactions between local agents. The observer monitors certain (raw) attributes of the system and aggregates them to situation parameters, which concisely characterize the observed situation from a global point of view, and passes them to the controller. The controller acts according to an evaluation of the observation. The energy management system for the smart home presented here is based on a hierarchical o/c architecture as shown in Fig. 1. Between the real hardware components of the household and the energy management system, represented as o/c architecture, a hardware abstraction layer (HAL) is implemented to make an abstraction from the manufacture specific protocols of the components. The HAL is more closely described in [2]. In the local o/c units the observed data

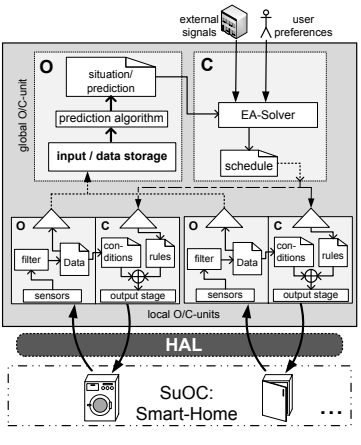


Fig. 1. o/c Architecture in the smart Home

of the hardware components are filtered and communicated to the global component, where the current state of the complete household is aggregated and a prediction for the next optimization horizon can be calculated [3]. Based on the prediction, the global controller calculates an optimized schedule for the components in the household. This schedule is communicated to the local o/c-units, where the local controllers have the final decision to change the state of the hardware components based on the more precise information given from the local observers. The present paper is now focused on the global controller part in Fig. 1 where the optimization is done.

There are quite a few other approaches to autonomous systems for energy management based on optimization techniques for the low voltage grid. Mainly, they discuss electrical or economical dispatch problems in the low voltage grid. On the other hand, some approaches are based on the in-house scenario as the *organic computing* [10] inspired approach in the present paper. An approach with a multi-agent model by Shadi Abras et al. [1] handles a load limitation scenario with a fix load limit on the one hand, and on the other hand a two step price-signal is proposed: a high rate tariff for daytime and a low rate tariff at night, as it is common in France and Germany, for example. The optimization problem is described as a Resource Constrained Project Scheduling Problem (RCPSP), a tabu search algorithm is provided for solving it. Since electrical heating devices are included in the considered scenario, the optimization process is also influenced by the current weather data.

3 Problem Description

As already discussed, external signals are sent periodically to the households reflecting the current grid state. As shown in Fig. 2 these signals may be a variable price rate over the day and a load limitation curve. The household appliances have a specific load profile (the gray boxes in Fig. 2) and temporal degree of freedom (tDoF). The tDoF is the span between the release time r_j and the deadline d_j in which the user defines when the appliance j may run. (For instance: the user fills up his washing machine with laundry before work in the morning and defines that it has to be done eight hours later after work.) The challenge of the household energy management is now to minimize the electrical energy costs for the consumer and to minimize the violation of the load limitation curve. So, the energy management has to calculate the optimal starting point for the different appliances.

The given problem can be seen as similar to the Time Constrained Project Scheduling Problem (TCPSP) as introduced in [6]. The TCPSP is derived from the Resource Constrained Project Scheduling Problem (RCPSP) with the difference that the resource limit is not a hard deadline and a penalty function for the violation of the limited resources is proposed. In the smart home scenario the limited resources are equivalent to the load limitation curve. The work item of a washing machine for example can be defined as a job with the tDoF of the appliance as constraint.

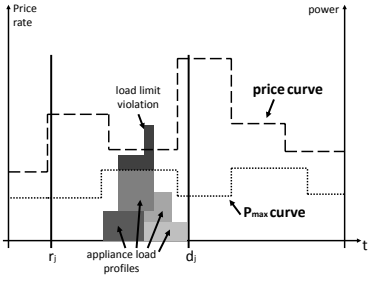


Fig. 2. External signals and load profiles

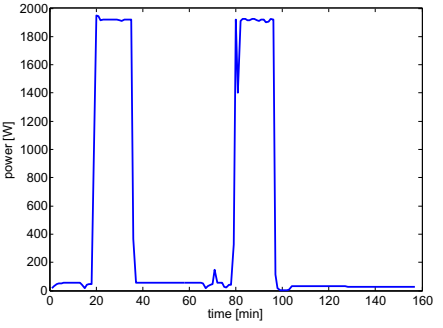
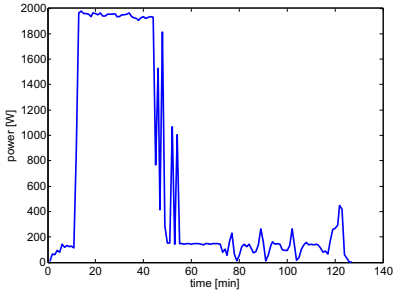


Fig. 3. Typical dish washer profile



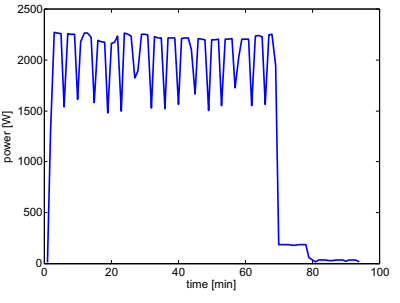


Fig. 4. Typical washing machine (left) and tumble dryer (right) profiles

The main difference between the problem described in [6] and the problem in the present paper is, that the cost caused by one job are not constant. For instance in the TCPSP formulation a job has a constant demand on manpower during the job. But a dishwasher for example has a characteristically variable and nonlinear electrical demand function as it can be seen in Fig. 3. Similar is the case for a washing machine and a tumble dryer (see Fig. 4). Due to these difficulties, the present problem cannot be modeled as an Integer Linear Program (ILP) (as opposed to the TCPSP formulation in [6]). This fact will be discussed in the next section.

4 Modeling the Problem

In this section, we model the electrical load management problem as nonlinear integer programming (NIP) problem. For this, as it is common practice in project scheduling, we assume to have a discrete time horizon with $t \in \{0, ..., T\}$ [6]. Every appliance respectively appliance program will be represented by one job. Thus with N jobs we get a set of jobs, $J := \{j_1, j_2, ..., j_N\}$. The starting time of every job j is constrained by the release time r_j which represents the earliest

starting time and the deadline d_j which represents the point in time when j has to be already finished. Together with r_j and d_j the processing time p_j of j defines the tDoF with $tDoF_j = d_j - r_j - p_j$. Furthermore, let the vector $q_{j,t}$ represents the amount of power job j requires in time slot t after its starting time. So, t in this case is relative to the starting time of job j making $q_{j,t}$ independent of t_j.

With this set of jobs a schedule for the appliances can be built. Let t_j denote the starting time of j. A schedule is represented by the two matrices $\widehat{S} = (s_{j,t})$ and $\widehat{X} = (x_{j,t})$ with $s_{j,t} = 1$ respectively $x_{j,t} = 1$ if job j starts respectively is active in t or $s_{j,t} = 0$ respectively $x_{j,t} = 0$ if not. For an easier calculation of the total costs a schedule also contains the vector H_t for the aggregated power load with $H_t = \sum_j x_{j,t} \cdot q_{j,t-t_j}$. The price per kWh can be resolved by the discrete function $P(t)$. The electrical load management problem can be modeled as the following (NIP):

$$\min \sum_{t \in T} P(t) \left(\sum_{j \in J} x_{j,t} q_{j,t-t_j} \right) + \delta P_f \sum_{t \in T} P(t) \left(\sum_{j \in J} x_{j,t} \cdot q_{j,t-t_j} - L(t) \right)$$

$$\text{subject to } \sum_{t=r_j}^{d_j-1} x_{j,t} = p_j, \qquad \forall j \in J; \tag{1}$$

$$\sum_{t=r_j}^{d_j-p_j} s_{j,t} = 1, \qquad \forall j \in J; \tag{2}$$

$$x_{j,r_j} = s_{j,r_j}, \qquad \forall j \in J \tag{3}$$

$$x_{j,t_j+l-1} + s_{j,t_j+l} \geq x_{j,t_j+l}, \quad \forall l \in \{1, 2, \ldots, d_j - p_j\}, \forall j \in J \tag{4}$$

$$s_{j,t} = 0, \qquad \forall t \notin \{r_j, ..., d_j - p_j\}; \tag{5}$$

$$x_{j,t} = 0, \qquad \forall t \notin \{r_j, ..., d_j - 1\}; \tag{6}$$

$$\sum_{j \in J} x_{j,t} q_{j,t-t_j} - L(t) \leq M\delta, \qquad \forall t \in T; \tag{7}$$

$$\sum_{j \in J} x_{j,t} q_{j,t-t_j} - L(t) \geq \delta - 1 \qquad \forall t \in T \tag{8}$$

$$s_{j,t} \in \{0, 1\} \qquad \forall j \in J, \forall t \in T \tag{9}$$

$$x_{j,t} \in \{0, 1\} \qquad \forall j \in J, \forall t \in T \tag{10}$$

$$\delta \in \{0, 1\}, \tag{11}$$

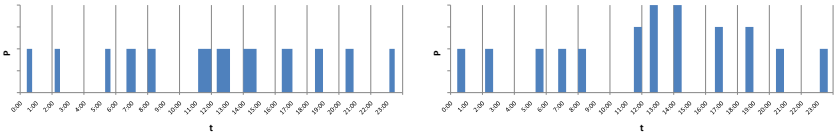
where $L(t)$ is the discrete load limitation function (for all $t \in T$), P_f is a penalty factor for the violation of the load limitation, and M is a large number. M is a constant coefficient representing an upper bound on $\sum_{j \in J} x_{j,t} \cdot q_{j,t-t_j} - L(t)$ and can be easily taken as: $M := N \max_{j \in J, t \in T} q_{j,t}$.

As the values $x_{j,t}$ in the matrix $\widehat{X}$ depend on the value of t_j, it can be seen that the objective function of the above program is a nonlinear function. The objective function represents the main goal, i.e., to minimize the total costs. The first term in the objective function is the price for energy consumption and the second term

(together with a penalty factor P_f) describes the extra costs as a result of the violation of the load curve. Constraint 1 ensures that the processing time for each job will be exactly as long as its program duration, while constraint 2 ensures the jobs to be started exactly once between their release time and deadline. Constraint 3 ensures that job is active in first possible time slot if it also starts in the same time slot. Constraint 4 ensures that in a particular time slot, the job can only be active if it starts in that slot or was active in the previous time slot. To fulfill the time restrictions of each job constraints 5 and 6 ensure that every $s_{j,t}$ and $x_{j,t}$ outside of the possible $tDoF_j$ respectively $tDoF_j + p_j$ is set to Zero. Constraints 7 and 8 impose the logical condition that ensures that δ is one whenever there the load limitation curve is violated (see the discussion in [11]). Constraints 9, 10 and 11 finally define the domain for the variables used.

The above problem is a nonlinear integer program and in principle could be solved using an exact mixed integer nonlinear program (MINLP) solver (see [7]). However, as we will see in Sec. 6, the lack of convexity, nonlinearities in the objective function, and a large number of variables makes this problem quite difficult to be solved to global optimality. Even the simpler problem consider in [6], where the cost caused by one job are assumed to be constant (in our problem the cost is variable and nonlinear), is NP-hard.

We will now show that the problem formulation presented above isn't restricted to periodically used household appliances (timed services) like washing machines or dryers. Even though a permanently running service (permanent services) like a deep freezer or an air conditioning seems to require more variables than the tuple $(r_j, p_j, d_j, q_{j,t})$, even these appliance types can be included in the model. A more detailed definition of timed services and permanent services is done in [2]. Every power curve for each of these permanent services has the same structure: The appliance will require a relevant amount of electrical energy only periodically. Therefore these appliances will be represented by one job for each start. The fridge in the aforementioned real smart home for example started periodically about 32 times a day for 10 minutes. So, in this case the time horizon will be split into 32 even intervals each containing 45 minutes. In general a permanently running appliance with N starts per day, a permanent processing time p and consumption vector q_t for each start will be represented by N tuples, $\left(\frac{T}{N}(k-1); p; \frac{T}{N} \cdot k\right)$ with $k \in \{1, ..., N\}$. An appliance which is more dependent on the time of day like the already mentioned air conditioning may also adapt each single job. Fig. 5(a) shows the possibility of extending the processing time with the same power consumption at high noon to compensate the stronger insulation. As an alternative Fig. 5(b) illustrates the possibility of increasing the cooling capacity as far as the appliance is capable of this feature. Of course, a combination of both adaptations is also possible. Another type of appliance which is included in the model is the electric vehicle with a variable power consumption. This variable power consumption is needed to respond flexibly on imbalances in the energy grid. Therefore, we assume that this kind of appliance can vary its power consumption in linearly increasing discrete steps. The total power consumption will be divided by the number of these steps.



(a) variable job duration (b) variable cooling capacity

Fig. 5. Power curve air of an conditioning device with 12 time-segments

Every appliance is then again represented by multiple jobs having the exactly same tuple (r, p, d, q_t). Fig. 6 shows a possible behavior for the loading process of an electric vehicle. The total power consumption is known, in this case 20 kWh and it is assumed that the power can only be increased by 1 kW for each step.

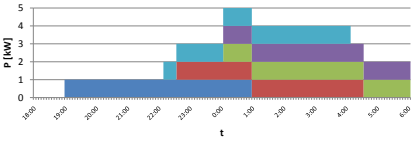


Fig. 6. Power curve of an electric car with 5 power steps

5 An Evolutionary Algorithm with Local Search

Evolutionary algorithms (EA) have already proven to efficiently solve similar scheduling problems like for example [9]. One elementary advantage of EAs is their memory and time requirements which are basically independent of the number of jobs even though the search space increases exponentially as it is in the present case.

5.1 Solution Representation

As we will not allow interrupts in each job program, the only variable that can be directly manipulated is the starting point represented in the matrix $\widehat{S}$. With $\widehat{S}$ and all job information it is possible to construct $\widehat{X}$ as well as the vector H_t. The fitness function will be defined as the total costs which are caused by $H_t \; \forall \, t \in \{0, ..., T\}$. Hence, the matrix $\widehat{S}$ will be used as the genetic representation of the solution domain. The matrix $\widehat{S}$ is of size $N \times T$ and every row of the matrix has exactly one nonzero element. For example, if we consider 4 jobs, $\widehat{S}$ may have the following structure:

$$\widehat{S} = \begin{pmatrix} 0\,0\,0\,1\,0\,0\,0\,0\,0\,0 \cdots 0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \cdots 1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0 \cdots 0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \cdots 0\,0 \end{pmatrix}.$$

5.2 Selection and Reproduction Scheme

Based on the lack of necessity for numeric scale and usually good overall performance, a rank-based selection procedure is used. All individuals of the current population will be ranked by their fitness value. The probability of choosing one individual with rank k is defined by $P(k) = \frac{N-k+1}{\sum_{i=1}^{N} i}$. This method is used for the selection of parents to reproduce new individuals and for the selection of individuals which will build the next generation.

Several combinations of evolution strategies have been tested (see Fig.8). The results have shown that for this case a $(\mu, \mu + \lambda)$-strategy shows a slightly faster convergence than a (μ, λ)-strategy while reaching equivalent or partially better solutions. The number of offspring is set to 2 for every parent individual. This parameter has shown a good overall performance especially when computing time is considered.

5.3 Customized Search Operators

Because the order of the inserted jobs is random, a uniform crossover operator is used as the recombination procedure. The method used in this case decides for every job whether the child will adopt the starting point from the first or the second parent. Therefore the starting points for all jobs in the first child-schedule are represented in the matrix $\widehat{S}_c = \left(s_{cj,t}\right)$ with

$$s_{cj,t} = \alpha_j \cdot s_{p1_{j,t}} + (1 - \alpha_j) \cdot s_{p2_{j,t}}, \forall\, j \in \{1, ..., N\}, \forall\, t \in \{0, ..., T\}, \alpha_j \in \{0, 1\}\,.$$

The variable α_j has to be randomly chosen for every job j. For this purpose we have chosen a uniform distribution between both jobs, therefore none of them is preferred to the other. The second child-schedule will contain all starting times which were not chosen for the first child-schedule, so none of the information of either parent will be lost through this operation. This recombination procedure even ensures that every newly generated schedule generated by two feasible schedules will also be feasible.

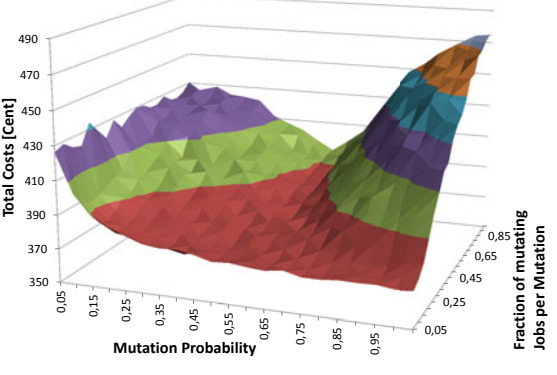


Fig. 7. Relation between mutation probability and fraction of mutating jobs per mutation

A mutating individual will randomly change a certain fraction of starting times. First the jobs whose variables will be changed are randomly selected. For each of these jobs the starting time will be randomly set to any point in time with respect to its tDoF: $t_j \in \{t | r_j \leq t \leq d_j - p_j, s_{j,t} = 1\}$. With this method it is once more ensured that newly generated individuals always satisfy the time constraints for each job.

Fig. 7 shows the relation between the mutation probability and the fraction of mutating jobs per mutation. With these results, the lowest costs where achieved with the mutation probability set to 0.5 and the fraction of mutating jobs set to 0.1.

5.4 Local Search

The EA will be combined with a local search technique to further optimize the best solution found. Therefore the local search algorithm will shift the starting point by 1 and by -1 for every job in the schedule. Only these shifts will be tested that will create a feasible schedule. Cost improvements of each single change will be stored. Finally, the single change of the starting time gaining the highest cost decrease will be performed. This procedure will be repeated until no cost improvements can be realized by changing a single starting time autonomously.

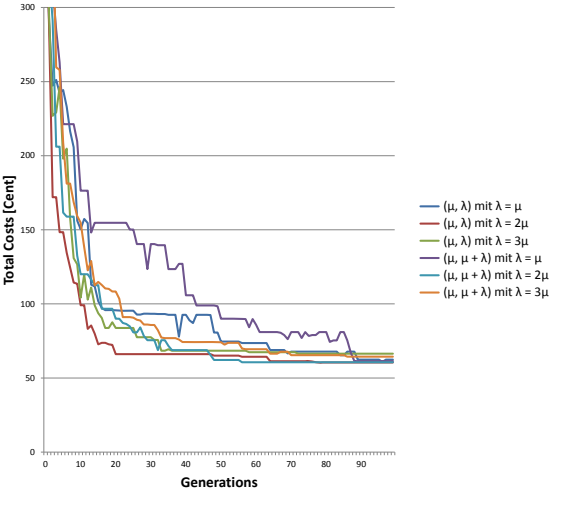


Fig. 8. Several combinations of evolutions strategies

6 Simulation Results

For the evaluation of the presented optimization methods 4000 problem instances based on the data from past living periods has been created. We have assumed

$T = 1440$ based on a one minute discretization of a day. Moreover, we have tested 32 jobs corresponding to each of the 4000 instance. The instances have been generated by changing the release time and the deadlines for every job.

The price curve was differentiated between summer and winter while the load limitation curve follows the H0 standard load profile of VDEW [8] for German households in the cyclic version and shows a contrary course of the curve in the anti-cyclic version. The latter is likely to animate residents to shift their electrical load to daytimes with lower overall load. The penalty factor for exceeding the power limit has been set to 5 to ensure noticeable extra costs but without the character of a hard limit. To a ensure replicability the same instances has been tested with three different solvers witch had to find a solution for the same problem.

The first simple solver optimizes each single job on its own by testing and evaluating every possibility. In doing so, the procedure won't consider coherence between two jobs and the probability that a job will be executed parallel to another one is relatively high. Therefore this solver will be called the *individual* solver. This procedure ignores the load limitation curve.

The EA has been tested in two configurations each with a population of 50 individuals: The short one with only 20 generations and a calculation time of 1 second and the long one with 1600 generations and a calculation time of 60 seconds. Both run times refer to an Intel Core i7 M 620 with 8 GB of memory. It should be noticed, that in practice a small energy saving computer system is proposed. So the duration for the optimization might increase to some minutes but this will be still feasible for solving the present problem.

We have also used a mixed-integer nonlinear programming (MINLP) optimizer that uses a branch-and bound search scheme. This solver is from the TomLab optimization suite (the function MINLPBB in Tomlab)[1] and was developed by Roger Fletcher and Sven Leyffer. Due to the nonlinearities present in the objective function, we were not able to use standard mixed-integer linear programming solvers like Cplex, Gurobi, and Xpress.

Table 1 shows the results of the simulation. For the parameter tests every pass used the same seed to be able to compare the results. Based on these instances an "individual solver", an exact solver (MINLP) and two configuration of an evolutionary algorithm has been evaluated. The solvers have been compared between each other and a reference value. This reference value in this case represents the costs which will occur when no optimization is performed and all appliances are started at their release time r_j. This procedure reflects the behavior in classical households because the release time represents the time when the appliance would be turned on.

The results show that the EA could decrease the costs reached by the other solvers in the short run. In the long run the EA found much better results. Above all the EA reached a much lower standard deviation especially in the anti-cyclic case. Even the results reached in one second of calculation are much more stable than those of the other methods while this stability can still be increased by

[1] http://tomopt.com/tomlab/optimization/minlp.php

passing more generations and thus prolonging the calculation time. The MINLP solver faced difficulties and was not able to find the global optima. The results of the MINLP solver were found to depend heavily on the starting point.

Table 1. Simulation results for the 4000 instances. The values are in cents.

(a) (Summer, cyclic)

	Reference	Individ.	MINLP Sol.	EA short	EA long
Best	1059.90	642.15	587.50	482.48	474.02
Median	1280.86	721.42	631.20	530.14	515.37
Std.-dev.	176.40	43.83	52.03	20.85	19.89
Worst	1839,08	883.68	709.42	594.26	576.36

(b) (Summer, anti-cyclic)

	Reference	Individ.	MINLP Sol.	EA short	EA long
Best	2371.96	428.69	589.12	406.51	399.95
Median	2616.17	512.03	614.56	454.77	448.57
Std-dev.	163.61	35.92	29.60	18.20	17.50
Worst	3119.75	660.39	670.20	552.50	531.05

(c) (Winter, cyclic)

	Reference	Individ.	MINLP Sol.	EA short	EA long
Best	1148.50	584.49	536.20	453.90	446.50
Median	1398.41	669.07	560.26	514.46	502.42
Std-dev.	200.27	40.16	26.83	23.19	21.34
Worst	2040.26	812.55	605.60	572.94	556.90

(d) (Winter, anti-cyclic)

	Reference	Individ.	MINLP Sol.	EA short	EA long
Best	2628.17	404.35	505.40	381.17	379.07
Median	2878.74	473.98	529.20	433.47	427.19
Std-dev.	189.51	34.97	38.04	18.20	17.45
Worst	3480.72	634.92	587.30	538.42	531.92

7 Conclusions and Outlook

In this paper a modeling approach for commonly used household appliances including electrical vehicles is presented. With the objective of reducing load peaks and supporting the integration of fluctuating renewable energy resources by the given external signals, an Evolutionary Algorithm is introduced to solve the energy management problem for private households. For comparison, the EA has been tested against the reference value and an exact optimization strategy. The performance of the EA even exceeded the other two even in the short run. Especially, the EA managed to gain a much lower standard deviation over all results. This implies that the results gained by the EA are much more stable.

To extend the current scenario there are further appliances that could be added. Analogously to the air conditioning a fridge and a deep freezer could be appended. It has been observed that the fridge in the smart home has an average of 32 starts per day, even more than the already implemented air conditioning. This would double the number of jobs used in the evaluation and it is likely that this leads to an increasing advance for the EA. Moreover, a combined heat and power plant should be considered. This again makes it necessary to consider thermal components. That makes it possible to simultaneously examine electrical and thermal energy. Besides the combined heat and power plant, there are also other appliances that are able to produce electrical energy. In the smart home, these are for example the photovoltaic panels and the electrical vehicle which is also able to feed back electrical energy into the energy grid or the smart home.

Besides extending the test set, the algorithm should be implemented in a real world scenario. Therefore, the simulation environment in the smart home can be easily exchanged with the real smart home [2]. In this way, the results gained by the simulation presented above could be verified by using real appliances

and time restrictions. As a future work, we plan to consider the multi-objective version of the electrical load management problem. It would be interesting to see trade-offs that occur due to the violation of the load curve. Another interesting idea would be to smoothen the load profiles and use exact algorithms [4,7,5].

References

1. Abras, S., Ploix, S., Pesty, S., Jacomino, M.: A multi-agent home automation system for power management. In: Informatics in Control Automation and Robotics. Lecture Notes in Electrical Engineering. Springer (2009)
2. Allerding, F., Schmeck, H.: Organic smart home - architecture for energy management in intelligent buildings. In: ICAC 2011 (2011)
3. Bao, K., Allerding, F., Schmeck, H.: User behavior prediction for energy management in smart homes. In: Proceedings of the FSKD 2011 (2011)
4. Fischer, A., Shukla, P.K.: A Levenberg-Marquardt algorithm for unconstrained multicriteria optimization. Oper. Res. Lett. 36(5), 643–646 (2008)
5. Fischer, A., Shukla, P.K., Wang, M.: On the inexactness level of robust levenberg–marquardt methods. Optimization 59(2), 273–287 (2010)
6. Guldemond, T., Hurink, J., Paulus, J., Schutten, J.: Time-constrained project scheduling. Journal of Scheduling (2008)
7. Li, D., Sun, X.: Nonlinear integer programming. International Series in Operations Research & Management Science, p. 84. Springer, New York (2006)
8. Meier, H., Fünfgeld, C., Schieferdecker, B.: Repräsentative VDEW-Lastprofile. Tech. rep., Verband der Elektrizitätswirschaft (1999)
9. Mesghouni, K., Hammadi, S.: Evolutionary algorithms for job-shop scheduling. Applied Mathematics and Computer Science (2004)
10. Müller-Schloer, C., Schmeck, H., Ungerer, T.: Organic Computing - A Paradigm Shift for Complex Systems. Birkhauser Verlag AG (2011)
11. Williams, H.P.: Model building in mathematical programming, 3rd edn. A Wiley-Interscience Publication, John Wiley & Sons Ltd, Chichester (1990)

Exact Computation of the Fitness-Distance Correlation for Pseudoboolean Functions with One Global Optimum

Francisco Chicano and Enrique Alba*

University of Málaga, Spain
{chicano,alba}@lcc.uma.es

Abstract. Landscape theory provides a formal framework in which combinatorial optimization problems can be theoretically characterized as a sum of a special kind of landscapes called elementary landscapes. The decomposition of the objective function of a problem into its elementary components can be exploited to compute summary statistics. We present closed-form expressions for the fitness-distance correlation (FDC) based on the elementary landscape decomposition of the problems defined over binary strings in which the objective function has one global optimum. We present some theoretical results that raise some doubts on using FDC as a measure of problem difficulty.

Keywords: Landscape theory, fitness landscapes, fitness-distance correlation.

1 Introduction

The theory of landscapes focuses on the analysis of the structure of the search space that is induced by the combined influences of the objective function of the optimization problem and the neighborhood operator [16]. In the field of combinatorial optimization, this theory has been previously used to characterize optimization problems [8], improve search algorithms [12], and obtain global statistics of the problems [20].

A *landscape* for a combinatorial optimization problem (COP) is a triple (X, N, f), where X is the set of *tentative solutions* of the COP, $f : X \mapsto \mathbb{R}$ defines the objective or *fitness function* and N is the *neighborhood operator*. There exists a special kind of landscapes, called *elementary landscapes* (EL), which are of particular interest due to their properties [22]. We define and analyze the elementary landscapes in Section 2, but we can advance that they are characterized by the *Grover's wave equation*:

$$\operatorname*{avg}_{y \in N(x)}\{f(y)\} = \frac{1}{d} \sum_{y \in N(x)} f(y) = f(x) + \frac{\lambda}{d}\left(\bar{f} - f(x)\right), \tag{1}$$

* Research partially funded by the Spanish Ministry of Science and Innovation and FEDER under contracts TIN2008-06491-C04-01 and TIN2011-28194 and the Andalusian Government under contract P07-TIC-03044.

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 111–123, 2012.

where d is the size of the neighborhood, $|N(x)|$, which we assume is the same for all the solutions in the search space, $\bar{f}$ is the average solution evaluation over the entire search space and λ is a characteristic constant. For a given problem instance whose objective function is elementary, the values $\bar{f}$ and λ can be easily computed in an efficient way, usually from the problem data. Thus, the wave equation makes it possible to compute the average value of the fitness function f evaluated over all of the neighbors of x using only the value $f(x)$, without actually evaluating any of the neighbors.

When the landscape is not elementary it is always possible to write the objective function as a sum of elementary components, called *elementary landscape decomposition* (ELD) of a problem [6]. In the case of binary strings with length n under the one-change neighborhood, the number of elementary components is at most n. Then, Grover's wave equation can be applied to each elementary component and all the results are summed to give the average fitness in the neighborhood of a solution. Furthermore, for some problems the average cannot be limited to the neighborhood of a solution, but it can be extended to the second-order neighrbors (neighbors of neighbors), third-order neighbors, and, in general, to any arbitrary region around a given solution, including the whole search space. Sutton *et al.* [19] show how to compute the averages over *spheres* and *balls* of arbitrary radius around a given solution in polynomial time using the elementary landscape decomposition of pseudoboolean functions.

Landscape theory has been proven to be quite effective computing summary statistics of the optimization problem. Measures like the autocorrelation length and the autocorrelation coefficient can be efficiently computed using the ELD of a problem [8]. Recently, Chicano and Alba [5] and Sutton and Whitley [18] have shown how the expected value of the fitness of a mutated individual can be exactly computed using the ELD. In short, landscape theory can be applied to any COP and thus is generally beneficial for the whole community in discrete optimization, representing a general and usable formalism in practice.

The main contribution of the present work is an exact expression for the Fitness-Distance Correlation (FDC) of COPs defined over a set of binary strings (pseudoboolean functions) having one global optimum. This expression is based on the ELD of the problem. We also analyze the expression in order to dicuss the usefulness of the FDC as a difficulty measure for a problem.

The remainder of the paper is organized as follows. In Section 2 we present the mathematical tools required to understand the rest of the paper. Section 3 presents the exact expression for the FDC and other theoretical results, while Section 4 validates FDC in practice with this theoretical background. Finally, we summarize our findings and future work in Section 5.

2 Background

In this section we present some fundamental results of landscape theory. We will only focus on the relevant information required to understand the rest of the paper. The interested reader can deepen on this topic in [15].

Let (X, N, f) be a landscape, where X is a finite set of candidate solutions, $f : X \to \mathbb{R}$ is a real-valued function defined on X and $N : X \to 2^X$ is the neighborhood operator. The pair (X, N) is called *configuration space* and induces a graph in which X is the set of nodes and an arc between (x, y) exists if $y \in N(x)$. The adjacency and degree matrices of the neighborhood N are:

$$A_{xy} = \begin{cases} 1 \text{ if } y \in N(x), \\ 0 \text{ otherwise;} \end{cases} \qquad D_{xy} = \begin{cases} |N(x)| \text{ if } x = y, \\ 0 \qquad \text{otherwise.} \end{cases} \tag{2}$$

We restrict our attention to regular neighborhoods, where $|N(x)| = d > 0$ for a constant d, for all $x \in X$. Then, the degree matrix is $D = dI$, where I is the identity matrix. The Laplacian matrix Δ associated to the neighborhood is defined by $\Delta = A - D$. In the case of regular neighborhoods it is $\Delta = A - dI$. Any discrete function, f, defined over the set of candidate solutions can be characterized as a vector in $\mathbb{R}^{|X|}$. Any $|X| \times |X|$ matrix can be interpreted as a linear map that acts on vectors in $\mathbb{R}^{|X|}$. For example, the adjacency matrix A acts on function f as follows

$$A\ f = \begin{pmatrix} \sum_{y \in N(x_1)} f(y) \\ \sum_{y \in N(x_2)} f(y) \\ \vdots \\ \sum_{y \in N(x_{|X|})} f(y) \end{pmatrix}, \quad (A\ f)(x) = \sum_{y \in N(x)} f(y). \tag{3}$$

Thus, the component x of $(A\ f)$ is the sum of the function values of all the neighbors of x. Stadler defines the class of *elementary landscapes* where the function f is an eigenvector (or eigenfunction) of the Laplacian up to an additive constant [16].

Definition 1. *Let (X, N, f) be a landscape and Δ the Laplacian matrix of the configuration space. The landscape is said to be elementary if there exists a constant b, which we call* offset*, and an eigenvalue λ of $-\Delta$ such that $(-\Delta)(f - b) = \lambda(f - b)$. When the neighborhood is clear from the context we also say that f is elementary.*

We use $-\Delta$ instead of Δ in the definition to avoid negative eigenvalues, since Δ is negative semidefinite. In connected neighborhoods, where the graph related to the configuration space (X, N) is connected, the offset b is the average value of the function over the whole search space: $b = \bar{f}$. Taking into account basic results of linear algebra, it can be proved that if f is elementary with eigenvalue λ, $af + b$ is also elementary with the same eigenvalue λ. Furthermore, in regular neighborhoods, if g is an eigenfunction of $-\Delta$ with eigenvalue λ then g is also an eigenfunction of A, the adjacency matrix, with eigenvalue $d - \lambda$. The average value of the fitness function in the neighborhood of a solution can be computed

using the expression $\text{avg}\{f(y)\}_{y\in N(x)} = \frac{1}{d}(A\,f)(x)$. If f is an elementary function with eigenvalue λ, then the average is computed as:

$$\begin{aligned}\underset{y\in N(x)}{\text{avg}}\{f(y)\} &= \underset{y\in N(x)}{\text{avg}}\{f(y)-\bar{f}\}+\bar{f} = \frac{1}{d}(A\,(f-\bar{f}))(x)+\bar{f}\\ &= \frac{d-\lambda}{d}(f(x)-\bar{f})+\bar{f} = f(x)+\frac{\lambda}{d}(\bar{f}-f(x)),\end{aligned}$$

and we get Grover's wave equation. In the previous expression we used the fact that $f-\bar{f}$ is an eigenfunction of A with eigenvalue $d-\lambda$.

The previous definitions are general concepts of landscape theory. Let us focus now on the binary strings with the one-change neighborhood, which is the representation and the neighborhood we use in the next section to compute the fitness-distance correlation. In this case the solution set X is the set of all binary strings of size n. Two solutions x and y are neighboring if one can be obtained from the other by flipping a bit, that is, if the Hamming distance between the solutions, denoted with $\mathcal{H}(x,y)$, is 1.

One relevant set of eigenvectors of the Laplacian in the binary representation is that of Walsh functions (4). Furthermore, the Walsh functions form an orthogonal basis of eigenvectors in the configuration space. Thus, they have been used to find the elementary landscape decomposition of problems with a binary representation like the MAXSAT [14]. Given the space of binary strings of length n, $\mathbb{B}^n$, a (non-normalized) Walsh function with parameter $w\in\mathbb{B}^n$ is defined as:

$$\psi_w(x) = \prod_{i=1}^{n}(-1)^{w_i x_i} = (-1)^{\sum_{i=1}^{n} w_i x_i}. \tag{4}$$

Two useful properties of Walsh functions are $\psi_w\cdot\psi_v = \psi_{w+v}$ where $w+v$ is the bitwise sum in $\mathbb{Z}_2$ of w and v; and $\psi_w^2 = \psi_w\cdot\psi_w = \psi_{2w} = \psi_0 = 1$. We define the *order* of a Walsh function ψ_w as the value $\langle w|w\rangle = \sum_{i=1}^{n} w_i$, that is, the number of ones in w. A Walsh function with order p is elementary with eigenvalue $\lambda = 2p$ [16]. The average value of a Walsh function of order $p>0$ is zero, that is, $\overline{\psi_w} = 0$ if w has at least one 1. The only Walsh function of order $p=0$ is $\psi_0 = 1$, which is a constant.

Since the Walsh functions form an orthogonal basis of $\mathbb{R}^{2^n}$, any arbitrary pseudoboolean function can be written as a weighted sum of Walsh functions in the following way: $f = \sum_{w\in\mathbb{B}^n} a_w\psi_w$, where the values a_w are called Walsh coefficients. We can group together the Walsh functions having the same order to find the elementary landscape decomposition of the function. That is: $f_{[p]} = \sum_{\substack{w\in\mathbb{B}^n\\ \langle w|w\rangle = p}} a_w\psi_w$, where each $f_{[p]}$ is an eigenvector of the Laplacian with eigenvalue $2p$, also called order-p elementary component of f. The function f can be written as a sum of the $n+1$ elementary components, that is: $f = \sum_{p=0}^{n} f_{[p]}$.

We define the sphere of radius k around a solution x as the set of all solutions lying at Hamming distance k from x [19]. In analogy to the adjacency matrix we define the sphere matrices of radius k as $S_{xy}^{(k)} = 1$ if $\mathcal{H}(x,y) = k$ and $S_{xy}^{(k)} = 0$ otherwise.

The sphere matrix of radius one is the adjacency matrix of the one-change neighborhood, A, and the sphere matrix of radius zero is the identity matrix, I. Each sphere matrix $S^{(k)}$ can be written as a polynomial in A (the adjacency matrix) [5]. Then, each eigenvector of A is an eigenvector of $S^{(k)}$, with a different eigenvalue. As a consequence, the eigenvectors of the Laplacian matrix Δ are eigenvectors of the sphere matrices $S^{(k)}$. Furthermore, an order-p function (having eigenvalue $2p$ for $-\Delta$) is eigenvector of the sphere matrix $S^{(k)}$ with eigenvalue $\mathcal{K}_{k,p}^{(n)}$, which is the (k,p) element of the n-th Krawtchouk matrix (see [19] for details). Krawtchouk matrices can be defined with the equation:

$$\mathcal{K}_{k,p}^{(n)} = \sum_{l=0}^{n} (-1)^l \binom{n-p}{k-l} \binom{p}{l}, \tag{5}$$

where $n \geq 0$, $0 \leq k, p \leq n$ and we consider that $\binom{a}{b} = 0$ when $b < 0$ or $b > a$.

The interested reader can deepen on Krawtchouk matrices in [7], here we only highlight their properties relevant to our mathematical derivations. One important property of the Krawtchouk matrices is:

$$(1+x)^{n-p}(1-x)^p = \sum_{k=0}^{n} x^k \mathcal{K}_{k,p}^{(n)}. \tag{6}$$

Proposition 1. *The following identity for the Krawtchouk matrices holds:*

$$\sum_{k=0}^{n} k\mathcal{K}_{k,p}^{(n)} = \begin{cases} n2^{n-1} & \text{if } p = 0, \\ -2^{n-1} & \text{if } p = 1, \\ 0 & \text{if } p > 1. \end{cases} \tag{7}$$

Proof. Taking the derivative of the two sides of (6) we have:

$$(n-p)(1+x)^{n-p-1}(1-x)^p - p(1+x)^{n-p}(1-x)^{p-1} = \sum_{k=1}^{n} x^{k-1} k \mathcal{K}_{k,p}^{(n)}. \tag{8}$$

If we set $x = 1$, the right hand side is the left hand side of (7). In the left hand side of (8) we can distinguish three cases:

- Case $p = 0$: the derivative polynomial evaluated in $x = 1$ is $n2^{n-1}$.
- Case $p = 1$: the derivative polynomial evaluated in $x = 1$ is -2^{n-1}.
- Case $p > 1$: the derivative polynomial evaluated in $x = 1$ is 0.

□

Each component $f_{[p]}$ of the elementary landscape decomposition of f is an eigenvector of the sphere matrix of radius k with eigenvalue $\mathcal{K}_{k,p}^{(n)}$. Thus, we can compute the sum of the fitness value in a sphere of radius k around x as:

$$\sum_{\substack{y \in \mathbb{B}^n \\ \mathcal{H}(x,y)=k}} f(y) = \sum_{p=0}^{n} \mathcal{K}_{k,p}^{(n)} f_{[p]}(x) \tag{9}$$

3 Fitness-Distance Correlation

The Fitness-Distance Correlation (FDC) is a measure introduced by Jones and Forrest [10] to measure problem difficulty. Given all the solutions in the search space, it computes the correlation coefficient between the fitness values of these solutions and the Hamming distances of the solutions to their nearest global optimum.

Definition 2. *Given a function $f : \mathbb{B}^n \mapsto \mathbb{R}$ the fitness-distance correlation for f is defined as*

$$r = \frac{Cov_{fd}}{\sigma_f \sigma_d}, \tag{10}$$

where Cov_{fd} is the covariance of the fitness values and the distances of the solutions to their nearest global optimum, σ_f is the standard deviation of the fitness values in the search space and σ_d is the standard deviation of the distances to the nearest global optimum in the search space. Formally:

$$\begin{aligned} Cov_{fd} &= \frac{1}{2^n} \sum_{x \in \mathbb{B}^n} (f(x) - \overline{f})(d(x) - \overline{d}), \\ \overline{f} = \frac{1}{2^n} \sum_{x \in \mathbb{B}^n} f(x), \quad \sigma_f &= \sqrt{\frac{1}{2^n} \sum_{x \in \mathbb{B}^n} (f(x) - \overline{f})^2}, \\ \overline{d} = \frac{1}{2^n} \sum_{x \in \mathbb{B}^n} d(x), \quad \sigma_d &= \sqrt{\frac{1}{2^n} \sum_{x \in \mathbb{B}^n} (d(x) - \overline{d})^2}, \end{aligned} \tag{11}$$

where the function $d(x)$ is the Hamming distance between x and its nearest global optimum.

The FDC r is a value between -1 and 1. The lower the absolute value of r, the more difficult the optimization problem is supposed to be. The exact computation of the FDC using the previous definition requires the evaluation of the complete search space. It is required to determine the global optima to define $d(x)$ and compute the statistics for d and f. If the objective function f is a constant function, then the FDC is not well-defined, since $\sigma_f = 0$.

In the following we will focus on the case in which there exists one only global optimum x^* and we know the elementary landscape decomposition of f. The following lemma provides an expression for $\overline{d}$ and σ_d in this case.

Lemma 1. *Given an optimization problem defined over $\mathbb{B}^n$, if there is only one global optimum x^*, then the distance function $d(x)$ defined in Definition 2 is the Hamming distance between x and x^* and its average and standard deviation in the whole search space are given by*

$$\overline{d} = \frac{n}{2}, \quad \sigma_d = \frac{\sqrt{n}}{2}. \tag{12}$$

Proof. Since there is only one global optimum, the function $d(x)$ is defined as $d(x) = \mathcal{H}(x, x^*)$. Given an integer number $0 \leq k \leq n$, the number of solutions at distance k from x^* is $\binom{n}{k}$. Then we can compute the two first raw moments of $d(x)$ over the search space as:

$$\alpha_1 = \overline{d} = \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} k = \frac{n2^{n-1}}{2^n} = \frac{n}{2},$$

$$\alpha_2 = \overline{d^2} = \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} k^2 = \frac{n(n+1)2^{n-2}}{2^n} = \frac{n(n+1)}{4}.$$

Using these moments we can compute the standard deviation as $\sqrt{\alpha_2 - \alpha_1^2}$, which yields:

$$\sigma_d = \sqrt{\frac{n(n+1)}{4} - \frac{n^2}{4}} = \sqrt{\frac{n}{4}} = \frac{\sqrt{n}}{2}. \tag{13}$$

□

Now we are ready to prove the main result of this work.

Theorem 1. *Let f be an objective function whose elementary landscape decomposition is $f = \sum_{p=0}^{n} f_{[p]}$, where $f_{[0]}$ is the constant function $f_{[0]}(x) = \overline{f}$ and each $f_{[p]}$ with $p > 0$ is an order-p elementary function with zero offset. If there exists only one global optimum in the search space x^*, the FDC can be exactly computed as:*

$$r = \frac{-f_{[1]}(x^*)}{\sigma_f \sqrt{n}}. \tag{14}$$

Proof. Let us expand the covariance as

$$\begin{aligned} Cov_{fd} &= \frac{1}{2^n} \sum_{x \in \mathbb{B}^n} f(x)d(x) - \overline{f}\,\overline{d} = \frac{1}{2^n} \sum_{k=0}^{n} k \sum_{\substack{x \in \mathbb{B}^n \\ \mathcal{H}(x,x^*)=k}} f(x) - \overline{f}\frac{n}{2} \\ &= \frac{1}{2^n} \sum_{k=0}^{n} k \sum_{\substack{x \in \mathbb{B}^n \\ \mathcal{H}(x,x^*)=k}} \sum_{p=0}^{n} f_{[p]}(x) - f_{[0]}\frac{n}{2} = \frac{1}{2^n} \sum_{k=0}^{n} k \sum_{p=0}^{n} \mathcal{K}_{k,p}^{(n)} f_{[p]}(x^*) - f_{[0]}\frac{n}{2} \\ &= \sum_{p=0}^{n} \left(\frac{1}{2^n} \sum_{k=0}^{n} k\mathcal{K}_{k,p}^{(n)} \right) f_{[p]}(x^*) - f_{[0]}\frac{n}{2} = \frac{n}{2} f_{[0]} - \frac{1}{2} f_{[1]}(x^*) - f_{[0]}\frac{n}{2} \\ &= -\frac{1}{2} f_{[1]}(x^*), \end{aligned} \tag{15}$$

where we used the result in Proposition 1. Substituting in (10) we obtain (14). □

The previous theorem shows that the only thing we need to know on the global optimum is the value of the first elementary component. With this information

we can exactly compute the FDC. Some problems for which we know the elementary landscape decomposition based on the numerial data defining a problem instance are MAX-SAT, 0-1 Unconstrained Quadratic Optimization (UQO), the Subset Sum problem (SS), the NK-landscapes, etc. For all of them we could provide expressions for their FDC.

The result of the previous theorem starts an interesting discussion. Some works on landscape analysis claim that the ruggedness of a landscape is related to its hardness [2]. The *autocorrelation coefficient* ξ and the *autocorrelation length* ℓ of a problem are two measures of the ruggedness of the problem proposed to characterize an objective function in a way that allows one to estimate the performance of a local search method: the lower their value the higher their ruggedness. Angel and Zissimopoulos [1] have studied the relationship between the performance of a local search and the autocorrelation coefficient. Also a relationship has been noticed between the autocorrelation length and the expected number of local optima of a problem [8]. Furthermore, the *autocorrelation length conjecture* [17] claims that the higher the value of ξ and ℓ, the lower the number of local optima and, as a consequence, the better could be the performance of a local search method. In summary, empirical and theoretical results support the hypothesis that a rugged landscape is more difficult than a problem with a smooth landscape.

In the case of the elementary functions defined over binary strings, the functions with higher order are more rugged than the ones with lower order. The order-1 elementary landscapes are the smoothest landscapes and, in fact, they can always be solved in polynomial time. Following this chain of reasoning, in a general landscape, the elementary components with order $p > 1$ are the ones that make the problem difficult. However, from Theorem 1 we observe that only the order-1 elementary component of a function f is taken into account in the computation of the FDC. This fact poses some doubts on the value of the FDC as a measure of difficulty of a problem, since FDC is shown to neglect the rest of information captured in the higher order components. This is true under the assumption that one single global optimum exists in the search space. We defer to future work the analysis of the general case. The doubts on FDC as being a difficulty indicator have also been raised by other authors. Two examples are the work by Tomassini et al. [21] focused on genetic programming and the one by Bierwirth et al. [3] based on the Job Shop Scheduling.

3.1 Fitness-Distance Correlation for Elementary Landscapes

If the objective function is elementary, then the expression of the exact FDC is specially simple, as the following corollary proves.

Corollary 1. *Let f be an elementary function of order $p > 0$ with one only global optimum x^*, then the fitness-distance correlation can be exactly computed using the following expression:*

$$r = \begin{cases} \frac{\overline{f} - f(x^*)}{\sigma_f \sqrt{n}} & \text{if } p = 1 \\ 0 & \text{if } p > 1 \end{cases} \tag{16}$$

Proof. An elementary function $f(x)$ of order $p > 0$ can always be written as the sum of the two eigenvectors of the adjacency matrix: $f(x) = f_{[0]} + f_{[p]}(x)$ where $f_{[0]} = \overline{f}$. Applying the result of Theorem 1 we obtain (16). □

The previous corollary states that only elementary landscapes with order $p = 1$ have a nonzero FDC. Furthermore, the FDC does depend on the value of the objective function in the global optimum $f(x^*)$ and the average value $\overline{f}$, but not on the solution x^* itself. We can also observe that if we are maximizing, then $f(x^*) > \overline{f}$ and the FDC is negative, while if we are minimizing $f(x^*) < \overline{f}$ and the FDC is positive.

Interestingly, the order-1 elementary landscapes can always be written as linear functions and they can be optimized in polynomial time. That is, if f is an order-1 elementary function then it can be written in the following way:

$$f(x) = \sum_{i=1}^{n} a_i x_i + b. \tag{17}$$

where a_i and b are real values. The following proposition provides the average and the standard deviation for this family of functions.

Proposition 2. *Let f be an order-1 elementary function, which can be written as (17). Then, the average and the standard deviation of the function values in the whole search space are:*

$$\overline{f} = b + \frac{1}{2}\sum_{i=1}^{n} a_i, \quad \sigma_f = \frac{1}{2}\sqrt{\sum_{i=1}^{n} a_i^2}. \tag{18}$$

Proof. Using the linearity property of the average we can write: $\overline{f} = \sum_{i=1}^{n} a_i \overline{x_i} + b$, and $\overline{f}$ in (18) follows from the fact that $\overline{x_i} = 1/2$. Now we can compute the variance of f as:

$$\begin{aligned} Var[f] &= \overline{(f(x) - \overline{f})^2} = \overline{\left(\sum_{i=1}^{n} a_i x_i - \frac{1}{2}\sum_{i=1}^{n} a_i\right)^2} = \overline{\left(\sum_{i=1}^{n} a_i \left(x_i - \frac{1}{2}\right)\right)^2} \\ &= \sum_{i,j=1}^{n} a_i a_j \overline{\left(x_i - \frac{1}{2}\right)\left(x_j - \frac{1}{2}\right)} = \sum_{i,j=1}^{n} a_i a_j \left(\overline{x_i x_j} - \frac{1}{2}\overline{x_i} - \frac{1}{2}\overline{x_j} + \frac{1}{4}\right) \\ &= \sum_{i,j=1}^{n} a_i a_j \left(\overline{x_i x_j} - \frac{1}{4}\right) = \sum_{i,j=1}^{n} a_i a_j \left(\delta_i^j \frac{1}{4} + \frac{1}{4} - \frac{1}{4}\right) = \frac{1}{4}\sum_{i=1}^{n} a_i^2, \end{aligned} \tag{19}$$

where we used again $\overline{x_i} = \overline{x_j} = 1/2$ and $\overline{x_i x_j} = 1/4(\delta_i^j + 1)$, being δ_i^j the Kronecker delta. The expression for σ_f in (18) follows from (19). □

Using Proposition 2 we can compute the FDC for the order-1 elementary landscapes.

Proposition 3. *Let f be an order-1 elementary function written as (17) such that all $a_i \neq 0$. Then, it has one only global optimum and its FDC (assuming maximization) is:*

$$r = \frac{-\sum_{i=1}^{n} |a_i|}{\sqrt{n \sum_{i=1}^{n} a_i^2}}, \tag{20}$$

which is always in the interval $-1 \leq r < 0$.

Proof. The global optimum x^* has 1 in all the positions i such that $a_i > 0$ and the maximum fitness value is:

$$f(x^*) = b + \sum_{\substack{i=1 \\ a_i > 0}}^{n} a_i. \tag{21}$$

Using Proposition 2 we can write:

$$\overline{f} - f(x^*) = \left(b + \frac{1}{2} \sum_{i=1}^{n} a_i \right) - \left(b + \sum_{\substack{i=1 \\ a_i > 0}}^{n} a_i \right) = -\frac{1}{2} \sum_{i=1}^{n} |a_i|. \tag{22}$$

Replacing the previous expression and σ_f in (16) we prove the claimed result. □

When all the values of a_i are the same, the FDC computed with (20) is -1. This happens in particular for the Onemax problem. But if there exist different values for a_i, then we can reach any arbitrary value in $[-1, 0)$ for r. The following theorem provides a way to do it.

Theorem 2. *Let ρ be an arbitrary real value in the interval $[-1, 0)$, then any linear function $f(x)$ given by (17) where $n > 1/\rho^2$, $a_2 = a_3 = \ldots = a_n = 1$ and a_1 is*

$$a_1 = \frac{(n-1) + n|\rho| \sqrt{(1-\rho^2)(n-1)}}{n\rho^2 - 1} \tag{23}$$

has exactly FDC $r = \rho$.

Proof. The expression for a_1 is well-defined since $n\rho^2 > 1$. Replacing all the a_i in (20) we get $r = \rho$. □

Theorem 2 provides a solid argument against the use of FDC as a measure of the difficulty of a problem. In effect, we can always build an optimization problem based on a linear function, which can be solved in polynomial time, with an FDC as near as desired to 0 (but not zero), that is, as "difficult" as desired according to the FDC. However, we have to highlight here that for a given FDC value ρ we need at least $n > 1/\rho^2$ variables. Thus, an FDC nearer to 0 requires more variables.

4 FDC, Autocorrelation Length and Local Optima

The autocorrelation length ℓ [8] has also been used as a measure of the difficulty of a problem. Chicano and Alba [4] found a negative correlation between ℓ and the number of local optima in the 0-1 Unconstrained Quadratic Optimization problem (0-1 UQO), an NP-hard problem [9]. Kinnear [11] also studied the use of the autocorrelation measures as problem difficulty, but the results were inconclusive. In this section we investigate which of the two measures, ℓ or the absolute value of FDC, is more correlated to the number of local optima for some random instances of the 0-1 UQO. In particular, we have randomly generated 1650 UQO instances using the Palubeckis instance generator [13]. The size of the instances varies between $n = 10$ and $n = 20$ and the density (percentage of nonzero elements in the coefficients matrix) varies from 10 to 90 in steps of 20. For each n and density, 30 random instances were generated by randomly selecting the nonzero elements of the coefficients matrix from the interval $[-100, 100]$. For all the instances we computed the autocorrelation length ℓ, the absolute value of the FDC $|r|$ and the number of local optima (minima) by complete enumeration of the search space. In Table 1 we show the Spearman rank correlation coefficient between the number of local optima and ℓ and $|r|$. The correlations are computed using all the instances with the same size n.

Table 1. Spearman correlation coefficient for the number of local optima against the autocorrelation length (ℓ) and the absolute value of the FDC ($|r|$).

n	10	11	12	13	14	15
ℓ	−0.5467	−0.5545	−0.5896	−0.4796	−0.4725	−0.5511
$\|r\|$	−0.1407	−0.1843	−0.0787	−0.1203	−0.1944	−0.0538
n	16	17	18	19	20	
ℓ	−0.4959	−0.5740	−0.5872	−0.5249	−0.4829	
$\|r\|$	−0.1251	−0.1791	−0.1339	−0.3310	−0.0338	

We can observe a high inverse correlation (around −0.5) between the number of local optima and the autocorrelation length, supporting the autocorrelation length conjecture. However, the correlation between the number of local optima and FDC is low, again supporting the hypothesis that FDC is not an appropriate measure of the difficulty of a problem (this time, from an experimental point of view).

5 Conclusion

We have applied landscape theory to exactly compute the Fitness-Distance Correlation of combinatorial optimization problems defined over sets of binary strings. The result is valid in the case in which one single global optimum exists in the landscape. We defer to future work the analysis of the general case.

The expression for the FDC takes only into account the order-1 elementary component of the objective function, while previous work suggests that the components making a problem difficult are the higher order elementary components. This fact questions the use of FDC as a measure of difficulty of the problem. We prove that there exist polynomial time solvable problems with an FDC arbitrarily near to zero. An experimental study over random instances of the 0-1 UQO shows a low correlation between FDC and the number of local optima, supporting the hypothesis that FDC fails to capture the problem difficulty.

References

1. Angel, E., Zissimopoulos, V.: On the landscape ruggedness of the quadratic assignment problem. Theoretical Computer Science 263, 159–172 (2000)
2. Barnes, J.W., Dimova, B., Dokov, S.P.: The theory of elementary landscapes. Applied Mathematics Letters 16, 337–343 (2003)
3. Bierwirth, C., Mattfeld, D., Watson, J.P.: Landscape Regularity and Random Walks for the Job-Shop Scheduling Problem. In: Gottlieb, J., Raidl, G.R. (eds.) EvoCOP 2004. LNCS, vol. 3004, pp. 21–30. Springer, Heidelberg (2004)
4. Chicano, F., Alba, E.: Elementary landscape decomposition of the 0-1 unconstrained quadratic optimization. Journal of Heuristics (10.1007/s10732-011-9170-6)
5. Chicano, F., Alba, E.: Exact computation of the expectation curves of the bit-flip mutation using landscapes theory. In: GECCO, pp. 2027–2034 (2011)
6. Chicano, F., Whitley, L.D., Alba, E.: A methodology to find the elementary landscape decomposition of combinatorial optimization problems. Evolutionary Computation 19(4), 597–637 (2011)
7. Feinsilver, P., Kocik, J.: Krawtchouk polynomials and krawtchouk matrices. In: Recent Advances in Applied Probability, pp. 115–141. Springer, US (2005)
8. García-Pelayo, R., Stadler, P.: Correlation length, isotropy and meta-stable states. Physica D: Nonlinear Phenomena 107(2-4), 240–254 (1997)
9. Glover, F., Alidaee, B., Rego, C., Kochenberger, G.: One-pass heuristics for large-scale unconstrained binary quadratic problems. EJOR 137(2), 272–287 (2002)
10. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: GECCO, pp. 184–192. Morgan Kaufmann (1995)
11. Kinnear Jr., K.E.: Fitness landscapes and difficulty in genetic programming. In: IEEE CEC, vol. 1, pp. 142–147 (June 1994)
12. Lu, G., Bahsoon, R., Yao, X.: Applying elementary landscape analysis to search-based software engineering. In: Proceedings of SSBSE (2010)
13. Palubeckis, G.: Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. Annals of Oper. Research 131, 259–282 (2004)
14. Rana, S., Heckendorn, R.B., Whitley, D.: A tractable walsh analysis of SAT and its implications for genetic algorithms. In: Proceedings of AAAI, pp. 392–397 (1998)
15. Reidys, C.M., Stadler, P.F.: Combinatorial landscapes. SIAM Review 44(1), 3–54 (2002)
16. Stadler, P.F.: Toward a theory of landscapes. In: López-Peña, R., Capovilla, R., García-Pelayo, R., Waelbroeck, H., Zertruche, F. (eds.) Complex Systems and Binary Networks, pp. 77–163. Springer (1995)
17. Stadler, P.F.: Fitness Landscapes. In: Biological Evolution and Statistical Physics, pp. 183–204. Springer (2002)

18. Sutton, A.M., Whitley, D., Howe, A.E.: Mutation rates of the (1+1)-EA on pseudo-boolean functions of bounded epistasis. In: GECCO, pp. 973–980. ACM (2011)
19. Sutton, A.M., Whitley, L.D., Howe, A.E.: Computing the moments of k-bounded pseudo-boolean functions over hamming spheres of arbitrary radius in polynomial time. Theoretical Computer Science (10.1016/j.tcs.2011.02.006)
20. Sutton, A.M., Whitley, L.D., Howe, A.E.: A polynomial time computation of the exact correlation structure of k-satisfiability landscapes. In: Proceedings of GECCO, pp. 365–372. ACM, New York (2009)
21. Tomassini, M., Vanneschi, L., Collard, P., Clergue, M.: A study of fitness distance correlation as a difficulty measure in genetic programming. Evolutionary Computation 13(2), 213–239 (2005)
22. Whitley, D., Sutton, A.M., Howe, A.E.: Understanding elementary landscapes. In: Proceedings of GECCO, pp. 585–592. ACM, New York (2008)

Genetic Algorithms for Scheduling Devices Operation in a Water Distribution System in Response to Contamination Events

Marco Gavanelli, Maddalena Nonato, Andrea Peano,
Stefano Alvisi, and Marco Franchini

EnDiF, Università degli Studi di Ferrara
via G. Saragat 1 – 44122, Ferrara, Italy
{marco.gavanelli,maddalena.nonato,andrea.peano,
stefano.alvisi,marco.franchini}@unife.it

Abstract. This paper heuristically tackles a challenging scheduling problem arising in the field of hydraulic distribution systems in case of a contamination event, that is, optimizing the scheduling of a set of tasks so that the consumed volume of contaminated water is minimized. Each task consists of manually activating a given device, located on the hydraulic network of the water distribution system. In practice, once contamination has been detected, a given number of response teams move along the network to operate each device on site. The consumed volume of contaminated water depends on the time at which each device is operated, according to complex hydraulic laws, so that the value associated to each schedule must be evaluated by a hydraulic simulation.

We explore the potentials of Genetic Algorithms as a viable tool for tackling this optimization-simulation problem. We compare different encodings and propose ad hoc crossover operators that exploit the combinatorial structure of the feasible region, featuring hybridization with Mixed Integer Linear Programming.

Computational results are provided for a real life hydraulic network of average size, showing the effectiveness of the approach. Indeed, we greatly improve upon common sense inspired solutions which are commonly adopted in practice.

Keywords: Hybrid Genetic Algorithms, Simulation-Optimization, Scheduling.

1 Problem Description

The geo-political scenario arising from 9/11 has spurred research concerning infrastructures protection policies and recovery procedures from intentionally induced service disruptions, e.g., because of a terrorist attack. Water distribution systems are among the most vulnerable infrastructures, due to the distributed physical layout of their networks, and to how critical is the commodity they supply: drinking water. People rely on water quality for a number of crucial activities, such as cooking, washing and bathing in private households, while clear water is essential in operating restaurants, hospitals, and some manufacturing. Adding deadly contaminant into a hydraulic network can rapidly cause huge damage in terms of human losses, since contaminant quickly

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 124–135, 2012.

spreads through the network and is consumed by the users. In this framework, monitoring and promptly recovering is more viable than securing the whole water distribution system, which often has a vast planimetric extent, e.g., a small city network may reach $200km$ and a thousand of pipes and nodes. The common policy followed in the deployment of contamination warning systems consists of installing several sensors along the network, strategically located according to optimization procedures [10], and periodically checking water quality. As soon as a sensor has detected a contaminant, an ad hoc recovery procedure is started, in order to mitigate the impact on the population. Besides population alerting, two kinds of operations can be performed on network devices: opening *hydrants* in order to expel contaminated water, and isolating pipes by closing their *isolation valves* in order to prevent contaminated water to flow toward densely populated areas. The objective is to minimize the impact on the population, usually measured as the volume of contaminated water consumed by the users during a given period after contamination. This value, which heavily depends on devices activation times, can neither be computed nor approximated by simple analytical calculus, whereas it can be simulated. A simulator such as EPANET [11] takes as input the network configuration, the open/closed status of devices, and the time at which each device is operated, and returns the volume of consumed contaminated water. For real world networks, each simulation may take various seconds of computing time on a modern computer, ($5''$ for a network of about 800 nodes and 1100 main pipes) so that the total number of simulations cannot exceed some threshold to be practically usable, even in an off line procedure such as ours. In most networks, devices can only be operated manually, so teams of workers are dispatched on the network to open hydrants and close valves on site. This introduces significant delays and forbids to operate a large number of devices. The hydraulic engineering literature provides several approaches to select the most suitable subset of devices, given the location of the first alerted sensor: both [1] and [8] propose a multi-objective approach minimizing the number n of operated devices as well as the impact on the population. However, the next major decision concerning the actual schedule of devices activations has never been fully addressed. Indeed, [1] supposes to activate all the selected devices instantaneously and simultaneously, while [8] builds a schedule heuristically according to common sense criteria. However, there is no assurance that this approach gives a (near) optimum scheduling, i.e., a scheduling that minimizes the volume of consumed contaminated water.

This problem has some similarities with the multiple Traveling Salesman Problem ($mTSP$), where m salesmen visit the nodes of a graph minimizing total traveled distance. However, while the $mTSP$ objective function is easily computed, being the sum of traveled distances, ours requires an expensive simulation. Moreover, while $mTSP$'s good quality solutions tend to visit the nodes as soon as possible, in our problem, the early closure of a valve may divert contaminant towards high consumption/demand areas, so that a delay in the schedule sometimes improves the objective function value.

In this work, that extends [2], we propose a genetic algorithm that addresses explicitly the problem of assigning devices to teams (for a given number m of teams) and scheduling the teams operations, in order to minimize the volume of contaminated water consumed by the users. Let us call this problem Response to Contamination Problem (RCP). The genetic algorithm is coupled with a hydraulic simulator, that computes the

objective function. We implemented three different chromosome representations and corresponding genetic operators. One representation is original and it is built on the mathematical models developed for the *mTSP* and for vehicle routing problems (VRP) [12], while the other two come from the literature on the *mTSP*, namely the Two Chromosome and the Two Part representations [5]. The latter has been extended to insert pauses in the schedules while the new one encompasses pauses naturally. We experimentally compare all these representations on the real instance of a medium sized city.

2 Genetic Algorithms for the Scheduling of Operations

Defining a Genetic Algorithm (GA) basically amounts to define the structure of chromosomes, the selection operator, the recombination operators (crossover and mutation), besides fitness measures and termination conditions. In RCP, the evaluation of an individual fitness requires a long hydraulic simulation, so the main obstacle to obtaining good solutions is limited computing time. Therefore, our termination condition is a fixed number of invocations to the hydraulic simulator. Since each call is expensive, we store the input/output data of each call in a sort of caching mechanism with respect to a unique coding of the solution, the activation times vector $t^{\mathcal{F}}$. If the objective function has been invoked before with the same arguments, its value is not re-computed but retrieved from the cache. Thus, the number of invocations is not proportional to the number of generations. Other features common to all the *GA* families further introduced are: a classical *roulette wheel* procedure for parent selection, an *elitist generational replacement scheme*, mutation of clones, and random generation of the initial population.

2.1 A Genetic Algorithm Based on Sequences

As mentioned, RCP shares the feasibility structure of an *mTSP* defined on a graph where the mobilization point corresponds to the depot d and each client node to one of the n devices to operate. Then we can borrow from the encodings used for the *mTSP*. One of the first TSP encodings representing the sequence of the visited nodes in a vector extends to the case of m teams by adding a second row, the team identifiers.

3	4	1	2	8	5	7	6
1	2	1	3	2	3	2	2

(1)

In the chromosome shown in (1), team number 1 visits nodes 3 and 1 (in this order); team 2 visits 4, 8, 7, and 6, while team 3 visits 2 and 5. This representation is named *two chromosome technique*, we call the related *GA* 2*C*, and the size of its solution space is $n!m^n$ [5]. This encoding, as all those based on permutations, is affected by redundancy which slows down *GA*'s convergence. In fact, the first row of the 2*C* encoding describes a total order on the nodes but it gets decoded into a partial order, which is total only within each route. So, any total order complying with this partial order yields the same activation sequence. So far with the cons. Regarding the pros, this encoding supports simple crossover operators, thanks to the representation into a linear data structure. For example, one can use the *one-point ordered crossover* [7]. Given two parents, f and m, for each integer i in the interval $[1,n]$ two offsprings are generated as follows: the first

child inherits the first i columns from f and fills the other columns with the remaining elements taken from m in that order. In the example depicted in (2), i is equal to 4 so the first 4 columns of the child are inherited from f, while the remaining devices, namely 7, 3, 8, and 5, are taken from m in such order, together with the team information.

$$f = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 6&4&1&2&8&5&7&3\\\hline 1&2&1&3&2&1&2&2\\\hline\end{array} \qquad m = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 2&7&3&8&4&6&1&5\\\hline 2&1&3&3&1&2&1&2\\\hline\end{array} \quad \Rightarrow \quad \underbrace{\begin{array}{|c|c|c|c|}\hline 6&4&1&2\\\hline 1&2&1&3\\\hline\end{array}}_{f} \underbrace{\begin{array}{|c|c|c|c|}\hline 7&3&8&5\\\hline 1&3&3&2\\\hline\end{array}}_{m} \tag{2}$$

The idea behind this operator is the following. The aim of a good crossover operator is having offspring inherit those features that made its ancestors successful. We have no information about what influences the value of our objective function, lacking a simple analytic formulation: we can only make reasonable assumptions. A possible assumption is that the sequence of activations could influence such value. So, if a sequence is successful, keeping parts of this sequence could make the offspring successful as well. Note that, using a single point crossover, the offspring always inherits the first i elements from one of its parents. This is done on purpose, since devices operated as first strongly influence contaminant spreading, and the first i elements of the sequence are likely to determine which devices are operated first, at least for one team. Figure 1 shows the tree representation of the offspring in (2): in the child tree, the rooted subtree in bold, T_d, comes from f, while the routes of m, after the shrink due to the deletion of the already selected nodes, are randomly appended to T_d according to the team naming adopted in m. Symmetrically, the second child is generated by inheriting the first i columns from m while the remaining devices are activated in the order and by the teams as in f. Each solution (each tree) is associated with an equivalence class of individuals, each with a different chromosome representation, and this representation impacts on the crossover results. In order to reduce this impact, before crossover we shuffle the columns of each parent while preserving the partial order. In other words, we randomly pick another representative for the same tree in the equivalence class. A further level of redundancy comes from team names; by renaming teams we get different representations of the same solution. To deal with this symmetry, that may generate very different offspring from very similar parents, we adopt a standard team naming approach: the team operating device 1 takes name 1; the team that operates the device with smallest identifier amongst the remaining devices takes name 2, and so on.

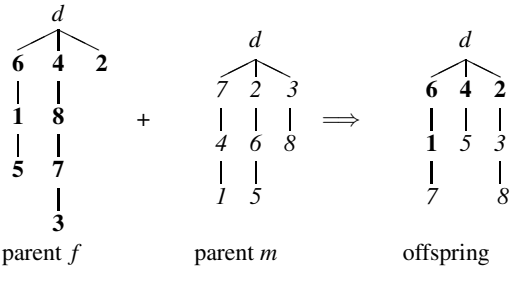
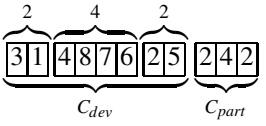


Fig. 1. An example of the tree representation of a crossover

2.2 Two-Part Chromosome

In [5] a so called two-part chromosome is proposed, with lower redundancy with respect to the other encodings so far proposed in the literature. The permutation part of the chromosome, C_{dev}, made of n integers as usual, is followed by a second part, C_{part}, being a string of m integers summing up to n. Its k^{th} value tells how many elements are part of the k^{th} tour. For example, the same solution depicted in (1) would become (3).



(3)

This way, the size of the representation space is lowered to the order of $n!\binom{n-1}{m-1}$. As in [5], we adopt the above mentioned one-point ordered crossover for the first chromosome part C_{dev}, and a single point asexual crossover (a random rotation) for the second one C_{part}. Both are closed with respect to this encoding and yield feasible solutions.

As already mentioned, in RCP introducing *delays* in the schedule may improve the objective function value. To this purpose, a vector C_{pause} is added to the chromosome assigning a pause to each device, ranging from 0 to an upper bound U. This can be equivalently thought of as the teams moving at *Variable Speed*. For this reason the related *GA* is named $2P^{VS}$, as opposed to the constant speed version called $2P^{CS}$. In case of variable speed, also the third part C_{pause} is handled by one-point ordered crossover.

While this encoding has a lower redundancy if compared to traditional permutation based encodings, redundancy can not be completely avoided. Indeed, redundancy is inherent into this kind of representation, since the encoding distinguishes among salesmen in the representation space, while they are all identical in the solution space.

In $2C^{CS}$ (i.e., the basic $2C$), $2P^{CS}$ and $2P^{VS}$ *GA*s, we adopt the same mutation operator, i.e., swapping two columns of the chromosome, applied randomly with given probability. Such probability has a base value of 2%, it is increased of 1% in case of no improvement for 3 consecutive generations, and reset to the base value in case of improvement.

2.3 A Genetic Algorithm Based on Activation Times

The previous encodings support schedule feasibility since they encode *mTSP* solutions, and any such solution identifies a feasible schedule. However, they do not allow to directly propagate the activation time of a device, that is, the basic piece of information in our problem, which can not be transmitted unless the whole sequence is inherited. A straightforward encoding, which emphasizes the scheduling information, encodes activation times directly in the chromosome, with gene i^{th} modeling activation time of device i. Such encoding, being the direct representation of the solution, is redundancy free. The absence of redundancy, however, goes to the detriment of feasibility, which is no longer guaranteed and must be explicitly restored after crossover and mutation. Indeed, a generic vector of activation times does not carry along with it any knowledge of the tours followed in the graph, nor the number of teams, therefore there is no straightforward crossover operator which can preserve feasibility since the encoding itself lacks

the necessary information. Consider for example the well known uniform crossover operator (UX), which selects genes from the two parents based on a randomly generated binary mask. A time-based *GA* based on UX may yield vectors spanning the whole space R^n (the most obvious relaxation of the feasible region) but the returned solution may not only be infeasible but also quite different from the closest feasible one. Thus, restoring feasibility after the application of each genetic operator ensures that each individual during the search represents a feasible scheduling, and only feasible schedulings are allowed to reproduce (Algorithm 1). In the following, we introduce a Mixed Integer Linear Programming (MILP) model mapping any vector of activation times to its closest feasible point. It will be used to restore feasibility at every step after the UX crossover, and this approach will be denoted as *UX with a posteriori feasibility restore* (*UXPF*). Furthermore, we extend this idea and integrate the MILP model directly within the genetic operator, giving raise to a second approach denoted as *MILPX*.

Algorithm 1. A genetic algorithm that restores feasibility through a MILP solver

```
Pop ← generate initial population;
while not(termination condition) do
    for N_pop/2 times do
        Select a pair (f,m) of population individuals;
        Temp_1 ← Crossover(f,m);
        Child_1 ← call_MILP_solver(Temp_1);
        Temp_2 ← Crossover(m,f);
        Child_2 ← call_MILP_solver(Temp_2);
        randomly apply mutation to individual I;
        I ← call_MILP_solver(I);
    Pop ← {Child_1,...,Child_Npop};
return best;
```

An Integer Programming Model to Restore Feasibility. Let t be a generic vector of activation times. If t is not feasible, i.e., it cannot be obtained by any scheduling of the teams, we propose to repair it by turning it into the feasible point $t^{\mathcal{F}}$ closest to t by norm L_1. As an example, consider a small network with 4 devices plus the mobilization point d, 2 teams and the following traveling time matrix τ:

$$\tau = \begin{array}{c} \\ d \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} d & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} - & 1 & 1 & 1 & 1 \\ 1 & - & 1 & 3 & 1 \\ 1 & 1 & - & 4 & 7 \\ 1 & 3 & 4 & - & 3 \\ 1 & 1 & 7 & 3 & - \end{pmatrix} \end{array}$$

Vectors $m = [1,1,4,8]$ and $f = [2,5,1,1]$ model feasible schedules but the UX operator, by using the binary mask [1,1,0,0], yields the infeasible child $t = [1,1,1,1]$; the restoring procedure returns $t^{\mathcal{F}} = [2,1,1,3]$ as the closest feasible vector, which is indeed at 3 units distance from t by L_1.

Several MILP models can be adopted to find $t^{\mathscr{F}}$, building on those developed for the *mTSP* [4] and routing problems in general, among which the following *2-index flow-based* formulation [12]. The constraints extend the *mTSP* model with traveling times information, the objective function minimizes the distance from t. The unknowns are:

- X a matrix $(n+1)\times(n+1)$ of 0-1 variables. $x_{ij}=1$ iff j is activated right after i by the same team; i is activated first by its team iff $x_{di}=1$; $x_{ii}=0 \forall i$ (no self loop arcs).
- $t^{\mathscr{F}}$ a vector of $n+1$ activation times; $t_i^{\mathscr{F}}$ is the time at which device i is activated, and $t_d^{\mathscr{F}}$ is the departure time from the depot d.
- δ a vector of n differences: it is defined as $\delta_i = t_i - t_i^{\mathscr{F}}$.

The input parameters are:

- t a vector of n ideal activation times.
- τ a matrix $(n+1)\times(n+1)$; τ_{ij} represents the time that a team takes to move at a given constant speed from the location of device i to that of device j.

The constraints:

$$\forall i \in \{1..n\} \qquad t_i^{\mathscr{F}} \geq \tau_{di} \tag{4}$$

$$\forall i \in \{1..n\} \qquad \delta_i = t_i - t_i^{\mathscr{F}} \tag{5}$$

$$t_d^{\mathscr{F}} = 0 \tag{6}$$

$$\sum_{i\in\{1..n\}} x_{di} = m \tag{7}$$

$$\forall i \in \{1..n\} \qquad \sum_{j\in\{1..n\}\cup d} x_{ij} = \sum_{h\in\{1..n\}\cup d} x_{hi} \tag{8}$$

$$\forall i \in \{1..n\} \qquad \sum_{j\in\{1..n\}\cup d} x_{ij} = 1 \tag{9}$$

$$\forall i \in \{1..n\} \qquad t_i^{\mathscr{F}} \leq M + x_{di}(\tau_{di} + U - M) \tag{10}$$

$$\forall i,j \in \{1..n\} \qquad t_i^{\mathscr{F}} + \tau_{ij} \leq t_j^{\mathscr{F}} + (1-x_{ij})M + x_{ji}(\tau_{ij} + \tau_{ji} + U - M) \tag{11}$$

Constraint (4) says that device i can be activated no earlier than the time it takes to reach it from d. Eq. (5) is the definition of δ. Teams leave the depot at time 0 (6). All teams depart from the depot (Eq. (7)). For each node i, the total number of teams arriving to i is equal to the number of teams leaving i, (8), the so called flow balance constraints. All nodes except d are visited exactly once (9). Constraint (10) is the linearization of the implication $x_{di}=1 \implies t_i^{\mathscr{F}} \leq \tau_{di} + U$ (where M is a sufficiently large positive number and U the upper bound for the potential pause before each activation), so that, together with (4), it imposes that the starting time of the first devices be equal to their traveling time from d plus a potential pause. Constraint (11) links the activation times $t^{\mathscr{F}}$ to the ordering between devices given by matrix X; indeed, (11) linearises the implications:

$$x_{ij}=1 \implies t_i^{\mathscr{F}} + \tau_{ij} \leq t_j^{\mathscr{F}} \qquad x_{ij}=1 \implies t_i^{\mathscr{F}} + \tau_{ij} + U \geq t_j^{\mathscr{F}}.$$

If $U=0$ then (11) imposes that the arrival time at device j equals the starting time from i plus the traveling time from i to j, thus implementing the constant speed variant of the time-based *GA*. Conversely, if $U>0$ the same constraint allows for a maximum pause of

U, thus implementing the variable speed variant of the algorithm. The objective function associated to problem (4-11) is the minimization of $||\delta||_1$, i.e. $min\left(\sum_{i\in\{1..n\}}|\delta_i|\right)$. To linearise this function, we introduce new unknowns δ^+ that represent the absolute value of δ, and minimize their sum.

Tighter Integration GA-MILP. Restoring feasibility after crossover may yield children quite different from their parents, since feasibility restoring could disrupt those patterns responsible for parents' fitness. For this reason, we moved the call to the MILP solver *inside* the crossover operator, giving raise to a new operator that we call *MILPX*. In this way, *MILPX* generates directly a new individual proven to be feasible and, at the same time, resembling the most to its parents among all their feasible children.

More precisely, given the chromosomes of the mating individuals $f \equiv (f_1,\ldots,f_n)$ and $m \equiv (m_1,\ldots,m_n)$, we generate the child c that minimizes the quantity

$$\sum_{i=1}^{n} min(|c_i - f_i|, |c_i - m_i|).$$

Stated otherwise, we can consider each chromosome as a point in a n-dimensional space. The two chromosomes f and m of the mating individuals define a hyper-parallelepiped that has m and f as two verteces, and with sides parallel to the coordinate axes (Figure 2). The MILP solver selects the feasible point in the n-space closest to any vertex of the hyper-parallelepiped. In this way, if there exists a feasible point in the n-space that inherits each coordinate from one of the two parents, it will be generated (or, if there exist more points with such feature, one of them is definitely generated as a spawn). Otherwise, the feasible point closest to one of such points is the spawned individual. This is implemented by slightly modifying the MILP model (4-11), by introducing a vector of unknowns W to range on the verteces of the hyper-parallelepiped; $w_i = 1$ iff the i-th coordinate of child c is inherited from f (i.e., $c_i = f_i$) and $w_i = 0$ otherwise (if $c_i = m_i$). The definition (5) of the displacement δ becomes (12)

$$\forall i \in \{1..n\} \qquad \delta_i = f_i w_i + m_i(1 - w_i) - t_i^{\mathscr{F}} \tag{12}$$

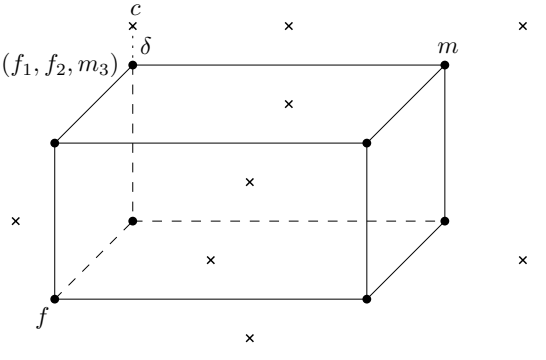


Fig. 2. Graphic representation of the crossover in a 3D space. Crosses represent feasible points, m and f are the mating individuals; c is the closest feasible point (at distance δ) to a vertex of the parallelepiped.

Summing up, for the *GA* based on activation times, we propose two crossovers, *UXPF* and *MILPX*, which can be used within the same algorithm, being invoked with different probability, yielding the so called *time-based Hybrid GAs*. Finally, mutation is applied when a generated offspring already belongs the current population (a clone), and consists of swapping the activation time of two devices, restoring feasibility if necessary.

3 Computational Results

We applied the presented *GA*s to the water distribution network of Ferrara, Italy, population 130,000. A previous work on the same network [8], selected the set of devices to be operated after contamination detection by way of a multi-criteria *GA*, targeting both minimal number of devices and minimal volume of consumed contaminated water, supposing to have as many teams as devices, all departing at the same time. From the Pareto front provided in [8], a point associated with a good trade-off was selected, yielding the $n = 13$ devices to be operated. Commonly, the response procedure starts as soon as a sensor raises the alarm. As stated in [8], an alarm event detects a dangerous toxicity plausibly due to several contamination's locations and times; in our case, 42 contamination scenarios exist which can be simulated and then optimized. Among those, we selected the 5 the most equally spread w.r.t. to the objective function value associated to the scheduling computed according to the *as soon as possible* criterion (ASAP). This scheduling, in turn, is obtained by solving a MILP model for the *mTSP* with constraints (4), (6-11), and $U = 0$, minimizing the maximum among the devices activation times $\{t_i^{\mathcal{F}}, i \in 1..n\}$, which is also called the *makespan*.

CBC COIN-OR [6] is the MILP Solver used to tackle the optimization problems in the Hybrid *GA*s and to compute the makespan. The hydraulic simulations were performed by EPANET [11], an open-source software developed by the U.S. Environmental Protection Agency (EPA). Each simulation requires on average about 5 seconds. Since we use a cutoff of 500 invocations to the hydraulic simulator, the average computational time of each *GA* is 5×500 seconds. Other parameters are the population size $N_{pop} = 20$, and the team number $m = 3$ (a value set by the managers of the utility company operating the Ferrara network). With these parameters, CBC running time is negligible w.r.t. EPANET.

Overall, we ran 13 *GA*s. The first 10 belong to the time-based Hybrid *GA*s family (section 2.3) and differ from each other regarding speed configuration, i.e. constant speed (CS) and variable speed (VS), and the chance of using the *MILPX* method rather than *UXPF* as the crossover operator at the current iteration. More specifically, we tested five *MILPX* probability values, namely $\{0, 25, 50, 75, 100\}\%$. The other three *GA*s belong to the sequence-based family (section 2.1), namely, $2C^{CS}$, $2P^{CS}$, and $2P^{VS}$ *GA*s. For each scenario, we run each *GA* 10 times. Each *GA* at each run shares the same initial population as the other *GA*s. Fig. 3 shows, for each *GA* in ascending order, the average of the objective function values achieved in the 5×10 runs.

All the time-based Hybrid *GA*s rank first, preceding all the sequence-based ones. The last stack, at the far right of the histogram, represents the average cost of the five solutions returned by the ASAP policy computed on the 5 chosen scenarios.

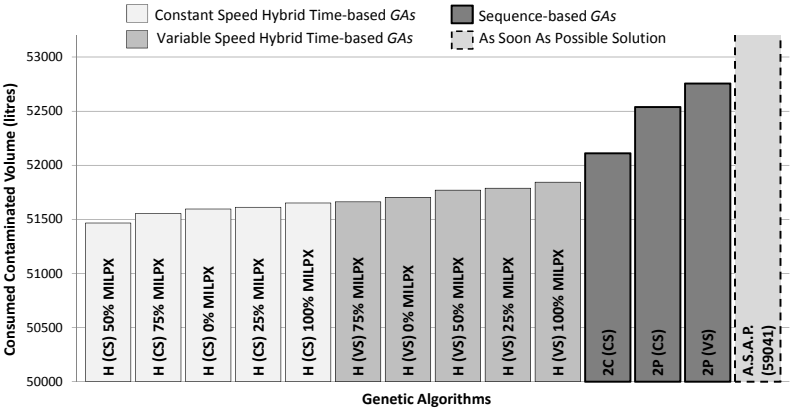


Fig. 3. Average of all the optimal candidates for each *GA* configurations and of the A.S.A.P. known solution

The histogram in Figure 3 allows us for several important considerations:

(a) the ASAP policy neither provides a lower bound for RCP nor a near-optimal solution, since its average volume of consumed contaminated water is much higher than any proposed *GA*s.
(b) Time-based encodings find better solutions than those based on sequences; this is probably due to the fact that the time-based Hybrid *GA*s work in the same solution space of the hydraulic simulator.
(c) Constant speed encodings find better solutions than variable speed encodings; this may be due to the fact that the search space of the variable speed encodings is quite larger due to pauses. Nevertheless, all variable speed Hybrid *GA*s outperform the constant speed sequence-based *GA*s.
(d) For both variable speed and constant speed, a mixed $UXPF$ and $MILPX$ policy for the Hybrid *GA*s finds, on average, better solutions than totally unbalanced policies.

The histogram in Figure 3 shows that hybrid MILP-*GA*s based on a time representation have on average a better performance with respect to the encodings known in the literature. However, in principle this result could be due to luck: indeed, we can only experiment on a limited number of instances and all algorithms are based on some randomness. For this reason there exists a probability that the new time-based algorithm is worse than the others, despite of its better performance in the finite number of the performed experiments. In order to disprove such conjecture, one should use a *significance test* [3]. We apply it to the five algorithms described earlier, namely the H^{CS}, H^{VS}, $2C^{CS}$, $2P^{CS}$ and $2P^{VS}$. For the first two, we selected the best configuration with respect to the percentage of the two crossovers, i.e., 50% for the H^{CS} and 75% for the H^{VS}. A common test used to compare multiple algorithms is the Friedman test [9]. In our case it affirms, with a confidence less than 10^{-4}, that there are some algorithms which perform significantly differently. In order to find the significantly different pairs, one should use the so-called *post-hoc analysis*; we adopted the Nemenyi procedure [9], that consists of pair-wise tests within the whole set of groups. For each pair of algorithms, Table 4

	H^{CS}	H^{VS}	$2C^{CS}$	$2P^{VS}$	$2P^{CS}$
H^{CS}	1	0.0193	$<$ **0.0001**	$<$ **0.0001**	$<$ **0.0001**
H^{VS}	0.0193	1	0.0499	$<$ **0.0001**	$<$ **0.0001**
$2C^{CS}$	$<$ **0.0001**	0.0499	1	0.0054	**0.0016**
$2P^{VS}$	$<$ **0.0001**	$<$ **0.0001**	0.0054	1	0.7043
$2P^{CS}$	$<$ **0.0001**	$<$ **0.0001**	**0.0016**	0.7043	1

Fig. 4. The $p-value$ for each pair-wise Nemenyi test. The bold $p-values$ are less than α^*.

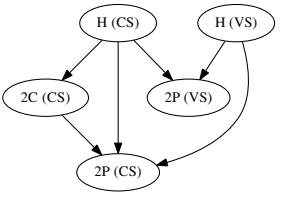


Fig. 5. The dominance graph

reports the confidence level (the so-called p-value) when assuming the two algorithms have different behaviour. As we can see, such confidence is very low, and in many cases below 10^{-4}.

However, although single confidence levels are low, the probability of having *at least one error* increases with the number of comparisons (i.e., $N_{comp} = \frac{5\times4}{2} = 10$). Conventionally, p-values are considered significant when they are below $\alpha = 5\%$. In order to ensure that the whole table contains no errors with p-value below α, Bonferroni [9] suggests to take as significant in Table 4 only those pairs for which the significance is below $\alpha^* = \alpha/N_{comp} = 0.05/10 = 0.005$.

Results in Fig. 4 are graphically depicted in Fig. 5, where an arrow from algorithm A to B means that algorithm A dominates B with a p-value below 0.5%. Accordingly with the histogram in Fig. 3, the dominance graph in Fig. 5 confirms that the Hybrid time-based *GA*, with a very little margin of error, achieves lower volumes of consumed contaminated water with respect to all sequence-based *GA*s.

4 Conclusions

In this study, we addressed an important problem in the security of water distribution systems: the near-optimal planning of the response to an event of contamination.

We tackled the problem by way of genetic algorithms which optimize the value of a black-box objective function, computed through a hydraulic simulator. We implemented two crossover operators taken from the literature on multiple traveling salesman problem, then we proposed and implemented two new crossover operators that exploit a mixed-integer linear programming solver, obtaining a hybrid GA-MILP algorithm.

We ran an extensive experimentation, in which we compared 13 variants of the various algorithms on 5 scenarios for 10 runs each. Considering that each invocation of the black-box function takes about 5 seconds on a modern computer and that we used a cutoff of 500 invocations, we have a total computing time of about 19 days.

All the proposed *GA*s improve on the common sense inspired solution. This confirms that the actual scheduling times impact on the solution value and should be explicitly taken into account by any recovery procedure. Comparing their average behaviour, we observed that the new, hybrid, algorithms outperform all the others. A significance test confirms this result, with a confidence level below 5%.

In future work, we plan to experiment on other scenarios and additional devices, and to test the effect of variable speed in 2*C GA*.

Acknowledgements. This work was partially supported by EU project *ePolicy*, FP7-ICT-2011-7, grant agreement 288147. Possible inaccuracies of information are under the responsibility of the project team. The text reflects solely the views of its authors. The European Commission is not liable for any use that may be made of the information contained in this paper.

References

1. Alfonso, L., Jonoski, A., Solomatine, D.: Multiobjective optimization of operational responses for contaminant flushing in water distribution networks. Journal of Water Resources Planning and Management 136(1), 48–58 (2010)
2. Alvisi, S., Franchini, M., Gavanelli, M., Nonato, M.: Near-optimal scheduling of device activation in water distribution systems to reduce the impact of a contamination event. Journal of Hydroinformatics (in press) doi:10.2166/hydro.2011.147
3. Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.): Experimental Methods for the Analysis of Optimization Algorithms. Springer (2010)
4. Bektas, T.: The multiple traveling salesman problem: An overview of formulations and solution procedures. Omega 34(3), 209–219 (2006)
5. Carter, A.E., Ragsdale, C.T.: A new approach to solving the multiple traveling salesperson problem using genetic algorithms. European Journal of Operational Research 175(1), 246–257 (2006)
6. Forrest, J., Lougee-Heimer, R.: CBC User Guide. Computational Infrastructure for Operations Research, http://www.coin-or.org/Cbc/cbcuserguide.html
7. Goldberg, D.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley (1989)
8. Guidorzi, M., Franchini, M., Alvisi, S.: A multi-objective approach for detecting and responding to accidental and intentional contamination events in water distribution systems. Urban Water 6(2), 115–135 (2009)
9. Hollander, M., Wolfe, D.: Nonparametric Statistical Methods, 2nd edn. Wiley (1999)
10. Murray, R., Hart, W., Phillips, C., Berry, J., Boman, E., Carr, R., Riesen, L.A., Watson, J.P., Haxton, T., Herrmann, J., Janke, R., Gray, G., Taxon, T., Uber, J., Morley, K.: US environmental protection agency uses operations research to reduce contamination risks in drinking water. Interfaces 39(1), 57–68 (2009)
11. Rossman, L.: EPANET 2 users manual. National Risk Management Research Laboratory, Office of research and development, U.S. Environmental Protection Agency, USA
12. Toth, P., Vigo, D.: The vehicle routing problem. SIAM (2002)

HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search

Gabriela Ochoa[1], Matthew Hyde[1], Tim Curtois[1], Jose A. Vazquez-Rodriguez[1], James Walker[1], Michel Gendreau[2], Graham Kendall[1], Barry McCollum[3], Andrew J. Parkes[1], Sanja Petrovic[1], and Edmund K. Burke[4]

[1] School of Computer Science, University of Nottingham, UK
[2] CIRRELT, University of Montreal, Canada
[3] School of Electronics and Computer Science, Queen's University, UK
[4] Department of Computing Science and Mathematics, University of Stirling, UK

Abstract. This paper presents *HyFlex*, a software framework for the development of cross-domain search methodologies. The framework features a common software interface for dealing with different combinatorial optimisation problems and provides the algorithm components that are problem specific. In this way, the algorithm designer does not require a detailed knowledge of the problem domains and thus can concentrate his/her efforts on designing adaptive general-purpose optimisation algorithms. Six hard combinatorial problems are fully implemented: maximum satisfiability, one dimensional bin packing, permutation flow shop, personnel scheduling, traveling salesman and vehicle routing. Each domain contains a varied set of instances, including real-world industrial data and an extensive set of state-of-the-art problem specific heuristics and search operators. HyFlex represents a valuable new benchmark of heuristic search generality, with which adaptive cross-domain algorithms are being easily developed and reliably compared.This article serves both as a tutorial and a as survey of the research achievements and publications so far using HyFlex.

1 Introduction

There is a renewed and growing research interest in techniques for automating the design of heuristic search methods. The goal is to reduce the need for a human expert in the process of designing an effective algorithm to solve a search problem and consequently raise the level of generality at which search methodologies can operate. Evolutionary algorithms and metaheuristics have been successfully applied to solve a variety of real-world complex optimisation problems. Their design, however, has become increasingly complex. In order to make these methodologies widely applicable, it is important to provide self-managed systems that can configure themselves 'on the fly'; adapting to the changing problem (or search space) conditions, based on general high-level guidelines provided by their users.

Researchers pursuing these goals within combinatorial optimisation, are often limited by the number of problem domains available to them for testing their

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 136–147, 2012.

adaptive methodologies. This can be explained by the difficulty and effort required to implement state-of-the-art software components, such as the problem model, solution representation, objective function evaluation and search operators; for many different combinatorial optimisation problems. Although several benchmark problems in combinatorial optimisation are available; they contain mainly the data of a set of instances and their best known solutions. They generally do not incorporate the software necessary to encode the solutions and calculate the objective function, let alone existing search operators for the given problem. It is the researcher who needs to provide these in order to later test their high-level adaptive search method. To overcome such limitations, we propose *HyFlex*, a modular and flexible Java class library for designing and testing iterative heuristic search algorithms. It provides a number of problem domain modules, each of which encapsulates the problem-specific algorithm components. Importantly, only the high-level control strategy needs to be implemented by the user, as HyFlex provides an easy-to-use interface with which the problem domain modules can be linked.

A number of research themes within operational research, computer science and artificial intelligence would benefit (and are already benefiting) from the proposed framework. Among them: hyper-heuristics [6,20], adaptive memetic algorithms [17,21], adaptive operator selection [11], reactive search [1], variable neighborhood search and its adaptive variants [19]; and generally the development of adaptive parameter control strategies in evolutionary algorithms [10]. HyFlex can be seen as a unifying and extended benchmark for combinatorial optimisation, with which the performance of different cross-domain adaptive techniques can be reliably assessed and compared. Practitioners can also gain even if they are only interested in one specific domain, because they could have available a large range of hyper-heuristics. A simple test process could determine the hyper-heuristic that best exploits the underlying domain and so allows practitioners to quickly and easily obtain their results without having to implement a complex search control process themselves.

HyFlex was used to support an international research competition: the first Cross-Domain Heuristic Search Challenge [16]. The challenge is analogous to the athletics Decathlon event, where the goal is not to excel in one event at the expense of others, but to have a good general performance on each. Competitors submitted one Java class file using HyFlex representing their hyper-heuristic or high-level search strategy. This ensures that the competition is fair, because all of the competitors must use the same problem representation and search operators. Moreover, due to the common interface, the competition considered not only hidden instances, but also two hidden domains.

The purpose of this article is to describe the HyFlex framework as a benchmark tool for research in hyper-heuristics and adaptive/autonomous heuristic search. A detailed analysis of the 2011 CHeSC competition is beyond the scope of this article and will be discussed elsewhere. The next section describes the antecedents and architecture of the HyFlex framework. It also includes a detailed account of how to implement and run hyper-heuristics within the framework

and a brief summary of the problem domains currently implemented. Section 3 presents a survey of published research and achievements made possible by the framework so far. Finally, section 4 summarises our contribution and suggests directions for future research.

2 The HyFlex Framework

HyFlex (Hyper-heuristics Flexible framework) is a software framework designed to enable the development, testing and comparison of iterative general-purpose heuristic search algorithms (such as hyper-heuristics). To achieve these goals it uses modularity and the concept of decomposing a heuristic search algorithm into two main parts:

1. A general-purpose part: the algorithm or hyper-heuristic.
2. The problem-specific part: provided by the HyFlex framework.

In the hyper-heuristics literature, this idea is also referred to as the domain barrier between the problem-specific heuristics and the hyper-heuristic [5,8]. HyFlex extends the conceptual domain-barrier framework by maintaining a population (instead of a single incumbent solution) in the problem domain layer. Moreover, it provides a richer variety of problem specific heuristics and search operators (i.e. it includes crossover and 'ruin-recreate' heuristics). Another relevant antecedent to HyFlex is PISA [2], a text-based software interface for multi-objective evolutionary algorithms. PISA provides a division between the application-specific and the algorithm-specific parts of a multi-objective evolutionary algorithm. In HyFlex, the interface is given by an abstract Java class. This allows a more tight coupling between the modules and overcomes some of the speed limitations encountered in PISA. Moreover, HyFlex provides a richer variety of fully implemented combinatorial optimisation problems including real-world instance data.

The framework is written in Java which is familiar to and commonly used by many researchers. It also benefits from object orientation, platform independence and automatic memory management. At the highest level, the framework consists of just two abstract classes: ProblemDomain and HyperHeuristic. The structure of these classes is shown in the class diagram of figure 1. In the diagram, the signatures adjacent to circles are public methods and fields and the signatures adjacent to diamonds are protected. Abstract methods are denoted by italics and the implementations of these methods are necessarily different for each instantiation of problem domain or hyper-heuristic.

2.1 The `ProblemDomain` Class

As shown in figure 1, an implementation of the `ProblemDomain` class provides the following elements, each of which is easily accessed and managed with one or more methods.

1. A user-configurable memory (a population) of solutions, which can be managed by the hyper-heuristic through methods such as `setMemorySize` and `copySolution`.
2. A routine to randomly initialise solutions, `initialiseSolution(`i`)`, where i is the index of the solution array in the memory.
3. A set of problem specific heuristics, which are used to modify solutions. These are called with the `applyHeuristic(`i, j, k`)` method, where i is the index of the heuristic to call, j is the index of the solution in memory to modify and k is the index in memory where the resulting solution should be placed. Solution j is not changed by this operation. Each problem-specific heuristic in each problem domain is classified into one of four groups, shown below. The heuristics belonging to a specific group can be accessed by calling `getHeuristicsOfType(`*type*`)`.
   - Mutational or perturbation heuristics: perform a small change on the solution, by swapping, changing, removing, adding or deleting solution components.
   - Ruin-recreate (destruction-construction) heuristics: partly destroy the solution and rebuild or recreate it afterwards. These heuristics can be considered as large neighbourhood structures. They are, however, different from the mutational heuristics in that they can incorporate problem specific construction heuristics to rebuild the solutions
   - Hill-climbing or local search heuristics: iteratively make small changes to the solution, only accepting non-deteriorating solutions, until a local optimum is found or a stopping condition is met. These heuristics differ from mutational heuristics in that they incorporate an iterative improvement process and they guarantee that a non-deteriorating solution will be produced.
   - Crossover heuristics: take two solutions, combine them and return a new solution.
4. A varied set of instances that can be easily loaded using the method: `loadInstance(`a`)`, where a is the index of the instance to be loaded.
5. A fitness function, which can be called with the `getFunctionValue(`i`)` method, where i is the index of the required solution in the memory. HyFlex problem domains are always implemented as minimisation problems, so a lower fitness is always better. The fitness of the best solution found so far in the run can be obtained with the `getBestSolutionValue()` method.
6. Two parameters: α and $\beta, (0 <= [\alpha, \beta] <= 1)$, which are the 'intensity' of mutation and 'depth of search', respectively, that control the behaviour of some search operators.

2.2 The `HyperHeuristic` Class

The HyperHeuristic class is designed to allow algorithms which implement this class to be compared and benchmarked across one or more of the problem domains available (for example, in a competition). Users create cross-domain

Fig. 1. Class diagram for the HyFlex framework

heuristic algorithms by creating implementations of this abstract class. Each class must contain a `toString()` method, to give the methodology a name. It must also contain a `solve()` method, in which the functionality of the particular methodology is written.

The `solve()` method would normally contain a loop, which continues while the time limit (defined by the user) has not been exceeded. In the loop, the code should provide a mechanism for selecting between the available problem-specific heuristics and choose to which solutions in memory to apply the heuristics. This class could choose to work with a memory size of 1 for a single point search, or a large memory could be maintained for a population based approach. The memory can be easily defined and maintained through calling methods of the `ProblemDomain` class, where the memory is stored. A hyper-heuristic class automatically records the length of time for which it has been running and this can be monitored through methods such as `hasTimeExpired()` and `getElapsedTime()`.

The solve method is the only method which must be implemented. All other common functionality is provided by the HyFlex software, such as the timing function and the recording of the best solution.

2.3 Running a Hyper-heuristic

Algorithm 1 shows the ease with which a hyper-heuristic can be run on a problem domain. An object is created for the problem domain (in this example MAX-SAT) and for the hyper-heuristic, each with a random seed. Then a problem instance is loaded from the selection available in the problem domain object. In this example we choose the instance with index 0. The problem domain is now set up for the hyper-heuristic.

We set the time for which the hyper-heuristic will run, in milliseconds. Then the hyper-heuristic object is given a reference to the problem domain object. Now that the setup is complete, the `run()` method of the hyper-heuristic is called to start the search process. The hyper-heuristic will run for 60 seconds in this example and the best solution found during that time is retrievable with the `getBestSolutionValue()` method, as shown in Algorithm 1. This (or indeed any) hyper-heuristic can be run on the 5 other problem domains by changing just one line of code.

Algorithm 1. Java code for running a hyper-heuristic on a problem domain

```
ProblemDomain problem = new SAT(seed1);
HyperHeuristic HHObject = new ExampleHyperHeuristic1(seed2);
problem.loadInstance(0);
HHObject.setTimeLimit(60000);
HHObject.loadProblemDomain(problem);
HHObject.run();
System.out.println(HHObject.getBestSolutionValue());
```

2.4 An Example Hyper-heuristic

This section provides an example hyper-heuristic, to illustrate the ease with which a hyper-heuristic can be created. This is done by extending the Hyper-Heuristic abstract class and implementing only one method. All of the common functionality is provided by the HyFlex software, such as the timing function and the recording of the best solution. This example demonstrates exactly how to use certain elements of HyFlex functionality, including the solution memory.

After the `run()` method of the hyper-heuristic is called (see section 2.3), the hyper-heuristic abstract class performs some housekeeping tasks, such as initialising the timer and then calls the `solve` method of the chosen hyper-heuristic. In Algorithm 1, this is an object of the class `ExampleHyperHeuristic1`. Algorithm 2 shows the code for the `solve()` method in `ExampleHyperHeuristic1`. It shows that very few lines of code are necessary in order to implement a hyper-heuristic method with the HyFlex framework. Algorithm 2 is written in pseudocode, but each line corresponds to no more than one line of actual Java code. The `solve()` method is the only substantial method which needs to be

implemented. Indeed the only other necessary method is `toString()`, which requires one line to give the hyper-heuristic a name.

From Algorithm 2, we can see that the `solve()` method takes the problem domain object as an argument and checks for the number of search operators available within it. We also initialise a value to store the current objective function value. It is also necessary to initialise at least one solution in the memory. The default memory size is 2 and we initialise the solution at index 0, which means we build an initial solution with the method specified in the problem domain (generally a fast randomised constructive heuristic). The solution at index 1 remains uninitialised and therefore has a value of `null`.

An implemented hyper-heuristic must always contain a while loop which checks if the time limit has expired. The code within the loop specifies the main functionality of the hyper-heuristic. In this example, we choose a random operator and then apply it to the solution at index 0. The modified solution is put in the memory at index 1 (previously not initialised). Note that a random number generator `rng` is provided by the HyperHeuristic abstract class. This is created when the hyper-heuristic object's constructor is called and is the reason why that constructor requires a random seed.

If the new solution is superior to the old solution, it is accepted and the new solution overwrites the old one in memory. The `copySolution` method of the problem domain class is employed to manage this. If the new solution is not superior, then the new solution is accepted with 0.5 probability.

Algorithm 2. Pseudocode for the solve method of `ExampleHyperHeuristic1`. This is called when the `run()` method of the hyper-heuristic is called (see Algorithm 1).

```
Require: A ProblemDomain object, problem
  int numberOfHeuristics = problem.getNumberOfHeuristics
  double currentObjValue = Double.POSITIVE-INFINITY
  problem.initialiseSolution(0)
  while hasTimeExpired = FALSE do
    int h = rng.nextInt(numberOfHeuristics)
    double newObjValue = problem.applyHeuristic(h, 0, 1)
    double delta = currentObjValue - newObjValue
    if delta > 0 then
      problem.copySolution(1, 0)
      currentObjValue = newObjValue;
    else
      if rng.nextBoolean = TRUE then
        problem.copySolution(1, 0)
        currentObjValue = newObjValue;
      end if
    end if
  end while
```

2.5 HyFlex Problem Domains

Currently, six problem domain modules are implemented. From these, 4 were given as test domains for the CHeSC competition: maximum satisfiability (MAX-SAT), one-dimensional bin packing, permutation flow shop and personnel scheduling. Two additional domains, namely, the traveling salesman and the capacitated vehicle routing problem [22], were later implemented and used as hidden domains in the competition. Each domain includes 10 training instances from different sources and a number of problem-specific heuristics of the types discussed in section 2.1. The six domains together with technical reports describing them in detail, including the problem formulation, solution initialisation, instance date and low-level heuristics, can be found on the competition site [16]. Due to space constraints we only present a summary describing the solution initialisation, the total number of low-level heuristics and the number of heuristics of each type (Table 1).

Table 1. HyFlex problem domains, indicating initialisation, the total number of low-level heuristics and the number of heuristics per type

Domain	*Initialisation*	*Total*	*Mut.*	*R&R*	*Xover.*	*LS.*
Max-SAT	Random bit-string	9	4	1	2	2
Bin Packing	Randomised first-fit heuristic [13]	8	3	2	1	2
Flow Shop	Randomised NEH procedure [15]	15	5	2	3	4
Pers. Sched.	Randomised hill climbing heuristic	12	1	3	3	4
TSP	Random permutation	15	5	1	3	6
VRP	Randomised constructive heuristic	12	4	2	2	4

3 HyFlex Achievements

HyFlex was made available in August 2010 when the CHeSC competition was launched at the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010)[1]. In May 2011, a web statistics counter was added to the website and since then, up to January 30^{th} 2012, it has recorded 4,721 visits and 9,451 page views. This section briefly surveys the research achievements and publications made possible with HyFlex so far.

The first article implementing hyper-heuristics using HyFlex was published in 2010 [3], where several hyper-heuristics combining two heuristic selection mechanisms and three acceptance criteria were compared. A multiple neighbourhood iterated local search was also implemented and found to outperform the other approaches as a general optimiser. This iterated local search hyper-heuristic contains a perturbation stage, during which a neighborhood move is selected uniformly at random (from the available pool of mutation and ruin-recreate heuristics) and applied to the incumbent solution; followed by a greedy improvement stage (using all the local search heuristics). The approach is extended in

[1] http://www.cs.qub.ac.uk/~B.McCollum/patat10/

[4] by substituting the uniform random selection of neighbourhoods in the perturbation stage, by online learning strategies. Two strategies were implemented: *choice function* [8] (from the hyper-heuristics literature) and *extreme value based adaptive operator selection* [11] (from the evolutionary computation literature), with the latter producing better overall results. This last implementation was the best performing hyper-heuristic before the competition started.

In [12], the authors used reinforcement learning for heuristic selection and explored several variants for the rewards, policy and learning functions. Different ways of modeling the agents' states and actions were also explored. The results reported are preliminary and do not compare well with those generated by other HyFlex hyper-heuristics.

In [18], the authors implement a multi-stage hyper-heuristic, combining a greedy stage with a random descent stage, followed by a simple solution acceptance mechanism. During the greedy stage, all the available low-level heuristics are applied during a number of steps and a subset of the best performing heuristics (active set) is constructed using a dominance-based strategy. The subsequent random descent stage, randomly selects a heuristic from the active set and applies it repeatedly until no improvement is achieved. The transition between stages is controlled by a probability parameter. This relatively simple approach produces very good results when compared with previous HyFlex hyper-heuristics.

HyFlex was used to support the CHeSC 2011 competition. The event successfully attracted the interest and participation of universities and academic institutions across the six continents; 43 registrations and 20 submissions were received. We received several positive and encouraging comments regarding both HyFlex and the competition, from the registered participants. The competition results and brief technical reports describing the participant entries can be found in the website [16]. Here, we briefly summarise the top 4 hyper-heuristics:

1. *AdapHH*: Mustafa Misir, University KaHo Sint-Lieven, Belgium.
   A 'traditional' selective hyper-heuristic that includes two stages: heuristic selection and solution acceptance. Heuristic selection is done by learning dynamic heuristic sets and effective pairs of heuristics. The algorithm also incorporates adaptation of the heuristic parameters and an adaptive threshold acceptance. This approach was presented in [14].
2. *VNS-TW*: Ping-Che Hsiao, National Taiwan University, Taiwan.
   A variable neighborhood search algorithm that orders perturbation heuristics according to strength. It includes two stages: diversification and intensification and incorporates an adaptive technique to adjust the strength of the local search heuristics.
3. *ML*: Mathieu Larose, Montreal University,Canada.
   An adaptive iterated local search algorithm with three stages: diversification, intensification and a simple adaptive move acceptance. Reinforcement learning is used for selecting heuristics.
4. *PHUNTER*: Fan Xue, Hong Kong Polytechnic University, Hong Kong.
   A hyper-heuristic that can assemble different iterated local search algorithms. The authors use the metaphor of pearl hunting; there is a diver-

sification stage (surface and change target area) and an intensification stage (dive and find pearl oysters). The algorithm also uses offline learning to identify search modes. This approach was presented in [7].

Several new best-known solutions have been found for personnel scheduling instances [9] using HyFlex (see Table 2). This is an interesting result, considering that HyFlex was designed to implement general-purpose search heuristics.

Table 2. Personnel scheduling best-known solutions obtained by the PHUNTER HyFlex hyper-heuristic

Instance name	HyFlex best-known	Previous best-known	staff	Shift types	days
CHILD-A2	1095	1111	41	5	42
ERRVH-A	2142	2197	51	8	42
ERRVH-B	3121	6859	51	8	42
MER-A	9017	9915	54	12	42

A special session on Cross-domain Heuristic Search was held as part of the Learning and Intelligent OptimizatioN conference (LION 2012)[2]. Seven papers were accepted and presented using HyFlex for implementing cross-domain hyper-heuristics. HyFlex is also being used as a tool for teaching modules in meta-heuristics and evolutionary algorithms. Finally, HyFlex is potentially useful from the point of view of practitioners. If they provide their problem-specific software components following the interface, they will have at their disposal a growing number of hyper-heuristics and adaptive search controllers ready to use.

4 Discussion and Future Work

HyFlex is a software framework which enables cross-domain algorithms to be easily developed and reliably compared. It currently provides 6 problem domains, each containing a set of problem instances and search operators to apply. Therefore, it represents a novel extension of the notion of benchmark for combinatorial optimisation. Researchers from different communities and themes within computer science, artificial intelligence and operational research, can benefit from HyFlex, as it provides a common benchmark in which to test the performance and behavior of single-point and population-based self-configuring search heuristics. When using HyFlex, researchers can concentrate their efforts on designing their adaptive methodologies, rather than implementing the required set of problem domains. There is currently ample evidence that HyFlex is being used by the research community for both research and teaching.

Different algorithm design ideas have been implemented and tested using HyFlex. Some successful design principles start to emerge such as the use of

[2] http://intelligent-optimization.org/LION6

diversification and intensification phases, the use of reinforcement learning for heuristic selection, adaptation of the heuristic parameters and the use of adaptive acceptance criteria. Interesting emerging ideas are the use of co-evolution and evolution of heuristic sequences. The use of a population is starting to be valued within selective hyper-heuristic research, which has traditionally focused on single-point search algorithms. It is our vision that the HyFlex framework will continue to facilitate and increase international interest in developing hyper-heuristics and adaptive heuristic search methodologies that can find wider application in practice.

HyFlex can be extended to include new domains, additional instances and operators in existing domains and multi-objective and dynamic problems. The current software interface can also be extended to incorporate additional feedback information from the domains to guide the adaptive search controllers. In particular, parameterised low-level heuristics and diversity metrics can be included. We plan to host a new edition of the competition and maintain a repository of results and updates to the HyFlex framework.

Acknowledgements. The authors wish to thank Dr. Ender Özcan for interesting discussions and his contagious enthusiasm for hyper-heuristic research.

References

1. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Operations Research/Computer Science Interfaces Series, vol. 45. Springer (2009)
2. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA – A Platform and Programming Language Independent Interface for Search Algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
3. Burke, E.K., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vazquez-Rodriguez, J.A., Gendreau, M.: Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms. In: IEEE Congress on Evolutionary Computation (CEC 2010), Barcelona, Spain, pp. 3073–3080 (July 2010)
4. Burke, E.K., Gendreau, M., Ochoa, G., Walker, J.D.: Adaptive iterated local search for cross-domain optimisation. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011, pp. 1987–1994. ACM, New York (2011)
5. Burke, E.K., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 457–474. Kluwer (2003)
6. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.: A Classification of Hyper-heuristic Approaches. In: Handbook of Metaheuristics. International Series in Operations Research & Management Science, ch. 15, vol. 146, pp. 449–468. Springer (2010)
7. Chan, C.Y., Xue, F., Ip, W.H., Cheung, C.F.: A hyper-heuristic inspired by pearl hunting. In: International Conference on Learning and Intelligent Optimization (LION 6). LNCS, Springer (to appear, 2012)

8. Cowling, P., Kendall, G., Soubeiga, E.: A Hyperheuristic Approach to Scheduling a Sales Summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
9. Curtois, T.: Staff rostering benchmark data sets. Website (2011), http://www.cs.nott.ac.uk/~tec/NRP/
10. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter Control in Evolutionary Algorithms. In: Parameter Setting in Evolutionary Algorithms, pp. 19–46. Springer (2007)
11. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Extreme Value Based Adaptive Operator Selection. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN X 2008. LNCS, vol. 5199, pp. 175–184. Springer, Heidelberg (2008)
12. Di Gaspero, L., Urli, T.: A reinforcement learning approach for the cross-domain heuristic search challenge. In: Proceedings of the 9th Metaheuristics International Conference (MIC 2011), Udine, Italy, July 25-28 (2011)
13. Johnson, D., Demers, A., Ullman, J., Garey, M., Graham, R.: Worst-case performance bounds for simple one-dimensional packaging algorithms. SIAM Journal on Computing 3(4), 299–325 (1974)
14. Misir, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G.: An intelligent hyper-heuristic framework for chesc 2011. In: International Conference on Learning and Intelligent Optimization (LION 6). LNCS. Springer (to appear, 2012)
15. Nawaz, M., Enscore Jr., E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA-International Journal of Management Science 11(1), 91–95 (1983)
16. Ochoa, G., Hyde, M.: The Cross-domain Heuristic Search Challenge (CHeSC 2011). Website (2011), http://www.asap.cs.nott.ac.uk/chesc2011/
17. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Transactions on Systems, Man, and Cybernetics, Part B 36(1), 141–152 (2006)
18. Ozcan, E., Kheiri, A.: A hyper-heuristic based on random gradient, greedy and dominance. In: Proceedings of the 26th International Symposium on Computer and Information Sciences ISCIS 2011, London, UK, July 25-28 (2011)
19. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Computers and Operations Research 34, 2403–2435 (2007)
20. Ross, P.: Hyper-heuristics. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, ch.17, pp. 529–556. Springer (2005)
21. Smith, J.E.: Co-evolving memetic algorithms: A review and progress report. IEEE Transactions in Systems, Man and Cybernetics, part B 37(1), 6–17 (2007)
22. Walker, J.D., Ochoa, G., Gendreau, M., Burke, E.K.: Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: International Conference on Learning and Intelligent Optimization (LION 6). LNCS. Springer (to appear, 2012)

Hyper-Heuristic Based on Iterated Local Search Driven by Evolutionary Algorithm

Jiří Kubalík

Department of Cybernetics
Czech Technical University in Prague
Technická 2, 166 27 Prague 6, Czech Republic
kubalik@labe.felk.cvut.cz

Abstract. This paper proposes an evolutionary-based iterative local search hyper-heuristic approach called Iterated Search Driven by Evolutionary Algorithm Hyper-Heuristic (ISEA). Two versions of this algorithm, ISEA-chesc and ISEA-adaptive, that differ in the re-initialization scheme are presented. The performance of the two algorithms was experimentally evaluated on six hard optimization problems using the HyFlex experimental framework and the algorithms were compared with algorithms that took part in the CHeSC 2011 challenge. Achieved results are very promising, the ISEA-adaptive would take the second place in the competition. It shows how important for good performance of this iterated local search hyper-heuristic is the re-initialization strategy.

Keywords: hyper-heuristic, optimization, evolutionary algorithm.

1 Introduction

Hard optimization problems cannot be solved to optimality for large instance sizes due to extreme computational demands. Hence, either approximation or metaheuristic algorithms [9] are often used to find if not optimal solutions then at least solutions of good quality in reasonable time. Recently, hyper-heuristics[1], described as "heuristics to choose heuristics", that are search methods or learning mechanisms for selecting or generating heuristics to solve computational search problems were proposed [2].

1.1 Hyper-Heuristics

There are two main classes of hyper-heuristic approaches – *heuristic selection* and *heuristic generation*. Heuristic selection approaches, for given particular problem instance and a set of problem-specific low-level heuristics (LLHs), select and apply the most suitable LLH at each problem solving state. Heuristic generation methods are used to automatically generate new heuristics, suited for the given problem, from components of pre-existing heuristics [1].

[1] Comprehensive bibliography of Hyper-heuristics:
http://www.cs.nott.ac.uk/~gxo/hhbibliography.html

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 148–159, 2012.

The heuristic selection methods can be divided into *constructive* and *local search* hyper-heuristics based on the type of used LLHs. Constructive hyper-heuristics start with an empty solution and gradually build a complete solution, by iteratively selecting appropriate low-level construction heuristic (chosen from a pool of LLHs proposed for the problem at hand) and using it to enhance the developed solution. Local search heuristics start from a complete solution, then they try to iteratively improve the current solution by selecting appropriate neighborhood structure and/or simple local searcher and applying it to the current solution.

Typically, the heuristic selection methods operate on sequences of LLHs where an optimal sequence that produces the best solution (in case of constructive methods) or improves the most the initial solution (in case of local search methods) is sought. Since the construction of the optimal sequence of LLHs is not a trivial task, high-level heuristic or meta-heuristic techniques such as ILS, VNS, simulated annealing, tabu search or EAs are often used as search strategies across the search space of heuristics [2]. Some methods make use of information learned through the problem solving process, such as performance of individual LLHs, for selecting particular heuristic at given decision point. An association of LLHs to characteristic problem solving states can also be used. For example, at some point local search and mutation heuristics can be very effective while in other problem solving state more disturbing ruin-recreate heuristics can be the only way to escape from current local optimum solution [3].

Recent analysis show that the heuristic search space is likely to contain large plateaus, i.e. regions of many heuristic sequences producing solutions of the same quality. On the other hand, the heuristics search space have in a great picture a globally convex structure with the optimal solution surrounded by many local optima [2]. In this work we propose hyper-heuristic approach based on iterative local search algorithm called POEMS [6] that might be well suited for searching this kind of search space since it can make use of structured improving moves when trying to improve the current solution.

According to the classification provided in Section 1.1, the proposed hyper-heuristic belongs to the heuristic selection approaches. Specifically, to the local search hyper-heuristics as the goal of the hyper-heuristic is to iteratively improve starting complete solution by iteratively selecting and applying local search, mutation and ruin-recreate low level heuristics to it.

1.2 Proposed Hyper-Heuristic

Iterated Search Driven by Evolutionary Algorithm Hyper-Heuristic (ISEA) presented in this paper is based on an evolutionary-based iterative local search algorithm called POEMS [6,7,8]. POEMS is an optimization algorithm that operates on a single candidate solution called a *prototype* and tries to improve it in an iterative process. In each iteration, it runs an evolutionary algorithm (EA) that seeks for the most valuable modification to the prototype. The

modifications are represented as fixed length sequences of elementary actions, i.e. sequences of problem-specific variation or mutation operators. Such action sequences, produced by the EA, can be viewed as evolved *structured mutations*. Action sequences are assessed based on how well/badly they modify the current prototype, which is passed as an input parameter to the evolutionary algorithm. Besides actions that truly modify the prototype, there is also a special type of action called *nop* (no operation). The nop actions are interpreted as void actions with no effect on the prototype. Action sequences can contain one or more nop actions. This way a variable effective length of action sequences is realized. After the EA finishes, the best evolved action sequence is checked for whether it worsens the current prototype or not. If an improvement is achieved or the modified prototype is at least as good as the current one, then the modified prototype is considered as a new prototype for the next iteration. Otherwise, the current prototype remains unchanged. The iterative process stops after a specified number of iterations.

POEMS takes the best of the two worlds of the *single-state* and *population-based* metaheuristics. It iteratively improves the current solution, but contrary to the single-state metaheuristics it searches much bigger neighborhood structure defined by fixed length sequence of elementary actions (not just a single variation operator). Such a neighborhood structure is effectively searched by means of the EA, where the EA is capable of finding both the local fine-tuning move as well as the perturbation move.

Therefore, the exploration capability of POEMS should be better than the standard single-state techniques. Moreover, the EA run in each iteration searches a limited space of the current prototype's modifications instead of the whole space of all candidate solutions to the given problem that is searched by the traditional EAs. Thus, minimal resources (i.e. small population size and small number of generations) can be sufficient to effectively search the space of structured mutations.

Two versions of the ISEA hyper-heuristic were implemented and tested with the use of the HyFlex framework [4]. The two ISEA versions were evaluated and compared with the algorithms that took place in the competition of the Cross-domain Heuristic Search Challenge CHeSC 2011[2]. The performance of the algorithms was assessed based on the point scoring system used for the competition. The competition results and the program for calculating the hyper-heuristics' score being kindly provided by the organizers of the CHeSC 2011. The presented analysis shows how important for good performance of the iterated local search ISEA hyper-heuristic is the re-initialization strategy.

Map of the paper: In Section 2 an original ISEA version is proposed in the form that took part in CHeSC 2011 challenge. Section 3 presents the version with an adaptive re-initialization rate. Section 4 presents experimental evaluation of the two ISEA algorithms. Last section concludes the paper and proposes future work directions.

[2] `http://www.asap.cs.nott.ac.uk/chesc2011/index.html`

2 Original ISEA Algorithm

The general-purpose ISEA hyper-heuristic is based on the POEMS approach, but differs in several aspects. In the following paragraphs the main structure of the ISEA algorithm and its key components will be described.

Memory Solutions. Generally, the ISEA maintains a set of three solutions to the given problem via methods provided by the HyFlex abstract class `ProblemDomain`. The meaning of the three solutions is as follows:

- *Evaluation solution* is used for evaluating candidate sequences of low level heuristics (LLHs) defined for the problem at hand.
- *Working solution* stores intermediate solutions and the final solution that are successively generated from the starting evaluation solution by applying individual LLHs of the evaluated candidate sequence of LLHs. Thus, at the beginning of the candidate sequence of LLHs evaluation process the *working solution* is set to the *evaluation solution* and then it is modified step-by-step as the LLHs are applied to it one by one.
- *Best-so-far solution* is the best solution found so far in the whole ISEA run.

Low Level Heuristics and the Prototype. Unlike other traditional applications of POEMS, ISEA operates on a prototype that does not directly represent a solution to the given problem. Instead, the prototype is a fixed-length sequence of LLHs that is to be modified by evolved action sequences. Despite the HyFlex framework provides four types of LLHs, the ISEA considers only local search, mutation and ruin-recreate heuristics for the generated sequences of LLHs. When initialized, LLHs are initially assigned to conform with the following schema:

- At the first position of the prototype, only local search LLH can be generated.
- All of the remaining positions but the last one can be initialized with any type of LLH, where all of the three types of LLH have equal probability to be chosen.
- Only local search LLH can be assigned to the last position of the prototype.

The idea behind this scheme is that when looking for better solution in the neighborhood of the *working solution*, one might try to locally optimize it first, then any type of LLH can be chosen multiple times and finally some local search heuristics can be used to finalize the new solution.

Actions. Note, that the prototype is to be modified by evolved action sequences. Following three simple actions are proposed for the ISEA:

- `addLLH(position, type, parameters)` – adds a new LLH of certain type with specified parameters to the prototype at given position.
- `removeLLH(position)` – removes the LLH that is at given position in the prototype.
- `changeLLH(position, parameters)` – modifies the LLH at given position in the prototype according to the new parameters. It can change either the type of the LLH (mutation / local search / ruin-recreate) and/or its parameters.

Obviously, the length of the prototype can vary as some LLH can be added to or removed from it. At the beginning of the EA, the prototype's length is set to p (that is one of the user defined parameters of the algorithm).

The task for the EA is to permanently evolve modifications to the current prototype that will, if possible, produce sequences of LLHs capable of improving the *evaluation solution* at any moment of the optimization process. Thus, it is assumed that at some stages the evolved action sequences will be able to adjust the prototype to realize rather disturbing moves (in order to escape a local optimum of the solved problem instance represented by the current *evaluation solution*) while at other stages the action sequences can transform the prototype so as to realize fine-tuning moves (in case there is still some room for improvement of the *evaluation solution* that can be effectively attained just by using the local search heuristics). In other words, the EA realizes the adaptation of the optimization process to the current status of the *evaluation solution*.

Assessing Quality of Action Sequences. In order to assess the performance of an action sequence, the resulting sequence of LLHs, obtained by modifying the prototype with the action sequence, is applied to the *evaluation solution* and the quality of the best solution generated through successive applying the LLHs of the sequence is taken as the quality of the candidate action sequence.

Perturbation. The *evaluation solution* is re-initialized from time to time so that it is assigned a solution that is created by perturbing the *best-so-far solution*. The perturbation is realized by mutation and ruin-recreate heuristics. This re-initialization operation is invoked as soon as the following two conditions are fulfilled

1. more than T_1 seconds has elapsed since the last perturbation action, and
2. there has been no improvement of the *evaluation solution* observed for T_2 seconds,

indicating that there was no progress observed for considerable amount of time. The parameters T_1 and T_2 are static in a sense that they stay constant during the whole ISEA run. The question is how to set the parameters in order to provide the ISEA procedure with an efficient re-initialization scheme.

2.1 ISEA Pseudo Code

At the beginning of the ISEA run, the *prototype*, *evaluation solution*, *best-so-far solution*, and starting population of action sequences are initialized. Also the counter of calculated fitness evaluations and the variables storing the time of the last improvement of the *evaluation solution* and the time of the last perturbation action are reset.

Then the algorithm repeats steps of the main loop until the maximal specified time has expired (steps 8-18 in Fig. 1). In the first step of this block one generation of the EA is carried out. This means that a pair of new action sequences is generated either by means of crossover and/or mutation. The newly generated action sequences are evaluated on the *evaluation solution*. Whenever the new

```
input:  set of local search, mutation and ruin-recreate heuristics
output: best solution found
1       lastPerturbationTime ← currentTime
2       lastImprovementTime ← currentTime
3       evaluations ← 0   // the number of evaluations calculated in one EA
4       initialize(evaluation_solution)
5       best_so_far_solution ← evaluation_solution
6       initialize(prototype)
7       initialize(populationAS)      // initialize population of action sequences
8       while (!hasTimeExpired())
9           calculateGenerationEA()        // calculates one generation of the EA
10          evaluations ← evaluations + 2   // two new solutions are generated
11          if(((currentTime − lastPerturbationTime) > T1) and
              ((currentTime − lastImprovementTime) > T2))
12              evaluation_solution ← perturb(best_so_far_solution)
13              lastPerturbationTime ← currentTime
14          if((evaluations) > N)          // start new EA
15              evaluations ← 0
16              initialize(prototype)
17              initialize(populationAS)
18      end while
19      return best_so_far_solution          // return best-so-far solution
```

Fig. 1. Pseudo code of ISEA hyper-heuristic

solution is at least as good as the *evaluation solution* the new solution replaces the *evaluation solution*. Similarly, when the new solution outperforms the *best-so-far solution*, the *best-so-far solution* is updated as well (all these operations are done in step 9 in Fig. 1). If the conditions for perturbation are fulfilled, the perturbation is carried out and the $lastPerturbationTime$ variable is updated accordingly (steps 11-13 in Fig. 1).

If the number of fitness evaluations exceeds the specified number of fitness evaluations allocated for one EA, the prototype and the population of action sequences are re-initialized (steps 14-17 in Fig. 1). Once the time allocated for the whole run has expired the *best so far solution* is returned as the final solution.

2.2 ISEA Control Parameters' Setting

Before the main procedure of the ISEA starts, it is checked whether low time demand or high time demand executions of the LLHs are expected for the problem instance at hand. This is estimated by applying several LLHs (five LLHs were used in this work) on the initial *evaluation solution*. Based on the estimate one of two possible configurations is chosen. The configuration for instances with low time demand LLHs uses longer prototypes, longer action sequences, bigger

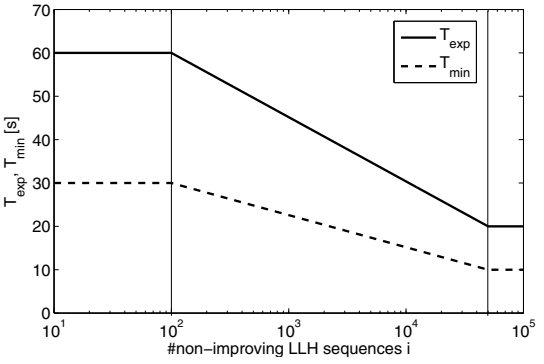


Fig. 2. Mapping the estimated total number of non-improving LLH sequences to the T_{exp} and T_{min}

population of action sequences and larger number of fitness evaluations calculated in one EA. The configuration for instances with high time demand LLHs uses shorter prototype, shorter action sequences, smaller population of action sequences and smaller number of fitness evaluations in one EA. Note, that just the time complexity of the LLHs for the problem instance at hand is estimated. There is no attempt to disclose the type of the problem at hand.

3 ISEA with Adapted Re-initialization Rate

A variant of ISEA algorithm described in this section does not use the hard-wired re-initialization strategy based on the rule, where the perturbation action takes place if and only if the two crisp time-based conditions for the action are fulfilled. Here, a time interval, (T_{min}, T_{exp}), is determined within which the re-initialization of the *evaluation solution* can take place with certain probability, P_{reinit}. Both, the boundaries of the time interval and the probability P_{reinit} are recalculated anew after each re-initialization action. Then, if a new solution is generated such that it is not better than the current *evaluation solution* (i.e. the candidate sequence of LLHs did not improve the *evaluation solution*) and one of the two following conditions is fulfilled

1. at least T_{exp} seconds has elapsed since the last re-initialization action,
2. at least T_{min} seconds has elapsed since the last re-initialization action and a random number from the interval (0.0, 1.0) smaller than P_{reinit} has been generated,

the new *evaluation solution* is calculated by perturbing the *best-so-far solution*.

Clearly, the parameters T_{min}, T_{exp} and P_{reinit} are crucial for proper functioning of the re-initialization strategy. Their values are derived from the estimated total number of non-improving LLH sequences generated during the whole ISEA run. The idea behind setting T_{min} and P_{reinit} is such that when solving less

time-demanding problem instance, for which we can evaluate large number of candidate LLH sequences per time unit, the re-initialization frequency can be quite high since the algorithm can converge to some local optimum in short time-period. On the other hand, when solving heavy time-demanding problem instance, for which we can evaluate only very small number of candidate LLH sequences per time unit, the re-initialization frequency should be rather low since the algorithm might not be able to converge to a local optimum fast enough.

This idea is realized by an adaptation procedure, which is invoked repeatedly during the ISEA run, consisting of the following steps:

1. A total number of non-improving LLH sequences generated within the whole run, $N_{non-imp}$, is estimated based on the number of non-improving LLH sequences observed so far.
2. Log-linear transformation is used to map $N_{non-imp}$ to the *expected time-interval between two consecutive re-initialization actions*, T_{exp}, see the red line in Fig. 2. The figure shows that for "easy" instances (i.e. instances for which the heuristics are less time-demanding) the ISEA is allowed to re-initialize the *evaluation solution* in shorter time, T_{exp}, while for "hard" instances the expected time interval between two re-initializations gets longer. There are two boundary values of the $N_{non-imp}$ considered for the transformation – the lower boundary of 100 and the upper boundary of 5×10^5. If the value of $N_{non-imp}$ smaller than 100 is estimated then the T_{exp} is set to 60s. For any value of $N_{non-imp}$ greater than 5×10^5 the T_{exp} is set to 20s.
3. Given the values of T_{exp} and $N_{non-imp}$ determined in steps 1. and 2., a number of non-improving LLHs, N_{Texp}, expected within time interval of T_{exp} seconds is estimated.
4. The probability of realizing the re-initialization action is $P_{reinit} = 1/N_{Texp}$. In order to eliminate rapid re-initialization of the evaluation solution, a minimal time-interval between two consecutive re-initializations, T_{min}, is set to $T_{exp}/2$, see Fig. 2.

Note, the adaptation procedure is hardware-oriented as it relies on absolute time constants, T_{min} and T_{exp}, in seconds. These can easily be replaced with hardware-independent time values given as fractions of the total running time.

4 Experiments

This section presents experiments carried out with the two versions of the hyper-heuristic approach ISEA

- ISEA-chesc – this version, described in Section 2, took part in the final competition CHeSC 2011.
- ISEA-adaptive – this is the version with adapted re-initialization rate.

The two versions of the ISEA hyper-heuristic were implemented and tested with the use of the HyFlex framework [4]. The framework provides experimental environment with the following six hard optimization problems implemented:

- Max-SAT problem (MSAT),
- Bin packing problem (BP),
- Personnel scheduling problem (PS),
- Flow shop problem (FS),
- Traveling salesman problem (TSP),
- Vehicle routing problem (VRP).

The two ISEA algorithms were compared with the algorithms that took place in the competition of the Cross-domain Heuristic Search Challenge CHeSC 2011[3]. Likewise in the CHeSC 2011 competition, 31 runs per instance were conducted with each of the two ISEA algorithm variants. The two ISEA algorithms were compared in an indirect way so that each of them was added separately to the set of CHeSC 2011 competition algorithms and the following performance indicators were used for their mutual comparison:

- The final rank of each algorithm and total number of points received in the competition.
- Individual scores of each algorithm per domain.
- Median of the best objective values calculated for each instance from the set of 31 runs.

The competition results and the program for calculating the hyper-heuristics' score being kindly provided by the organizers of the CHeSC 2011 competition.

4.1 Experimental Setup

ISEA-chesc configuration for easy and hard instances, see Section 2.2:

- common parameters:
  - total running time: 600 [s]
  - probability of crossover: 75%
  - probability of mutation: 25%
  - tournament size: 2
- parameters for easy instances:
  - initial prototype length (i.e. a length of the LLHs sequence): 5
  - population size:50
  - action sequence length: 5
  - max. number of evaluations in one EA: 200
  - $T_1 = 45$ [s]; $T_2 = 10$ [s]
- parameters for hard instances:
  - initial prototype length: 3
  - population size: 30
  - action sequence length: 3
  - max. number of evaluations in one EA: 50
  - $T_1 = 60$ [s]; $T_2 = 10$ [s]

[3] `http://www.asap.cs.nott.ac.uk/chesc2011/index.html`

Table 1. Comparisons of ISEA-chesc and ISEA-adaptive on CHeSC 2011 test suite

algorithm		Total	SAT	BP	PS	FS	TSP	VRP
ISEA-chesc	rank	8	9	2	5	11	7	13
	points	71.0	6.0	30.0	14.5	3.5	12.0	5.0
ISEA-adaptive	rank	**2**	14	**1**	**4**	**2-3**	**3**	**3**
	points	**145.75**	0.25	**42.0**	**22.5**	**34.0**	**24.0**	**23.0**

The same configuration was used for the ISEA-adaptive with one difference – parameters T_1 and T_2 were replaced with the parameters used for adapting the probability P_{reinit}, see Fig. 2 and accompanying text.

4.2 Results

Results are summarized in Table 1 and Table 2. The main observation is that ISEA-adaptive significantly improved the final score over ISEA-chesc. ISEA-adaptive was ranked as the second best hyper-heuristic with the total score of 141 points among all competition hyper-heuristics while ISEA-chesc placed at eighth position with the total number of 71 points, see Table 1[4]. We can see that ISEA-adaptive outperformed ISEA-chesc on all problems but the Max-SAT one.

Table 2. Results obtained by ISEA-chesc and ISEA-adaptive on CHeSC 2011 test suite. Median values from 31 final objective values per instance are presented. Statistically significant improvements of the ISEA-adaptive over the ISEA-chesc are emphasized in bold as confirmed by the Wilcoxon rank sum test at the 1% level.

problem	algorithm	instance 1	2	3	4	5
SAT	ISEA-chesc	5	11	4	9	11
	ISEA-adaptive	7	12	5	11	10
BP	ISEA-chesc	0.034223	0.003284	0.003655	0.108625	0.006400
	ISEA-adaptive	0.034025	**0.002936**	**0.001669**	**0.108386**	**0.002487**
PS	ISEA-chesc	20	9966	3308	1660	315
	ISEA-adaptive	20	9841	3274	**1587**	315
FS	ISEA-chesc	6262	26844	6366	11419	26663
	ISEA-adaptive	**6242**	**26811**	**6325**	**11364**	**26636**
TSP	ISEA-chesc	48194.9	20868203.1	6832.6	67282.1	54129.2
	ISEA-adaptive	**48194.9**	**20827231.2**	6832.8	**66983.3**	55248.6
VRP	ISEA-chesc	70471.7	13339.8	149149.6	20657.2	150474.0
	ISEA-adaptive	67582.3	13338.1	**145280.0**	**20653.8**	**148476.1**

[4] Recall, that the results presented in Table 1 come from two separate competitions.

The same trend can be seen in Table 2, where ISEA-adaptive outperformed ISEA-chesc on most of the competition instances with exception of the Max-SAT ones. On BP, PS, FS, TSP and VRP problems we can observe significant improvement of the median best value and corresponding score gain.

The results are very promising and indicate that a proper re-initialization scheme is crucial for optimal performance of the algorithm. It shows that the scheme with adapted re-initialization probability is better suited for this kind of iterated local search algorithm than the rather hard-wired re-initialization strategy that is dependent on properly set user-defined parameters T_1 and T_2.

The question is why ISEA-adaptive performed almost consistently worse than ISEA-chesc on SAT instances. One reason for this observation could be that the time-intervals T_{exp} and T_{min} (and subsequently the P_{accept}), derived based on the expected number of non-improving LLH sequences, were too short so that there was not enough time to converge to a local optimum before new re-initialization was carried out.

5 Conclusions

This paper presented two versions of an evolutionary-based iterative local search hyper-heuristic called Iterated Search Driven by Evolutionary Algorithm Hyper-Heuristic (ISEA). The two versions differ in the re-initialization scheme. The first one, ISEA-chesc, relies on strictly defined time-based conditions that if fulfilled than the re-initialization of the current evaluation solution takes place. The second one, ISEA-adaptive, uses a probability P_{reinit} to determine whether the evaluation solution will be re-initialized or not.

The performance of the two algorithms was experimentally evaluated on six hard optimization problems using the HyFlex experimental framework [4] and the two algorithms were compared using the results of the algorithms that took part in the CHeSC 2011 challenge [10].

The achieved results are very promising, the ISEA-adaptive variant would place second in the CHeSC 2011 final competition. It also shows that the re-initialization scheme used in ISEA-adaptive is better suited for this optimization algorithm as the ISEA-chesc placed on eighth place and attained almost consistently worse results than ISEA-adaptive.

There are several directions for future research:

- In order to avoid cycles in the search process we plan to incorporate a Tabu list containing several recent *evaluation solutions*. Any newly generated solution that appears in the Tabu list must not be accepted as the new *evaluation solution*.
- Another subject of research is how to implement the perturb operator so that it enables sampling regions of attraction of local optima in the vicinity of the current local optimum. We would like to investigate a potential of adaptive perturb operators.
- Third main direction of our research is towards utilization of information gathered during the optimization process such as which low level heuristics

or their combinations appeared frequently in improving LLH sequences and in which situations. Such information can be used for example in biased sampling of low level heuristics to the LLH sequences.

Acknowledgement. The work presented in this paper has been supported by the research program No. MSM 6840770038 "Decision Making and Control for Manufacturing III" of the CTU in Prague. I would like to thank Dr. Gabriela Ochoa for kindly providing me with complete outcome data of the ISEA hyper-heuristic calculated for the CHeSC 2011 competition.

References

1. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: Exploring hyper-heuristic methodologies with GP. Artificial Evolution 1, 177–201 (2009)
2. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A Classification of Hyper-heuristic Approaches. In: Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 146, pp. 449–468 (2010)
3. Burke, E.K., Curtois, T., Hyde, M.R., Kendall, G., Ochoa, G., Petrovic, S., Vázquez Rodríguez, J.A., Gendreau, M.: Iterated Local Search vs. Hyper-heuristics: Towards General-Purpose Search Algorithms. In: IEEE Congress on Evolutionary Computation CEC 2010, pp. 1–8 (2010)
4. Burke, E., Curtois, T., Hyde, M., Ochoa, G., Vazquez-Rodriguez, J.A.: HyFlex: A Benchmark Framework for Cross-domain Heuristic Search, ArXiv e-prints, arXiv:1107.5462v1 (July 2011)
5. Garrido, P., Riff, M.C.: DVRP: A Hard Dynamic Combinatorial Optimisation Problem Tackled by an Evolutionary Hyper-Heuristic. Journal of Heuristics 16(6), 795–834 (2010)
6. Kubalik, J., Faigl, J.: Iterative Prototype Optimisation with Evolved Improvement Steps. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 154–165. Springer, Heidelberg (2006)
7. Kubalik, J.: Solving the Sorting Network Problem Using Iterative Optimization with Evolved Hypermutations. In: Genetic and Evolutionary Computation Conference 2009 (CD-ROM), pp. 301–308. ACM, New York (2009)
8. Kubalik, J.: Efficient stochastic local search algorithm for solving the shortest common supersequence problem. In: Proceedings of the 12th Genetic and Evolutionary Computation Conference, pp. 249–256. ACM, New York (2010) ISBN 978-1-4503-0073-5
9. Luke, S.: Essentials of Metaheuristics. Lulu (2009), http://cs.gmu.edu/~sean/book/metaheuristics/
10. The results of the first Cross-domain Heuristic Search Challenge, CHeSC (2011), http://www.asap.cs.nott.ac.uk/chesc2011/index.html

Intensification/Diversification-Driven ILS for a Graph Coloring Problem

Samir Loudni

Université de Caen Basse-Normandie,
UMR 6072 GREYC, F-14032 Caen, France

Abstract. This paper presents an extension of the ILS algorithm, called `ID-ILS`, by introducing new local search devices that enforce an efficient tradeoff of intensification and diversification. Experiments performed on the DIMACS benchmarks show that our method is competitive with the best coloring algorithms.

1 Introduction

The *Graph Coloring Problem* (GCP) is to find the minimum number of colors required to color the vertices of a graph so that no edge has both endpoints with the same color. The GCP has received much attention in the literature, not only for its direct applications to many other real world problems [1,2], but also for its difficulty from complexity point of view. In fact, although many exact algorithms have been proposed for this problem (see [3]), such algorithms can only be used to solve small instances (up to 100 vertices). Therefore, heuristic algorithms are needed for larger instances. The best performing heuristic algorithms are local search methods (e.g., [4,5,6,7]) and hybrid algorithms that combines a local search with a population based method (e.g. [8,9,10,11]).

Iterated local search (ILS) [12] is a simple and effective type of metaheuristic that has been successfully applied on a wide range of problems. The basic principle of ILS consists in successively applying perturbations and local search to the current solution. The perturbation step plays a primary role because it drives ILS to explore different regions of the search space, in order to escape from the basin of attraction of the most recently visited local optima (*diversification effort*), while the goal of local search step is to focus more intensively within each promising region to converge towards a local optimum (*intensification effort*). However, as explained in [13], most LS algorithms handle diversity and intensity as two opposite objectives : as one gets more intensity, one can lose diversity. So, more coordination/balance is required between these two main objectives. The `Aspiration Plus` CLS (*Candidate List Strategy*) [14,15] is a promising mechanism proposed for the Tabu Search. It restricts the number of neighbors to examine for the next move, in order to control the intensification effort.

The goal of this work is to propose a new extension of the ILS algorithm, noted `ID-ILS`, by introducing in both steps of ILS new local search devices that enforce an efficient tradeoff of intensification and diversification. We performed

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 160–171, 2012.

experiments on a set of challenging DIMACS graphs [16], and we have compared our results to six local search methods, as well as to three hybrid evolutionary algorithms HCA [9], MACOL [10] and MMT [11]. These results show that, our method clearly dominates the local search approaches and is competitive compared to the hybrid ones. Section 2 gives a synthetic overview of the GCP and presents the best performing algorithms for solving it. Section 3 describes our resolution approach ID-ILS and details their main components. Section 4 is devoted to experimentations. Finally, we conclude and draw some perspectives.

2 Graph Coloring Problem

2.1 Definitions and Notations

Given a graph $G = (V, E)$ with vertex set V and edge set E, and given an integer k, a k-coloring of G is a function $c : V \rightarrow \{1, \ldots k\}$. The value $c(u)$ of a vertex u is called the color of u. An edge $(u, v) \in E$ is said *conflicting* if its vertices u and v have the same color. A k-coloring without conflicting edges is said *legal*, otherwise it is *illegal*. Let $s = [c(1), \ldots, c(|V|)]$ be a legal k-coloring, s can be represented by a partition of V into k disjoint subsets V^1, ..., V^k. We say that V^r is the *color class* r induced by s (i.e., the set of vertices having a color r in s). The objective function f counts the number of conflicting edges induced by s. The GCP is to determine the chromatic number $\chi(G)$ of G, i.e. the minimum value of k for which there is a k-coloring s of G such that $f(s) = 0$.

2.2 Metaheuristic Approaches to the GCP

TABUCOL [5] is one of the most famous local search algorithms proposed for the GCP. Morgenstern [7] proposed a complex algorithm, called MOR, based on *partial k-colorings*. A solution is a partition of vertices of G into k disjoint color classes $\{V^1, \ldots, V^k\}$. A specific class (i.e., V^{k+1}) is used to represent the set of uncolored vertices. A neighbor solution is obtained by moving an uncolored vertex u from V^{k+1} to a pre-existing color class V^h, and by moving to V^{k+1} all vertices in V^h that are adjacent to u. The complete legal k-coloring is obtained by emptying V^{k+1}.

In [9], an evolutionary algorithm, called HCA, combining an improved version of TABUCOL with a *Greedy Partitioning Crossover* operator (GPX) was proposed. GPX builds a partial legal k-coloring $\{V^1, \ldots, V^k\}$ by alternatively selecting from two parent solutions the class of maximum size to become color class V^i of the offspring. All vertices in this color class are then deleted from the parents. The remaining vertices are then assigned to a class randomly chosen. In [10], a similar approach was proposed, called MACOL, which extends GPX to use more than two parents for generating color classes of the offspring.

In [4], two TS methods (DYN-P.COL and FOO-P.COL), based on *partial k-colorings* were proposed. Finally, Hertz et al. [6] proposed an extension of the VNS algorithm, called *Variable Search Space* (VSS). The idea of VSS is to completely change the search space and to consider different objective functions for each space. They proposed VSS-Col, which moves between three different search spaces.

Algorithm 1. Pseudo-code algorithm for `ID-ILS`

```
   function ID-ILS(maxIter, maxneigh, maxMoves, nextneigh) ;
   begin
1      s ← genRandomSol(), i ← 1, b ← b_max;
2      while (i ≤ maxIter) do
3          i ← i + 1;
4          s' ← Perturbation(s, b) ;
5          s' ← LS(s', maxneigh, maxMoves, nextneigh) ;
6          if f(s') < f(s) then
7              s ← s', i ← 1;
8              b ← b_max;
9          else b ← updatePertubationSize(i);
10     return s;
   end
```

3 Intensification/Diversification-Driven ILS

3.1 Main Scheme of `ID-ILS`

`ID-ILS` extends `ILS` [12] by introducing new local search devices that enforce an efficient tradeoff of intensification and diversification. To achieve this goal, we first define an adaptive scheme to control the size for the perturbation (cf. Sect 3.2). Second, we make use a *candidate list strategy*, endowed with a diversification mechanism to exit from local minima (cf. Sect. 3.3). Algorithm 1 presents its pseudo-code. We denote by b_{max} the maximum perturbation size. It starts from an initial solution s which is randomly generated. The loop in lines 2 to 9 is performed until i number of consecutive iterations performed without improving s reaches `maxIter`. A new local optimum s' is obtained by the combination of a perturbation move of size b applied to the current solution s (line 4) with a local search procedure applied to the so obtained perturbed solution (line 5). If s' is better than s, it becomes the new current solution and i is reset to 1 (line 7);

3.2 Perturbation Step

Adaptive Perturbation. As explained in [12], the perturbation is just a collection of moves that complement those carried out by the local search. A weak perturbation is likely to get rapidly stuck in a deep local optima, whereas a strong perturbation is prone to be slow in convergence and similar to a randomized search. Achieving such a delicate balance is a challenge and certainly it is the key to success in ILS. We propose to exploit the search history to determine the perturbation size (i.e., b). In our approach, b proportionally decreases according to the value of i (see Algorithm 1). The main idea is to perform large perturbations each time the current solution s is improved, and to favor gradually small perturbations when s has not been improved for a long time. In fact, in our experiments we observed that the space of solutions has very distant solutions

that are nearly as good as the optimum. So, after getting a best coloring in one's neighbor solution by the local search procedure (i.e., intensification effort), one must go explore other regions of locally optimal solutions. This is achieved by using *large* perturbations (i.e., diversification effort). Initially, b is set to b_{max}. During the search, each time s is improved b is reset to b_{max} (line 8); otherwise it is decreased whenever i is increased, until reaching the value b_{min} (line 9).

Perturbation Schemes. Our perturbation operator consists of changing the color of some *conflicting vertices* in s. Let us note by $neighbors(u)$ the set of all vertices adjacent to u and by $\mathcal{X}(s)$ the set of conflicting vertices in s. We randomly select a first vertex v_O in $\mathcal{X}(s)$ and move its original color to the best possible other one (i.e. the new color is chosen among those producing the smallest number of conflicts). Let s_1 be the new perturbed solution. If s_1 increases the number of conflicts, we randomly select a new vertex among conflicted ones in $\mathcal{X}(s_1) \setminus \mathcal{X}(s)$ and assign to it the best possible new color. This process is repeated until a non-deteriorating move (i.e., that does not increase the number of conflicts) is found. In this way, we only accept moves that deacrese as small as possible the solution quality. To favor the diversification capability, we prevent changing the color of a vertex more than once. This sequence of moves are successively applied with b different v_O. This perturbation operator is noted *ConflictVar* (P_1). This operator, which is based on random choices, will change the current solution in an unpredictable way. The result will be, most likely, a worse solution, and many times, much worse. We propose to exploit information from the topology of the constraints graph to guide the perturbation operator towards more promising regions [17,18]. We propose new perturbations :

a) *ConflictVar Chain* (P_2): We first randomly select an initial vertex v_{init} in $\mathcal{X}(s)$ and move it into the best possible other color class V^i. Let s_1 be the new perturbed solution. If s_1 increases the number of conflicts, we randomly select a new vertex u among conflicted ones in $(\mathcal{X}(s_1) \setminus \mathcal{X}(s)) \cap V^i$ and assign to it the best possible new color class V^j. This sequence of moves is achieved until a non-deteriorating move is found. This process is repeated by successively applying such sequences of changes with b different v_{init}.

b) *ConflictVar Connected Centers* (P_3): We randomly select a first vertex v_{cc} noted "connected center" in $\mathcal{X}(s)$ and move it into the best possible other color class V^i. Then, for each conflicting vertex v_c in $neighbors(v_{cc}) \cap V^i$, we move it into the best possible other color class V^j, and we assign the best possible color to every new conflicting vertex in $neighbors(v_c) \cap V^j$. This sequence of moves are successively applied with b different v_{cc}.

c) *Conflict ColorClass* (P_4): We first select a color class V^i having the highest number of conflicting vertices (ties are randomly broken) and move each of its vertices into the best possible other color class. Let s_1 be the new perturbed solution. Then, we select randomly b new conflicting vertices from $\mathcal{X}(s_1) \setminus \mathcal{X}(s)$ and move each of them into a best possible other color. A *tabu list* is used to forbid selecting the color class V^i for the next l iterations of `ID-ILS`, with $l = 0.6 \times |\mathcal{X}(s)| + rand(0, 9)$ [9], where $rand(0, 9)$ is a function providing a random number in $\{0, \ldots, 9\}$.

3.3 Local Search Step

Our local search procedure is an extension of `TABUCOL`. Algorithm 2 shows its pseudo-code. We use a neighborhood defined by the 1-flip move, which consists of changing the original color class $V^{c(v)}$ of a single conflicting vertex v to its *best* possible new color class V^i ($c(v) \neq i$). (s, v, i) will denote this move (lines 7 to 8). Once a move is performed, vertex v is forbidden to move back to its previous color $c(v)$ for the next T iterations (line 19). After preliminary experiments, the *tabu tenure* T was fixed to 20. At each iteration, it determines the best neighbor s' of the current solution s such that either s' is a non-tabu solution or $f(s') < f(s^*)$, where s^* is the best solution found so. Our stopping condition is based on a total number of iterations (`stopIter`). Initially, `stopIter` = `maxMoves` (line 1). Each time a best solution is found, `stopIter` is increased by the current number of iterations performed (lines 20 to 21).

The most critical part for local search methods concerns the neighborhood exploration, and more exactly: (i) the number of candidate neighbors to visit, and (ii) the way of selecting the next move among these candidates. Indeed, the number of candidates should be large enough to focus more intensively on regions found to be good. However, it should be small enough to prevent examining a large set of candidates and thus to allow the search to exit from local optima more quickly. Moreover, the way of selecting the next move should drive the search towards unexplored regions in the solution space. To address these issues, we make use a *candidate list strategy* (CLS) to manage the neighborhood exploration. Two parameters are defined : (a) `maxneigh` which is the maximum number of candidate neighbors studied in every move. This CLS manages the `maxneigh` candidates so as to obtain a good tradeoff between intensification and diversification efforts; and (b) `nextneigh` which is a diversification devise to jump out of local minima. Two variants perform this diversification process. In the first variant, `ID-ILS(first)`, where `nextneigh` is set to `first`, the first neighbor among the `maxneigh` non-accepted candidates is selected (i.e., `s_first`). In the second variant, `ID-ILS(best)`, where `nextneigh` is set to `best`, the best neighbor with a lowest cost among the `maxneigh` non-accepted candidates is selected (i.e., `s_best`). The loop in lines 6 to 16 is performed until a better solution s' which improves s^* is obtained or no improvement has been made after `maxneigh` iterations. According to the value of `nextneigh`, the next neighbor solution is selected in lines 17 and 18 from the rejected candidates.

4 Experimental Results

In this section, we report experimental results over different graph types from the DIMACS benchmarks. For some very difficult graphs, we considered a set of k-coloring instances for different values of k. As experiments have been run on various machines, we will report (when it is possible), normalized[1] CPU times.

[1] For a machine κ times slower than ours, reported CPU times will be divided by κ.

Algorithm 2. Pseudo-code algorithm for LS

```
   function LS (s, maxneigh, maxMoves, nextneigh) ;
   begin
1      stopIter ← maxMoves;
2      s_first ← ∅,  s_best ← ∅,  f(s_best) ← ⊤,  i ← 1,  s* ← s;
3      while (i ≤ stopIter) do
4          i ← i + 1,  nbtries ← 1;
5          firstfound ← false, done ← false;
6          while (nbtries ≤ maxneigh) and not(done) do
7              v ← randomConflictVertex(s);
8              s' ← getNeighbor(s, v);
9              if not(firstfound) then
10                 firstfound ← true,  s_first ← s';
11             if (not(done) and (v, s'[v]) ∉ tabulist) or (f(s') < f(s*)) then
12                 s ← s',  done ← true;
13             else
14                 if not(done) and (nextneigh = best) and (f(s') < f(s_best))
                   then
15                     s_best ← s' ;
16             nbtries ← nbtries + 1;
17         if not(done) and (nextneigh = first) then s ← s_first ;
18         if not(done) and (nextneigh = best) then s ← s_best ;
19         insert (v, c(v)) in tabulist and make (v, c(v)) tabu for T iterations;
20         if f(s) < f(s*) then
21             stopIter ← i + maxMoves,  s* ← s;
22     return s*
   end
```

4.1 Problem Instances and Experimental Protocol

We experimented our algorithm on the following difficult graphs [16]:

- **8 DSJC$n.y$ graphs:** DSJCs are random graphs with n vertices and a density equal to $0.y$. We selected those with $n \in \{250, 500, 1000\}$ and $y \in \{1, 5, 9\}$.
- **2 DSJR$n.r$ graphs:** DSJRs are geometric random graphs. We selected those with $n = 500$ and $r \in \{1, 5\}$.
- **5 flatn_x_0 graphs:** flat graphs are quasi-random graphs. We selected the $flat300\_x\_0$, with $x \in \{26, 28\}$ and the $flat1000\_x\_0$, with $x \in \{50, 60, 76\}$.
- **4 len_x graphs:** the Leighton graphs are derived from scheduling, and have 450 vertices. We selected instances c and d, with $x \in \{15, 25\}$.
- **one latin square** graph ($latin\_square\_10$).

Based on preliminary testing, we used the following parameter settings: maxIter=$2*n$, maxMoves=200,000, maxneigh$\in\{50, 100, 150, 175, 200\}$, b_{min}=20,

and b_{max}=50 (except for P_1, where b_{max}=100). A set of 20 (or 10) runs per k-coloring instance has been performed on a 2GHz Intel Core 2 DUO with 2GB of RAM. We report the value of k for which a k-coloring was found, the number of successful runs ("succ. runs/total runs"), the average CPU time in seconds for successful runs and the average cost over the total runs. `ID-ILS` has been implemented in C++.

Table 1. Comparing the different perturbation schemes. The best results are in bold.

Instance	k	P_i	MaxN.	Succ.	Time	Avg
DSJC250.5	28	P_1	200	18/20	113.5	0.1
n=250		P_2	175	18/20	148.1	0.1
m=15668		P_3	150	20/20	118.9	0
k^*=28		P_4	**150**	**20/20**	**76.4**	**0**
DSJC250.9	72	P_1	50	20/20	8.5	0
n=250		P_2	100	20/20	9.8	0
m=27897		P_3	**50**	**20/20**	**8.4**	**0**
k^*=72		P_4	50	20/20	10.86	0
DSJC500.1	12	P_1	175	17/20	142.4	0.15
n=500		P_2	175	17/20	448.9	0.2
m=24916		P_3	100	13/20	303.4	0.5
k^*=12		P_4	**200**	**20/20**	**121.9**	**0**
DSJC500.5	48	P_1	200	1/10	2077.6	1.5
n=500		P_2	**100**	**2/10**	**4762.6**	**1.2**
m=125248		P_3	100	2/10	6445.6	1.6
k^*=48		P_4	50	1/10	2820.6	2.1
DSJC500.9	126	P_1	**150**	**18/20**	**2451**	**0.15**
n=500		P_2	50	11/20	3513	0.45
m=224874		P_3	175	14/20	5348.8	0.3
k^*=126		P_4	175	12/20	6093.4	0.4
DSJR500.1c	85	P_1	150	0/10	-	2
n=500		P_2	200	0/10	-	2
m=121275		P_3	50	1/10	152.2	1.8
k^*=85		P_4	**50**	**10/10**	**1713.5**	**0**
DSJR500.5	125	P_1	150	0/10	-	2.7
n=500		P_2	100	0/10	-	3.1
m=58862		P_3	100	0/10	-	1.4
k^*=122		P_4	**100**	**9/10**	**2702.6**	**0.15**
	124	P_1	175	0/10	-	3.6
		P_2	100	0/10	-	3.9
		P_3	175	0/10	-	1.8
		P_4	**175**	**1/10**	**297.1**	**1.4**
DSJC1000.1	21	P_1	50	20/20	3.67	0
n=1000		P_2	50	20/20	3.1	0
m=49629		P_3	50	20/20	3.3	0
k^*=20		P_4	**50**	**20/20**	**3.08**	**0**
DSJC1000.5	88	P_1	150	20/20	1546.4	0
n=1000		P_2	50	20/20	1322.9	0
m=499652		P_3	**100**	**20/20**	**852.6**	**0**
k^*=83		P_4	50	20/20	1116.1	0
	86	P_1	150	0/10	-	5.5
		P_2	**100**	**2/10**	**4998.6**	**1.7**
		P_3	50	0/10	-	19.3
		P_4	175	2/10	27,071.2	4.8
le450_15c	15	P_1	25	20/20	0.6	0
n=450		P_2	25	20/20	0.6	0
m=16680		P_3	25	20/20	0.4	0
k^*=**15**		P_4	**25**	**20/20**	**0.3**	**0**

Instance	k	P_i	MaxN.	Succ.	Time	Avg
le450_15d	15	P_1	150	20/20	2.9	0
n=450		P_2	50	20/20	6.4	0
m=16750		P_3	150	19/20	6.7	0.05
k^*=**15**		P_4	50	20/20	3.2	0
le450_25c	26	P_1	100	7/25	19.9	0.8
n=450		P_2	**100**	**25/25**	**10**	**0**
m=17343		P_3	150	18/25	12	0.32
k^*=**25**		P_4	100	12/25	19.4	0.68
le450_25d	26	P_1	100	11/20	61	0.56
n=450		P_2	**100**	**20/20**	**11.3**	**0**
m=17425		P_3	100	17/20	9	0.4
k^*=**25**		P_4	150	7/20	17.2	0.88
flat300_26_0	26	P_1	50	20/20	4.6	0
n=300		P_2	50	20/20	4.2	0
m=21633		P_3	**50**	**20/20**	**3.2**	**0**
k^*=**26**		P_4	50	20/20	10.3	0
	31	P_1	150	15/20	192	0.25
		P_2	150	17/20	160.9	0.15
		P_3	175	17/20	239.1	0.15
		P_4	**200**	**20/20**	**185**	**0**
Flat300_28_0	30	P_1	150	10/20	1399.43	1.7
n=300		P_2	**100**	**10/20**	**1273.6**	**2**
m=21695		P_3	150	9/20	1932.9	2
k^*=**28**		P_4	150	8/20	1344.8	2.2
	29	P_1	50	2/20	547.3	9.35
		P_2	200	4/20	2075.8	8.3
		P_3	**200**	**4/20**	**1612.7**	**8.2**
		P_4	200	3/20	1190.9	8.75
flat1000_50	50	P_1	50	0/20	2379.8	-
n=1000		P_2	50	0/20	2305.1	-
m=224874		P_3	50	0/20	1977.4	-
k^*=**50**		P_4	**50**	**20/20**	**2858.1**	**0**
flat1000_60	60	P_1	50	0/20	-	-
n=1000		P_2	50	0/20	-	-
m=245830		P_3	50	0/20	-	-
k^*=**60**		P_4	**50**	**20/20**	**13,854**	**0**
flat1000_76_0	86	P_1	100	2/20	5750.6	**1.77**
n=1000		P_2	100	1/20	29,765	2.8
m=246708		P_3	150	0/20	-	9.88
k^*=82		P_4	100	**17/20**	**20,579**	0.15
DSJC1000.9	224	P_1	175	8/10	31,461	0.2
n=1000		P_2	50	5/10	13,384	0.7
m=449449		P_3	50	3/10	20,671	1.2
k^*=223		P_4	**150**	**9/10**	**32,598**	**0.1**
latin_square	100	P_1	50	0/20	-	3.35
n=900		P_2	50	0/20	-	3.35
m=307350		P_3	50	0/20	-	3.75
k^*=98		P_4	**100**	**15/20**	**12,812.1**	**0.3**

4.2 Comparing the Different Perturbation Schemes

Our first experiment aims to evaluate the effectiveness of our perturbation schemes. Table 1 reports the detailed results of `ID-ILS(first)`. Column 1 gives the features of each instance: its name, the number of vertices (n), the number of edges (m) and the value of the best known coloring (k^*) (in bold when it is the proven optimal value). Column 3 denotes the different perturbations (P_i). Column 4 reports the best setting for `maxneigh`. A score (b-s-w) is assigned to each P_i, corresponding to the number of k-colorings for which P_i gets

Table 2. Comparison among ID/TS and VSS-Col. Best results are in bold. Column P_2 (resp. P_4) refers to the results obtained by ID-ILS(first) with P_2 (resp. P_4).

Instance	k^*	k	ID-ILS(first+P_4)		ID-ILS(first+P_2)		VSS-Col		ID/TS		
			Succ.	Time	Succ.	Time	Succ.	Time	k_{best}	Time	Avg.
DSJC250.5	28	28	20/20	76	18/20	148	-	-	28 (1)	1241	0.8
DSJC250.9	72	72	20/20	11	20/20	10	-	-	72 (5)	15	0
DSJC500.1	12	12	20/20	122	17/20	449	10/10	97	12 (5)	1465	0
DSJC500.5	48	48	1/10	2820	2/10	4762	**3/10**	1331	50 (3)	2378	0.4
		49	12/20	1894	14/20	889	**10/10**	162			
DSJC500.9	126	126	12/20	6094	11/20	3513	**8/10**	1686	127 (1)	3435	1
		127	20/20	194	20/20	173	10/10	169			
DSJR500.1c	85	85	**10/10**	1713	0/10	-	**9/10**	736	85 (0)	-	1.4
DSJR500.5	122	124	**1/10**	297	0/10	-	0/10	-	125 (0)	-	3.4
		125	**9/10**	2702	0/10	-	0/10	-			
DSJC1000.1	20	20	0/20	-	0/20	-	**3/10**	2396	21 (5)	4	0
		21	20/20	3	20/20	3	10/10	11			
DSJC1000.5	83	86	**2/10**	27,071	**2/10**	4998	0/10	-	90 (1)	2711	1.2
		88	20/20	1116	20/20	1323	8/10	2028			
DSJC1000.9	223	224	**9/10**	32,598	**5/10**	13,384	1/10	3326	228 (1)	5707	1.2
		225	**20/20**	20,816	**20/20**	1546	5/10	1484			
le450_15c	15	15	20/20	0	20/20	0	10/10	6	15 (5)	3	0
le450_15d	15	15	20/20	3	20/20	6	10/10	44	15 (5)	5	0
le450_25c	25	26	12/25	19	**25/25**	10	10/10	1	26 (3)	6	0.4
le450_25d	25	26	7/20	17	**20/20**	11	10/10	1	26 (1)	279	0.8
flat300_28_0	28	29	**3/20**	1191	**4/20**	2075	1/10	867	31 (1)	486	1.4
		30	**8/20**	1344	**10/20**	1273	2/10	2666			
		31	**20/20**	185	17/20	160	10/10	39			
flat1000_50_0	50	50	**20/20**	2858	0/20	-	10/10	318	60 (0)	-	484.4
flat1000_60_0	60	60	**20/20**	13,854	0/20	-	10/10	694	70 (0)	-	223.8
flat1000_76_0	82	86	**17/20**	20,579	1/20	29,765	0/10	-	90 (5)	1190	0
		87	**20/20**	1204	**20/20**	1780	4/6	1689			
		88	NA	NA	NA	NA	10/10	1155			
			P_2		P_4		P_4	P_2	P_4	P_2	
Better			**4**		0		**3**	2	**10**	8	
Equal			**14**		14		**12**	9	8	8	
Worse			**0**		4		**1**	4	**0**	0	

respectively better (2^{nd} better in parentheses), equal (100%) and worse success rates than the other perturbations. From Table 1, the following remarks are drown :

- Perturbations based on the topology of the constraints graph (except P_3) are clearly more relevant. Both P_1 (*score*: 2(5)-6-10) and P_3 (*score*: 3(6)-6-9) perform similarly, with a slight advantage to P_3. This can be explained by the fact that P_3 performs perturbations only on a very limited part of the graph, whereas the random character of P_1 allows more diversification in the perturbation step, which helps to find better solutions.
- P_2 (*score*: 6(3)-6-7) and P_4 (*score*: 12(2)-6-4) outperform P_3. Both perturbations find solutions with better success rates respectively for 6 and 12 coloring instances among 24, whereas P_3 obtains best success rates for 3 coloring instances. Indeed, perturbations P_2 and P_4 allow a more "aggressive" diversification by performing perturbations on different connected subparts of the graph.
- Finally, P_4 clearly dominates P_2: P_2 obtains better success rates on 5 k-coloring instances while P_4 outperforms P_2 on 11 k-coloring instances.

4.3 Comparison with Two Local Search Methods

We have compared ID-ILS(first) using the two best perturbations P_2 and P_4, with two local search methods:

(i) ID/TS, a variant of TS endowed with our CLS. We made experiments with nextneigh set to first and maxneigh $\in \{50, 100, 150, 175, 200\}$. For each value of k, and each trial, ID/TS is run 20 times with maxMoves set to $1,500,000$. If no legal k-coloring is found, then it is run 10 times with maxMoves equal to $5,000,000$. A set of 5 trials per k-coloring is performed.
(ii) VSS-Col which is one of the most performing among local search coloring algorithms [6]. The reason for comparing ID-ILS with VSS-Col is that both methods are very close. However, VSS-Col considers different search spaces, each one being associated with a set of neighborhoods.

Table 2 compares performances of the four methods. Results for VSS-Col are taken from [6] and correspond to those obtained with a time limit of 1h on a 2 GHz Pentium 4, with 512 MB of RAM. For ID/TS, we report the best value of k (k_{best}) found, the number of successful runs (in parentheses), the average CPU time in seconds for successful runs and the average cost among the five trials. The last three rows show the summary of the comparisons. The rows *better*, *equal* and *worse* gives respectively the number of graphs for which our method gets better, equal and worse colorings than the other algorithms.

ID-ILS(first+P_4) is clearly the best one. From these results, we observe that the two variant of ID-ILS(first) outperform ID/TS, particularly on large graphs, where better colorings have been found on at least eight large graphs. For the two flat1000_50&60, ID/TS was not able to find a legal coloring even for high values of k. On the eight remaining graphs, the two methods find solutions of the same quality, but ID-ILS(first) provides better success rates.

When comparing with the results of VSS-Col, ID-ILS(first+P_4) gets better solutions on three graphs, with colorings using respectively 2, 2 and 1 less colors, and it is worse on one graph. Both methods obtain the same colorings on 12 graphs. However, if we compare the success rates, ID-ILS(first+P_4) performs better than VSS-Col on three graphs. Both algorithms find the same colorings, with the same success rate on five graphs, but VSS-Col is generally faster, except for le450_15c&15d, where ID-ILS(first+P_4) find optimal colorings very quickly. So, ID-ILS(first+P_4) can be considered as more effective than VSS-Col.

4.4 Comparison with the Most Effective Algorithms

In this section we compare our method with the most performing algorithms for the GCP: four local search algorithms (MOR [7], ILS [19] and DYN/FOO-P.COL [4]) and three hybrid evolutionary methods (HCA [9], MACOL [10] and MMT [11]). However, we do not report the CPU times because the conditions of experimentation are not equivalent. So, comparisons must be done with care. Results are given so that the reader may have a baseline by which he may evaluate ID-ILS.

If we compare the results of ID-ILS(first+P_4) with those of local search methods, one easily observes that our method clearly dominates these local search algorithms (see last three rows of Table 3). Indeed, our method obtains worse results for **at most three graphs** while better results are obtained for **at least six graphs**, except for ILS, where our approach obtains better results

Table 3. Comparison with the state-of-the-art algorithms

Instance	MOR	ILS	HCA		FOO-P.COL		DYN-P.COL		MACOL		MMT	ID-ILS	
	k_{best}	k_{best}	Succ.	k	Succ.	k	Succ.	k	Succ.	k	k_{best}	Succ.	k
DSJC250.5	28	28	9/10	28					20/20	8	28	20/20	28
DSJC250.9	-	-							10/10	72	72	20/20	72
DSJC500.1	12	12			23/50	12	50/50	12	20/20	12	12	20/20	12
DSJC500.5	**49**	**49**	5/10	48	50/50	**50**	1/50	**49**	20/20	48	48	1/10	48
DSJC500.9	**128**	126			48/50	**128**	1/50	127	20/20	126	**127**	12/20	126
DSJR500.1c	85	-			50/50	85	3/50	85	20/20	85	85	10/10	85
DSJR500.5	*123*	124			24/50	**128**	28/50	**126**	11/20	*122*	*122*	1/10	124
DSJC1000.1	21	-	-	*20*	50/50	21	3/50	*20*	20/20	20	*20*	20/20	21
DSJC1000.5	**88**	**89**	-	*83*	5/50	**89**	6/50	**89**	20/20	*83*	*84*	2/10	86
DSJC1000.9	**226**	-	-	224	30/50	**228**	30/50	**228**	18/20	*223*	**225**	9/10	224
le450_15c	15	15	6/10	15	50/50	15	50/50	15	20/20	15	15	20/20	15
le450_15d	15	15			50/50	15	50/50	15	20/20	15	15	20/20	15
le450_25c	*25*	26	10/10	26	50/50	**27**	50/50	**27**	20/20	*25*	*25*	12/20	26
le450_25d	*25*	26			50/50	**27**	50/50	**27**	20/20	*25*	*25*	7/20	26
flat300_26_0	26	26							20/20	26	26	20/20	26
flat300_28_0	**31**	**31**	6/10	**31**	35/50	*28*	13/50	*28*	15/20	29	**31**	3/20	29
flat1000_50_0	50	-			50/50	50	50/50	50	20/20	50	50	20/20	50
flat1000_60_0	60	-			50/50	60	50/50	60	20/20	60	60	20/20	60
flat1000_76_0	**89**	-	4/5	*83*	10/50	**88**	9/50	**88**	20/20	*82*	*83*	17/20	86
latin_square	-	*99*							5/20	*99*	**101**	15/20	100
Better	**6**	**3**		**1**		**8**		**8**		**0**	**4**		
Equal	9	9		5		7		6		12	10		
Worse	**3**	**1**		**3**		**1**		**2**		**8**	**6**		

for **three graphs**. For DSJC500.9 (resp. DSJC1000.9), ID-ILS(first+P_4), ILS and VSS-Col are the only algorithms that can reach 126-coloring. Detailed comparisons are given bellow:

- ID-ILS(first+P_4) is better than MOR on six graphs and worse on three graphs (DSJR500.5, le450_25c and le450_25d).
- ID-ILS(first+P_4) is better than ILS on three graphs and worse on one graph (latin_square). This comparison is very informative as well and shows the importance of our perturbation scheme P_4 as well as of the CLS.
- ID-ILS(first+P_4) is better than DYN/FOO-P.COL on eight graphs and worse on two/one graphs. For flat300_28, there are only few algorithms in the literature that can reach 28-coloring.
- ID-ILS(first+P_4) is better than VSS-Col on three graphs (six graphs if we consider the success rates) and worse on two graphs.

When comparing with the results of the two hybrid evolutionary algorithms HCA and MMT, one observes that ID-ILS(first+P_4) is competitive. Indeed, our method is better than HCA on flat300_28_0 and worse on three graphs, better than MMT on four graphs (DSJC500.9, DSJC1000.9, flat300_28_0 and latin_square) and worse on six graphs. If we compare with the results of MACOL, one easily observes

Table 4. Impact of the CLS and the `nextneigh` parameter on the performance of `ID-ILS`. We report in parentheses the best cost for the unsuccessful runs.

Instance	k	P_i	ID-ILS(first)			ID-ILS(best)			ILS/TS		
			Succ.	Time	Avg.	Succ.	Time	Avg.	Succ.	Time	Avg.
le450_15c	15	P_2	20/20	0.6	0	10/20	9.1	2.15	5/20	286	7.5
		P_4	20/20	0.3	0	7/20	17.4	3.1	0/20	-	24.7(2)
le450_15d	15	P_2	20/20	6.4	0	11/20	76.4	1.05	2/20	468.5	11.7
		P_4	20/20	3.2	0	5/20	43.6	2.7	0/20	-	31.5(13)
le450_25c	26	P_2	25/25	10	0	0/20	-	6.1(4)	5/20	295.6	2.2
		P_4	12/25	19.4	0.68	0/20	-	6.6(5)	0/20	-	6.6(2)
le450_25d	26	P_2	20/20	11.3	0	0/20	-	5.4(3)	7/20	233	2.2
		P_4	7/20	17.2	0.88	0/20	-	6.6(5)	2/20	79.5	5.6
DSJC250.5	28	P_2	18/20	148.1	0.1	0/20	-	3.9(2)	3/20	1436	2.7
		P_4	20/20	76.4	0	0/20	-	5(4)	1/20	149	5.5
DSJC500.9	127	P_2	20/20	173.3	0	0/20	-	3.95(2)	1/20	5523	3.85
		P_4	19/20	194.2	0.05	0/20	-	5.05(3)	1/20	687	5.3

that `MACOL` clearly outperforms `ID-ILS(first+`P_4`)`. However, our method should be considered as a simple local search method which uses very simple diversification devices, while `HCA`, `MMT` and `MACOL` are much more complex algorithms, with sophisticated ingredients finely tuned.

4.5 Analysis of the Parameters of `ID-ILS`

The aim of this section is to study the impact of the two local search devices, on the performance of `ID-ILS`.

(a) Impact of `nextneigh`. Table 4 compares the results of the two variants of `ID-ILS`. The impact of setting `nextneigh` to `first` has a strong influence on the effectiveness of `ID-ILS`, particularly on le450_25 and DSJC500.9, for which several orders of magnitude are gained. The poor results of `ID-ILS(best)` are probably due to the fact that selecting the best neighbor among the `maxneigh` non-accepted candidates leads `ID-ILS` to get stuck in a deep local optimum looping between already visited areas. This surprising result shows that setting `nextneigh` to `first` clearly favors diversification.

(b) Impact of the CLS. In this study, we compared `ID-ILS` with a version of `ILS` using `TABUCOL` without any CLS (noted `ILS/TS`). As showed in Table 4, `ID-ILS(first)` is clearly more relevant. We can observe that the impact of the CLS when `nextneigh` = `best` is not systematic. Indeed, compared to `ILS/TS`, `ID-ILS(best)` performs very well on le450_15c&d, but on the other instances, `ILS/TS` is very effective. These results highlight the importance of the CLS device for enforcing a tradeoff between intensification and diversification.

5 Conclusions

In this paper, we have proposed a new extension of the `ILS` algorithm, noted `ID-ILS`, by introducing new devices that enforce an efficient tradeoff between intensification and diversification. For the graph coloring problem, we have defined new perturbation schemes that exploit information from the topology of the constraints graph. Experimentations, carried out on a set of DIMACS graphs show that our method is very competitive with the current best hybrid approaches. Let

us mention that our approach is generic (with some adaptations for perturbation schemes) and could be applied to other difficult optimization problems. We are currently investigating such a direction on Radio link frequency assignment (RLFAP), and Car Sequencing problems. We also intend to study the impact of the perturbation size on intensification/diversification.

References

1. Burke, E.K., Marecek, J., Parkes, A.J., Rudová, H.: A supernodal formulation of vertex colouring with applications in course timetabling. Annals OR 179(1), 105–130 (2010)
2. Gamache, M., Hertz, A., Ouellet, J.O.: A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. Computers & OR 34(8), 2384–2395 (2007)
3. Malaguti, E., Toth, P.: A survey on vertex coloring problems. Intl. Trans. in Op. Res. 17(1), 1475–3995 (2010)
4. Blöchliger, I., Zufferey, N.: A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers & OR 35(3), 960–975 (2008)
5. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing 39(4), 345–351 (1987)
6. Hertz, A., Plumettaz, M., Zufferey, N.: Variable space search for graph coloring. Discrete Applied Mathematics 156(13), 2551–2560 (2008)
7. Morgenstern, C.: Distributed coloration neighborhood search. DIMACS Series, vol. 26, pp. 335–357. Providence, RI (1996)
8. Fleurent, C., Ferland, J.: Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63(3), 437–461 (1996)
9. Galinier, P., Hao, J.K.: Hybrid evolutionary algorithms for graph coloring. J. Comb. Optim. 3(4), 379–397 (1999)
10. Lü, Z., Hao, J.K.: A memetic algorithm for graph coloring. European Journal of Operational Research 203(1), 241–250 (2010)
11. Malaguti, E., Monaci, M., Toth, P.: A metaheuristic approach for the vertex coloring problem. INFORMS Journal on Computing 20(2), 302–316 (2008)
12. Lourenço, H.R., Martin, O., Stützle, T.: Iterated local search: Framework and applications. In: Handbook of Metaheuristics, vol. 146, pp. 363–397. Springer, New York (2010)
13. Linhares, A., Yanasse, H.: Search intensity versus search diversity: a false trade off? Appl. Intell. 32(3), 279–291 (2010)
14. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers (1997)
15. Neveu, B., Trombettoni, G., Glover, F.: ID Walk: A Candidate List Strategy with a Simple Diversification Device. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 423–437. Springer, Heidelberg (2004)
16. Trick, M.: Computational symposium: Graph coloring and its generalizations. Cornell University, Ithaca, NY (2002), http://mat.gsia.cmu.edu/COLOR02/
17. Avanthay, C., Hertz, A., Zufferey, N.: A variable neighborhood search for graph coloring. European Journal of Operational Research 151(2), 379–388 (2003)
18. Loudni, S., Boizumault, P., Levasseur, N.: Advanced generic neighborhood heuristics for vns. Eng. Appl. of AI 23(5), 736–764 (2010)
19. Chiarandini, M., Stützle, T.: An application of iterated local search to graph coloring. In: Proceedings of the Comput. Symposium on Graph Coloring and its Generalizations, Ithaca, New York, USA, pp. 112–125 (2002)

Iterated Greedy Algorithms for the Maximal Covering Location Problem

Francisco J. Rodriguez[1], Christian Blum[2],
Manuel Lozano[1], and Carlos García-Martínez[3]

[1] Department of Computer Science and Artificial Intelligence,
University of Granada, Granada, Spain
[2] ALBCOM Research Group, Technical University of Catalonia, Barcelona, Spain
[3] Department of Computing and Numerical Analysis,
University of Córdoba, Córdoba, Spain
fjrodriguez@decsai.ugr.es, cblum@lsi.upc.edu,
lozano@decsai.ugr.es, cgarcia@uco.es

Abstract. The problem of allocating a set of facilities in order to maximise the sum of the demands of the covered clients is known as the maximal covering location problem. In this work we tackle this problem by means of iterated greedy algorithms. These algorithms iteratively refine a solution by partial destruction and reconstruction, using a greedy constructive procedure. Iterated greedy algorithms have been applied successfully to solve a considerable number of problems. With the aim of providing additional results and insights along this line of research, this paper proposes two new iterated greedy algorithms that incorporate two innovative components: a population of solutions optimised in parallel by the iterated greedy algorithm, and an improvement procedure that explores a large neighbourhood by means of an exact solver. The benefits of the proposal in comparison to a recently proposed decomposition heuristic and a standalone exact solver are experimentally shown.

Keywords: iterated greedy algorithm, large neighbourhood search, maximal covering location problem.

1 Introduction

The *maximal covering location problem* MCLP [4] considers a predefined number of facilities that have to be allocated such that the demand of the clients covered by these facilities—given a maximum service distance—is maximal. This problem has several real-world applications in different fields, including the planning of service locations such us health-care centres, fire stations, and emergency centres.

More precisely, let M be the set of m potential facility locations and N the set of n clients to be covered. $D(i, j)$ denotes the distance between each pair of nodes $i \in N$ and $j \in M$, U is the maximum service distance, and w_i is the demand of client i. Note that a client i is covered by a facility installed at location $j \in M$ iff $D(i, j) \leq U$. The objective is to maximise the sum of the demands of all the

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 172–181, 2012.

clients covered by any of the p installed facilities. The MCLP may be formulated as the following zero-one integer programming problem [4]:

$$\textbf{max } z = \sum_{i=1}^{n} w_i \cdot x_i \tag{1}$$

$$\textbf{subject to: } \sum_{j \in S_i} y_j \geq x_i \quad \text{for } i \in N \tag{2}$$

$$\sum_{j \in M} y_j = p \tag{3}$$

$$x_i \in \{0,1\} \quad \text{for } i \in N \tag{4}$$

$$y_j \in \{0,1\} \quad \text{for } j \in M \tag{5}$$

Hereby, x_i is a binary variable indicating whether client i is covered by a facility, y_j is a binary variable that attests whether location j has been chosen to install a facility, and S_i is the set composed by all potential facility locations that cover client i, that is, $S_i = \{j \in M : D(i,j) \leq U\}$.

1.1 Previous Work

The MCLP is an NP-hard problem [9] that has received quite some attention since it was presented, having resulted in a variety of proposals for tackling the problem. The latter include exact algorithms for relaxations of the problem [4,6,7,8], greedy heuristics [4], and several metaheuristics such as genetic algorithms [2,14], tabu search [14], and simulated annealing [14]). Recently, Senne et al. [13] presented a decomposition heuristic to perform a cluster partitioning, resulting in smaller subproblems (clusters) that can be solved independently by exact methods (LagClus). The results obtained by this approach are compared in terms of quality and computational time required with regards to those of a commercial solver (CPLEX [1]), indicating that the proposed decomposition approach can substantially reduce the time for providing good-enough solutions to large problem instances.

1.2 Our Contribution

In this work, we propose two iterated greedy (IG) algorithms [5,11] for solving the MCLP. IG algorithms, generally, try to iteratively refine a solution by removing elements from this solution by means of a destructive procedure and reconstructing the resulting partial solution using a greedy constructive procedure. The first one of the proposed IG variants extends the basic IG idea by considering a population of solutions that are improved in parallel by means of the standard IG procedure. The resulting algorithm is labelled *population-based iterated greedy* (PBIG). Second, we propose a hybrid algorithm that combines PBIG with an exact solver. In particular, CPLEX is employed for this purpose. The idea consists in completing the solutions provided by the destruction procedure of IG

by means of the application of CPLEX. The latter only optimises a predefined number of components of each solution, while the remaining components of the solution are fixed to the values of the partial solutions received as input. This strategy is known as *large neighbourhood search* (LNS) [3,12]. The basic idea is to combine the advantages of a large neighbourhood, which usually enhances the exploration of a local search method, with an exhaustive tree-search exploration which is faster than enumeration. Our second IG approach is labelled *population based iterated greedy with large neighbourhood search* (PBIG+LNS).

1.3 Paper Organization

The remainder of this paper is organized as follows. In Section 2, we present in detail the two proposed IG variants for the MCLP. In Section 3, we present a empirical study that compares the behaviour of the two proposed IG algorithms with regards to those of the most recent proposal from the literature, LagClus, and a standalone CPLEX procedure. Finally, in Section 4, we discuss conclusions and further work.

2 Proposed IG Variants for the MLCP

In this section, we describe the two proposed IG variants for the MLCP. First, let us focus on the PBIG scheme. It extends IG by working on a population of solutions which is managed in the style of evolution strategies. The resulting algorithm is outlined in Figure 1. It starts by initialising the population P with t solutions generated by a probabilistic greedy constructive procedure (as outlined below). Inside the main loop, each solution $s \in P$ is optimised by means of a destruction/re-construction procedure, generating a new population P_n of solutions. The destruction step consists in randomly removing n_d elements from the considered solution s, resulting in a partial solution s_d. This solution is reconstructed by means of the same probabilistic greedy constructive procedure that was used to generate the initial population. This step results in a (possibly new) complete solution s_c which is then added to P_n. After applying this process to all solutions from P, the new population P_n is added to P, resulting in a new set P of size $2 \cdot t$. The last step of each iteration consists in choosing the best t solutions from P for the population of the next iteration. The proposed algorithm iterates through these phases until a computation limit t_{max} is reached.

2.1 The Probabilistic Greedy Procedure

The probabilistic greedy constructive procedure used for the initialisation of the population and the re-construction of partial solutions works as follows. At each step, it considers placing a new facility in any of the locations that is not yet occupied by an already installed facility. For each of these options, it calculates the contribution to the objective function value, that is, the increase in the function value caused by the respective option. The two options which cause the

```
   Input: t_max, t, n_d, prob
   Output: s
1  P ←GenerateInitialPopulation(t);
2  while computation time limit t_max not reached do
3      P_n ← ∅ ;
4      foreach s ∈ P do
5          s_d ← Destruction(s, n_d);
6          if PBIG() then s_c ← Construction(s_d, prob) // PBIG ;
7          else s_c ← LNS(s_d) // PBIG+LNS ;
8          P_n ← P_n ∪ {s_c};
9      end
10     P ← P ∪ P_n;
11     P ← SelectBestSolutions(P, t) ;
12 end
```

Fig. 1. PBIG and PBIG+LNS scheme

highest increase are identified. Finally, the best option is chosen with probability *prob*, which is an input parameter of the algorithm. Otherwise the second-best option is chosen. The procedure stops once p facilities are installed.

2.2 PBIG+LNS

As mentioned already in the introduction, PBIG+LNS modifies PBIG by replacing the constructive step with a large neighborhood search method applied to the solutions generated by the destructive step of PBIG (see line 7 of the algorithm from Figure 1). This procedure is performed by an exact solver (CPLEX), whereby the size of the neighbourhood is determined by fixing the components provided by the current partial solution s_d. In particular, a binary variable y_j (see the definition of the MCLP) is fixed to 1 if location j is selected as a facility in s_d. In the same way, a variable x_i is fixed to 1 if client i is covered by any of the fixed facility locations. This means that CPLEX will try to find an allocation for the n_d unallocated facilities. In this way, CPLEX—which is already very efficient for problem instances with a small number of clients and facilities [13]—can be used as a sub-ordinate procedure for tackling large-size problem instances. The complete pseudocode of the LNS method is shown in Figure 2.

3 Computational Experiments

This section describes the computational experiments performed to assess the performance of the two IG algorithms presented in the previous section. Both PBIG and PBIG+LNS were coded in Java and the tests were conducted on a computer with a 3.2 GHz Intel i7 processor with 12 GB of RAM running Fedora Linux V15.

Input: s_c
Output: s_c
$Y_{fixed} \leftarrow \emptyset$;
$X_{fixed} \leftarrow \emptyset$;
foreach $j \in M$ **do**
  if IsSelectedAsFacility(j, s_d) **then** $Y_{fixed} \leftarrow Y_{fixed} \cup \{y_j\}$ **end**;
end
foreach $i \in N$ **do**
  if IsCovered(i, s_d) **then** $X_{fixed} \leftarrow X_{fixed} \cup \{x_i\}$ **end**;
end
s_c = CPLEX(Y_{fixed}, X_{fixed}) // Neighbourhood restricted to the set of free binary variables x_i and y_j ;

Fig. 2. Procedure LNS() of PBIG+LNS

3.1 Problem Instances

We have employed two different sets of problem instances:

1. Real case instances for facility location problems in Sao Jose dos Campos, Brazil (SCJ instances). These instances are available for download at http://www.lac.inpe.br/lorena/instancias.html.
2. An instance which was created on the basis of instance PCB3038 available from the TSPLIB [10].

3.2 Tuning Experiments

In order to perform a fine-tuning of the two proposed IG algorithms we first conducted tuning experiments in order to find values for the following algorithm parameters:

1. **Population size** t**:** values from $\{1, 2, 10, 10, 50, 100\}$ were considered.
2. **Destruction size** n_d**:** For the percentage of elements dropped from the current solution during the destructive step values from $\{5\%, 10\%, 20\%, 50\%\}$ were considered.
3. **Degree of determinism for the solution construction** *prob***:** for the probability of accepting the option with the best contribution during the greedy constructive procedure values from $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$) were considered. Note that $prob = 1.0$ corresponds to a completely deterministic solution (re-)construction.

For each combination of values for the three parameters (full factorial design), we applied both PBIG (and PBIG+LNS) to a subset of the real-case instances. The computation time limits were chosen dependent on the number of clients of the instance (50 seconds for instances with 324, respectively 500, clients and 100 seconds for those with 818 clients). A rank-based analysis was applied to the results. The parameter combination with the best average rank over all

Table 1. Parameters values

Parameter	PBIG	PBIG+LNS
Population size (t)	100	20
Elements dropped (n_d)	20%	50%
Degree of determinism ($prob$)	0.8	0.8

testing instances is shown in Table 1. Although PBIG and PBIG+LNS share core funcionality, we have performed their parameter analysis separately. The tuning outcome shows that, interestingly, the two algorithm versions reach their best performance with quite different values for two out of the three parameters.

3.3 Experimental Results

In the following we describe the outcome of a comparative analysis between the results of PBIG and PBIG+LNS and those of LagClus, which is the currently best heuristic method, and the standalone CPLEX procedure. The reported results for LagClus and CPLEX are the ones provided in [13]. They were obtained on a computer with an Intel Core 2 Duo 2.0 GHz processor and 2.0 GB RAM, running Windows XP. Tables 2 and 3 show the results for the SCJ instances, considering a service distance of 150 and 200, respectively. For each instance—which is determined by a number of clients, n, and a number of facilities, p—we provide the optimal solution obtained by CPLEX and the $gap = 100 \cdot ((Result - Optimal)/Optimal)$ for each algorithm. Moreover, for each algorithm we show the computational time needed to obtain the corresponding result. The results reported for PBIG and PBIG+LNS correspond to the average over 10 independent applications to each problem instance. Concerning the computation time limits, we used the same ones as for the tuning procedure.

The results of the considered algorithms as shown in Tables 2 and 3 allow us to make the following observations:

- Comparing the results of PBIG and PBIG+LNS, we can observe that PBIG+LNS clearly outperforms PBIG. In addition, analysing the computational time used by the two algorithms, we can observe that PBIG+LNS reduces the computational time requirements with respect to PBIG, especially when larger problem instances are concerned. This indicates that the hybridisation with LNS seems to be a decisive element to improve not only the quality of the results but also for the reduction of the computation time requirements.
- Concerning the comparison of PBIG+LNS with LagClus, we can observe that PBIG+LNS exceeds or equals the results of LagClus for all instances with $U = 150$ and—with one exception—also for all instances with $U = 200$. It is noticeable that PBIG+LNS is able to obtain optimal solutions in all 10 runs in 37 out of 46 cases, which is indicated by an average gap of 0.000. This fact shows that—in addition to reaching high quality solutions—PBIG+LNS is characterized by a very stable behaviour. Concerning computation time,

Table 2. Results for the SJC instances, U=150

n	p	Optimal	LagClus	PBIG	PBIG+LNS
		Result (time s)	*Gap (time s)*	*Gap (time s)*	*Gap (time s)*
324	20	7302 (0.015)	**0.000** (2.543)	**0.000** (0.391)	**0.000** (0.381)
	30	9127 (0.047)	0.027 (24.650)	0.072 (9.633)	**0.000** (5,082)
	40	10443 (0.188)	0.108 (25.985)	**0.000** (26.553)	**0.000** (10.320)
	50	11397 (0.391)	0.138 (24.452)	0.005 (20.313)	**0.000** (11.164)
	60	11991 (0.235)	0.024 (44.514)	0.097 (26.896)	**0.000** (10.576)
	80	12152 (0.031)	**0.000** (8.876)	**0.000** (9.181)	**0.000** (3.732)
	108	12152 (0.016)	**0.000** (1.595)	**0.000** (0.761)	**0.000** (0.672)
500	40	13340 (0.047)	**0.000** (3.453)	0.247 (4.560)	**0.000** (3.857)
	50	14773 (0.047)	**0.000** (4.938)	0.093 (22.381)	**0.000** (6.529)
	60	15919 (0.063)	**0.000** (8.233)	0.092 (31.052)	**0.000** (12.073)
	70	16908 (0.031)	**0.000** (3.723)	0.002 (41.019)	**0.000** (8.339)
	80	17749 (0.015)	**0.000** (5.406)	0.038 (44.786)	**0.000** (10.938)
	100	18912 (0.109)	**0.000** (10.276)	0.310 (46.737)	**0.000** (24.574)
	130	19664 (0.297)	0.015 (30.827)	0.230 (47.735)	**0.000** (30.642)
	167	19707 (0.047)	0.003 (14.600)	**0.000** (1.057)	**0.000** (0.931)
818	80	23325 (0.140)	0.003 (45.564)	0.293 (94.376)	**0.000** (28.637)
	90	24455 (0.266)	0.041 (56.388)	0.348 (91.452)	**0.000** (37.726)
	100	25435 (0.344)	0.012 (87.279)	0.306 (94.742)	**0.000** (40.067)
	120	26982 (0.297)	0.015 (69.658)	0.446 (93.713)	**0.000** (33.781)
	140	28802 (0.359)	0.095 (52.966)	0.597 (94.131)	**0.002** (44.736)
	160	28699 (0.391)	0.107 (58.453)	0.612 (93.477)	**0.004** (74.202)
	200	29153 (0.234)	0.011 (61.531)	0.096 (92.225)	**0.000** (40.068)
	273	29168 (0.031)	**0.000** (3.343)	**0.000** (1.439)	**0.000** (1.271)

it seems that the requirements of LagClus and PBIG+LNS are of the same order of magnitude. All in all, this indicates that PBIG+LNS is a new state-of-the-art method for what concerns heuristics for the MCLP.

– For what concerns the comparison to CPLEX, we must observe that CPLEX is able to solve all problem instances in very little computation time. Therefore, PBIG+LNS must be considered inferior to CPLEX for the SJC instances.

Finally, in Table 4 we show the results of all analysed algorithms for the set of instances derived from the TSPLIB instance PCB3038. PBIG and PBIG+LNS consider a computation time limit of 1500 seconds for each run. It is important to highlight that the number of clients considered in this case is much higher than in the case of the SCJ instances, which imposes a more complicated environment for the analysed algorithms. In fact, as shown in Table 4, CPLEX is unable to confirm the optimality of solutions within the predefined time limit of 20000 seconds for instances with more than 18 facilities. These cases are marked by an asterisk. Therefore, large-size instances are the ones for which PBIG+LNS is an interesting alternative. Concerning the results of Table 4, we can conclude the following:

Table 3. Results for the SJC instances, U=200

n	p	Optimal	LagClus	PBIG	PBIG+LNS
		Result (time s)	*Gap (time s)*	*Gap (time s)*	*Gap (time s)*
324	20	9670 (0.172)	0.243 (19.293)	0.307 (9.633)	**0.000** (11.064)
	30	11737 (0.484)	0.060 (28.943)	0.098 (25.676)	**0.000** (12.580)
	40	12151 (0.094)	0.008 (31.066)	0.023 (35.910)	**0.004** (19.813)
	50	12152 (0.015)	**0.000** (9.926)	**0.000** (0.494)	**0.000** (0.469)
	60	12152 (0.047)	**0.000** (4.575)	**0.000** (0.620)	**0.000** (0.506)
	80	12152 (0.016)	**0.000** (3.670)	**0.000** (0.637)	**0.000** (0.663)
	108	12152 (0.031)	0.248 (11.343)	**0.000** (0.643)	**0.000** (0.763)
500	40	17077 (0.233)	0.387 (24.668)	0.255 (45.223)	**0.012** (25.679)
	50	18361 (0.109)	0.003 (39.109)	0.326 (43.199)	**0.000** (24.485)
	60	19153 (0.063)	**0.005** (52.639)	0.352 (47.504)	0.017 (31.036)
	70	19551 (1.078)	0.069 (43.946)	0.397 (46.511)	**0.006** (30.314)
	80	19703 (0.156)	0.008 (35.495)	0.124 (45.296)	**0.001** (22.716)
	100	19707 (0.078)	**0.000** (16.624)	**0.000** (1.050)	**0.000** (0.832)
	130	19707 (0.047)	**0.000** (1.986)	**0.000** (1.025)	**0.000** (1.017)
	167	19707 (0.016)	0.016 (22.379)	**0.000** (0.954)	**0.000** (1.026)
818	80	27945 (0.203)	0.069 (57.835)	0.858 (94.841)	**0.000** (51.136)
	90	28519 (1.141)	0.071 (114.145)	0.828 (95.977)	**0.013** (75.486)
	100	28910 (1.391)	0.036 (88.885)	0.801 (91.365)	**0.000** (59.345)
	120	29165 (1.234)	0.002 (55.710)	0.185 (90.345)	**0.001** (56.270)
	140	29168 (0.125)	**0.000** (11.643)	**0.000** (79.766)	**0.000** (10.313)
	160	29168 (0.062)	**0.000** (9.738)	**0.000** (1.087)	**0.000** (1.087)
	200	29168 (0.032)	**0.000** (5.762)	**0.000** (1.288)	**0.000** (1.276)
	273	29168 (0.031)	0.207 (24.689)	**0.000** (1.491)	**0.000** (1.513)

Table 4. Results for the TSPLIB instance PCB3038, U=400

n	p	Optimal	LagClus	PBIG	PBIG+LNS
		Result (time s)	*Gap (time s)*	*Gap (time s)*	*Gap (time s)*
3038	17	125320 (802)	0.205 (844)	0.992 (354)	**0.125** (357)
	18	130004 (10265)	0.372 (817)	0.481 (384)	**0.236** (556)
	19	134262* (20000)	0.382 (1483)	0.552 (573)	**0.265** (602)
	20	138028* (20000)	0.698 (1712)	0.165 (941)	**0.033** (518)
	21	141279* (20000)	0.024 (3117)	0.085 (913)	**0.000** (583)
	22	143809* (20000)	0.024 (6656)	7.155 (573)	**0.010** (602)

- PBIG+LNS is able to significantly improve over the gap of LagClus for all considered instances, while requiring even less computation time.
- PBIG+LNS generates solutions close to the ones of CPLEX, while reducing substantially—especially for instances with more than 17 facilities—the time needed to obtain high-quality solutions.

4 Conclusions and Future Work

In this paper, we have proposed two IG algorithms for the maximum covering location problem. The proposed algorithms add two novel components to the

basic IG technique. In the first place, PBIG incorporates a population of solutions evolved in parallel by means of the classic destruction/re-construction procedure of IG algorithms. In the second place, PBIG+LNS extends PBIG by incorporating an exact solver to complete the solutions generated by the destructive procedure of IG, following the ideas of large neighborhood search. The resulting hybrid algorithm, PBIG+LNS, has proved to be superior to a recently proposed decomposition heuristic. Moreover, in the case of large-scale instances, where the computation time requirements of CPLEX explode, PBIG+LNS arises as a tool of choice to face this kind of problems.

We believe that the IG frameworks presented in this paper are an interesting contribution, worthy of further study. We will mainly focus on the following avenues of possible research: (1) study of the behaviour of the proposed PBIG+LNS for what concerns new instances of the problem and (2) adapting the PBIG+LNS approach for its application to other challenging optimisation problems, especially when large-size instances are concerned.

Acknowledgements. This work was supported by grants TIN2007-66523 and TIN2008-05854 of the Spanish government and by grant P08-TIC-4173 of the Andalusian regional goverment. Moreover, Christian Blum acknowledges support from the *Ramón y Cajal* program of the Spanish Ministry of Science and Innovation.

References

1. IBM ILOG CPLEX optimizer (November 2011), http://www-01.ibm.com/software/integration/optimization/cplexoptimizer/
2. Arakaki, R.G.I., Lorena, L.A.N.: A constructive genetic algorithm for the maximal covering location problem. In: Proceedings of the 4th Metaheuristics International Conference (MIC 2001), pp. 13–17 (2001)
3. Blum, C., Puchinger, J., Raidl, G.R., Roli, A.: Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing 11(6), 4135–4151 (2011)
4. Church, R., Velle, C.R.: The maximal covering location problem. Papers in Regional Science 32(1), 101–118 (1974)
5. Culberson, J.C., Luo, F.: Exploring the k-colorable landscape with iterated greedy. Dimacs Series in Discrete Mathematics and Theoretical Computer Science, pp. 245–284. American Mathematical Society (1996)
6. Galvao, R.D., Espejo, L.G.A., Boffey, B.: A comparison of lagrangean and surrogate relaxations for the maximal covering location problem. European Journal of Operational Research 124(2), 377–389 (2000)
7. Galvao, R.D., ReVelle, C.: A lagrangean heuristic for the maximal covering location problem. European Journal of Operational Research 88(1), 114–123 (1996)
8. Lorena, L.A., Pereira, M.A.: A lagrangean/surrogate heuristic for the maximal covering location problem using hillsman's edition. International Journal of Industrial Engineering 9, 57–67 (2001)
9. Megiddo, N., Zemel, E., Hakimi, S.L.: The maximum coverage location problem. SIAM Journal on Algebraic and Discrete Methods 4(2), 253–261 (1983)

10. Reinelt, G.: The traveling salesman: computational solutions for TSP applications. Springer, Heidelberg (1994)
11. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177(3), 2033–2049 (2007)
12. Shaw, P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: Maher, M., Puget, J.-F. (eds.) CP 1998. LNCS, vol. 1520, pp. 417–431. Springer, Heidelberg (1998)
13. Senne, E.L.F., Pereira, M.A., Lorena, L.A.N.: A decomposition heuristic for the maximal covering location problem. Advances in Operations Research 2010 (2010)
14. Xia, L., Xie, M., Xu, W., Shao, J., Yin, W., Dong, J.: An empirical comparison of five efficient heuristics for maximal covering location problems. In: IEEE/INFORMS International Conference on Service Operations, Logistics and Informatics (SOLI 2009), pp. 747–753 (2009)

Multiobjectivizing the HP Model for Protein Structure Prediction

Mario Garza-Fabre, Eduardo Rodriguez-Tello, and Gregorio Toscano-Pulido

Information Technology Laboratory, CINVESTAV-Tamaulipas
Parque Científico y Tecnológico TECNOTAM
Km. 5.5 carretera Cd. Victoria-Soto La Marina
Cd. Victoria, Tamaulipas 87130, México
{mgarza,ertello,gtoscano}@tamps.cinvestav.mx

Abstract. The hydrophobic-polar (HP) model for protein structure prediction abstracts the fact that hydrophobic interactions are a dominant force in the protein folding process. This model represents a hard combinatorial optimization problem, which has been widely addressed using evolutionary algorithms and other metaheuristics. In this paper, the multiobjectivization of the HP model is proposed. This originally single-objective problem is restated as a multiobjective one by decomposing the conventional objective function into two independent objectives. By using different evolutionary algorithms and a large set of test cases, the new alternative formulation was compared against the conventional single-objective problem formulation. As a result, the proposed formulation increased the search performance of the implemented algorithms in most of the cases. Both two- and three-dimensional lattices are considered. To the best of authors' knowledge, this is the first study where multiobjective optimization methods are used for solving the HP model.

Keywords: Multiobjectivization, protein structure prediction, HP model.

1 Introduction

Proteins, the working molecules of the cell, are linear chains composed from up to 20 different building blocks called amino acids. The specific sequence of amino acids determines how proteins fold into unique three-dimensional structures which allow them to carry out their biological functions [1]. The *protein structure prediction* problem (PSP) can be defined as the problem of finding the functional conformation for a protein given only its amino acid sequence.

The hydrophobic-polar (HP) model [12] is an abstraction of the PSP. This model captures the fact that hydrophobicity is one of the main driving forces in protein folding. The prediction of protein structures using the HP model is a hard combinatorial optimization problem which has been demonstrated to be $\mathcal{NP}$-complete [3, 7]. A variety of metaheuristic approaches have been applied to this problem, including genetic algorithms [16, 31], memetic and hybrid algorithms [6, 17], ant colony optimization [29], immune-based algorithms [9], particle

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 182–193, 2012.

swarm optimization [5], differential evolution [25] and estimation of distribution algorithms [24]. Some of the work in this regard is reviewed in [22, 33].

Multiobjectivization concerns the reformulation of single-objective optimization problems in terms of two or more objective functions [20]. This transformation introduces fundamental changes in the search landscape, potentially allowing algorithms to perform a more efficient exploration [4, 15]. Multiobjectivization has been successfully used to deal with difficult optimization problems. Among them, there can be mentioned well-known combinatorial problems such as the traveling salesman problem [18–20], shortest path and minimum spanning tree problems [23], job-shop scheduling [19, 21] and bin packing problems [28], as well as important problems in the fields of mobile communications [26, 27] and computer vision [32]. Multiobjectivization approaches have also been proposed for the PSP [2, 8, 10, 14, 30]. However, it was not until the present study that this concept is applied to the particular HP model of this problem.

In this paper, the multiobjectivization for the HP model is proposed. The conventional HP model's energy function is decomposed into two separate objectives based on the parity of amino acid positions in the protein sequence. The suitability of this approach is investigated by comparing it with respect to the conventional single-objective formulation. Different evolutionary algorithms (EAs) and a large set of test cases were adopted for this sake. Results are provided for both the two-dimensional square lattice and the three-dimensional cubic lattice.

This paper is organized as follows. Background concepts are given in Section 2. In Section 3, the proposed multiobjectivization is described. Section 4 details the implemented EAs and the performance assessment methodology. Results are presented in Section 5. Finally, Section 6 provides the conclusions of this study.

2 Background and Notation

2.1 The HP Model for Protein Structure Prediction

Amino acids can be classified either as *hydrophobic* (H) or *polar* (P) on the basis of their affinity for water. In the hydrophobic-polar (HP) model [12], proteins are abstracted as chains of H and P beads. Protein sequences, originally defined over a 20-letters alphabet, are thus of the form $S \in \{H, P\}^L$, where L is the number of amino acids. Valid conformations are modeled as *Self-Avoiding Walks* of the HP chain on a lattice. That is, each lattice node can be assigned to at most one amino acid and consecutive amino acids in S are to be also adjacent in the lattice.

By emulating the hydrophobic effect, the HP model aims to maximize the interaction among H amino acids. Two H amino acids $s_i, s_j \in S$ are said to form a *hydrophobic topological contact*, denoted by $htc(s_i, s_j)$, if they are nonconsecutive in S (*i.e.*, $|j - i| \geq 2$) but adjacent in the lattice. Following the notation of the field, an energy minimization function $E : C \to \mathbb{R}$ is defined as the negative of the total number of hydrophobic topological contacts; C is the set of all valid protein conformations. Formally, the energy of a conformation $c \in C$ is given by:

$$E(c) = \sum_{s_i, s_j \in S | i<j} e(s_i, s_j) \tag{1}$$

where

$$e(s_i, s_j) = \begin{cases} -1 \text{ if } htc(s_i, s_j) \\ 0 \quad \text{otherwise} \end{cases}$$

The protein structure prediction problem using the HP model can be formally stated as the problem of finding the conformation $c^* \in C$ such that $E(c^*) = \min\{E(c) \mid c \in C\}$. An example conformation for an HP chain of length $L = 20$ on the two-dimensional square lattice is shown in Figure 1.

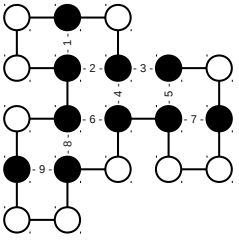


Fig. 1. Black and white beads denote H and P amino acids, respectively. Hydrophobic topological contacts have been numbered. The energy is $E(c) = -9$.

2.2 Single-objective and Multiobjective Optimization

Without loss of generality, a *single-objective optimization problem* can be stated as the problem of minimizing an *objective function* $f : \mathcal{F} \to \mathbb{R}$, where $\mathcal{F}$ denotes the set of all feasible solutions. The aim is to find the solution(s) $x^* \in \mathcal{F}$ yielding the optimum value for the objective function; that is, $f(x^*) = \min\{f(x) \mid x \in \mathcal{F}\}$.

Similarly, a *multiobjective optimization problem* is the problem of minimizing an *objective vector* $\mathbf{f}(x) = [f_1(x), f_2(x), \ldots, f_k(x)]^T$, where $f_i : \mathcal{F} \to \mathbb{R}$ is the i-th objective function, $i \in \{1, \ldots, k\}$. Rather than searching for a single optimal solution, the task in multiobjective optimization is to identify a set of trade-offs among the, usually conflicting, objectives. More formally, the goal is to find a set of *Pareto-optimal solutions* $\mathcal{P}^* \subset \mathcal{F}$, such that $\mathcal{P}^* = \{x^* \in \mathcal{F} \mid \nexists x \in \mathcal{F} : x \prec x^*\}$. The symbol "$\prec$" denotes the *Pareto-dominance* relation, which is given by:

$$x \prec y \Leftrightarrow \forall i \in \{1, \ldots, k\} : f_i(x) \leq f_i(y) \quad \wedge \\ \exists j \in \{1, \ldots, k\} : f_j(x) < f_j(y) \tag{2}$$

If $x \prec y$, x is said to *dominate* y. Otherwise ($x \not\prec y$), y is said to be *nondominated* by x. The image of $\mathcal{P}^*$ in the objective space is called the *Pareto-optimal front*.

2.3 Multiobjectivization

Multiobjectivization refers to the process of reformulating a single-objective optimization problem as a multiobjective one [20]. Two different approaches are possible. On the one hand, additional information can be incorporated and used as *supplementary* (also called artificial or helper) objectives [4, 19]. On the other

hand, in the *decomposition* approach the original objective is fragmented into several different components, each to be treated as an objective function under the new alternative formulation [15, 20]. In either approach, the idea is to alter the search landscape in order to enable a more efficient exploration, but the goal remains to solve the original problem. Therefore, the original optima are to be also Pareto-optimal with regard to the multiobjectivized version of the problem.

This work is based on the decomposition approach. More formally, a single-objective problem, with a given objective function $f : \mathcal{F} \to \mathbb{R}$, is restated in terms of $k \geq 2$ objectives $f_i : \mathcal{F} \to \mathbb{R}, i \in \{1, \ldots, k\}$ such that for all $x \in \mathcal{F}$ it holds that $f(x) = \sum_{i=1}^{k} f_i(x)$. As the only possible effect [15], plateaus may be introduced in the search landscape. That is, originally comparable solutions may become incomparable (mutually nondominated) with regard to the decomposed formulation. This can be seen as a potential strategy to escape from local optima [15, 20].

3 Multiobjectivization Proposal: The Parity Decomposition

In the two-dimensional square and the three-dimensional cubic lattices, adjacencies (topological contacts) are only possible between amino acids whose sequence positions are of opposite parity. Based on this fact and following the multiobjectivization by decomposition approach (Section 2.3), a two-objective formulation $\mathbf{f}(c) = [f_1(c), f_2(c)]^T$ is defined over the set of feasible conformations $c \in C$:

$$f_1(c) = \sum_{s_i, s_j \in S | i<j} e_p(s_i, s_j, 0) \tag{3}$$

$$f_2(c) = \sum_{s_i, s_j \in S | i<j} e_p(s_i, s_j, 1) \tag{4}$$

where both $f_1(c)$ and $f_2(c)$ are to be minimized and

$$e_p(s_i, s_j, \rho) = \begin{cases} -1 & \text{if } htc(s_i, s_j) \wedge i \equiv \rho \pmod 2 \\ 0 & \text{otherwise} \end{cases}$$

That is, the objective function f_1 accounts only for hydrophobic topological contacts $htc(s_i, s_j)$ where i, the sequence position of amino acid s_i, is even. On the contrary, f_2 is defined for those cases where such the i-th sequence position is odd. Note that the sum of the two proposed objectives equals the conventional energy function defined in Section 2.1 (*i.e.*, $E(c) = f_1(c) + f_2(c)$ for all $c \in C$), which is in accordance with the decomposition approach for multiobjectivization.

4 Experimental Setup

4.1 Algorithms

Several evolutionary algorithms (EAs) are used to investigate the suitability of the proposed multiobjectivization. The so-called (1+1) EA is described in Algorithm 1. First, an initial individual c is generated at random. At each

generation, a new individual c' is created by means of mutation. If c' is at least as good as c, then c' is accepted as the starting point for the next generation. Depending on the problem formulation, this acceptance criterion is to be based either on the conventional energy evaluation or on the Pareto-dominance relation.

Algorithm 1. Basic (1+1) evolutionary algorithm

```
1: choose c ∈ C uniformly at random
2: repeat
3:     c' ← mutate(c)
4:     if c' not worse than c then
5:         c ← c'
6:     end if
7: until < stop condition >
```

A variant of the above described (1+1) EA is presented in Algorithm 2. An external archive stores the nondominated solutions found along the evolutionary process. The archive influences the behavior of the algorithm in such a way that the mutant c' is only accepted if it is not dominated by any archived individual. If accepted, c' is included in the archive and all individuals dominated by c', and those mapping to the same objective vector $\mathbf{f}(c')$, are removed. Note that the use of this external archive makes only sense for the multiobjectivized formulation.

Algorithm 2. Archiving (1+1) evolutionary algorithm

```
1: choose c ∈ C uniformly at random
2: A ← {c}
3: repeat
4:     c' ← mutate(c)
5:     if ∄ĉ ∈ A : ĉ ≺ c' then
6:         A ← {ĉ ∈ A : c' ⊀ ĉ ∧ f(ĉ) ≠ f(c')} ∪ {c'}
7:         c ← c'
8:     end if
9: until < stop condition >
```

It was also considered a genetic algorithm (GA) whose general structure is given in Algorithm 3. First, an initial parent population P of size N is randomly generated. At each generation, the fittest individuals in P are selected for mating (*selection-for-variation*). Then, a children population P' is created by applying the genetic operators. Finally, parents and children compete for a place in the new population (*selection-for-survival*). When applied to the single-objective formulation, selection is driven by the conventional energy value of the candidate conformations. For the multiobjective formulation, the discrimination among individuals is to be based on *nondominated sorting* and *crowding distance* [11].

Algorithm 3. Genetic algorithm

```
1: choose P ⊂ C : |P| = N uniformly at random
2: while < stop condition > do
3:     P̂ ← selection-for-variation(P)
4:     P' ← variation(P̂)
5:     P ← selection-for-survival(P ∪ P')
6: end while
```

An internal coordinates representation with absolute moves was adopted in all cases. Conformations are encoded as sequences in $\{U, D, L, R, F, B\}^{L-1}$, denoting the up, down, left, right, forward and backward possible lattice locations for an amino acid with regard to the preceding one (the position of the first amino acid is fixed). Only directions $\{U, D, L, R\}$ hold for the two-dimensional lattice. The implemented genetic operators are as follows. One-point crossover (only for the GA) is applied with a given probability p_c. In mutation, each encoding position is randomly and independently perturbed with probability p_m. In all cases, only valid solutions are accepted during the search process.

4.2 Test Cases and Performance Assessment

A total of 30 HP benchmark sequences were used (15 for the two-dimensional square lattice and 15 for the three-dimensional case). Due to space limitations, details of these instances are not provided here, but they are available online.[1]

For all the experiments, 100 independent executions were performed. Results are evaluated in terms of the best obtained energy value (β), the number of times this solution was found (f) and the arithmetic mean (μ). Additionally, the *overall average performance* (OAP) measure [13] was defined as the average ratio of the obtained mean values to the optimum (E^*). Formally, $\text{OAP} = \frac{100}{|T|}\left(\sum_{t \in T} \frac{\mu(t)}{E^*(t)}\right)$, where T denotes the set of all test cases. OAP is expressed as a percentage. Thus, a value of OAP = 100% suggests the ideal situation where the optimum solution for each benchmark was reached during all the performed executions.

Statistical significance analysis was performed for all the experiments. First, *D'Agostino-Pearson's omnibus* K^2 test was used to evaluate the normality of data distributions. For normally distributed data, either *ANOVA* or the *Welch's t* parametric tests were used depending on whether the variances across the samples were homogeneous (*homoskedasticity*) or not. This was investigated using the *Bartlett's* test. For non-normal data, the nonparametric *Kruskal-Wallis* test was adopted. A significance level of $\alpha = 0.05$ has been considered.

Most of the results are presented in tables, where values **marked ▲** highlight a statistically significant increase in performance achieved by the proposed formulation with regard to the conventional one. Conversely, values **marked ▼** indicate that a statistically significant performance decrease was obtained as a consequence of using the new alternative formulation. Additionally, the best average performance (μ) for each test instance has been **shaded** in these tables.

5 Results

5.1 Results for the (1+1) Evolutionary Algorithm

In this section, the (1+1) EA is used for comparing the conventional single-objective HP model formulation with respect to the proposed parity decomposition. Results are also provided for the archiving (1+1) EA, which applies only for the proposed formulation. A fixed mutation probability of $p_m = \frac{1}{L-1}$ and a

[1] http://www.tamps.cinvestav.mx/~mgarza/HPmodel/

Table 1. Results for the (1+1) EA on two-dimensional benchmarks

			Single-objective		Parity decomposition		Parity dec. - archive	
Seq.	L	E^*	$\beta\ (f)$	μ	$\beta\ (f)$	μ	$\beta\ (f)$	μ
2d1	**18**	**-4**	-4 (4)	-2.70	-4 (6)	**-2.71**	-4 (5)	-2.69
2d2	**18**	**-8**	-8 (18)	-6.81	-8 (24)	**-7.04**	-8 (21)	-7.00
2d3	**18**	**-9**	-8 (11)	-7.00	-8 (48)	**-7.45** ▲	-8 (24)	-7.12
2d4	**20**	**-9**	-9 (8)	-6.84	-9 (4)	**-6.95**	-9 (6)	-6.88
2d5	**20**	**-10**	-9 (3)	-6.92	-10 (2)	**-7.08**	-9 (1)	-6.99
2d6	**24**	**-9**	-8 (14)	-6.81	-9 (1)	-6.87	-9 (1)	**-6.89**
2d7	**25**	**-8**	-7 (26)	-5.79	-8 (6)	**-5.90**	-8 (5)	-5.80
2d8	**36**	**-14**	-13 (1)	-9.97	-13 (1)	**-10.23**	-13 (1)	-10.12
2d9	**48**	**-23**	-18 (5)	-14.23	-19 (2)	**-15.20** ▲	-18 (5)	**-15.02** ▲
2d10	**50**	**-21**	-18 (2)	-13.79	-18 (1)	**-14.06**	-17 (4)	-13.76
2d11	**60**	**-36**	-30 (2)	-24.39	-30 (7)	**-25.43** ▲	-31 (1)	**-25.32** ▲
2d12	**64**	**-42**	-29 (1)	-23.82	-30 (1)	**-25.12** ▲	-30 (1)	**-24.63** ▲
2d13	**85**	**-53**	-41 (1)	-33.81	-41 (1)	**-34.54**	-42 (1)	-34.18
2d14	**100**	**-48**	-41 (1)	-30.80	-39 (3)	**-32.18** ▲	-41 (1)	**-31.72** ▲
2d15	**100**	**-50**	-40 (1)	-31.71	-40 (3)	**-32.70** ▲	-40 (1)	-32.57
OAP			69.22%		**71.39%**		70.47%	

Table 2. Results for the (1+1) EA on three-dimensional benchmarks

			Single-objective		Parity decomposition		Parity dec. - archive	
Seq.	L	E^*	$\beta\ (f)$	μ	$\beta\ (f)$	μ	$\beta\ (f)$	μ
3d1	**20**	**-11**	-11 (57)	-10.48	-11 (69)	**-10.64**	-11 (64)	-10.51
3d2	**24**	**-13**	-13 (23)	-11.30	-13 (34)	**-11.70** ▲	-13 (27)	-11.59
3d3	**25**	**-9**	-9 (57)	-8.48	-9 (70)	**-8.65** ▲	-9 (62)	-8.51
3d4	**36**	**-18**	-18 (10)	-15.19	-18 (13)	**-15.74** ▲	-18 (8)	-15.30
3d5	**46**	**-32**	-30 (2)	-23.87	-30 (1)	**-25.38** ▲	-30 (1)	-24.56
3d6	**48**	**-31**	-29 (1)	-22.79	-29 (2)	**-24.42** ▲	-28 (3)	**-23.64** ▲
3d7	**50**	**-32**	-25 (6)	-20.64	-27 (1)	**-22.07** ▲	-27 (1)	-21.22
3d8	**58**	**-44**	-35 (1)	-27.34	-36 (1)	**-29.02** ▲	-35 (1)	-27.96
3d9	**60**	**-52**	-46 (1)	-37.20	-47 (1)	**-40.03** ▲	-47 (1)	**-38.81** ▲
3d10	**64**	**-55**	-45 (1)	-35.59	-46 (1)	**-37.69** ▲	-43 (2)	-36.51
3d11	**67**	**-56**	-38 (2)	-30.17	-39 (2)	**-32.65** ▲	-38 (2)	-31.17
3d12	**88**	**-72**	-47 (1)	-36.22	-49 (1)	**-39.85** ▲	-48 (1)	**-38.09** ▲
3d13	**103**	**-56**	-40 (1)	-29.97	-41 (1)	**-31.31** ▲	-38 (1)	-29.94
3d14	**124**	**-71**	-43 (4)	-34.51	-48 (1)	**-36.97** ▲	-47 (1)	-35.04
3d15	**136**	**-80**	-51 (1)	-37.26	-52 (1)	**-42.11** ▲	-50 (1)	**-40.43** ▲
OAP			68.31%		**72.20%**		70.00%	

stopping condition of 100,000 evaluations were adopted. Tables 1 and 2 present the obtained results for the two- and three-dimensional test cases, respectively.

Without using the archiving strategy, the parity decomposition improved the average performance of the algorithm in all the 15 two-dimensional test cases (see Table 1). For 6 out of them, such an improvement was statistically significant with regard to the conventional formulation, leading to an OAP increase of $(71.39 - 69.22) = 2.17\%$. The use of the nondominated solutions archive seems not to be favorable for the proposed multiobjectivization. However, even in this case it was possible to score better results than the conventional single-objective formulation for most of the instances, with a statistically important difference in 4 of them. Also, an increase of 1.25% for the OAP measure has been obtained.

As shown in Table 2, the proposed decomposition reached the lowest average energy for all the three-dimensional instances when using the basic, non-archiving, (1+1) EA. Statistical analysis indicates a significant outperformance

over the conventional single-objective formulation in all but one of the test cases. This was also reflected as an OAP increase of 3.89%. Again, the advantages of the multiobjective formulation were not as impressive when using the archiving (1+1) EA. Even so, the results were improved in most cases with regard to the conventional formulation. This performance increase was found to be statistically significant in 4 of the instances. The OAP measure was improved by 1.69%.

5.2 Results for the Genetic Algorithm

In this section, the obtained results regarding the implemented genetic algorithm (GA) are analyzed. The behavior of this algorithm is sensitive to several parameters. Therefore, different parameter settings have been considered in order to identify the most convenient adjustment for the compared approaches.

Three different recombination and mutation probabilities were considered: $p_c = \{0.8, 0.9, 1.0\}$ and $p_m = \{\frac{1}{L-1}, 0.01, 0.05\}$. Also, the effects of preventing duplicate individuals (clones) from the population are analyzed. This leads to a total of 18 different parameter configurations for the GA. The population size was fixed to $N = 100$ in all cases, and the algorithm was allowed to run until a maximum number of 100,000 function evaluations was reached. Figure 2 presents the overall average performance (OAP) for both the conventional and the proposed formulations when using the different GA parameter settings.

From this figure, it is possible to note that there was a performance difference in favor of the proposed decomposition, in all the cases. On the one hand, the algorithm seemed not to be seriously affected when varying the recombination probability (p_c). On the other hand, it responded positively to the increased mutation rate, being $p_m = 0.05$ the fixed value which provided the best performance in all the cases. Finally, it can be seen that an important performance increase was achieved in all cases when duplicates avoidance was enabled.

In order to provide a more detailed analysis, the parameters adjustment which allowed each of the approaches to reach the highest OAP value has been selected. For the two-dimensional instances, a recombination probability of $p_c = 0.8$ was chosen for the conventional formulation and $p_c = 1.0$ for the proposed one. For

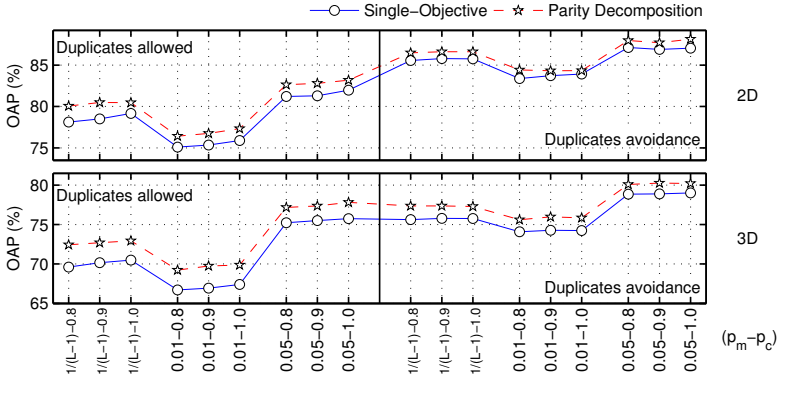


Fig. 2. Performance for all configurations of the GA

the three-dimensional test cases, $p_c = 1.0$ and $p_c = 0.9$ were respectively selected. A mutation probability of $p_m = 0.05$ and enabled duplicates avoidance hold for all cases. The obtained results are presented in Tables 3 and 4.

Table 3. Results for the GA on two-dimensional benchmarks (best settings)

			Single-objective		Parity decomposition	
Seq.	L	E^*	β (f)	μ	β (f)	μ
2d1	18	-4	-4 (69)	-3.69	-4 (78)	**-3.78**
2d2	18	-8	-8 (92)	**-7.92**	-8 (91)	-7.91
2d3	18	-9	-9 (68)	-8.68	-9 (73)	**-8.73**
2d4	20	-9	-9 (99)	**-8.99**	-9 (93)	**-8.93** ▼
2d5	20	-10	-10 (87)	-9.75	-10 (94)	**-9.89**
2d6	24	-9	-9 (62)	-8.60	-9 (69)	**-8.69**
2d7	25	-8	-8 (47)	-7.40	-8 (49)	**-7.47**
2d8	36	-14	-13 (12)	-11.45	-14 (2)	**-11.49**
2d9	48	-23	-21 (2)	-17.85	-23 (1)	**-18.30**
2d10	50	-21	-21 (4)	-18.27	-21 (1)	**-18.54**
2d11	60	-36	-34 (1)	-30.27	-34 (1)	**-30.54**
2d12	64	-42	-36 (2)	**-30.94**	-35 (3)	-30.75
2d13	85	-53	-49 (1)	-41.75	-48 (1)	**-42.57** ▲
2d14	100	-48	-44 (1)	-36.74	-43 (1)	**-37.74** ▲
2d15	100	-50	-43 (2)	-37.14	-43 (1)	**-38.28** ▲
OAP			87.13%		**88.13%**	

As shown in Table 3, the parity decomposition increased the average performance of the algorithm for 12 out of the 15 two-dimensional test cases. Such an increase was statistically significant for the three largest sequences. The single-objective formulation performed best for the remaining three instances, with a statistically important difference in one of them. An increase of $(88.13 - 87.13) = 1\%$ in the OAP measure was obtained by using the proposed formulation.

Table 4. Results for the GA on three-dimensional benchmarks (best settings)

			Single-objective		Parity decomposition	
Seq.	L	E^*	β (f)	μ	β (f)	μ
3d1	20	-11	-11 (100)	**-11.00**	-11 (100)	**-11.00**
3d2	24	-13	-13 (95)	**-12.94**	-13 (97)	**-12.94**
3d3	25	-9	-9 (72)	-8.71	-9 (87)	**-8.87** ▲
3d4	36	-18	-18 (12)	-15.91	-18 (31)	**-16.54** ▲
3d5	46	-32	-32 (1)	-27.72	-32 (1)	**-28.12**
3d6	48	-31	-31 (1)	-26.59	-30 (3)	**-26.89**
3d7	50	-32	-30 (1)	-26.43	-29 (12)	**-26.70**
3d8	58	-44	-37 (1)	-32.39	-37 (3)	**-33.03** ▲
3d9	60	-52	-50 (1)	-43.46	-50 (1)	**-44.56** ▲
3d10	64	-55	-52 (1)	-46.12	-53 (1)	**-46.15**
3d11	67	-56	-41 (1)	-36.39	-43 (1)	**-37.36** ▲
3d12	88	-72	-50 (5)	-44.02	-54 (1)	**-44.85** ▲
3d13	103	-56	-41 (1)	-34.99	-43 (1)	**-35.78** ▲
3d14	124	-71	-51 (1)	-41.83	-50 (1)	**-42.80** ▲
3d15	136	-80	-52 (2)	-45.51	-56 (2)	**-46.43**
OAP			79.01%		**80.26%**	

Regarding the three-dimensional instances, it can be seen from Table 4 that the best average performance of the algorithm was obtained in all cases when using the proposed multiobjectivization. Statistical analysis has shown that for 8 out of the 15 test cases, the achieved improvement was significant with regard to the conventional formulation. The OAP measure was increased by 1.25%.

6 Conclusions and Future Work

The multiobjectivization of the HP model for protein structure prediction was proposed. An alternative two-objective formulation for this problem was defined by means of the decomposition of the original objective function. This approach, called the parity decomposition, is based on the fact that hydrophobic interactions in the lattice are only possible between amino acids of opposite parity.

Experiments were conducted using different evolutionary algorithms and a total of 30 HP instances. Both two- and three-dimensional lattices were explored. As the main finding, the proposed parity decomposition increased the average performance of the implemented algorithms in most of the cases. Thus, the suitability of this approach was demonstrated. The obtained results support previous evidence regarding the effectiveness of multiobjectivization to overcome search difficulties such as that of becoming trapped in local optima [15, 20].

Although still competitive, the proposed multiobjectivization was negatively affected by the use of the nondominated solutions archive within the (1+1) EA. This is contrary to what is expected in multiobjective optimization, where it is the goal to converge towards different trade-offs among the problem objectives. Nevertheless, the addressed problem of this study is actually a single-objective problem, so that maintaining an approximation set of nondominated solutions becomes not as important. In addition, the archiving strategy influences the acceptance criterion, restricting the exploration behavior of the algorithm.

Even when the performance of the GA was increased in most cases by using the proposed formulation, such an increase was not as remarkable as that achieved for the (1+1) EA. This can be explained by the fact that population-based approaches are inherently less susceptible to get stuck in local optima. On the other hand, the multiobjectivized formulation enabled diversity promotion in the objective space, thus enhancing the exploration capabilities of the algorithm.

To the best of authors' knowledge, this is the first study on the application of multiobjective optimization techniques to the HP model for protein structure prediction. It is important to remark that the aim was not to improve the state-of-the-art results, but rather to evaluate the impact of using the proposed multiobjectivization on the resolution of this problem. The findings of this study motivate further research in this direction. An important issue would be to investigate whether the proposed parity decomposition can be incorporated in order to improve the performance of established state-of-the-art algorithms. Also, the conflicting relationship between the objectives of the proposed formulation needs to be analyzed. Finally, the multiobjectivization of the HP model by means of the addition of supplementary objectives has not been addressed yet.

References

1. Anfinsen, C.: Principles that Govern the Folding of Protein Chains. Science 181(4096), 223–230 (1973)
2. Becerra, D., Sandoval, A., Restrepo-Montoya, D., Nino, L.: A Parallel Multi-Objective Ab Initio Approach for Protein Structure Prediction. In: IEEE International Conference on Bioinformatics and Biomedicine, pp. 137–141 (2010)

3. Berger, B., Leighton, T.: Protein Folding in the Hydrophobic-Hydrophilic (HP) Model is NP-complete. In: International Conference on Research in Computational Molecular Biology, pp. 30–39. ACM, New York (1998)
4. Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: Do Additional Objectives Make a Problem Harder? In: Genetic and Evolutionary Computation Conference, pp. 765–772. ACM, London (2007)
5. Băutu, A., Luchian, H.: Protein Structure Prediction in Lattice Models with Particle Swarm Optimization. In: Dorigo, M., Birattari, M., Di Caro, G.A., Doursat, R., Engelbrecht, A.P., Floreano, D., Gambardella, L.M., Groß, R., Şahin, E., Sayama, H., Stützle, T. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 512–519. Springer, Heidelberg (2010)
6. Chira, C.: A Hybrid Evolutionary Approach to Protein Structure Prediction with Lattice Models. In: IEEE Congress on Evolutionary Computation, New Orleans, LA, USA, pp. 2300–2306 (2011)
7. Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A., Yannakakis, M.: On the Complexity of Protein Folding. In: ACM Symposium on Theory of Computing, pp. 597–603. ACM, Dallas (1998)
8. Cutello, V., Narzisi, G., Nicosia, G.: A Multi-Objective Evolutionary Approach to the Protein Structure Prediction Problem. Journal of The Royal Society Interface 3(6), 139–151 (2006)
9. Cutello, V., Nicosia, G., Pavone, M., Timmis, J.: An Immune Algorithm for Protein Structure Prediction on Lattice Models. IEEE Transactions on Evolutionary Computation 11(1), 101–117 (2007)
10. Day, R., Zydallis, J., Lamont, G.: Solving the Protein structure Prediction Problem Through a Multi-Objective Genetic Algorithm. In: IEEE/DARPA International Conference on Computational Nanoscience, pp. 32–35 (2002)
11. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
12. Dill, K.: Theory for the Folding and Stability of Globular Proteins. Biochemistry 24(6), 1501–1509 (1985)
13. Garza-Fabre, M., Rodriguez-Tello, E., Toscano-Pulido, G.: Comparing Alternative Energy Functions for the HP Model of Protein Structure Prediction. In: IEEE Congress on Evolutionary Computation, New Orleans, LA, USA, pp. 2307–2314 (2011)
14. Handl, J., Lovell, S.C., Knowles, J.: Investigations into the Effect of Multiobjectivization in Protein Structure Prediction. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 702–711. Springer, Heidelberg (2008)
15. Handl, J., Lovell, S.C., Knowles, J.: Multiobjectivization by Decomposition of Scalar Cost Functions. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 31–40. Springer, Heidelberg (2008)
16. Hoque, M., Chetty, M., Lewis, A., Sattar, A.: Twin Removal in Genetic Algorithms for Protein Structure Prediction Using Low-Resolution Model. IEEE/ACM Transactions on Computational Biology and Bioinformatics 8(1), 234–245 (2011)
17. Islam, M., Chetty, M., Murshed, M.: Novel Local Improvement Techniques in Clustered Memetic Algorithm for Protein Structure Prediction. In: IEEE Congress on Evolutionary Computation, New Orleans, LA, USA, pp. 1003–1011 (2011)

18. Jähne, M., Li, X., Branke, J.: Evolutionary Algorithms and Multi-Objectivization for the Travelling Salesman Problem. In: Genetic and Evolutionary Computation Conference, pp. 595–602. ACM, Montreal (2009)
19. Jensen, M.: Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation. Journal of Mathematical Modelling and Algorithms 3, 323–347 (2004)
20. Knowles, J.D., Watson, R.A., Corne, D.W.: Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 269–283. Springer, Heidelberg (2001)
21. Lochtefeld, D., Ciarallo, F.: Helper-Objective Optimization Strategies for the Job-Shop Scheduling Problem. Applied Soft Computing 11(6), 4161–4174 (2011)
22. Lopes, H.S.: Evolutionary Algorithms for the Protein Folding Problem: A Review and Current Trends. In: Smolinski, T.G., Milanova, M.G., Hassanien, A.-E. (eds.) Comp. Intel. in Biomed. and Bioinform. SCI, vol. 151, pp. 297–315. Springer, Heidelberg (2008)
23. Neumann, F., Wegener, I.: Can Single-Objective Optimization Profit from Multi-objective Optimization? In: Multiobjective Problem Solving from Nature. Natural Computing Series, pp. 115–130. Springer, Heidelberg (2008)
24. Santana, R., Larranaga, P., Lozano, J.: Protein Folding in Simplified Models With Estimation of Distribution Algorithms. IEEE Transactions on Evolutionary Computation 12(4), 418–438 (2008)
25. Santos, J., Diéguez, M.: Differential Evolution for Protein Structure Prediction Using the HP Model. In: Ferrández, J.M., Álvarez Sánchez, J.R., de la Paz, F., Toledo, F.J. (eds.) IWINAC 2011, Part I. LNCS, vol. 6686, pp. 323–333. Springer, Heidelberg (2011)
26. Segredo, E., Segura, C., Leon, C.: A Multiobjectivised Memetic Algorithm for the Frequency Assignment Problem. In: IEEE Congress on Evolutionary Computation, New Orleans, LA, USA, pp. 1132–1139 (2011)
27. Segura, C., Segredo, E., González, Y., León, C.: Multiobjectivisation of the Antenna Positioning Problem. In: Abraham, A., Corchado, J.M., González, S.R., De Paz Santana, J.F. (eds.) International Symposium on DCAI. AISC, vol. 91, pp. 319–327. Springer, Heidelberg (2011)
28. Segura, C., Segredo, E., León, C.: Parallel Island-Based Multiobjectivised Memetic Algorithms for a 2D Packing Problem. In: Genetic and Evolutionary Computation Conference, pp. 1611–1618. ACM, Dublin (2011)
29. Shmygelska, A., Hoos, H.: An Ant Colony Optimisation Algorithm for the 2D and 3D Hydrophobic Polar Protein Folding Problem. BMC Bioinformatics 6(1), 30 (2005)
30. Soares Brasil, C., Botazzo Delbem, A., Ferraz Bonetti, D.: Investigating Relevant Aspects of MOEAs for Protein Structures Prediction. In: Genetic and Evolutionary Computation Conference, pp. 705–712. ACM, Dublin (2011)
31. Unger, R.: The Genetic Algorithm Approach to Protein Structure Prediction. In: Applications of Evolutionary Computation in Chemistry, Structure & Bonding, vol. 110, pp. 2697–2699. Springer, Heidelberg (2004)
32. Vite-Silva, I., Cruz-Cortés, N., Toscano-Pulido, G., de la Fraga, L.G.: Optimal Triangulation in 3D Computer Vision Using a Multi-objective Evolutionary Algorithm. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 330–339. Springer, Heidelberg (2007)
33. Zhao, X.: Advances on Protein Folding Simulations Based on the Lattice HP models with Natural Computing. Applied Soft Computing 8(2), 1029–1040 (2008)

Multi-Pareto-Ranking Evolutionary Algorithm

Wahabou Abdou, Christelle Bloch, Damien Charlet, and François Spies

University of Franche-Comté / FEMTO-ST Institute
1 Cours Leprince-Ringuet, 25201 Montbéliard, France
firstname.lastname@univ-fcomte.fr

Abstract. This paper proposes a new multi-objective genetic algorithm, called GAME, to solve constrained optimization problems. GAME uses an elitist archive, but it ranks the population in several Pareto fronts. Then, three types of fitness assignment methods are defined: the fitness of individuals depends on the front they belong to. The crowding distance is also used to preserve diversity. Selection is based on two steps: a Pareto front is first selected, before choosing an individual among the solutions it contains. The probability to choose a given front is computed using three parameters which are tuned using the design of experiments. The influence of the number of Pareto fronts is studied experimentally. Finally GAME's performance is assessed and compared with three other algorithms according to the conditions of the CEC 2009 competition.

Keywords: Multiobjective optimization, genetic algorithm, Pareto ranking, multiple fronts.

1 Introduction

Many multi-objective evolutionary algorithms (MOEAs) use an elitist archive based on Pareto domination. The main difficulty of these MOEAs often lies in maintaining diversity. Thus, various solutions have been proposed as the crowding distance [6], metrics based on diversity [18] or specific definitions of fitness [8]. In the same way, the algorithm proposed in this paper, named Genetic Algorithm with Multiple parEto sets (GAME), rests on the concept of Pareto ranking. But it uses several Pareto fronts to rank individuals. Besides, the formula used to compute the fitness of individuals varies according to the set they belong to. Finally selection operators (for reproduction and for replacement) work in two steps: selecting one of the Pareto sets and selecting one individual in this set. The probability that each Pareto set is selected both depends on its rank and its size. Individuals belonging to the selected Pareto set are then compared using a fitness-based tournament. Defining various kinds of fitness depending on the non-domination rank preserves both quality and diversity. Section 2 quickly describes previous work related to this subject : some classical MOEAs based on Pareto archives, tuning parameters and managing constraints in such algorithms. Section 3 details the proposed algorithm itself. Section 4 first explains how the parameters used to define the selection probability of each Pareto front

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 194–205, 2012.

are tuned using the design of experiments. This section then experimentally studies the benefit obtained by using several Pareto fronts. Finally that section presents the experiments performed to validate this approach, particularly the algorithms compared with it in accordance with the CEC2009 competition conditions, and the assessment of GAME efficiency. Section 5 concludes by highlighting and gives the main prospects of this work.

2 Multi-Objective Evolutionary Algorithms (MOEAs) in Literature

2.1 Pareto-Based MOEAs

Among several evolutionary algorithms which archive non-dominated solutions, we can cite Strength Pareto Evolutionary Algorithm 2 (SPEA2) [18] or a newer method such as the Fast Pareto Genetic Algorithm (FastPGA) [8].

SPEA2 is an elitist multi-objective evolutionary algorithm that relies on Pareto-based and archiving techniques. SPEA2 assigns a strength to each S solution of the current population and the archive. This strength represents the number of solutions dominated by S. The fitness of each solution equals the summation of the strength of solutions that it is dominated by. When two solutions have the same fitness value, they are discriminated according to a metric based on the density (an adaptation of the k-th nearest neighbor method) [13].

FastPGA divides the solutions of the current population into two fronts (non-dominated and dominated sets) using the Pareto dominance principles. The fitness of non-dominated solutions is calculated using the crowding distance proposed by Deb et al. [5]. For solutions in the dominated front, FastPGA uses an extension of SPEA2's fitness assignment method. Indeed, the fitness of each dominated solution equals the summation of the strength values of all the solutions it dominates minus the summation of the strength values of all the solutions by which it is dominated. This reduces the chances that two solutions may have the same value of fitness. FastPGA also uses a specific mechanism to regulate population size over generations.

2.2 Tuning of MOEA Parameters

Evolutionary algorithms include several parameters the tuning of which is generally difficult. Their values might significantly impact the quality of the solutions provided by the algorithm. A high mutation probability for instance would favor diversity and limit the exploitation of zones where good solutions may have been found. The algorithm would then nearly behave like a random search strategy. Such parameters are often tuned empirically. But it is difficult to find values which are both suitable and robust (i.e. achieving good enough performance without being too sensitive to the tackled optimization problems). This approach often requires a great number of executions and the knowledge of a human decision-maker to select appropriate values. That is why, in recent years, several approaches have been proposed to design and/or tune EAs statistically in order to better guarantee their performance and/or robustness [9] [2].

2.3 Constraint-Handling in MOEAs

Some optimization problems include constraints associated with the search space (set of variables) or the objective space (set of objective functions). Only points that satisfy these constraints are considered as feasible solutions of the optimization problem. Michalewicz and Schoenauer [12] define the types of approaches to handle such constraints in EAs. There are methods that preserve solution feasibility whenever possible. They use some operators (crossover and mutation) which only create solutions belonging to the set of feasible solutions. There also exists methods that include a penalty function in the calculation of the fitness of each solution. This allows to weaken the fitness of solutions that violate constraints. Another approach to handle constraints combine evolutionary algorithms and other optimization methods. They may repeatly use EAs. For instance, Surry et al. [15] rank individuals according to both objective values and constraint violations, which amounts to adding some objectives to the problem.

The genetic algorithm proposed in this paper uses a penalty-based approach to manage the constraints (see section 3.1). In addition, this algorithm uses both the computation of several Pareto fronts like NSGA-II and a definition of fitness which is different for each front like SPEA 2. The proposed solution combines the advantages of these two famous algorithms. Nevertheless contrary to NSGA-II structuring the population in several Pareto fronts is not only used for the replacement selection, it also impacts the recombination process. These characteristics have led to the implementation of a new selection strategy that takes into account the multiplicity of fronts and the fitness functions that depend on rank of the front. The benefit related to the number of fronts and the tuning of this new selection process parameters have been studied statistically.

The following section describes the proposed algorithm while particularly detailing its main characteristics.

3 A Multiple-Pareto-Ranking Genetic Algorithm

The proposed algorithm includes a wide-ranking process and a specific method of fitness assignment. The main goal of these operations is both to ensure a good exploration of the space search and the convergence toward the optimal solution(s). This algorithm is named Genetic Algorithm with Multiple parEto sets (GAME). It rests on a classical global scheme and is quite like other MOEAs developed in literature. Initial population is generated randomly. Individuals represent solutions which are made of several integer or float variables, using binary simulated encoding. This representation is quite classical [14]: each variable is represented as a binary chain. The chain length (number of bits) depends on the wished precision. For instance, when representing a variable that takes values in the interval [0; 10], with a precision equals to 10^{-2}, requires 10 bits. The chains 0000000000 and 1111111111 will respectively represent 0 and 10. 1,23 will be encoded as 0001111110.

Crossover and mutation operators are also commonly used in this kind of representation. They are 2-point crossover and bitflip mutation [5]. GAME also uses parallel , based on master-slave model [4].

Besides, GAME differs from well-known MOEAs in several ways. These differences deal with the following steps: the ranking of individuals, based on Pareto dominance (3.1), fitness assignment (3.2), and the associated selection strategies for both recombination and replacement (3.3).

3.1 Ranking

GAME uses the objective values computed at the evaluation step to rank individuals. This ranking is based on the principle of non-dominated sorting (Pareto dominance). Pareto solutions are those for which improvement in one objective implies the worsening of at least one other objective. All the non-dominated individuals are added to the first Pareto front PF_1. This process is repeated successively three times with the remaining subset of individuals to build the second, the third and the fourth Pareto fronts (PF_2, PF_3 and PF_4). Finally, the remaining subset solutions are gathered in the set of dominated solutions PF_5.

When the problem has some constraints, they are handled during the construction of the Pareto fronts. GAME uses a penality-based method to take constraints intoaccount. In order to preserve the primacy of solutions which respect the limits, those which violate them are banned in the first Pareto front. However, when PF_1 is empty, solutions that do not satisfy the constraints can be accepted. This situation often occurs in the first generations. But it is quickly corrected as the solutions which respect the constraints have a higher probability of reproduction and thus transmit their characters to their offspring.

3.2 Fitness Assignment

Since it is not obvious to compare two solutions in the same Pareto front (improvement in an objective is always to the detriment of at least one other), a fitness value is assigned to each solution.

The computation of the value of fitness depends on the Pareto rank of the solution. GAME distinguishes three types of fitness: one for the best solutions (the archive of non-dominated solutions and the first Pareto front), the second for the solutions of the second Pareto front and the third one for the other fronts. These fitness values are only used within the same front. The fitness of two individuals of different fronts cannot be compared. In this case, the comparison will only focus on the ranks of their respective fronts.

Fitness for the Archive and the First Pareto Front Solutions. During the execution of GAME, the first Pareto front (PF_1) contains the best solutions of the current population. The archive of non-dominated solutions contains all the best solutions evaluated since the first generation. Note that a solution of PF_1 can dominate some solutions in the archive. These two sets represent solutions that are closest to the optimal solutions (PF^*). While trying to get closer to

PF^*, the algorithm must preserve diversity in the best solutions set. GAME assigns to the archived solutions and PF_1 a fitness based on the *crowding distance* [6] which is a diversity indicator. This indicator provides information on the distribution of solutions in the front. To avoid bias due to the sizes of ranges of the objective functions, GAME uses a normalized crowding distance.

Fitness for the Second Pareto Front Solutions. The solutions of the second Pareto front (PF_2) are the second best set of solutions in a population. GAME favors solutions of the PF_2 that are closest to PF_1. The convergence indicator used to measure the proximity of solutions of the two fronts is the Generational distance. Using this indicator as fitness enables the solutions of PF_2 that are closest to PF_1 to have a greater probability to be selected for recombination and/or replacement.

Fitness for the Remaining Solutions. For solutions that do not belong to any of the three previous sets (archive of non-dominated solutions, first and second Pareto fronts), a new indicator is used to calculate the values of fitness. This is the *gain* (see Equation 1).

The gain of an $\overrightarrow{x_i}$ solution with respect to $\overrightarrow{x_j}$ in accordance with the f_k objective function is called $gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k)$. It expresses the improvement provided by $\overrightarrow{x_i}$ in comparison to $\overrightarrow{x_j}$ in the selected function. The gain varies from -1 to 1. A negative value indicates that the first solution is worse than the second one according to f_k. A null gain means that both solutions are equivalent.

$$gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k) = \frac{\lambda(f_k(\overrightarrow{x_i}) - f_k(\overrightarrow{x_j}))}{Max(f_k(\overrightarrow{x_i}), f_k(\overrightarrow{x_j}))} \tag{1}$$

where λ is a coefficient: 1 for a maximization problem; -1 for a minimization.

The fitness of each individual is the sum of gains (for all objectives) compared to other solutions in the same front (see Equation 2).

$$fitness(\overrightarrow{x_i}) = \sum_{j=1}^{|PF|} \sum_{k=1}^{m} gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k) \tag{2}$$

where $|PF|$ is the Pareto front size and m the number of objectives functions.

3.3 Selection Operator

To ensure both exploration and exploitation, a kind of fitness is associated to each Pareto front. This is actually a selection probability of the front. Indeed, GAME selects individuals in two steps. The first one consist in selecting a Pareto front within the five built. Then, a solution is chosen from those belonging to the front using a binary tournament selection.

The selection of Pareto fronts is based on a biased roulette wheel where the part of each front depends on both its rank and its cardinality. The probability of selection associated with each front is given by equation 3.

$$P(PF_i) = \frac{\delta(PF_i) * |PF_i|}{\sum_{i=1}^{n}[\delta(PF_i) * |PF_i|]} \quad (3)$$

where:
$P(PF_i)$ is the probability of selection associated with the i^{th} Pareto front.
$|PF_i|$ is the cardinality of PF_i.
n is the total number of Pareto fronts. $\delta(PF_i)$ indicates a priority level associated with each front (see Equation 4).

$$\delta(PF_i) = a(n - i) + b \quad (4)$$

The coefficients a and b must be carefully chosen to preserve diversity within the sets of solutions (successive populations and archive), while improving individuals over generations. These parameters may differ for recombination and replacement. Therefore, the coefficients associated with recombination selection are named a and b, whereas those used in replacement selection are a' and b' (see Section 4.1).

Using both $\delta(PF_i)$ and the cardinality ensures the selection of the best solutions, prevents the dominated solutions from having very low values of fitness and preserves the diversity of the successive populations.

The utility and the efficiency of the proposed ranking mode and selection process were experimentally validated. This experimentation first studied the tuning of a, b, a' and b' for a given number of Pareto fronts. Secondly, a preliminary test phase focused on the sensitivity of GAME to the used number of Pareto Fronts, for the tuned values of $\delta(PF_i)$.

It is worth noting that besides its specific characteristics, presented in this section, GAME uses a parallel evaluation process based on a master-slave model [4]. The parallelization allows GAME to tackle problems with complex objective functions or with objective functions requiring simulation to assess their values. A variant of GAME, named ESBEA, has been applied, for instance, to a constrained optimization problem containing four objective functions evaluated by simulation, in the field of ad hoc mobile communication networks [1].

In this paper, the performance of GAME was assessed by comparison with previous results in literature. The following sections describe experimental conditions and summarize the main results.

4 Validation and Experimental Results

The experiments described in the following subsections aim at:

- tuning the probabilities associated with Pareto sets;
- validating the interest to have multiple Pareto fronts;
- assessing the performance of GAME compared with other methods presented in literature.

All experiments were performed on the optimization problems proposed in the CEC 2009 [17] competition, more particularly bi-objective constrained problems, according to the experimental conditions defined in this competition. These problems are detailed in [17].

GAME has been implemented using jMetal [7], a platform for the development, experimentation and study of metaheuristics. A population size equals to 100 and 300 generations were used to perform the 30,000 evaluations required for each execution in the CEC 2009 competition. Crossover rate and mutation rate were respectively equal to values recommended in literature: 0.8 and $\frac{1}{Number\ of\ variables}$ (in order to limit the number of parameters in the parameter study and focus on those involved in the proposed mechanisms).

4.1 Tuning GAME's Parameters

A parametric study based on the design of experiments (DOEs) has been performed to tune the priority assigned to each Pareto front built by GAME. DOEs permit to get a lot of information on parameters[1] and study their influence, while carrying out a minimum of tests [3].

The whole campain of tests has been based on JMP (9 Pro version)[2], a statistical software for expert data analysis, to create DOE and to analyze the provided results. This software has been used to create some test tables, in which each line corresponds to a given combination of the factors (in this case, a, b, a' and b' presented in Section 3.3) defining a given test. For each of these tests, two indicators (*IGD* and *Maximum Spread* [5]) have been used to measure the quality of the studied combination of parameters[3].

This parameter tuning was performed with a number of Pareto fronts set at 5, by varying a, b, a' and b' between 0 and 10 (integer values). JMP has generated a partial DOE table consisting of 44 tests to be performed. The benchmark used for these tests was Constrained Problem 1.

Figure 1 shows the results returned by JMP. For IGD and *Maximum Spread* indicators, the first four curves show the variations of these indicators based on the values taken by each factor. The last column gives a desirability function (for each indicator) with values between 0 and 1. A desirability value of 1 indicates that the used parameter values enable to get the optimal level for the considered indicator. In this parameter tuning, the desirability according to the IGD decreases when the IGD reaches high values. That is the inverse of the *Maximum Spread*. The last row of curves shows the levels of desirability with respect to both the IGD and *Maximum Spread* indicators.

Based on these results, the best value of desirability is achieved when $a = 6$, $b = 5$, $a' = 5$ and $b' = 4$.

[1] In DOEs these parameters are called factors.
[2] http://www.jmp.com
[3] The measured values are also called responses.

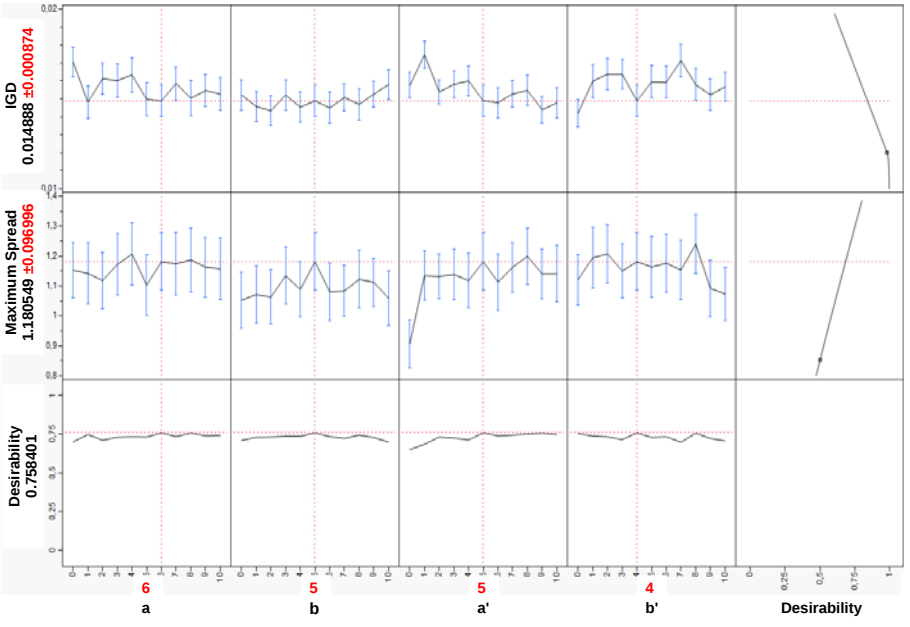


Fig. 1. Tuning the selection coefficients

4.2 Influence of Multiple Pareto Fronts

To measure the benefits of multiple Pareto fronts, four versions of GAME are compared, using the values of $\delta(PF_i)$ tuned with JMP. The first version builds two Pareto fronts (like classical algorithms). This version will serve as a reference. The other three versions build 3, 4 and 5 fronts. These values were chosen to study a steady increase of the number of fronts while limiting the duration of experimentations. They are compared to the "2 front" version in the constrained problem 1 of the CEC 2009 competition. The comparison focuses on convergence (IGD) and diversity (*Maximum Spread*) indicators.

Since the generation of the initial population of GAME does not use specific heuristics, individuals are created randomly. Therefore, the best of them often are of fair quality. This results in a relatively high value of the IGD in the first generations (see Figure 2). Fast enough, solutions that are better suited to the problem are produced (by recombination) and eliminate the worst individuals in the archive. This explains the sharp decrease in the IGD in the first 25 generations. Thereafter, the curves of the IGD continue to decline steadily but with a much lower slope. Beyond this behavior common to all scenarios, these results allow us to observe that the increase in the number of fronts improves the quality of final solutions in terms of proximity to the optimum.

During the first few generations, the *Maximum Spread* has relatively high values (see Figure 3). This can be explained by the fact that the archive contains

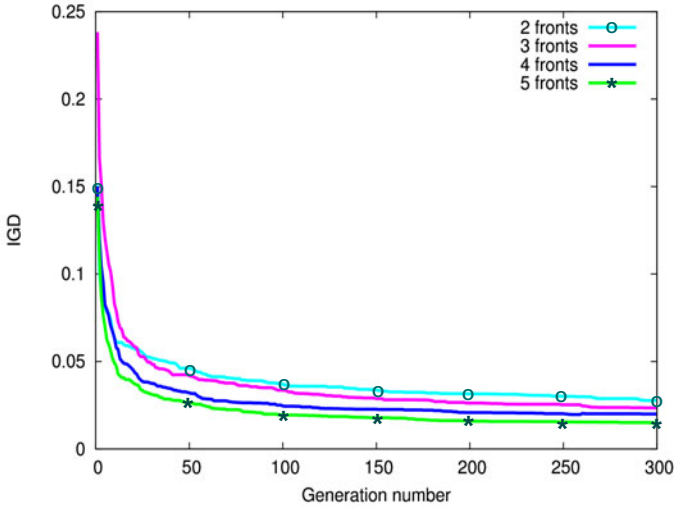


Fig. 2. IGD

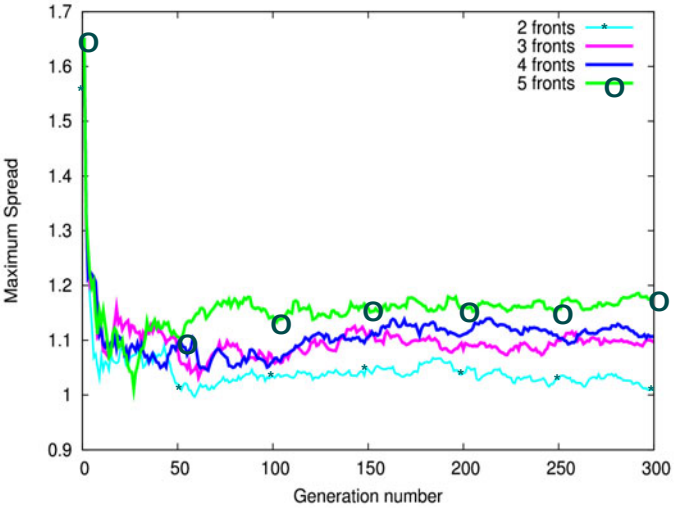


Fig. 3. Maximum spread

solutions that do not meet constraints. Indeed, the inclusion of such solutions stretches the front beyond the boundaries that are set by the constraints of the problem. The archive built by GAME is then refined, when solutions which respect the constraints and have good values of objective functions are found. This explains the drop in the *Maximum Spread* in the early generations. After the fall phase, the *Maximum Spread* is growing slightly. In addition, the slight variations in the curves of *Maximum Spread* over generations indicates that the archive is updated regularly. New non-dominated solutions are continually being

discovered. If initially all scenarios have the same behavior, one can observe that after a few dozen generations, the impact of multiple fronts becomes substantial.

4.3 Performance Assessment

The performance of GAME was assessed as if it had taken part in the CEC 2009 competition, in the "Constrained problems" category. This category includes bi-objective problems where the two objectives are to be minimized. Each problem includes 10 real-valued decision variables. GAME was compared to the three algorithms which got the best results in this category, using the same experimental conditions and the same performance criterion:

- for each constrained problem, 30 independent executions limited to 30,000 evaluations must be performed for each algorithm;
- then the mean value of IGD indicator (computed in the sets of final solutions for the 30 independent executions) must be used as comparison criterion. The IGD is to be minimized.

Table 1 provides, for the exhaustive list of bi-objective problems of the competition, the average of the IGD values obtained by the three compared algorithms: DMOEA-DD [11], LiuLi [10] and MTS [16]. These results permitted these algorithms to respectively reach the first rank, the second rank and the third rank during the CEC 2009 competition.

DMOEAD-DD (Dynamical Multiobjective Evolutionary Algorithm - Domain Decomposition) improves DMOEA [19] that used an aggregated *fitness* function including the notion of Pareto dominance, entropy and density (based on *crowding distance*). In the improved variant, authors split the search space into several subsets. DMOEA computes Pareto fronts for each of them. Genetic operators permit information exchange between these subsets.

LiuLi (concatenation of the authors'name : Hai-Lin Liu and Xueqiang Li) splits the search space into sub-areas in order to reduce algorithm complexity. Genetic operations, particularly reproduction, are performed in a single sub-area. Information exchange between areas is based on children, because they may be assigned to other areas.

Table 1. Mean values of IGD returned by GAME and the algorithms which participated to CEC2009 competition

Problem	**GAME**	**DMOEADD**	**LiuLi**	**MTS**
Constrained Problem 1	0.01489	0.01131	0.00085	0.01918
Constrained Problem 2	0.00042	0.0021	0.004203	0.02677
Constrained Problem 3	0.03462	0.056305	0.182905	0.10446
Constrained Problem 4	0.00742	0.00699	0.014232	0.01109
Constrained Problem 5	0.01227	0.01577	0.10973	0.02077
Constrained Problem 6	0.00181	0.01502	0.013948	0.01616
Constrained Problem 7	0.00545	0.01905	0.10446	0.02469

MTS (Multiple Trajectory Search) is an algorithm based on three local search methods. For each solution, MTS determines the method which corresponds to its neighborhood. This algorithm begins with a large search. The size of neighborhood is progressively reduced until it reaches a given minimal size. Then its size is set to its initial value and the regression re-starts.

Besides, GAME has been tested in the same conditions and the results it provided were added in table 1. This study shows that, globally, GAME reaches solutions which are quite close to the reference front. This results in low IGD values in table 1. Moreover, in a large majority of cases, GAME provides better results than those returned by the three best algorithms in the competition (in the "constrained problems" category).

5 Conclusion

To sum up, GAME is an elitist multi-objective genetic algorithm, based on the building of multiple Pareto fronts. This ranking strategy and the associated 2-step selection provided gains both in terms of proximity with optimal solutions and in terms of diversity in the set of final solutions returned by GAME. Finally, using the experimental conditions of the CEC 2009 competition showed that it would have been quite well ranked in this competition, which constitutes a promising result. In addition, GAME uses a parallel evaluation procedure based on a master-slave model. Although this model allowed the reduction in terms of the duration of the experimentations in previous work in the field of mobile networks [1]. Nevertheless, designing an asynchronous version of GAME would be very interesting. Such a version would no longer rely on an architecture where the master must wait until all the slaves have finished their task before building the next generation. Two other prospects are planned for this work. First, GAME should be applied in various real-life fields to test its sensitivity to the tackled problems more widely. This would permit to prove the genericity of the results provided in this paper. Besides, the authors are working on an adaptive version of GAME. This aims at making it more robust with regards to problem characteristics, rather than using statistically tuned parameters without varying the algorithm behavior during execution.

References

1. Abdou, W., Henriet, A., Bloch, C., Dhoutaut, D., Charlet, D., Spies, F.: Using an evolutionary algorithm to optimize the broadcasting methods in mobile ad hoc networks. Journal of Network and Computer Applications 34(6), 1794–1804 (2011)
2. Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M.: Experimental Methods for the Analysis of Optimization Algorithms. Springer, Germany (2010)
3. Box, G.E.P., Hunter, J.S., Hunter, W.G.: Statistics for experimenters. Design, innovation, and discovery, 2nd edn. Wiley Series in Probability and Statistics, xvii, p. 633. John Wiley & Sons, Hoboken (2005)
4. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers (2000)

5. Deb, K.: Multi-objective Optimization Using Evolutionary Computation. John Wiley & Sons (2003)
6. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans. Evol. Computation 6(2), 182–197 (2002)
7. Durillo, J.J., Nebro, A.J.: jmetal: A java framework for multi-objective optimization. Advances in Engineering Software 42(10), 760–771 (2011)
8. Eskandari, H., Geiger, C.D., Lamont, G.B.: FastPGA: A Dynamic Population Sizing Approach for Solving Expensive Multiobjective Optimization Problems. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 141–155. Springer, Heidelberg (2007)
9. Hippolyte, J.L., Bloch, C., Chatonnay, P., Espanet, C., Chamagne, D., Wimmer, G.: Tuning an evolutionary algorithm with taguchi methods. In: 5th Int. Conf. on Soft Computing as Transdisciplinary Science and Technology, CSTST 2008, pp. 265–272. ACM Press (2008)
10. Liu, H.L., Li, X.: The multiobjective evolutionary algorithm based on determined weight and sub-regional search. In: Proc. of 11th Conf. on Congress on Evolutionary Computation, CEC 2009, Trondheim, Norway, pp. 1928–1934 (2009)
11. Liu, M., Zou, X., Chen, Y., Wu, Z.: Performance assessment of dmoea-dd with cec 2009 moea competition test instances. In: IEEE Congress on Evolutionary Computation, pp. 2913–2918 (2009)
12. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evol. Comput. 4, 1–32 (1996)
13. Silverman, B.W.: Density estimation: for statistics and data analysis. Chapman and Hall, London (1986)
14. Srinivas, M., Patnaik, L.M.: Genetic algorithms: A survey. IEEE Computer 27(6), 17–26 (1994)
15. Surry, P., Radcliffe, N., Boyd, I.: A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks: The Comoga Method. In: Fogarty, T.C. (ed.) AISB-WS 1995. LNCS, vol. 993, pp. 166–180. Springer, Heidelberg (1995)
16. Tseng, L.Y., Chen, C.: Multiple trajectory search for unconstrained/constrained multi-objective optimization. In: Proc. of 11th Conf. on Congress on Evolutionary Computation, CEC 2009, pp. 1951–1958. IEEE Press, Trondheim (2009)
17. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the cec 2009 special session and competition. Tech. rep., School of Computer Science and Electronic Engineering (2009)
18. Zitzler, E., Laumanns, M., Thiele, L.: Spea2: Improving the strength pareto evolutionary algorithm. Tech. rep., Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich (2001)
19. Zou, X., Chen, Y., Liu, M., Kang, L.: A new evolutionary algorithm for solving many-objective optimization problems. IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics: a Publication of the IEEE Systems, Man, and Cybernetics Society 38(5), 1402–1412 (2008)

Pareto Local Search Algorithms for Anytime Bi-objective Optimization

Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{jeremie.dubois-lacoste,manuel.lopez-ibanez,stuetzle}@ulb.ac.be

Abstract. Pareto local search (PLS) is an extension of iterative improvement methods for multi-objective combinatorial optimization problems and an important part of several state-of-the-art multi-objective optimizers. PLS stops when all neighbors of the solutions in its solution archive are dominated. If terminated before completion, it may produce a poor approximation to the Pareto front. This paper proposes variants of PLS that improve its anytime behavior, that is, they aim to maximize the quality of the Pareto front at each time step. Experimental results on the bi-objective traveling salesman problem show a large improvement of the proposed anytime PLS algorithm over the classical one.

1 Introduction

Multi-objective optimization deals with problems for which solutions are evaluated according to multiple criteria. These criteria are often in conflict and, hence, different solutions can show different possible trade-offs between the objectives that must be optimized. If there is no *a priori* knowledge about the preferences of the decision maker, multi-objective problems are usually tackled by determining the set of all Pareto optimal solutions from which the decision maker can select, *a posteriori*, one solution or extract some conclusions.

Pareto local search (PLS) is an extension of iterative improvement algorithms for single-objective problems to the multi-objective case [13]. PLS produces high-quality results when used as a stand-alone algorithm [13], and even better results when used as a component of hybrid algorithms, where PLS starts from a good set of initial solutions. In fact, PLS is a crucial component of the best algorithms known for the multi-objective traveling salesman problem [12] with two and three objectives, various bi-objective permutation flowshop problems [4], and the bi-objective multi-dimensional knapsack problem [11].

Even when starting from a good approximation to the Pareto front, PLS can require a long time to terminate [13,4]. Hence, many applications define an additional termination criterion, either in terms of a maximum number of solutions to be explored or computation time. In many real-life situations, however, the available computation time may be unknown, and it is therefore desirable to obtain the best possible output whenever the algorithm is stopped. Unfortunately, PLS was originally designed without taking into account its *anytime behavior* [15].

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 206–217, 2012.

Although there is much work on anytime algorithms for single-objective optimization problems [15,9], there is little research on anytime multi-objective optimization algorithms [5]. Some papers on multi-objective evolutionary algorithms report the quality obtained along time among other evaluations of the performance, to show the convergence ability of the algorithms, but their goal is not to obtain a good anytime behavior. By focusing on improving the anytime behavior of PLS, similarly to [5], our aim is also to better formalize the notion of anytime behavior for multi-objective algorithms.

In this paper, we study alternatives to the algorithmic components of the classical PLS with the goal of improving its anytime behavior. In particular, we propose variants that switch strategies during the run of the algorithm. We consider the bi-objective traveling salesman problem (bTSP) as a case study. The traveling salesman problem (TSP) is a well-known $\mathcal{NP}$-hard combinatorial problem widely used to assess the performance of optimization algorithms and meta-heuristics. In its bi-objective variant, two cost values are assigned to each edge of a graph, and each of the two objective functions is computed with respect to the corresponding cost value. The bTSP is a prominent benchmark problem in the study of multi-objective optimization algorithms [7,14,5].

2 Anytime Pareto Local Search

PLS is an iterative improvement method for solving multi-objective combinatorial optimization problems. The acceptance criterion in PLS is based on Pareto dominance. Given two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^2$, we say that $\boldsymbol{u}$ *dominates* $\boldsymbol{v}$ ($\boldsymbol{u} \prec \boldsymbol{v}$) iff $\boldsymbol{u} \neq \boldsymbol{v}$ and $u_i \leq v_i,\ i = 1, 2$, assuming, without loss of generality, that both objectives must be minimized. Algorithm 1 illustrates the PLS framework. Given is an initial set of mutually nondominated solutions, called *archive*. The solutions are initially marked as unexplored (line 2). PLS then iteratively applies the following steps. First, a solution s is chosen randomly among all unexplored ones (*selection step*, line 5). Then, the neighborhood of s, $\mathcal{N}(s)$, is fully explored and all neighbors that are nondominated w.r.t. s and any solution in the archive $\mathcal{A}$ are added to $\mathcal{A}$ (lines 6 to 11). Solutions in $\mathcal{A}$ that are dominated by the newly added solutions are removed (procedure Update in line 9). Once the neighborhood of s has been fully explored, s is marked as explored (line 10). When all solutions have been explored, and no more nondominated solutions can be discovered, the algorithm stops. Next we discuss the three main algorithmic components of PLS.

Selection Step. In many combinatorial optimization problems, solutions that are close to each other in the solution space are also close in the objective space. Transferring this fact to multi-objective problems, we may expect to fill existing "gaps" around the image of a solution in the Pareto front by exploring the neighborhood of it. In the original PLS, the next solution to be explored is selected randomly among the ones not explored, without considering possibly existing "gaps". Instead, PLS could select the solution whose neighborhood has the largest potential of improving the current Pareto front approximation.

Algorithm 1. Pareto Local Search

```
Input: An initial set of nondominated solutions A_0
explored(s) := FALSE   ∀s ∈ A_0
A := A_0
repeat
    s := select randomly a solution from A_0      // Selection step
    for each s' ∈ N(s) do      // Neighborhood exploration
        if s ⊀ s' then      // Acceptance criterion
            explored(s') := FALSE
            A := Update(A, s')
    explored(s) := TRUE
    A_0 := {s ∈ A | explored(s) = FALSE}
until A_0 = ∅
Output: A
```

A measure of the quality of a single solution is given by its hypervolume contribution. The hypervolume measures the volume of the objective space weakly dominated by a set of solutions [17]. The larger the hypervolume, the better the quality of the set. The hypervolume contribution of a single solution measures how much a solution contributes to the total hypervolume of a set. However, in the selection step of PLS we are not concerned about the hypervolume contribution of the solutions in the current archive, but on the potential hypervolume contribution of solutions that are generated by exploring the neighborhood of solutions in the archive. Since the actual hypervolume contribution that can be generated by neighborhood exploration is unknown, we "optimistically" estimate it. Given two solutions s and s' (in the biobjective case), we define the *optimistic hypervolume contribution* (ohvc) as the hypervolume contribution of the local ideal point defined by s and s' in the objective space:

$$\mathsf{ohvc}(s, s') = (f_1(s) - f_1(s')) \cdot (f_2(s') - f_2(s)), \tag{1}$$

where $f_1(\cdot)$ and $f_2(\cdot)$ are the normalized objective values for the first and second objective, respectively.

We expect to find new nondominated solutions in the region between the current solution and its closest neighbors in the objective space, and, thus, we define the *optimistic hypervolume improvement* (OHVI) of a solution s as

$$\mathsf{OHVI}(s) = \begin{cases} 2 \cdot \mathsf{ohvc}(s, s_{\text{sup}}) & \text{if } \nexists s_{\text{inf}}, \\ 2 \cdot \mathsf{ohvc}(s_{\text{inf}}, s) & \text{if } \nexists s_{\text{sup}}, \\ \mathsf{ohvc}(s, s_{\text{sup}}) + \mathsf{ohvc}(s_{\text{inf}}, s) & \text{otherwise,} \end{cases} \tag{2}$$

where s_{sup} and s_{inf} are the closest neighbors of s in the objective space from the current archive $\mathcal{A}_0$ defined as

$$\begin{aligned} s_{\text{sup}} &= \arg\min_{s_i \in \mathcal{A}_0} \{ f_2(s_i) \mid f_2(s_i) > f_2(s) \} \\ s_{\text{inf}} &= \arg\max_{s_i \in \mathcal{A}_0} \{ f_2(s_i) \mid f_2(s_i) < f_2(s) \}. \end{aligned} \tag{3}$$

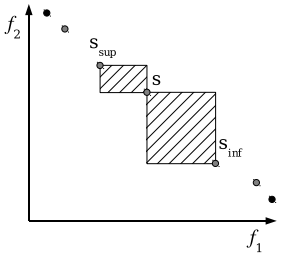


Fig. 1. Representation of the normalized objective space. The optimistic hypervolume improvement (OHVI) of a solution s is the sum of the two areas between s and its closest neighbors in the objective space, s_{inf} and s_{sup}.

Either s_{sup} or s_{inf} may not exist if s is the best solutions for $f_2()$ or $f_1()$, respectively. In such a case, we define the OHVI to be two times the existing ohvc, in order to avoid a strong bias against extreme solutions. Figure 1 graphically illustrates the computation of the OHVI of a solution.

When using OHVI for selection, the unexplored solution with the maximum OHVI value is chosen. It is, however, important to note that the above definition of OHVI makes some implicit assumptions. First, it assumes that by exploring the neighborhood of a solution new solutions are obtained only between the selected solution and the closest solutions in the objective space. This may not be necessarily the case and if the correlation between the distance of solutions in the solution space and the distance in the objective space is small, OHVI may not work better than random selection. Second, the above definition does not consider the area that dominates the current solution, and, hence, it assumes that no solutions are found that dominate the current solution. Again, this is not necessarily true, but given a good enough initial set, it is more common to find nondominated solutions than dominating ones.

Although the proposed OHVI selection is specifically designed for PLS, the concept of ohvc was previously proposed for improving the anytime behavior of Two-Phase Local Search [5]. Moreover, ohvc is related, but different to the hypervolume contribution used in some multi-objective evolutionary algorithms (MOEAs), e.g. in [1] where the hypervolume contribution of each solution with respect to the current archive is used to select or discard solutions. By contrast, ohvc estimates the potential contribution of solutions that *could* be found by the algorithm in the next iteration. The proposed OHVI selection is related to diversity measures used in other MOEAs, such as crowding distance [3] and the distance to the k-nearest neighbor [16], since it favors the exploration of the less crowded regions of the objective space.

Acceptance Criterion. The original PLS accepts any new nondominated solution into its archive. This *nondominating acceptance criterion* favors exploration but it also leads to a fast archive growth, which in turn slows down the algorithm and makes it more difficult to select the best solutions for further exploration. By contrast, a *dominating acceptance criterion*, that is, accepting only neighbors that dominate the currently explored solution, may allow PLS to reach faster the

optimal Pareto front. There are two reasons for this. First, the size of the archive stays more limited; second, the search becomes more focused on moving the current archive closer to the optimal Pareto front. The downside of a dominating acceptance criterion is that exploration is limited: nondominated solutions that may fill gaps between other solutions are not accepted and possible paths to other dominating or nondominated solutions are cut.

To avoid poor quality solutions caused by early termination of PLS when using the dominating acceptance criterion, we propose to change between the two acceptance criteria in the following manner. Initially, the dominating acceptance criterion is used as described above. If no dominating neighbor is found, the neighborhood of the current solution is explored again, this time accepting nondominated solutions. In this way, a switch from the dominating to the nondominated acceptance criterion happens on an adaptive, per-solution basis.

Neighborhood Exploration. The neighborhood exploration component decides when to stop exploring the neighborhood of the current solution. In the original PLS, the neighborhood of a solution is always fully explored. Alternatively, the exploration of the neighborhood may be stopped as soon as an "improved" solution is found, where the meaning of "improved" is defined by the acceptance criterion (see above) as being either a dominating or a nondominated solution. Different levels of neighborhood exploration may be defined by allowing a partial exploration of the neighborhood beyond finding a first acceptable solution. Here, we focus on the two extreme cases of either a *full* exploration (as in the original PLS) or a *first-accepted* exploration. By first-accepted exploration one cannot find as many solutions around the current one as with full exploration, but it terminates earlier and, thus, may allow to move close to the Pareto front faster. Independent of which of the two rules is applied, in PLS a solution becomes marked as explored, as soon as its exploration is stopped.

When using first-accepted exploration, one may find improving solutions by completing the neighborhood exploration of solutions in the archive, even if they are marked as explored. To account for this possibility, we propose the following combination of the two rules: When all solutions in the archive have been explored using the first-acceptance rule, the whole archive is marked as unexplored and the algorithm switches to full neighborhood exploration.

Related Work. To the best of our knowledge, there has been no study of the anytime behavior of PLS subject to some algorithmic variations. Liefooghe et al. [8] study the effect on the quality of restricting the overall exploration by limiting the number of solutions to be selected for exploration, and by limiting the neighborhood exploration itself. Nonetheless, there are significant differences with our proposals here. First, they do not distinguish between acceptance criterion and neighborhood exploration. In particular, they do not examine the combination of the dominating acceptance criterion and full exploration. Second, when using the dominating acceptance criterion and the first-accepted neighborhood exploration, they propose to also add all nondominated solutions found during exploration to the current archive. Our dominating acceptance criterion is more aggressive, keeps a smaller archive size, and focuses the search on getting

closer to the Pareto front. Third, different from us they do not consider variants proposed here that switch strategies during a single run of PLS, such as from dominating to nondominated acceptance criterion.

3 Experimental Analysis

3.1 Experimental Setup and Performance Assessment

We test the proposed PLS variants on the bTSP. We generate 10 bTSP instances of size 200, 300, and 500. The two distance matrices of each instance are generated independently of each other and correspond to symmetric, Euclidean TSP instances [5]. Due to the stochasticity of the algorithms, we repeat each experiment 10 times using a different random seed leading to different initial sets for PLS. The algorithms are implemented in `C++`, compiled with gcc 4.4, and the experiments were run on a single core of Intel Xeon E5410 CPUs, running at 2.33 Ghz with a 6 MB cache under Cluster Rocks Linux version 4.2.1/CentOS 4. We use the hypervolume unary indicator as a measure of the quality of the Pareto front approximations. As the upper bounds used for normalization of the objective values, we sample 10 000 random solutions, and we record the worst objective value produced from this sampling. Note that we also check that the upper bounds are never attained in any result we obtained. The lower bounds are the optimal solutions for each distance matrix obtained from the exact Concorde solver, release 03.12.19. In order to study the anytime behavior of the algorithms, we store the approximation of the Pareto front at 100 steps during the run of each algorithm, where each step is the CPU time in seconds at the following points in time: $t_i = \exp(i \cdot \ln(1001)/100) - 1, \quad i \in 1, \ldots, 100$. We examine the anytime behavior of each variant by plotting the hypervolume of its archive at each time step as follows. First, we normalize all the objectives values to the range $[1, 2]$, using the bounds as described above. Next, we compute the hypervolume of each Pareto front approximation using the reference point $[2.1, 2.1]$. Finally, for each strategy, each instance and each time step, we plot the hypervolume averaged over the 10 independent runs of each variant and the 95% confidence intervals around the mean as a gray area.

PLS starts from a given set of solutions, and this set has a large impact on PLS' final results. We consider three common alternatives for this initial set:

1. One random solution, corresponds to use PLS as a stand-alone algorithm.
2. Two solutions, each of high quality for one objective. This corresponds to a scenario where high-performing single-objective algorithms are available for each objective. Here, each solution is generated by running the iterated local search (ILS) algorithm using a 3-opt neighborhood for 2 seconds.
3. A nondominated set of high-quality solutions. This setting corresponds to a scenario where we have an algorithm for generating a high-quality approximation to the Pareto front and PLS further improves this approximation. Here, we generate a set of 5 high-quality solutions by running the anytime two-phase local search (TPLS) algorithm [5]. Each weighted sum aggregation of the objective functions is tackled by the ILS algorithm during 2 seconds.

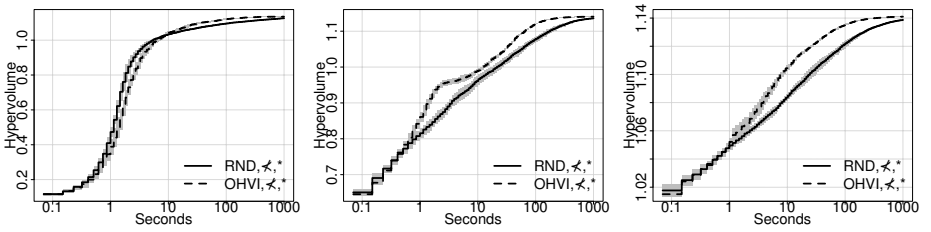


Fig. 2. PLS⟨RND⟩ vs PLS⟨OHVI⟩, starting from a random seed (left), 2 solutions (middle), and a set of solutions (right), respectively. Note the different ranges on the y-axis.

3.2 Experimental Results

First, we graphically explore the impact of one component at a time. Later, we compare the best combinations on larger instances. For reasons of space, we only provide results on one instance for each comparison; other instances show very similar behavior, and their plots are available as supplementary material [6]. As a last step, we carry out a statistical analysis over all instances.

Selection Step. We call the variant of PLS using a random selection PLS⟨RND⟩, and the variant using the OHVI for selection PLS⟨OHVI⟩. We compare in Fig. 2 these two variants. When starting from a random solution (*left*) the two variants show a very similar evolution; no curve clearly dominates the other. When the initial set is either two (*middle*) or a set of high-quality solutions (*right*), the curve corresponding to PLS⟨OHVI⟩ is most of the time above the one corresponding to PLS⟨RND⟩, which shows that PLS⟨OHVI⟩ generates a significantly better approximation set during most of the run time. Therefore, we choose OHVI as the selection step in the remainder of the paper.

Acceptance Criterion. We call the nondominated acceptance criterion PLS⟨$\nprec$⟩, the dominating one PLS⟨$\succ$⟩, and the combination of both PLS⟨$\succ\nprec$⟩. All variants use OHVI as the selection step. We compare these three variants in Fig. 3. When the initial set is a random solution (*left*), PLS⟨$\succ$⟩ improves the quality of the archive in a very short time, after which it stops because the whole archive has been explored. On the other hand, PLS⟨$\nprec$⟩ improves its archive at a slower rate, but it does not stop prematurely and it yields much better final results. Interestingly, the combination of the two strategies for the acceptance criterion used by PLS⟨$\succ\nprec$⟩ outperforms both individual strategies clearly: it combines the fast initial improvement of PLS⟨$\succ$⟩ with the much better final performance of PLS⟨$\nprec$⟩. When starting from two (*middle*) or a set of high-quality solutions (*right*), the behavior of PLS⟨$\succ$⟩ is particularly bad: it is not able to improve the initial set at all, which results in the flat line at the bottom of the plots. On the other hand, PLS⟨$\nprec$⟩ is able to significantly improve the initial solutions. The fact that the anytime behavior of PLS⟨$\succ\nprec$⟩ is better than the one of PLS⟨$\nprec$⟩ shows that the adaptive switching of the acceptance criteria on a per solution basis in PLS⟨$\succ\nprec$⟩ is highly beneficial.

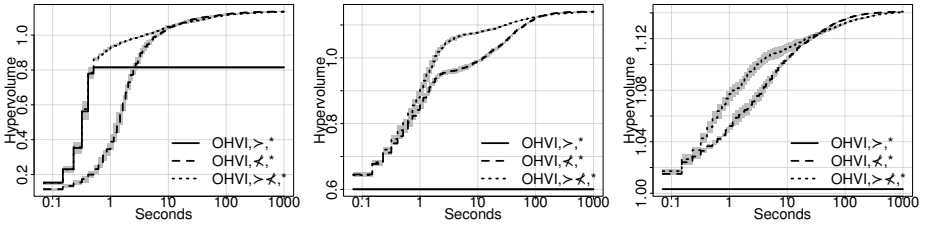


Fig. 3. PLS⟨OHVI, ≻⟩ vs PLS⟨OHVI, ≺⟩ vs PLS⟨OHVI, ≻≺⟩, starting from a random seed (left), 2 solutions (middle), and a set of solutions (right), respectively. Note the different ranges on the y-axis.

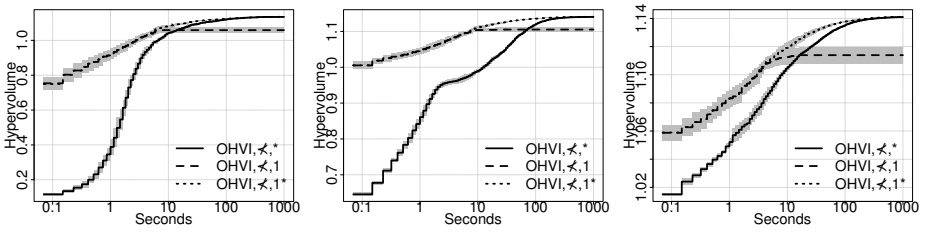


Fig. 4. PLS⟨∗⟩ vs PLS⟨1⟩ vs PLS⟨1∗⟩, starting from a random seed (left), 2 solutions (middle), and a set of solutions (right). Note the different ranges on the y-axis.

Neighborhood Exploration. In the following, PLS⟨∗⟩ denotes the full neighborhood exploration in PLS, PLS⟨1⟩ the first-accepted neighborhood exploration, and PLS⟨1∗⟩ the PLS variant that switches from the first-accepted to the full neighborhood exploration. All variants use OHVI selection and nondominating acceptance criterion. Fig. 4 shows that the anytime behavior of these three variants is very consistent, independently of the kind of initial set. PLS⟨1⟩ improves quickly the quality of the current archive in the first ten seconds, but then it terminates. PLS⟨∗⟩ requires more than ten seconds to reach the same quality as PLS⟨1⟩, but then improves further until the computation time limit. Finally, PLS⟨1∗⟩ initially matches the behavior of PLS⟨1⟩ (being actually the same algorithm) but then continues to progress due to the switch in the rule of the neighborhood exploration. Hence, PLS⟨1∗⟩ has a much better anytime behavior than each individual strategy for exploring the neighborhood.

Interactions between PLS Components. We now compare PLS variants that differ in more than one component in order to explore interactions between components. PLS⟨RND, ≺, ∗⟩ denotes the classical PLS using random selection, nondominated acceptance criterion and full neighborhood exploration; PLS⟨OHVI, ≻≺, ∗⟩ denotes the best variant obtained above when analyzing the acceptance criterion; PLS⟨OHVI, ≺, 1∗⟩ the best variant obtained above when analyzing the neighborhood exploration rule; and PLS⟨OHVI, ≻≺, 1∗⟩ the variant that combines the PLS⟨≻≺⟩ and PLS⟨1∗⟩ strategies. These four PLS variants are compared in Fig. 5. The results obtained by PLS⟨OHVI, ≻≺, ∗⟩ and PLS⟨OHVI, ≺, 1∗⟩ confirm that the newly proposed algorithms improve the anytime behavior of PLS w.r.t. to the

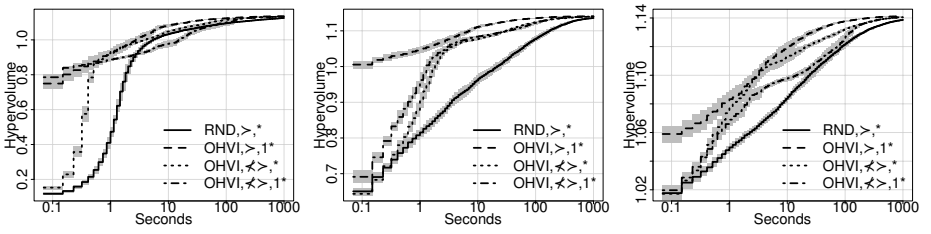


Fig. 5. PLS⟨RND, $\not\prec$, $*$⟩ vs PLS⟨OHVI, $\not\prec$, $1*$⟩ vs PLS⟨OHVI, $\succ\not\prec$, $*$⟩ and PLS⟨OHVI, $\succ\not\prec$, $1*$⟩, starting from a random seed (left), 2 solutions (middle), a set of solutions (right)

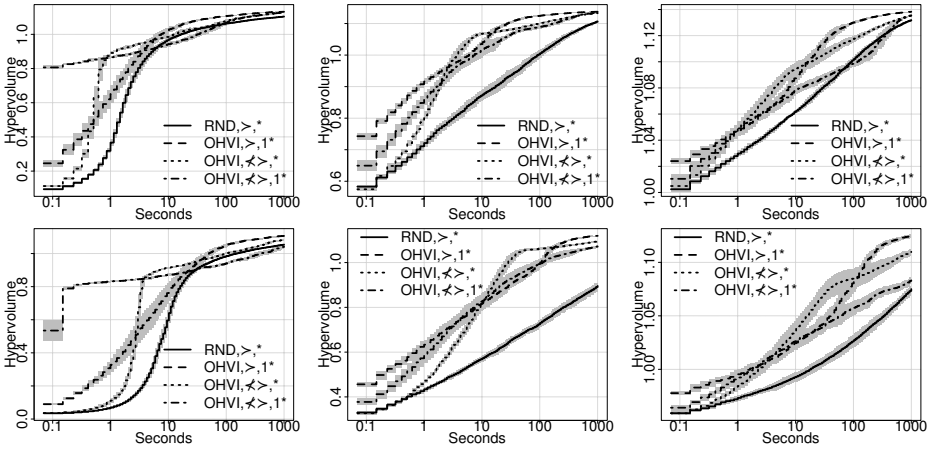


Fig. 6. PLS⟨RND, $\not\prec$, $*$⟩ vs PLS⟨OHVI, $\not\prec$, $1*$⟩ vs PLS⟨OHVI, $\succ\not\prec$, $*$⟩ and PLS⟨OHVI, $\succ\not\prec$, $1*$⟩, starting from a random seed (left), 2 solutions (middle), a set of solutions (right), and for two instance sizes: 300 (top) and 500 cities (bottom)

classical PLS⟨RND, $\not\prec$, $*$⟩. However, there is a somewhat surprising interaction between the components of PLS⟨OHVI, $\succ\not\prec$, $1*$⟩, since its anytime behavior is worse than that of PLS⟨OHVI, $\not\prec$, $1*$⟩ in general, and also worse than PLS⟨OHVI, $\succ\not\prec$, $*$⟩ for a high-quality initial set (right plot). Further work would be necessary to completely understand this behavior.

We also compare these four variants on larger bTSP instances, 10 instances of sizes 300 and 500, respectively. For tackling these larger instances, we use *candidate lists* obtained by Pareto-ranking of the edges to speed-up the neighborhood exploration [10]. We tested candidate lists of size 25, 50 and 100; they resulted in similar trends, and, here we only present results with size 50. The results provided by Fig. 6 do not allow us to declare an overall winner. When the initial solution is random (left plots), PLS⟨OHVI, $\succ\not\prec$, $1*$⟩ progresses much faster than other variants, but it is eventually outperformed by other variants. When the initial set are two or more high-quality solutions, PLS⟨OHVI, $\not\prec$, $1*$⟩ obtains the best hypervolume in the earlier and final stages. However, there is a range

Table 1. Statistical analysis of the best variants of PLS at different time steps. We chose 10.2 and 101.4 as they are the closest time steps from 10 and 100 seconds, respectively. For each time, PLS variants are ordered according to the sum of ranks obtained across all instances. The numbers in parenthesis are the differences of the sum of ranks relative to the best variant. PLS variants that are statistically significantly better than the best one are indicated in bold face. ΔR_α gives the difference of the sum of ranks that is significant.

Time	ΔR_α	**Strategies** (ΔR)
		Instance size 200, initial set: random solution.
1.0	12.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (34), ⟨OHVI,≻≯,1*⟩ (161), ⟨RND,≯,*⟩ (265)
10.2	13.2	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (116), ⟨RND,≯,*⟩ (192), ⟨OHVI,≻≯,1*⟩ (284)
101.4	13.5	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (145), ⟨OHVI,≻≯,1*⟩ (158), ⟨RND,≯,*⟩ (293)
1000.0	19.5	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (81.5), ⟨OHVI,≻≯,*⟩ (97), ⟨RND,≯,*⟩ (259.5)
		Instance size 200, initial set: two high-quality solutions.
1.0	3.9	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (103), ⟨OHVI,≻≯,*⟩ (197), ⟨RND,≯,*⟩ (300)
10.2	11.6	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (139), ⟨OHVI,≻≯,*⟩ (158), ⟨RND,≯,*⟩ (299)
101.4	10.6	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (131), ⟨OHVI,≻≯,*⟩ (169), ⟨RND,≯,*⟩ (300)
1000.0	13.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (115.5), ⟨OHVI,≻≯,*⟩ (157.5), ⟨RND,≯,*⟩ (291)
		Instance size 200, initial set: high-quality set.
1.0	17.2	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (117), ⟨OHVI,≻≯,1*⟩ (138), ⟨RND,≯,*⟩ (277)
10.2	10.7	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (76), ⟨OHVI,≻≯,1*⟩ (194), ⟨RND,≯,*⟩ (278)
101.4	15.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (130), ⟨OHVI,≻≯,1*⟩ (191), ⟨RND,≯,*⟩ (279)
1000.0	16.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (111.5), ⟨OHVI,≻≯,*⟩ (125.5), ⟨RND,≯,*⟩ (279)
		Instance size 300, initial set: random solution.
1.0	7.1	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≻≯,1*⟩ (78), ⟨OHVI,≯,1*⟩ (189), ⟨RND,≯,*⟩ (289)
10.2	14.2	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (84), ⟨RND,≯,*⟩ (171), ⟨OHVI,≻≯,1*⟩ (277)
101.4	16.9	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (122), ⟨OHVI,≻≯,1*⟩ (220), ⟨RND,≯,*⟩ (254)
1000.0	12.7	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (142), ⟨OHVI,≻≯,*⟩ (146), ⟨RND,≯,*⟩ (296)
		Instance size 300, initial set: two high-quality solutions.
1.0	12.4	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (102), ⟨OHVI,≻≯,*⟩ (169), ⟨RND,≯,*⟩ (289)
10.2	11.5	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (137), ⟨OHVI,≻≯,1*⟩ (160), ⟨RND,≯,*⟩ (299)
101.4	3.2	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (102), ⟨OHVI,≻≯,1*⟩ (198), ⟨RND,≯,*⟩ (300)
1000.0	10.4	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (129.5), ⟨OHVI,≻≯,*⟩ (170.5), ⟨RND,≯,*⟩ (300)
		Instance size 300, initial set: high-quality set.
1.0	22.3	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (35), ⟨OHVI,≻≯,1*⟩ (37), ⟨RND,≯,*⟩ (224)
10.2	13.4	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (101), ⟨OHVI,≻≯,1*⟩ (175), ⟨RND,≯,*⟩ (284)
101.4	9.9	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (100), ⟨RND,≯,*⟩ (225), ⟨OHVI,≻≯,1*⟩ (275)
1000.0	10.9	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (135.5), ⟨OHVI,≻≯,1*⟩ (164.5), ⟨RND,≯,*⟩ (300)
		Instance size 500, initial set: random solution.
1.0	0	⟨**OHVI,≻≯,1***⟩, ⟨OHVI,≯,1*⟩ (100), ⟨OHVI,≻≯,*⟩ (200), ⟨RND,≯,*⟩ (300)
10.2	12	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≻≯,1*⟩ (106), ⟨OHVI,≯,1*⟩ (193), ⟨RND,≯,*⟩ (285)
101.4	6.4	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (103), ⟨RND,≯,*⟩ (196), ⟨OHVI,≻≯,1*⟩ (297)
1000.0	16	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (120), ⟨OHVI,≻≯,1*⟩ (223), ⟨RND,≯,*⟩ (257)
		Instance size 500, initial set: two high-quality solutions.
1.0	9.1	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (71), ⟨OHVI,≻≯,*⟩ (186), ⟨RND,≯,*⟩ (283)
10.2	22.4	⟨**OHVI,≯,1***⟩, ⟨**OHVI,≻≯,*** **(18)**⟩, ⟨OHVI,≻≯,1*⟩ (46), ⟨RND,≯,*⟩ (220)
101.4	13.8	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (101), ⟨OHVI,≻≯,1*⟩ (157), ⟨RND,≯,*⟩ (286)
1000.0	0	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (100), ⟨OHVI,≻≯,1*⟩ (200), ⟨RND,≯,*⟩ (300)
		Instance size 500, initial set: high-quality set.
1.0	14.5	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,1*⟩ (65), ⟨OHVI,≻≯,*⟩ (154), ⟨RND,≯,*⟩ (273)
10.2	19.4	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≻≯,1*⟩ (63), ⟨OHVI,≯,1*⟩ (105), ⟨RND,≯,*⟩ (256)
101.4	11.7	⟨**OHVI,≻≯,***⟩, ⟨OHVI,≯,1*⟩ (59), ⟨OHVI,≻≯,1*⟩ (169), ⟨RND,≯,*⟩ (276)
1000.0	8.4	⟨**OHVI,≯,1***⟩, ⟨OHVI,≻≯,*⟩ (100), ⟨OHVI,≻≯,1*⟩ (216), ⟨RND,≯,*⟩ (284)

of time where other variants, in particular PLS$\langle$OHVI, $\succ\not\prec, *\rangle$, perform better. Nonetheless, the plots clearly show that the proposed PLS variants consistently outperform the classical PLS$\langle$RND, $\not\prec, *\rangle$.

Statistical Tests. We perform statistical tests to assess the behavior over the whole set of 10 instances of each size. We apply the Friedman test for analyzing non-parametric, unreplicated, complete block designs. Each block is a run using a particular seed (out of ten) on a single instance, and the different PLS variants are the treatment factors. Next, we rank the PLS variants per block according to the hypervolume, the lower the rank the better, and we calculate the difference (ΔR) between the sum of ranks of each variant and the best ranked one (with the lowest sum of ranks). Finally, we calculate the minimum difference between the sum of ranks of two variants that is statistically significant (ΔR_α), given a significance level of $\alpha = 0.05$, using the Friedman post-test for multiple comparisons [2]. We perform this test on the output of the algorithms at various time steps, for each instance size and for each type of initial set. Table 1 summarizes the result of one independent test in each row. We indicate in bold face the best variant (the one having the lowest sum of ranks) and those that are not significantly different from the best one. The tests confirm our conclusions that PLS$\langle$OHVI, $\not\prec, 1*\rangle$ is the best strategy overall for size 200. It is also often the best for sizes 300 and 500, but not always. Moreover, it shows that the classical PLS$\langle$RND, $\not\prec, *\rangle$ performs quite poorly when compared with the anytime PLS variants proposed here, being in some cases completely outranked (with a sum of ranks close to 300) by the other variants.

4 Conclusions

In this paper, we proposed alternative choices for algorithmic components of PLS that improve substantially its anytime behavior. In addition, we have proposed novel approaches that are based on switching strategies for the neighborhood exploration and the acceptance criterion; these switching strategies have proven to be essential for improving the anytime behavior. As a result, replacing the original PLS with one of our proposed variants in hybrid algorithms is likely to further improve the current state of the art in multi-objective optimization. However, none of the variants is clearly superior to all others, and further research is needed to understand in more detail the behavior of the proposed variants. In the future, we will extend the analysis to problems with more than 2 objectives, to other problems such as MKP [11] and permutation flowshop [4], and investigate other possibilities to improve anytime behavior.

Acknowledgments. This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium, and by the MIBISOC network, an Initial Training Network funded by the European Commission, grant PITN–GA–2009–238819. Thomas Stützle and Manuel López-Ibáñez acknowledge support from the Belgian F.R.S.-FNRS, of which they are a Research Associate and a Postdoctoral researcher, respectively.

References

1. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181(3), 1653–1669 (2007)
2. Conover, W.J.: Practical Nonparametric Statistics. John Wiley & Sons (1999)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 181–197 (2002)
4. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Computers & Operations Research 38(8), 1219–1236 (2011)
5. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Improving the anytime behavior of two-phase local search. Annals of Mathematics and Artificial Intelligence 61(2), 125–154 (2011)
6. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Supplementary Material: Pareto Local Search Variants for Anytime Bi-Objective Optimization (2012), http://iridia.ulb.ac.be/supp/IridiaSupp2012-004
7. Ehrgott, M., Gandibleux, X.: Approximative solution methods for combinatorial multicriteria optimization. TOP 12(1), 1–88 (2004)
8. Liefooghe, A., Mesmoudi, S., Humeau, J., Jourdan, L., Talbi, E.G.: On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. Journal of Heuristics (2011)
9. Loudni, S., Boizumault, P.: Combining VNS with constraint programming for solving anytime optimization problems. European Journal of Operational Research 191, 705–735 (2008)
10. Lust, T., Jaszkiewicz, A.: Speed-up techniques for solving large-scale biobjective TSP. Computers & Operations Research 37(3), 521–533 (2010)
11. Lust, T., Teghem, J.: The multiobjective multidimensional knapsack problem: a survey and a new approach. Arxiv preprint arXiv:1007.4063 (2010)
12. Lust, T., Teghem, J.: Two-phase Pareto local search for the biobjective traveling salesman problem. Journal of Heuristics 16(3), 475–510 (2010)
13. Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Gandibleux, X., et al. (eds.) Metaheuristics for Multiobjective Optimisation. LNEMS, vol. 535, pp. 177–200. Springer (2004)
14. Paquete, L., Stützle, T.: Design and analysis of stochastic local search for the multiobjective traveling salesman problem. Computers & Operations Research 36(9), 2619–2631 (2009)
15. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine 17(3), 73–83 (1996)
16. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control, pp. 95–100. CIMNE, Barcelona (2002)
17. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)

Pure Strategy or Mixed Strategy?

An Initial Comparison of Their Asymptotic Convergence Rate and Asymptotic Hitting Time

Jun He[1], Feidun He[2], and Hongbin Dong[3]

[1] Department of Computer Science, Aberystwyth University, Ceredigion SY23 3DB, U.K.
[2] School of Information Science and Technology, Southwest Jiaotong University Chengdu, Sichuan 610031, China
[3] College of Computer Science and Technology, Harbin Engineering University Harbin, Heilongjiang 150001, China

Abstract. Mixed strategy evolutionary algorithms (EAs) aim at integrating several mutation operators into a single algorithm. However no analysis has been made to answer the theoretical question: whether and when is the performance of mixed strategy EAs better than that of pure strategy EAs? In this paper, asymptotic convergence rate and asymptotic hitting time are proposed to measure the performance of EAs. It is proven that the asymptotic convergence rate and asymptotic hitting time of any mixed strategy (1+1) EA consisting of several mutation operators is not worse than that of the worst pure strategy (1+1) EA using only one mutation operator. Furthermore it is proven that if these mutation operators are mutually complementary, then it is possible to design a mixed strategy (1+1) EA whose performance is better than that of any pure strategy (1+1) EA using only one mutation operator.

Keywords: Mixed Strategy, Pure Strategy, Asymptotic Convergence Rate, Asymptotic Hitting Time, Hybrid Evolutionary Algorithms.

1 Introduction

Different search operators have been proposed and applied in EAs [1]. Each search operator has its own advantage. Therefore an interesting research issue is to combine the advantages of variant operators together and then design more efficient hybrid EAs. Currently hybridization of evolutionary algorithms becomes popular due to their capabilities in handling some real world problems [2].

Mixed strategy EAs, inspired from strategies and games [3], aims at integrating several mutation operators into a single algorithm [4]. At each generation, an individual will choose one mutation operator according to a strategy probability distribution. Mixed strategy evolutionary programming has been implemented for continuous optimization and experimental results show it performs better than its rival, i.e., pure strategy evolutionary programming which utilizes a single mutation operator [5,6].

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 218–229, 2012.

However no analysis has been made to answer the theoretical question: whether and when is the performance of mixed strategy EAs better than that of pure strategy EAs? This paper aims at providing an initial answer. In theory, many of EAs can be regarded as a matrix iteration procedure. Following matrix iteration analysis [7], the performance of EAs is measured by the asymptotic convergence rate, i.e., the spectral radius of a probability transition sub-matrix associated with an EA. Alternatively the performance of EAs can be measured by the asymptotic hitting time [8], which approximatively equals the reciprocal of the asymptotic convergence rate. Then a theoretical analysis is made to compare the performance of mixed strategy and pure strategy EAs.

The rest of this paper is organized as follows. Section 2 describes pure strategy and mixed strategy EAs. Section 3 defines asymptotic convergence rate and asymptotic hitting time. Section 4 makes a comparison of pure strategy and mixed strategy EAs. Section 5 concludes the paper.

2 Pure Strategy and Mixed Strategy EAs

Before starting a theoretical analysis of mixed strategy EAs, we first demonstrate the result of a computational experiment.

Example 1. Let's see an instance of the average capacity 0-1 knapsack problem [9,10]:

$$\begin{array}{ll} \text{maximize } \sum_{i=1}^{10} v_i b_i, & b_i \in \{0,1\}, \\ \text{subject to } \sum_{i=1}^{10} w_i b_i \leq C, & \end{array} \tag{1}$$

where $v_1 = 10$ and $v_i = 1$ for $i = 2, \cdots, 10$; $w_1 = 9$ and $w_i = 1$ for $i = 2, \cdots, 10$; $C = 9$.

The fitness function is that for $x = (b_1, \cdots, b_{10})$

$$f(x) = \begin{cases} \sum_{i=1}^{10} v_i b_i, & \text{if } \sum_{i=1}^{10} w_i b_i \leq C, \\ 0, & \text{if } \sum_{i=1}^{10} w_i b_i > C. \end{cases}$$

We consider two types of mutation operators:

- s1: flip each bit b_i with a probability 0.1;
- s2: flip each bit b_i with a probability 0.9;

The selection operator is to accept a better offspring only.

Three (1+1) EAs are compared in the computation experiment: (1) EA(s1) which adopts s1 only, (2) EA(s2) with s2 only, and (3) EA(s1,s2) which chooses either s1 or s2 with a probability 0.5 at each generation.

Each of these three EAs runs 100 times independently. The computational experiment shows that EA(s1, s2) always finds the optimal solution more quickly than other twos.

This is a simple case study that shows a mixed strategy EA performs better than a pure strategy EA. In general, we need to answer the following theoretical

question: whether or when do a mixed strategy EAs are better than pure strategy EAs?

Consider an instance of the discrete optimization problem which is to maximize an objective function $f(x)$:

$$\max\{f(x); x \in S\}, \tag{2}$$

where S a finite set. For the analysis convenience, suppose that all constraints have been removed through an appropriate penalty function method. Under this scenario, all points in S are viewed as feasible solutions. In evolutionary computation, $f(x)$ is called a *fitness function*.

The following notation is used in the algorithm and text thereafter.

- $x, y, z \in S$ are called *points* in S, or *individuals* in EAs or *states* in Markov chains.
- The *optimal set* $S_{\text{opt}} \subseteq S$ is the set consisting of all optimal solutions to Problem (2) and *non-optimal set* $S_{\text{non}} := S \setminus S_{\text{opt}}$.
- t is the generation counter. A random variable Φ_t represents the state of the t-th generation parent; $\Phi_{t+1/2}$ the state of the child which is generated through mutation.

The mutation and selection operators are defined as follows:

- A *mutation operator* is a probability transition from S to S. It is defined by a *mutation probability transition matrix* $\mathbf{P}_m$ whose entries are given by

$$P_m(x, y), \quad x, y \in S. \tag{3}$$

- A *strict elitist selection operator* is a mapping from $S \times S$ to S, that is for $x \in S$ and $y \in S$,

$$z = \begin{cases} x, \text{ if } f(y) \leq f(x), \\ y, \text{ if } f(y) > f(x). \end{cases} \tag{4}$$

A *pure strategy* (1+1) EA, which utilizes only one mutation operator, is described in Algorithm 1.

Algorithm 1. Pure Strategy Evolutionary Algorithm EA(s)

```
input: fitness function;
generation counter t ← 0;
initialize Φ_0;
while stopping criterion is not satisfied do
    Φ_{t+1/2} ← mutate Φ_t by mutation operator s;
    evaluate the fitness of Φ_{t+1/2};
    Φ_{t+1} ← select one individual from {Φ_t, Φ_{t+1/2}} by strict elitist selection;
    t ← t + 1;
end while
output: the maximal value of the fitness function.
```

The stopping criterion is that the running stops once an optimal solution is found. If an EA cannot find an optimal solution, then it will not stop and the running time is infinite. This is common in the theoretical analysis of EAs.

Let s1, ..., sκ be κ mutation operators (called *strategies*). Algorithm 2 describes the procedure of a *mixed strategy* (1+1) EA. At the t-th generation, one mutation operator is chosen from the κ strategies according to a *strategy probability distribution*

$$q_{s1}(x), \cdots, q_{s\kappa}(x), \tag{5}$$

subject to $0 \leq q_s(x) \leq 1$ and $\sum_s q_s(x) = 1$.

Write this probability distribution in short by a vector $\mathbf{q}(x) = [q_s(x)]$.

Algorithm 2. Mixed Strategy Evolutionary Algorithm EA(s1, ..., sκ)

1: **input**: fitness function;
2: generation counter $t \leftarrow 0$;
3: initialize Φ_0;
4: **while** stopping criterion is not satisfied **do**
5: choose a mutation operator sk from s1, ..., sκ;
6: $\Phi_{t+1/2} \leftarrow$ mutate Φ_t by mutation operator sk;
7: evaluate $\Phi_{t+1/2}$;
8: $\Phi_{t+1} \leftarrow$ select one individual from $\{\Phi_t, \Phi_{t+1/2}\}$ by strict elitist selection;
9: $t \leftarrow t + 1$;
10: **end while**
11: **output**: the maximal value of the fitness function.

Pure strategy EAs can be regarded a special case of mixed strategy EAs with only one strategy.

EAs can be classified into two types:

- A *homogeneous EA* is an EA which applies the same mutation operators and same strategy probability distribution for all generations.
- An *inhomogeneous* EA is an EA which doesn't apply the same mutation operators or same strategy probability distribution for all generations.

This paper will only discuss *homogeneous EAs* mainly due to the following reason:

- The probability transition matrices of an inhomogeneous EA may be chosen to be totally different at different generations. This makes the theoretical analysis of an inhomogeneous EA extremely hard.

3 Asymptotic Convergence Rate and Asymptotic Hitting Time

Suppose that a homogeneous EA is applied to maximize a fitness function $f(x)$, then the population sequence $\{\Phi_t, t = 0, 1, \cdots\}$ can be modelled by a

homogeneous Markov chain [11,12]. Let $\mathbf{P}$ be the probability transition matrix, whose entries are given by

$$P(x, y) = P(\Phi_{t+1} = y \mid \Phi_t = x), \quad x, y \in S.$$

Starting from an initial state x, the mean number $m(x)$ of generations to find an optimal solution is called the *hitting time* to the set S_{opt} [13].

$$\begin{aligned} \tau(x) &:= \min\{t; \Phi_t \in S_{\text{opt}} \mid \Phi_0 = x\}, \\ m(x) &:= E[\tau(x)] = \sum_{t=0}^{+\infty} tP(\tau(x) = t). \end{aligned}$$

Let's arrange all individuals in the order of their fitness from high to low: $x_1, x_2, \cdots$, then their hitting times are:

$$m(x_1), m(x_2), \cdots .$$

Denote it in short by a vector $\mathbf{m} = [m(x)]$.

Write the transition matrix $\mathbf{P}$ in the canonical form [14],

$$\mathbf{P} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ * & \mathbf{T} \end{pmatrix}, \tag{6}$$

where $\mathbf{I}$ is a unit matrix and $\mathbf{0}$ a zero matrix. $\mathbf{T}$ denotes the probability transition sub-matrix among non-optimal states, whose entries are given by

$$P(x, y), \quad x \in S_{\text{non}}, y \in S_{\text{non}}.$$

The part $*$ plays no role in the analysis.

Since $\forall x \in S_{\text{opt}}, m(x) = 0$, it is sufficient to consider $m(x)$ on non-optimal states $x \in S_{\text{non}}$. For the simplicity of notation, the vector $\mathbf{m}$ will also denote the hitting times for all non-optimal states: $[m(x)], x \in S_{\text{non}}$.

The Markov chain associated with an EA can be viewed as a matrix iterative procedure, where the iterative matrix is the probability transition sub-matrix $\mathbf{T}$. Let $\mathbf{p}_0$ be the vector $[p_0(x)]$ which represents the probability distribution of the initial individual:

$$p_0(x) := P(\Phi_0 = x), \quad x \in S_{\text{non}},$$

and $\mathbf{p}_t$ the vector $[p_t(x)]$ which represents the probability distribution of the t-generation individual:

$$p_t(x) := P(\Phi_t = x), \quad x \in S_{\text{non}}.$$

If the spectral radius $\rho(\mathbf{T})$ of the matrix $\mathbf{T}$ satisfies: $\rho(\mathbf{T}) < 1$, then we know [7]

$$\lim_{t \to \infty} \| \mathbf{p}_t \| = 0.$$

Following matrix iterative analysis [7], the asymptotic convergence rate of an EA is defined as below.

Definition 1. *The* asymptotic convergence rate *of an EA for maximizing* $f(x)$ *is*

$$R(\mathbf{T}) := -\ln \rho(\mathbf{T}) \tag{7}$$

where $\mathbf{T}$ *is the probability transition sub-matrix restricted to non-optimal states and* $\rho(\mathbf{T})$ *its spectral radius.*

Asymptotic convergence rate is different from previous definitions of convergence rate based on matrix norms or probability distribution [12].

Note: Asymptotic convergence rate depends on both the probability transition sub-matrix $\mathbf{T}$ and fitness function $f(x)$. Because the spectral radius of the probability transition matrix $\rho(\mathbf{P}) = 1$, thus $\rho(\mathbf{P})$ cannot be used to measure the performance of EAs. Becaue the mutation probability transition matrix is the same for all functions $f(x)$, and $\rho(\mathbf{P}_m) = 1$, so $\rho(\mathbf{P}_m)$ cannot be used to measure the performance of EAs too.

If $\rho(\mathbf{T}) < 1$, then the hitting time vector satisfies (see Theorem 3.2 in [14]),

$$\mathbf{m} = (\mathbf{I} - \mathbf{T})^{-1}\mathbf{1}. \tag{8}$$

The matrix $\mathbf{N} := (\mathbf{I}-\mathbf{T})^{-1}$ is called the *fundamental matrix* of the Markov chain, where $\mathbf{T}$ is the probability transition sub-matrix restricted to non-optimal states.

The spectral radius $\rho(\mathbf{N})$ of the fundamental matrix can be used to measure the performance of EAs too.

Definition 2. *The* asymptotic hitting time *of an EA for maximizing* $f(x)$ *is*

$$T(\mathbf{T}) = \begin{cases} \rho(\mathbf{N}) = \rho((\mathbf{I} - \mathbf{T})^{-1}), & \text{if } \rho(\mathbf{T}) < 1, \\ +\infty, & \text{if } \rho(\mathbf{T}) = 1. \end{cases}$$

where $\mathbf{T}$ *is the probability transition sub-matrix restricted to non-optimal states and* $\mathbf{N}$ *is the fundamental matrix.*

From Lemma 5 in [8], we know the asymptotic hitting time is between the best and worst case hitting times, i.e.,

$$\min\{m(x); x \in S_{\text{non}}\} \leq T(\mathbf{T}) \leq \max\{m(x); x \in S_{\text{non}}\}. \tag{9}$$

From Lemma 3 in [8], we know

Lemma 1. *For any homogeneous (1+1)-EA using strictly elitist selection, it holds*

$$\begin{aligned} &\rho(\mathbf{T}) = \max\{P(x,x); x \in S_{\text{non}}\}, \\ &\rho(\mathbf{N}) = \frac{1}{1-\rho(\mathbf{T})}, \qquad \text{if } \rho(\mathbf{T}) < 1. \end{aligned}$$

From Lemma 1 and Taylor series, we get that

$$R(\mathbf{T})T(\mathbf{T}) = \sum_{k=1}^{\infty} \frac{1}{k} \left(\frac{1}{T(\mathbf{T})}\right)^{k-1}.$$

If we make a mild assumption $T(\mathbf{T}) \geq 2$, (i.e., the asymptotic hitting time is at least two generations), then the asymptotic hitting time approximatively equals the reciprocal of the asymptotic convergence rate (see Figure 1).

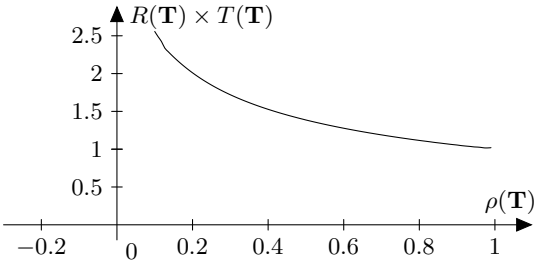


Fig. 1. The relationship between the asymptotic hitting time and asymptotic convergence rate: $1/R(\mathbf{T}) < T(\mathbf{T}) < 1.5/R(\mathbf{T})$ if $\rho(\mathbf{T}) \geq 0.5$

Example 2. Consider the problem of maximizing the One-Max function:

$$f(x) =\mid x \mid,$$

where $x = (b_1 \cdots b_n)$ a binary string, n the string length and $\mid x \mid := \sum_{i=1}^{n} b_i$. The mutation operator used in the (1+1) EA is to choose one bit randomly and then flip it.

Then asymptotic convergence rate and asymptotic hitting time are

$$1/n < R(\mathbf{T}) < 1/(n-1),$$
$$T(\mathbf{T}) = n.$$

4 A Comparison of Pure Strategy and Mixed Strategy

In this section, subscripts $\mathbf{q}$ and s are added to distinguish between a mixed strategy EA using a strategy probability distribution $\mathbf{q}$ and a pure strategy EA using a pure strategy s. For example, $\mathbf{T_q}$ denotes the probability transition sub-matrix of a mixed strategy EA; $\mathbf{T}_s$ the transition sub-matrix of a pure strategy EA.

Theorem 1. *Let* $s1, \cdots s\kappa$ *be* κ *mutation operators.*

1. *The asymptotic convergence rate of any mixed strategy EA consisting of these* κ *mutation operators is not smaller than the worst pure strategy EA using only one of these mutation operator;*
2. *and the asymptotic hitting time of any mixed strategy EA is not larger than the worst pure strategy EA using one only of these mutation operator.*

Proof. (1) From Lemma 1 we know

$$\begin{aligned}\rho(\mathbf{T_q}) &= \max\{\frac{1}{\kappa}\sum_{k=1}^{\kappa} P_{s_k}(x,x); x \in S_{\text{non}}\} \\ &\leq \frac{1}{\kappa}\sum_{k=1}^{\kappa}\rho(\mathbf{T}_{sk}) \\ &\leq \max\{\rho(\mathbf{T}_{sk}); k = 1, \cdots, \kappa\}.\end{aligned}$$

Thus we get that

$$R(\mathbf{T_q}) := -\ln \rho(\mathbf{T_q}) \geq \max\{-\ln \rho(\mathbf{T}_{sk}); k = 1, \cdots, \kappa\}.$$

(2) From Lemma 1, we know

$$\rho(\mathbf{N}) = \frac{1}{1 - \rho(\mathbf{T})},$$

then we get $\rho(\mathbf{N_q}) \leq \max\{\rho(\mathbf{N}_{s_k}); k = 1, \cdots, \kappa\}$. □

In the following we investigate whether and when the performance of a mixed strategy EA is better than a pure strategy EA.

Definition 3. *A mutation operator s1 is called* complementary *to another mutation operator s2 on a fitness function $f(x)$ if for any x such that*

$$P_{s1}(x, x) = \rho(\mathbf{T}_{s1}), \tag{10}$$

it holds

$$P_{s2}(x, x) < \rho(\mathbf{T}_{s1}). \tag{11}$$

Theorem 2. *Let $f(x)$ be a fitness function and EA(s1) a pure strategy EA. If a mutation operator s2 is complementary to s1, then it is possible to design a mixed strategy EA(s1,s2) which satisfies*

1. *its asymptotic convergence rate is larger than that of EA(s1);*
2. *and its asymptotic hitting time is shorter than that of EA(s1).*

Proof. (1) Design a mixed strategy EA(s1, s2) as follows. For any x such that

$$P_{s1}(x, x) = \rho(\mathbf{T}_{s1}),$$

let the strategy probability distribution satisfy

$$q_{s2}(x) = 1.$$

For any other x, let the strategy probability distribution satisfy

$$q_{s1}(x) = 1.$$

Because s2 is complementary to s1, we get that

$$\rho(\mathbf{T_q}) < \rho(\mathbf{T}_{s1}),$$

and then

$$-\ln \rho(\mathbf{T_q}) > -\ln \rho(\mathbf{T}_{s1}),$$

which proves the first conclusion in the theorem.

(2) From Lemma 1

$$\rho(\mathbf{N}) = \frac{1}{1 - \rho(\mathbf{T})}$$

we get that

$$\rho(\mathbf{N_q}) < \rho(\mathbf{N}_{sk}), \quad \forall k = 1, \cdots, \kappa,$$

which proves the second conclusion in the theorem. □

Definition 4. *κ mutation operators $s1, \cdots, s\kappa$ are called* mutually complementary *on a fitness function $f(x)$ if for any $x \in S_{\text{non}}$ and $sl \in \{s1, \cdots, s\kappa\}$ such that*

$$P_{sl}(x, x) \geq \min\{\rho(\mathbf{T}_{s1}), \cdots, \rho(\mathbf{T}_{s\kappa})\}, \tag{12}$$

it holds: $\exists sk \neq sl$,

$$P_{sk}(x, x) < \min\{\rho(\mathbf{T}_{s1}), \cdots, \rho(\mathbf{T}_{s\kappa})\}. \tag{13}$$

Theorem 3. *Let $f(x)$ be a fitness function and $s1, \cdots, s\kappa$ be κ mutation operators. If these mutation operators are mutually complementary, then it is possible to design a mixed strategy EA which satisfies*

1. *its asymptotic convergence rate is larger than that of any pure strategy EA using one mutation operator;*
2. *and its asymptotic hitting time is shorter than that of any pure strategy EA using one mutation operator.*

Proof. (1) We design a mixed strategy EA(s1, ..., sκ) as follows. For any x and any strategy $sl \in \{s1, \cdots, s\kappa\}$ such that

$$P_{sl}(x, x) \geq \min\{\rho(\mathbf{T}_{s1}), \cdots, \rho(\mathbf{T}_{s\kappa})\},$$

from the mutually complementary condition, we know $\exists sk \neq sl$, it holds

$$P_{sk}(x, x) < \min\{\rho(\mathbf{T}_{s1}), \cdots, \rho(\mathbf{T}_{s\kappa})\}.$$

Let the strategy probability distribution satisfy

$$q_{sk}(x) = 1.$$

For any other x, we assign a strategy probability distribution in any way.

Because the mutation operators are mutually complementary, we get that

$$\rho(\mathbf{T_q}) < \min\{\rho(\mathbf{T}_{s1}), \cdots, \rho(\mathbf{T}_{s\kappa})\},$$

and then

$$-\ln \rho(\mathbf{T_q}) > \min\{-\ln \rho(\mathbf{T}_{s1}), \cdots, -\ln \rho(\mathbf{T}_{s\kappa})\},$$

which proves the first conclusion in the theorem.
(2) From Lemma 1

$$\rho(\mathbf{N}) = \frac{1}{1 - \rho(\mathbf{T})},$$

we get that

$$\rho(\mathbf{N_q}) < \rho(\mathbf{N}_{sk}), \quad \forall k = 1, \cdots, \kappa,$$

which proves the second conclusion in the theorem. □

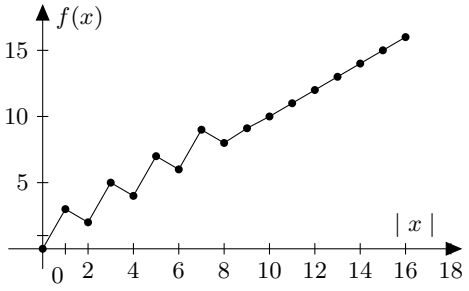


Fig. 2. The shape of the function $f(x)$ in Example 3 when $n = 16$

Example 3. Consider the problem of maximizing the following fitness function $f(x)$ (see Figure 2):

$$f(x) = \begin{cases} |\, x \,|, & \text{if } |\, x \,| < 0.5n \text{ and } |\, x \,| \text{ is even;} \\ |\, x \,| + 2, & \text{if } |\, x \,| < 0.5n \text{ and } |\, x \,| \text{ is odd;} \\ |\, x \,|, & \text{if } |\, x \,| \geq 0.5n. \end{cases}$$

where $x = (b_1 \cdots b_n)$ is a binary string, n the string length and $|\, x \,| := \sum_{i=1}^{n} b_i$.

Consider two common mutation operators:

- s1: to choose one bit randomly and then flip it;
- s2: to flip each bit independently with a probability $1/n$.

EA(s1) uses the mutation operator s1 only. Then $\rho(\mathbf{T}_{s1}) = 1$, and then the asymptotic convergence rate is $R(\mathbf{T}_{s1}) = 0$.

EA(s2) utilizes the mutation operator s2 only. Then

$$\rho(\mathbf{T}_{s2}) = 1 - \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}.$$

We have

$$\min\{\rho(\mathbf{T}_{s1}), \rho(\mathbf{T}_{s2})\} = 1 - \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}.$$

(1) For any x such that

$$P_{s1}(x, x) \geq 1 - \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1},$$

we have

$$P_{s1}(x, x) = 1,$$

and we know that

$$P_{s2}(x, x) < 1 - \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}.$$

(2) For any x such that

$$P_{s2}(x,x) = \rho(\mathbf{T}_{s2}) = 1 - \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1},$$

we know that

$$P_{s1}(x,x) = 1 - \frac{1}{n} < \rho(\mathbf{T}_{s2}) = 1 - \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}.$$

Hence these two mutation operators are mutually complementary.

We design a mixed strategy EA(s1,s2) as follows: let the strategy probability distribution satisfy

$$q_{s1}(x) = \begin{cases} 0, \text{ if } \mid x \mid \leq 0.5n; \\ 1, \text{ if } \mid x \mid > 0.5n. \end{cases}$$

According to Theorem 3, the asymptotic convergence rate of this mixed strategy EA(s1,s2) is larger than that of either EA(s1) or EA(s2).

5 Conclusion and Discussion

The result of this paper is summarized in three points.

- Asymptotic convergence rate and asymptotic hitting time are proposed to measure the performance of EAs. They are seldom used in evaluating the performance of EAs before.
- It is proven that the asymptotic convergence rate and asymptotic hitting time of any mixed strategy (1+1) EA consisting of several mutation operators is not worse than that of the worst pure strategy (1+1) EA using only one of these mutation operators.
- Furthermore, if these mutation operators are mutually complementary, then it is possible to design a mixed strategy EA whose performance (asymptotic convergence rate and asymptotic hitting time) is better than that of any pure strategy EA using one mutation operator.

An argument is that several mutation operators can be applied simultaneously, e.g., in a population-based EA, different individuals adopt different mutation operators. However in this case, the number of fitness evaluations at each generation is larger than that of a (1+1) EA. Therefore a fair comparison should be a population-based mixed strategy EA against a population-based pure strategy EA. Due to the length restriction, this issue will not be discussed in the paper.

Acknowledgement. J. He is partially supported by the EPSRC under Grant EP/I009809/1. H. Dong is partially supported by the National Natural Science Foundation of China under Grant No. 60973075 and Natural Science Foundation of Heilongjiang Province of China under Grant No. F200937, China.

References

1. Fogel, D., Michalewicz, Z.: Handbook of Evolutionary Computation. Oxford Univ. Press (1997)
2. Grosan, C., Abraham, A., Ishibuchi, H.: Hybrid Evolutionary Algorithms. Springer (2007)
3. Dutta, P.: Strategies and Games: Theory and Practice. MIT Press (1999)
4. He, J., Yao, X.: A Game-Theoretic Approach for Designing Mixed Mutation Strategies. In: Wang, L., Chen, K., Ong, Y.S. (eds.) ICNC 2005, Part III. LNCS, vol. 3612, pp. 279–288. Springer, Heidelberg (2005)
5. Dong, H., He, J., Huang, H., Hou, W.: Evolutionary programming using a mixed mutation strategy. Information Sciences 177(1), 312–327 (2007)
6. Shen, L., He, J.: A mixed strategy for evolutionary programming based on local fitness landscape. In: Proceedings of 2010 IEEE Congress on Evolutionary Computation, pp. 350–357. IEEE Press, Barcelona (July 2010)
7. Varga, R.: Matrix Iterative Analysis. Springer (2009)
8. He, J., Chen, T.: Population scalability analysis of abstract population-based random search: Spectral radius. Arxiv preprint arXiv:1108.4531 (2011)
9. Michalewicz, Z.: Genetic Algorithms + Data Structure = Evolution Program. Springer, New York (1996)
10. He, J., Zhou, Y.: A Comparison of GAs Using Penalizing Infeasible Solutions and Repairing Infeasible Solutions on Average Capacity Knapsack. In: Kang, L., Liu, Y., Zeng, S. (eds.) ISICA 2007. LNCS, vol. 4683, pp. 100–109. Springer, Heidelberg (2007)
11. Rudolph, G.: Convergence analysis of canonical genetic algorithms. IEEE Transactions on Neural Networks 5(1), 96–101 (1994)
12. He, J., Kang, L.: On the convergence rate of genetic algorithms. Theoretical Computer Science 229(1-2), 23–39 (1999)
13. He, J., Yao, X.: Towards an analytic framework for analysing the computation time of evolutionary algorithms. Artificial Intelligence 145(1-2), 59–97 (2003)
14. Iosifescu, M.: Finite Markov Chain and their Applications. Wiley, Chichester (1980)

Recurrent Genetic Algorithms: Sustaining Evolvability

Adnan Fakeih[1] and Ahmed Kattan[2]

[1] Futures Business Development Ltd., Saudi Arabia
[2] Department of Computer Science, Um Al-Qura University, Saudi Arabia
adnan@fbd.net, ajkattan@uqu.edu.sa

Abstract. This paper proposes a new paradigm, referred to as *Recurrent Genetic Algorithms (RGA)*, to sustain Genetic Algorithm (GA) evolvability and effectively improves its ability to find superior solutions. RGA attempts to continually recover evolvability loss caused by the canonical GA iteration process. It borrows the term Recurrent from the taxonomy of Neural Networks (NN), in which a Recurrent NN (RNN) is a special type of network that uses a feedback loop, usually to account for temporal information embedded in the sequence of data points presented to the network. Unlike RNN, the temporal dimension in our algorithm pertains to the sequential nature of the evolution process itself; and not to the data sampled from the problem solution space. Empirical evidence shows that the new algorithm better preserves the population's diversity, higher number of constructive crossovers and mutations. Furthermore, evidence shows that the RGA outperforms the standard GA on two NP problems and does the same on three continuous optimisation problems when aided by problem encoding information.

1 Introduction

The notion of "evolvability" is defined as *"the ability of a population to produce variants fitter than any yet existing"* [1]. Hence, the choice of selection, search operator and representation is vital to the performance of GA because they control the creation of new individuals throughout the evolutionary process. One aim of researchers in the Evolutionary Computation (EC) field is to discover new methods for increasing evolvability of evolutionary systems. The term evolvability does not only refer to how often offspring are fitter than their parents but also to the entire distribution of fitness values among offspring produced by a group of parents [1]. It should noted that even a random search can generate offspring that are fitter than their parents. Thus, to prove that a GA's performance is superior we need to show that the fitness distribution of the entire population is higher than that produced by a random search process. Obviously, for a successful evolvable search process not all parents in the population need to produce fitter offspring. It is usually those parents with higher than average fitness who carry the responsibility of making the search rewarding. This is because selection is naturally biased toward this slice of the population.

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 230–242, 2012.

To increase evolvability of individuals means to implicitly encourage search operators to produce a high correlation between parents and offspring fitness values in the next generation. This correlation has been explained by building block hypothesis in [4] as the correlation between parents and offspring fitnesses under the crossover operator. The building block is a sequence of genetic materials in a fit parent that is likely to produce fitter offspring upon joining a crossover process with other individuals.

In the standard form of GA, the fundamental idea that moves the search process is gleaned from the famous Darwinian theory of the *"survival of the fittest"* in which individuals that have superior fitness value (in relation to the problem to be solved) are considered fitter than inferior individuals and thus have a better chance to pass their good genetic materials (or more precisely, potentially good genetic materials) into the next generation. In this work, we consider another way of looking at the term *"fittest"* in which we ascribe this description to those individuals who are able to produce fitter offspring. Naturally, these individuals may not necessarily be the fittest in relation to solving the given problem. To this end, we propose a modification to the canonical GA where we evaluate individuals based on their parental abilities (more on this in Section 3). Evolvability refers to the potentiality of evolvement; rather than immediate improvements in fitness. Therefore, our algorithm works best when allowed to evolve for significantly higher number of generations.

2 Related Works

The concept of evolvability has been an active research area in both evolutionary biology and computer science for the past several decades. Hu and Banzhaf in [6] have argued that adopting new knowledge about natural evolution generated in areas such as molecular genetics, cell biology, developmental biology, and evolutionary biology would benefit the field of evolutionary computation. The authors discussed evolvability and methods for accelerating artificial evolution by introducing notions from biology and their potential in designing new algorithms in EC.

It has been recorded that the evolvability property has good effect on the search process. For example, in [2], the authors suggested that evolvability can effectively reduce the bloat in evolutionary algorithms that use variable length representations. In their work, the authors noted the similarity of bloat causes and evolvability theory, thus, they argue that reproductive operators with high evolvability will be less likely to cause bloat.

With the importance of evolvability as a research topic, several measurements have been proposed to quantify it. Wang and Wineberg [11], suggested two measures of evolvability one based on fitness improvement and the other based on the amount of genotypic change. The authors divided the population into three subpopulations, where the size of each sub-population is determined dynamically. The first sub-population uses selection based on fitness directly; the second subpopulation is based on the fitness-improvement-ratio; finally, selection for the

third sub-population is based on genotypic change. Each sub-population is filled by selecting chromosomes from the parent's generation under its own selection functions. Thereafter, the three sub-populations are merged, and the GA genetic standard operators are applied to form the next generation.

Unlike other works, in this paper, we propose a new paradigm for the evolutionary process to sustain population's evolvability and effectively improves its ability to find superior solutions. The main idea is based on rewarding the parents the fitness of their offspring. This is implemented by introducing an intermediate population (a feedback loop) to measure the ability of parents to reproduce. Upon re-evaluating parents' fitnesses, the algorithm proceeds as a standard GA; until next evaluation is due. (more details in Section 3).

3 Recurrent Genetic Algorithms

The proposed paradigm is broadly outlined in figure1. Firstly, as in standard GA procedures, we randomly generate an initial population t_i where $i \in \{0, 1, \ldots max\_generation\}$ and rank its individuals based on their fitness values. Using standard selection and genetic operators, the system generates population $\hat{t_i}$ from population t_i. Here, population $\hat{t_i}$ is used as feedback intermediate population (between population t_i and t_{i+1}) where it allows the system to discover the ability of individuals to produce fitter offspring. Once $\hat{t_i}$ is available, the algorithm uses its fitness values to reward parents in population t_i, thus, this intermediate population is used to evaluate individuals considering how much they effectively managed to push the search process forward, which, of course, may not coincide with the standard evaluation of individuals based on their fitness values.

Here $\hat{t_i}$ is used as a feedback loop as in the Recurrent Neural Networks (RNN) [5] where connections between units form a directed cycle used to allow it to exhibit dynamic temporal behaviour. Unlike feedforward Neural Networks, RNNs can use their internal memory to process arbitrary sequences of inputs. Although GA applications, in general, do not have temporal dimension as the basic focus lies in finding the best solution in a static "spacial" solution space, we use the the recurrencey concept in the intermediate population $\hat{t_i}$ to capture the temporal effects that the GA undergoes during the process of evolution. In other words, we use the recurrency notion to account for the evolvability of GA from one generation to the next.

Preliminary experiments show that an evaluation of individuals that is solely based on the fitness value of their offspring may not appropriately enhance the evolvability. This may be explained by several factors controlling the creation of any offspring; such as crossover/mutation point, original fitness value of the parent, selection pressure and the fitness value of the other parent in case of crossover. Therefore, if we simply allocate offspring fitness to parents, we would be neglecting all these important factors that contribute to creating the offspring. Also, this raises the question of which offspring to use when rewarding parents?. In [11] the authors used fittest offspring (i.e., the one with the highest fitness value) to measure the evolvability of an individual (assuming that it gives an

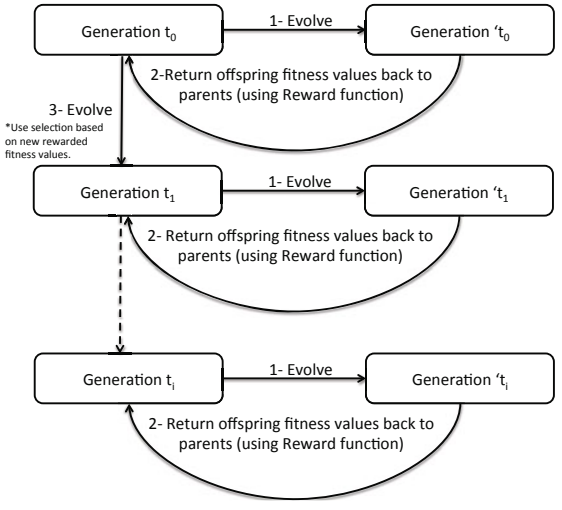


Fig. 1. RGA process outline

indication of the potential fitness upper limit that an individual can produce). This is not, however, entirely accurate estimation because the circumstances that resulted in creating this offspring are not necessarily to be repeated in future generations. For these reason, in this research, we opted for rewarding the parents based on the average fitness values of their offspring and the amount of genetic materials they passed onto their offspring. Here, we used a *Fitness Reward Function* (FRF) that rewards parents in population t_i based on their offspring's fitness values in the intermediate population $\hat{t_i}$.

For each crossover operator that parents P_x and P_y joined, where $x, y \in \{1, 2, \ldots, \mathit{population\_size}\}$, we use the following FRF:

$$FRF(Parent_x Fitness) = Offspring\_Fitness \times Parent_x\_contribution$$

$$FRF(Parent_y Fitness) = Offspring\_Fitness \times Parent_y\_contribution \quad (1)$$

where, *Offspring_Fitness* is the fitness value of the generated offspring, $Parent_x\_contribution$ and $Parent_y\_contribution$ are real numbers from the interval $(0, 1)$ to represent the proportion of genetic materials that each parent contributed when generating the offspring. Note that $Parent_x\_contribution + Parent_y\_contribution = 1$.

For each mutation operator that parent P_x joined, we use the following FRF:

$$FRF(Parent_x Fitness) = (Offspring\_Fitness \times Parent\_contribution_x) \quad (2)$$

Here, because the mutation operator is based on single parent, $Parent_x\_contribution$ is calculated as the amount of genetic materials that passed from the parent into the offspring.

The final rewarded fitness value is calculated as follows:

$$Parent_x\_Fitness = \frac{\sum FRF(Parent_x\_Fitenss)}{Number\_of\_offspring} \tag{3}$$

where, $\sum FRF(Parent_x\_Fitenss)$ is the summation of the FRFs for $Parent_x$ as shown in Equations 1 & 2 (whether it joined crossover and/or mutation operators) and *Number_of_offspring* is the total number of offspring produced by $Parent_x$ in population $\hat{t_i}$.

Note that once the RGA re-evaluates population t_i, it does not consider the original fitness of the parent any longer. Instead, all individuals are ranked based on their ability to produce offspring in relation to the amount of genetic materials they passed onto those offspring.

Naturally, the selection process may decides to leave some individuals unselected. Usually, those individuals have the most inferior fitness values of the whole population and therefore the selection process decides that they are not worthy to be allowed to be part of the next generation. We experimented with several alternatives as to how to evaluate parents that have not produced any offspring. One was to assign them the mean fitness value of the whole population, which resulted in poor performance. The best empirically based treatment was found to be allocating such parents "the least" fitness value allocated to individuals in the same population.

Despite the success of RGA (as will be shown in the experiments section), it suffers from an obvious disadvantage which is the extra computational cost introduced by producing and evaluating the intermediate population. Thus, it is fair to say that RGA has a slower convergence rate than standard GA. However, when comparing the performance of the two algorithms to each others, this disadvantage is mitigated by allowing RGA to evolve for only half the number of generations iterated by the standard GA.

3.1 Elitism

As explained previously, RGA uses FRF to reward parents based on the average fitness values of their offspring; and the amount of genetic materials they passed onto their offspring. Thus, for two parents who joined a crossover operator, the evaluation of their fitness values is dependant on the amount of chromosomes they passed to their offspring. This is both a strength and a weakness, though. On one hand, it is a strength in that the parent who passed more chromosomes into its offspring is more likely to pass a golden building-block of chromosomes that contributes in creating a fitter offspring therefore it receives bigger reward than the second parent. On the other hand, it is a weakness because if this golden building-block of chromosomes is a mixture of both parents (e.g., tail of the first parent concatenated with the head of the second parent) then our reward mechanism will not be fair. This unfairness of reward may divert the algorithm from pursing optimality by discarding already highly fit solutions. There is no practical way of knowing this information unless we have an explicit

knowledge about the problem. Therefore, since RGA has already invested in evaluating population $\hat{t_i}$ (the intermediate stage to measure the evolvability of population t_i) we copy the elite individuals from $\hat{t_i}$ into t_i. This has proven to improve the performance in some problems.

4 Experiments

The experiments have been designed to see whether RGA can sustain evolvability, and to see how diversity is closely related its behaviour. Our experiments covered three different problems, namely, NK-Landscape [8] *(unimodal problem)*, Hamming Centres [3], *(multimodal problem)*, and three different continuous optimisation problems.

4.1 NK Landscape

NK-Landscape was established by Stuart Kauffman in [8]. We investigated the performance of RGA under different values of N. Namely, we used $N = 20, 30, 40$ and 50. For each N value we tested three different K values. Thus, $K = \frac{N}{5}$ *(easy problem)*, $\frac{N}{2}$ *(hard problem)*, and $\frac{N}{2} + 5$ *(very hard problem)*.[1] For each N, K combination we tested the system using 20 independent runs. Results have been compared to the standard GA. As stated earlier, to allow fair comparison, we used exactly the same number of evaluations in both systems. Thus, we counted the number of evaluations in the intermediate generations and gave exactly the same to the standard GA. For both RGA and standard GA we used population of 100 individuals and evolved them through 500 generation (this includes the intermediate generations in RGA), crossover rate was 0.9 and mutation 0.1, tournament selection was of size 2. In RGA we applied 5% elitism (defined in sec. 3.1) and 5% standard elitism for the standard GA.

Table 1 summarises the results of 480 independent runs. As can be seen in the table, when the NK problems are easy (as in $NK(20, 5)$, $NK(30, 6)$, $NK(40, 8)$ and $NK(50, 10)$) both RGA and standard GA are even on average (where each system has better average in two out of four cases). Also, in these four easy problems the best solutions (across the 20 runs) achieved by standard GA are better than those achieved by RGA. This is because the problem's landscape is relatively smooth so standard GA search had a good chance to hit solutions near the global optima. Now, if we look at the hard and very hard problems, it is clear that RGA comes on the first place both in terms of average (i.e., average of best solutions in the 20 runs) and best (i.e., best achieved solution across the entire 20 runs). Thanks to the feedback loop, introduced through the intermediate populations RGA has higher evolvability than standard GA search.

To further compare the behaviour of RGA against its competitor, we measured the average of four different criteria for each system under each N, K

[1] In [7] the authors provided an indication of NK-landscape hardness under different settings.

Table 1. Summary of 480 independent runs (20 runs for each N,K combination with each system)

	RGA			GA		
	Mean	*Best*	*Std*	*Mean*	*Best*	*Std*
N20						
K=5	**0.76**	0.77	0.01	0.75	0.77	0.02
K=10	**0.76**	**0.78**	0.02	0.74	0.77	0.02
K=15	**0.75**	**0.79**	0.02	0.74	0.77	0.02
N30						
K=6	0.76	0.80	0.03	**0.78**	0.80	0.02
K=15	**0.72**	**0.77**	0.02	0.71	0.74	0.01
K=20	**0.71**	**0.74**	0.01	0.70	0.73	0.02
N40						
K=8	0.74	0.78	0.02	0.74	**0.79**	0.02
K=20	**0.70**	**0.72**	0.01	0.69	0.70	0.01
K=25	**0.69**	**0.71**	0.01	0.68	0.69	0.01
N50						
K=10	0.72	0.75	0.02	0.72	0.75	0.01
K=25	**0.69**	**0.74**	0.02	0.67	0.69	0.01
K=30	**0.67**	**0.72**	0.02	0.67	0.70	0.01

*__Bold__ numbers are the highest.

combination across the 20 runs. As can be seen in Figure 3, we compared first the average of best solution (generation by generation) for each system. Note that when the problems are easy both RGA and standard GA have almost the same performance. However, as the problem gets harder RGA gets better as its performance goes beyond the standard GA. What is impressive about these fitness curves is that they maintain an almost linear fitness increase for a prolonged period, and do so in a gradualistic manner, whilst standard GA reaches a plateau and no further improvement in the fitness is observed. Secondly, we compared the diversity of population (diversity was measured as the entropy of the population's fitness). RGA remarkably has higher diversity than standard GA in all experiments. It is worth noticing, though, that the diversity becomes higher as the problem gets harder. Finally, we compared the number of constructive crossovers, and mutations, by 'constructive' we refer to the crossover or mutation operator that resulted in a fitter offspring than its parents. It is clear that RGA has significantly a higher number of constructive crossovers and mutations than its competitor in all runs.

To compare the parents-offspring fitnesses across all generations we used the *fitness cloud* graph introduced by Vanneschi *et al.* in [10] to get a visual rendering of evolvability. Figure 2 illustrates the parent-offspring fitnesses in one of the runs versus their number of occurrence in all generations. As can be seen in standard GA fit parents are not able of producing fitter offspring all times. This

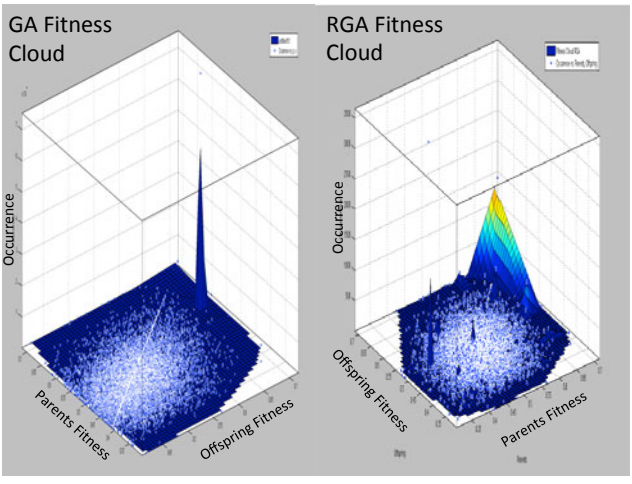


Fig. 2. Fitness Cloud: RGA vs. Standard GA NK(30,15)

is probably due to the destructive nature of the search operators (as it has been illustrated in figure 3 by the declining number of successful (i.e., constructive) crossover/mutations operators). We also noticed that in standard GA the search converges to a single solution dominating the whole population. This is clearly illustrated by the peak appearing in the figure, where a single parents-offspring fitness has high dominating number of occurrences. However, RGA shows that fit parents are able of producing fitter offspring most of the times. Also, the search does not allow a single individual to dominate the whole population as in standard GA, due to the diversity sustained by the RGA.

4.2 Hamming Centres

Hamming Centers is an NP-complete problem defined in [3] as follows. Lets a set S of k_i binary strings, where $i \in \{1, 2...I\}$, each of length n, and r is a positive integer. The objective is to find $n-$bits string y such that for every string k_i in S, the Hamming distance, $H(k_i, y) \leq r$.

We investigated the performance of RGA under different values of $n-$bits ($n =$ 20, 40, 60, 80, 100 and 120). For each n value we performed 20 independent runs. The size of the set S was 20 in all experiments. The fitness value was measured as the number of cases that string y satisfied the condition of $H(k_i, y) \leq r$. Thus, the optimal solution is equal to the size of S (which is 20 in our case). We used the same setting as in Section 4.1, except that the population's size was 500 and the number of generations was set to 1000 for each system.

Table 2 summarises the results of 240 independent runs. As can be seen in the table, RGA performance improves as the problem gets harder. To further compare the behaviour of RGA against its competitor, we measured the average of the four criteria (similar to the NK experiments in Section 4.1) for each system

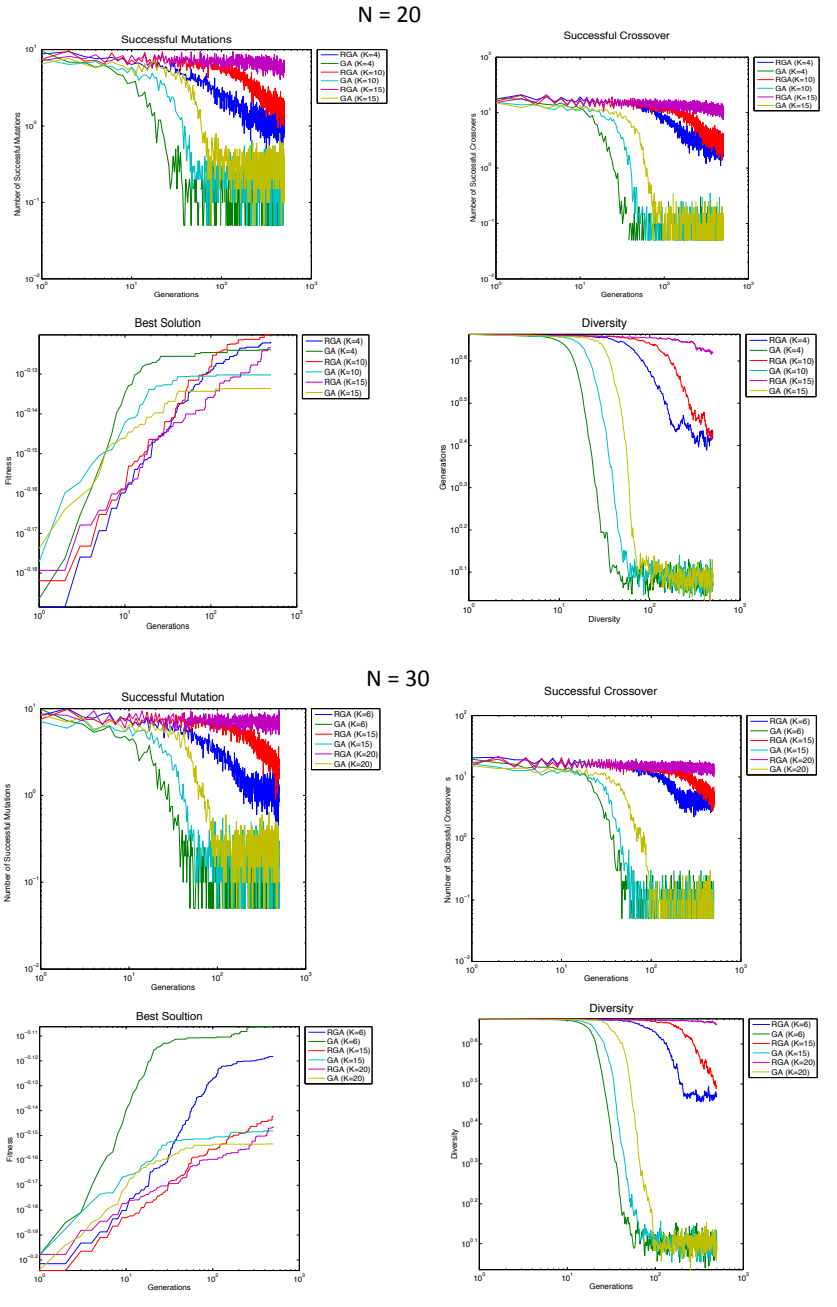


Fig. 3. RGA vs. Standard GA on NK landscape (N = 20, 30)

Table 2. Summary of 240 independent runs (20 runs for each n value)

	RGA			GA		
	Mean	*Best*	*Std*	*Mean*	*Best*	*Std*
n=20	**3**	3	0.00	2.75	3	0.43
n=40	**7.1**	8	0.71	6.95	8	0.70
n=60	**9.75**	**11**	0.73	8.7	10	0.69
n=80	**11.45**	**13**	1.00	10.9	12	0.84
n=100	**12.65**	**15**	1.20	12.1	14	0.90
n=120	12.6	**14**	0.94	**12.65**	13	0.56

*__Bold__ numbers are the highest.

under each n value across the 20 runs.[2] Looking at the average best solutions in each generation, we noticed that RGA behaves the same way it did in the NK problem. For $n = 20$ and 40 RGA and GA have almost similar performance. The performance gap increases in favour of the RGA in the remaining n values. Also, RGA has a higher diversity and higher number of constructive operators in all runs. Moreover, we noticed that these measures show better values as the problem gets harder. Despite the remarkable results obtained by the RGA it is fair to note that the difference in average best solutions in each generation between it and the standard GA is not as large in this problem as it was in the NK experiments, which may indicate that the RGA does not perform as well in multimodal problems as it does in unimodal ones.

4.3 Continuous Optimisation

We investigated RGA's performance in continuous optimisation problems. Three benchmark functions have been used in our experiments. Namely, Rastrigin function, Dixson & Price function, and Michalewics function. Functions' notations are defined in [9].

Here, we used the same setting as in 4.1, except that the number of generations was set to 5000 and population size was set to 500 for each system. Unlike the other two problems (i.e., NK-landscape and Hamming Centres), here individuals are coded as real numbers from the interval $(0, 5]$. Thus, each function receives n number of parameters and the RGA tries to optimise these parameters in such a way that maximises the function's output. In our experimentation we tried to maximise the test functions using RGA and GA under different number of parameters. Thus, we explored the systems performance at $n = 15$, 20, 25, and 30 for each function. For each n value we tested the systems using 20 independent runs. Table 3 summarises the results. As can be seen in the table, RGA has been outperformed by standard GA in most of the runs. We also noticed that averages of best solutions in each generation, diversity and the number of constructive crossovers/mutations have dropped drastically similar to the standard

[2] Due to the restriction on number of pages we are not able to present the full figures in this paper.

Table 3. Summary of 480 independent runs (20 runs for each n value) using original FRF

	RGA			GA		
	Mean	*Best*	*Std*	*Mean*	*Best*	*Std*
DixonPrice function						
$n = 15$	8227.37	8704.30	303.86	**8524.99**	**9078.43**	408.09
$n = 20$	14157.76	15211.50	591.62	**14954.73**	**16054.00**	779.36
$n = 25$	21715.74	23024.50	1051.07	**22759.18**	**25005.10**	1125.48
$n = 30$	30109.90	34750.80	2117.33	**31761.24**	**35665.00**	1835.55
Michalewics function						
$n = 15$	8.94	10.27	0.75	**9.30**	**10.87**	0.85
$n = 20$	10.57	12.95	1.08	**11.86**	**14.01**	1.16
$n = 25$	12.28	14.10	1.06	**13.77**	**16.22**	1.43
$n = 30$	13.97	16.30	1.19	**15.09**	**18.46**	1.72
Rrastrigin function						
$n = 15$	543.68	548.53	2.88	**545.95**	**554.09**	4.53
$n = 20$	725.72	732.41	3.34	**729.53**	**743.96**	6.28
$n = 25$	905.09	**916.32**	5.21	**907.34**	915.27	5.16
$n = 30$	1087.16	1096.39	4.80	**1090.50**	**1106.41**	7.84

*__Bold__ numbers are the highest.

GA (unlike the other two problems). However, RGA still maintaining slightly better diversity. These results were surprising given the superior performance by RGA in the previous two problems. We believe that the reason for this performance is largely attributed to the selection of the FRF, which needs to be defined differently for this type of problems. The standard FRF did not manage to reward parents in a favourable manner.

In an attempt to verify our assumption (i.e., degradation of performance in this problem is largely attributed to the selection of the FRF) we have introduced a slight modification in the FRF used for solving continuous optimisation problems. The FRF (defined in Section 3) assumes a linear dependency between impact on fitness and quantity of genetic material passed to the offspring, thus the fitness of the parent has been estimated as the weighted average of the fitness of the offspring with weights the chromosomes proportions of the offspring inherited from the parent. This assumption coincides with the building block hypothesis [4]. In other words, beneficial properties of parents are aggregated in (relatively) small code blocks at various locations within the genome.

This, however, and unlike the previous two test problems, does not work well in continuous optimisation problems where chromosomes are real numbers and not binary digits that can be summed up to represent the parents contribution to the formation of a specific offspring. To account for the different nature of the problem under investigation, we modified the FRF to consider the values contained in the chromosomes contributed by the parents rather than the mere number of chromosomes. So, the contribution of $Parent_x$ now is the sum of the values contained in the chromosomes contributed into the offspring. This

Table 4. Summary of 240 independent runs (20 runs for each n value) using modified FRF

	RGA		
	Mean	*Best*	*Std*
DixonPrice function			
$n = 15$	**8944.85**	**9509.38**	272.66
$n = 20$	**15645.78**	**16321.30**	475.21
$n = 25$	**23859.73**	**25401.10**	975.31
$n = 30$	**34229.60**	**36340.1**	1184.77
Michalewics function			
$n = 15$	**12.13**	**12.73**	0.44
$n = 20$	**16.29**	**17.42**	0.65
$n = 25$	**17.67**	**19.22**	0.93
$n = 30$	**20.19**	**23.16**	1.39
Rrastrigin function			
$n = 15$	**552.36**	**569.30**	6.51
$n = 20$	**732.04**	**742.12**	5.12
$n = 25$	**910.80**	**925.45**	5.16
$n = 30$	**1090.55**	1100.94	5.00

*__Bold__ numbers are the higher than standard GA.

modification did the trick, and the performance of the RGA was again superior to that of the GA in all test problems (see Table 4).

This modification confirmed our assumption, that the low performance in Table 3 was indeed due to the unsuitability of the FRF. Off course it can be argued that this enhancement is obtained by using additional knowledge about the problem under consideration. This is absolutely true, but it should also be remembered that the issue of "parent contribution" is in the heart of the RGA algorithm. In contrary to standard GA, this allows RGA to benefit from such problem "encoding" knowledge in a very simple and straightforward manner. So, although it is not fair to compare the performance of RGA to that of GA when the former benefits form problem encoding information, while the later does not; it is also unfair to deprive the former form so doing just because the later does not have the means to employ such useful information. In future research we will concentrate on this aspect of the algorithm.

5 Conclusions

This paper proposes a new paradigm, referred to as *Recurrent Genetic Algorithms (RGA)*, that attempts to sustain Genetic Algorithm (GA) evolvability and effectively improves its ability to find superior solutions. The main idea is formalised by simply introducing an intermediate population between subsequent generations. This intermediate population serves as a feedback loop where

the system reward individuals based on their abilities to produce better offspring. The idea of the feedback loop accounts for the temporal effects that the GA undergoes during the process of evolution.

As an experimental validation of the new paradigm on a non-trivial space and structured representation, we have considered two well-known NP benchmark problems: the NK-landscape problem, to show the RGA behaviour under unimodal problems and the Hamming Centres, to show RGA behaviour under mutlimodal problems. Moreover, we tested RGA using three continuous optimisation problems. Experimental evidence shows that RGA remarkably maintains higher diversity and increase the population's ability to produce fitter offspring in comparison to standard GA. Furthermore, empirical evidence shows that the new paradigm has outperformed standard GA on two NP problems and does the same on three continuous optimisation problems when aided by problem encoding information. This, indeed, shows that RGA has the potential to work well on real-world problems.

In future work we will test RGA on multi-objective optimisation problems.

References

1. Altenberg, L.: The evolution of evolvability in genetic programming. In: Kinnear Jr., K.E. (ed.) Advances in Genetic Programming, ch.3, pp. 47–74. MIT Press (1994)
2. Bassett, J.K., Coletti, M., De Jong, K.A.: The relationship between evolvability and bloat. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO 2009, pp. 1899–1900. ACM, New York (2009)
3. Frances, M., Litman, A.: On covering problems of codes. Theory of Computing Systems 30, 113–119 (1997), doi:10.1007/BF02679443
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
5. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice Hall, Upper Saddle River (1999)
6. Hu, T., Banzhaf, W.: Evolvability and speed of evolutionary algorithms in light of recent developments in biology. J. Artif. Evol. App. 2010, 1:1–1:28 (2010)
7. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Eshelman, L.J. (ed.) ICGA, pp. 184–192. Morgan Kaufmann (1995)
8. Kauffman, S., Levin, S.: Towards a general theory of adaptive walks on rugged landscapes. J. Theoret. Biol. 128(1), 11–45 (1987)
9. Molga, M., Smutnick, C.: Test functions for optimization needs. Test functions for optimization needs (2005)
10. Vanneschi, L., Clergue, M., Collard, P., Tomassini, M., Vérel, S.: Fitness Clouds and Problem Hardness in Genetic Programming. In: Deb, K., et al. (eds.) GECCO 2004, Part II. LNCS, vol. 3103, pp. 690–701. Springer, Heidelberg (2004)
11. Wang, Y., Wineberg, M.: The estimation of evolvability genetic algorithm. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2302–2309 (September 2005)

Splitting Method for Spatio-temporal Sensors Deployment in Underwater Systems

Mathieu Chouchane[1], Sébastien Paris[1],
François Le Gland[2], and Mustapha Ouladsine[1]

[1] LSIS, Aix-Marseille University, Domaine universitaire de Saint-Jérôme, Avenue Escadrille Normandie Niemen, 13397 Marseille Cedex 20, France
[2] INRIA Rennes, Campus de Beaulieu, 263 avenue du Général Leclerc, 35042 Rennes Cedex, France

Abstract. In this paper, we present a novel stochastic optimization algorithm based on the rare events simulation framework for sensors deployment in underwater systems. More precisely, we focus on finding the best spatio-temporal deployment of a set of sensors in order to maximize the detection probability of an intelligent and randomly moving target in an area under surveillance. Based on generalized splitting technique with a dedicated Gibbs sampler, our approach does not require any state-space discretization and rely on the evolutionary framework.

Key words: Evolutionary algorithm, Stochastic optimization, Generalized splitting, Genetic algorithm, Gibbs sampler.

1 Introduction

Consider a randomly moving target that tries to cross a closed area or to run away from a known position. We want to compute the best spatial and temporal deployment of sensors in order to maximize the detection probability of this intelligent and randomly moving target. Until recently, these types of problems were solved using operational research algorithms [8,12,14,19], which commonly model the constraints in both discrete time and space. In order conduct a continuous optimization, we have decided to use a novel probabilistic optimization algorithm based on the rare events simulation framework.

Probabilistic optimization algorithms are based on natural evolution processes of a population of individuals. Each of them represent a direction in the space of solutions to a problem. These algorithms have been developed by mimicking natural evolution and simplifying biological knowledge. All of them follow a simple scheme: (i) first, initialize arbitrarily a population and rate the fitness of each individual with a scoring function; (ii) using the individuals' fitness select a proportion of the population for reproduction. The selection process can be deterministic (for example select the best solutions) or completely random; (iii) then, generate a new population of solutions, from those selected previously, through a modification process that is often used to include recombination and/or mutation [1,13]. The new individual obviously shares many characteristics of its

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 243–254, 2012.

parents; (iv) we repeat the fitness-selection-reproduction process until certain stopping criteria are reached. Genetic algorithms have proven their efficiency for complex optimization problems. When the optimization is unconstrained or depends on "simple" or linear constraints, we can also use parametric methods such as covariance matrix adaptation [11] and its evolutions [6] or the cross entropy method [3,5] which is a parametric method based on the rare events simulation framework. However, when we are faced with a strongly constrained optimization problem, parametric optimization methods and other classic genetic algorithms cannot always handle the consistency of the solutions.

The goal of this paper is to present the method we have used to solve our real-world problem. This method is similar to the non-parametric genetic algorithms, but it is based on the generalized splitting method. This approach helped us to conduct a continuous optimization under a heterogeneous set of constraints. The splitting method mainly differs from a genetic algorithm in that a solution can remain in the pool for iterations before it is modified or excluded, while in most of evolutionary algorithms, the parents are ousted from the new generation. Moreover, in the reproduction step, we use a dedicated Gibbs sampler as a special mutation operator.

Along this paper, we will try to point out the commonalities and the differences between the two approaches. We will also try to justify why the splitting method may better suit these kind of problems.

Our article is organized as follows. The first section is devoted to the presentation of the real-world spatio-temporal scheduling problem. In the next two sections, we present our method in order to solve our optimization problem. We then apply and illustrate the splitting algorithm with the *flaming datum* [17,18] problem and offer some conclusions.

2 Problem Presentation: Spatio-temporal Search Efforts Planning

This section deals with the presentation of our real-world problem and its associated constraints.

Our goal is to maximize the detection probability of an intelligent and randomly moving target in an area Ω under surveillance. This can be achieved by optimizing the spatio-temporal deployment of a set of limited sensors during a period T. A spatio-temporal deployment consists of a list of sensors activations and a set of their associated position.

The target that we consider is not only smart but also reactive and its trajectory is unknown (and depends in practice on random variables).

2.1 The Solution Constraints

We have a set of P sensors $\boldsymbol{s}_i$ that we may spatially and temporally deploy in our operational theatre $\Omega \times [0, T]$ in order to detect a smart and reactive target. Let $\boldsymbol{X} \in \mathcal{X}$ a solution. So we can define

$$\boldsymbol{X} \triangleq \{\boldsymbol{s}_i(\boldsymbol{\tau}_i)\}_{i=1,\ldots,P} \text{ with } \boldsymbol{\tau}_i = \{t_{i,1}, \ldots, t_{i,j}\}_{j=1,\ldots,np_i}. \quad (1)$$

$\boldsymbol{s}_i$ corresponds to the sensor i ($i \in \{1 \dots P\}$) position, while $\boldsymbol{\tau}_i$ is a vector containing its np_i ($np_i \leq E_{max}$) instants of activation (in $[0, T]$). We also denote by $\mathcal{C}$ all the spatial and temporal constraints on the feasible solutions. As soon as a sensor is deployed (for example: dropped and set up), it is powered on and starts consuming energy; it becomes out of service when its battery is empty. Until that moment, it can send a limited number of pings (E_{max} times maximum).

Also, we assume that the sensors are only active and do not cooperate. In other words, they can only detect a target when they send a signal. Since they are not autonomous, they only ping when they are in the radio range of a moving commanding station that requests it. This constraint is denoted by the "operator" visibility parameter until time $t \leq T$, $\varphi_t(\boldsymbol{X})$.

2.2 The Target Constraints

A feasible trajectory of duration T is denoted by the random variable $\boldsymbol{Y}_T \in \mathcal{Y}$. We are only given an *a priori* on the starting point of the target trajectory. This *a priori* is weak if the starting point is randomly sampled in the search space Ω. On the contrary, a strong *a priori* means its initial position is sampled from a Gaussian pdf centred on the last seen position.

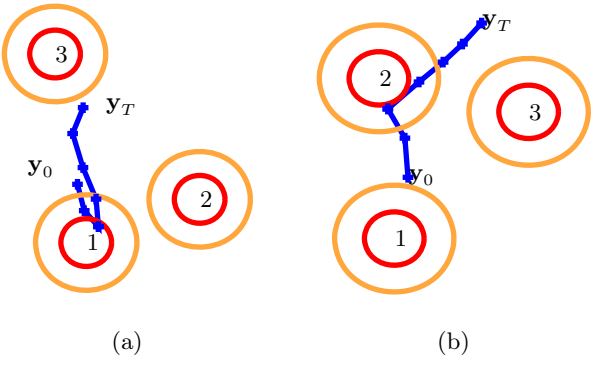


Fig. 1. (a) The target starts from $\boldsymbol{y}_0$. After a while, it is detected by an active sensor (sensor 1) and immediately escapes by following a radial course. (b) The target starts from $\boldsymbol{y}_0$. After a while, it detects an active sensor (sensor 2) and avoids it.

If the target detects a signal originating from a sensor, but is too far from it, it is able to avoid it before being detected while simultaneously memorizing its position. In our model, we define the target as detected if it enters the sensor's detection range (delimited by the red circle in figure 1). However, since the target is smart, if it comes close enough to a sensor which has just sent a signal (in the zone delimited by the red and orange circles in figure 1), it detects this sensor and learns all of its specifications. Thus, it may decide to avoid this threat or to come closer and start an avoidance later. In all cases, when the target is notified of the existence of a sensor, it changes its course before it enters the detection

range. The target trajectory controls (for the avoidance of a sensor) are then directly influenced by the partial visibility (or knowledge) of the target on the deployed search efforts $\mu_t(\boldsymbol{X})$.

3 The Generalized Splitting Framework

The purpose of this section is to discuss the generalized splitting framework in the context of unconstrained (or simple constrained) optimization. This will be used in the next section.

Consider a random object $\boldsymbol{X}$ (vector, random variable, etc.) that takes values in some set $\mathcal{X}$ and is distributed according to a pdf f. Also consider a real-valued function S on $\mathcal{X}$, a threshold level $\gamma \in \mathbb{R}$ and assume that sampling from f is easy. Let $\gamma^\star = S(\boldsymbol{X}^\star)$. Most optimization problems involve finding $\boldsymbol{X}^\star$, defined by

$$\boldsymbol{X}^\star \triangleq \arg\max_{\boldsymbol{X}\in\mathcal{X}} \{S(\boldsymbol{X})\}. \tag{2}$$

Splitting theory is based on the observation that maximizing $S(\boldsymbol{X})$ is similar to estimating probabilities of the form

$$\ell(\gamma) = \mathbb{P}(S(\boldsymbol{X}) \geq \gamma) = \int_{\mathcal{X}} \mathbb{1}_{\{S(\boldsymbol{X})\geq\gamma\}}\ f(\boldsymbol{X})\ d\boldsymbol{X}, \tag{3}$$

given that this probability reaches zero when γ converges toward the optimal score $S(\boldsymbol{X}^\star)$. Maximizing $S(\boldsymbol{X})$ is also similar to sampling the set

$$\mathcal{X}_\gamma = \{\boldsymbol{X} \in \mathcal{X} : S(\boldsymbol{X}) \geq \gamma\} \subset \mathcal{X}. \tag{4}$$

with the idea that this set decreases toward $\boldsymbol{X}^\star$ when γ increases toward the unknown value $\gamma^\star$. However, when γ goes to $\gamma^\star$, the event $\{S(\boldsymbol{X}) \geq \gamma\}$ becomes more rare, and consequently the CMC estimator

$$\ell(\gamma) = \frac{1}{C}\sum_{i=1}^{C} \mathbb{1}_{\{S(\boldsymbol{X}_i)\geq\gamma\}} \text{ where } \boldsymbol{X}_i \sim f(\boldsymbol{X}) \tag{5}$$

has a relative error

$$RE_{CMC}(\ell(\gamma)) = \frac{\sqrt{1-\ell(\gamma)}}{\sqrt{C\ell(\gamma)}} \tag{6}$$

that increases to infinity. In order to reduce this relative error, we should increase the sample size C, but then, we would be faced with a computationally intractable problem. Moreover, when γ goes to $\gamma^\star$, it becomes increasingly more difficult to produce samples from $f(\boldsymbol{X})$ that would be close to $\boldsymbol{X}^\star$.

To address this problem, a technique called generalized splitting [4] and derived from Diaconis, Holmes and Ross researches [9] on MCMC (Markov chain Monte Carlo) allows us to compute $\ell(\gamma)$ in an easier and more precise way. For an optimization problem, we will find at least one solution that maximizes our criteria among all the solutions sampled to compute our probability of interest.

If we define a sequence of increasing thresholds γ, such that $\gamma_0 \geq \gamma_1 \geq \ldots \geq \gamma_L$ (with $\gamma_L \leq \gamma^\star$), we can rewrite $\ell(\gamma)$ as the following product of conditional probabilities:

$$\ell(\gamma) = \mathbb{P}_f\left(S(\boldsymbol{X}) \geq \gamma_0\right) \prod_{l=1}^{L} \mathbb{P}_f\left(S(\boldsymbol{X}) \geq \gamma_l | S(\boldsymbol{X}) \geq \gamma_{l-1}\right) = c_0 \prod_{l=1}^{L} c_l \ . \quad (7)$$

where

$$c_l = \mathbb{P}_{g^\star_{l-1}}(S(\boldsymbol{X}) \geq \gamma_l) \ . \quad (8)$$

and where the importance sampling density [16]

$$g^\star_{l-1}(X; \gamma_{l-1}) = \frac{\mathbb{1}_{\{S(\boldsymbol{X}) \geq \gamma_{l-1}\}} f(\boldsymbol{X})}{\ell(\gamma_{l-1})} \quad (9)$$

is precisely the conditional density of $\boldsymbol{X}$, given that $S(\boldsymbol{X}) \geq \gamma_{l-1}$. It is worth noting that the support of this density $g^\star_{l-1}(\boldsymbol{X}; \gamma_{l-1})$ is precisely the set $\{\boldsymbol{X} \in \mathcal{X} : S(\boldsymbol{X}) \geq \gamma_{l-1}\}$.

If we know how to draw independent and identically distributed random variables $\boldsymbol{X}_i$ over $\mathcal{X}_{l-1} \in \mathcal{X}$ from this importance sampling function, $\ell(\gamma)$ can be rewritten:

$$\ell(\gamma) = \mathbb{P}_f\left(S(\boldsymbol{X}) \geq \gamma_0\right) \prod_{l=1}^{L} \mathbb{P}_{g^\star_{l-1}}\left(S(\boldsymbol{X}) \geq \gamma_l\right). \quad (10)$$

With a judicious choice or a fair estimation of the $\{\gamma_l\}$ sequence, the event $\{S(\boldsymbol{X}) \geq \gamma_l\}$ is no longer a rare event (generally, $c_l \in \left[10^{-3}, 10^{-2}\right]$) under the distribution $g^\star_{l-1}(\boldsymbol{X}, \gamma_{l-1}, \mathcal{C})$ and therefore the c_l quantities can now be well approximated through a CMC estimator. Hence, a CMC estimator of $\ell(\gamma)$ is:

$$\widehat{\ell}(\gamma) = \prod_{l=0}^{L} \widehat{c}_l, \quad (11)$$

where $\widehat{c}_l = \frac{1}{C} \sum_{i=1}^{C} \mathbb{1}_{\{S(\boldsymbol{X}_i) \geq \gamma_l\}}$ and where $\boldsymbol{X}_i \sim g^\star_{l-1}(\boldsymbol{X}; \gamma_{l-1})$.

4 Solving Our Real-World Problem

4.1 Evaluating the Detection Probability

According to what we have explained in the first part of the article, we want to maximize the detection probability of a target until time T. This quantity is denoted by $S_T(\boldsymbol{X})$ and is given by the following equation:

$$S_T(\boldsymbol{X}) \triangleq \int_{Y_T \in \mathcal{Y}} f\left(\boldsymbol{Y}_T | \varphi_T(\boldsymbol{X}); \mathcal{C}\right) p\left(\boldsymbol{Y}_T | \mu_T(\boldsymbol{X}); \mathcal{C}\right) \, d\boldsymbol{Y}_T \ . \quad (12)$$

Here, $f\left(\boldsymbol{Y}_T | \varphi_T(\boldsymbol{X}); \mathcal{C}\right)$ is a cookie-cutter cost function that takes the value 1 if the studied trajectory $\boldsymbol{Y}_T$ satisfies some defined criteria (such as a number of

detections and a number of avoidances) and 0 otherwise, *i.e.* $f\left(\boldsymbol{Y}_T|\varphi_T(\boldsymbol{X});\mathcal{C}\right) = \mathbb{1}_{\{\boldsymbol{Y}_T \in A(\boldsymbol{X},\mathcal{C})\}}$ where $A(\boldsymbol{X},\mathcal{C})$ is the set of trajectories which are detected by the solution $\boldsymbol{X}$ and which respect the constraints $\mathcal{C}$. It depends on the visibility of the solution $\varphi_T(\boldsymbol{X})$. $p\left(\boldsymbol{Y}_T|\mu_T(\boldsymbol{X});\mathcal{C}\right)$ is the conditional pdf used to generate the target trajectories and depends on the target intelligence $\mu_T(\boldsymbol{X})$. As this is not the goal of this article, we do not give any more information on how we have implemented this cost function. Unfortunately, $S_T(\boldsymbol{X})$ is an integral with respect to the probability distribution of the (random, solution-dependant) target trajectory $\boldsymbol{Y}$ and its analytical expression is not available. A first approach should be to use the crude Monte Carlo method to obtain an unbiased estimator of $S_T(\boldsymbol{X})$, $\widehat{S}_T(\boldsymbol{X})$:

$$\widehat{S}_T(\boldsymbol{X}) = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{Y}_T^i|\varphi_T(\boldsymbol{X});\mathcal{C}), \text{ where } \boldsymbol{Y}_T^i \sim p(\boldsymbol{Y}_T|\mu_T(\boldsymbol{X});\mathcal{C}). \tag{13}$$

The trajectories $\boldsymbol{Y}_T^i$ are recursively generated using a first order motion state equation [15] as defined in [7]. To be concrete, we generate a large number of feasible trajectories $\boldsymbol{Y}_T^i, i = 1,\ldots,N$ and evaluate $f\left(\boldsymbol{Y}_T^i|\varphi_T(\boldsymbol{X});\mathcal{C}\right)$.

Note that the relative error associated with $\widehat{S}_T(\boldsymbol{X})$ given by the CMC estimator is

$$RE_{CMC}(\widehat{S}_T(\boldsymbol{X})) = \frac{\sqrt{1 - S_T(\boldsymbol{X})}}{\sqrt{NS_T(\boldsymbol{X})}}\ . \tag{14}$$

Also remark that the smaller the probability to estimate, the larger the relative error. To reduce this error, we have to increase the number of trajectories N. Knowing the probability we are meeting is above 10^{-3} (if they were below, planning would be useless), we have chosen $N \geq 50000$.

4.2 The Splitting Algorithm

To solve our problem, a customized version of the splitting method is used. The algorithm we apply is called generalized splitting for research efforts scheduling and is detailed below. For the sequel, we define the function $q(.,\mathcal{C})$ which plays the role of $f(.)$ in the simple case. To begin the computation, we generate an initial pool of feasible solutions with $q(.;\mathcal{C})$. Since a solution and the carrier trajectory are closely linked, we use our trajectory generator to obtain a pool of initial solutions that respect the whole constraints set $\mathcal{C}$.

To ensure our algorithm will not converge and stay into a local extremum, we developed a simple heuristic. If the current maximum score and the current threshold do not increase for a chosen number of times, we automatically reduce the value of the threshold. Through the decrease of the threshold, we start again to accept the feasible solutions generated by the moves and therefore, we reintroduce some diversity in the pool of solutions.

Algorithm 1. The GSRES algorithm

Given parameter ρ, sample number C and number of burn-in iterations b_l of the Gibbs sampler, follow the forthcoming steps:

1: **Initialization**. Set a counter $l = 1$. Generate C feasible solutions $\{\boldsymbol{X}_i\}, i = 1, \ldots, C$ and denote $\mathcal{X}_0$ the set containing them. Note that $\boldsymbol{X}_i \sim q(\boldsymbol{X}; \mathcal{C})$. Evaluate scores $\mathcal{S}_0 = \{\widehat{S}_T(\boldsymbol{X}_i)\}$ and sort in decreasing order $\mathcal{S}_0$ such that $\widehat{S}_T(\boldsymbol{X}_{j(1)}) \geq \widehat{S}_T(\boldsymbol{X}_{j(2)}) \geq \ldots \geq \widehat{S}_T(\boldsymbol{X}_{j(C)})$. We obtain $\widehat{\gamma}_0 = \widehat{S}_T(\boldsymbol{X}_{j(C_0)})$ with $C_0 = \lfloor \rho C \rfloor$. Define $\widetilde{\boldsymbol{X}}_0 = \widetilde{\boldsymbol{X}}_{0:0} = \boldsymbol{X}_{j(1)}$, $\widetilde{\gamma}_l = \widehat{\gamma}_{0:0} = \widehat{S}_T(\boldsymbol{X}_{j(1)})$.

2: **Selection**. Let $\widetilde{\mathcal{X}}_{l-1} = \{\widetilde{\boldsymbol{X}}_1, ..., \widetilde{\boldsymbol{X}}_{C_{l-1}}\}$ be the subset of the population $\{\boldsymbol{X}_1, ..., \boldsymbol{X}_C\}$ for which $\widehat{S}_T(\widetilde{\boldsymbol{X}}_i) \geq \widehat{\gamma}_{l-1}$. $\widetilde{\mathcal{X}}_{l-1}$ contains $\rho\%$ of the population. Notice that $\widetilde{\boldsymbol{X}}_i \sim g^{\star}_{l-1}(\boldsymbol{X}; \gamma_{l-1}, \mathcal{C})$ for $i = 1, \ldots, C_{l-1}$.

3: **Repopulation**. Apply one of these methods:

- Bootstrapping: sample uniformly with replacement C times from the population $\widetilde{\mathcal{X}}_{l-1}$ to define the temporary set of C solutions $\mathcal{X}^{boot}_{l-1}$.
- ADAM Cloning: make $\left\lfloor \frac{C}{C_l} \right\rfloor + B_i (i = 1, \ldots, C_l)$ copies of each population sample $\widetilde{\mathcal{X}}_{l-1}$. Here each $B_1, \ldots, B_{C_l}$ are $Ber(1/2)$ random variables conditional on $\sum_{i=1}^{C_l} B_i = C \mod C_l$. We then define the temporary set of C solutions $\mathcal{X}^{clon}_{l-1}$.

4: **Gibbs sampler**. Apply a random Gibbs sampler $\pi_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_{l-1}; \mathcal{C}) = \frac{1}{C_{l-1}} \sum_{i=1}^{C_{l-1}} \kappa_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i; \mathcal{C})$ with b_l burn-in iterations and the transition density κ_{l-1} to each sample of $\mathcal{X}^{boot/clon}_{l-1}$ (see section 4.3) to obtain $\mathcal{X}_l = \{\boldsymbol{X}_i\}$ such that $\boldsymbol{X}_i \sim g^{\star}_{l-1}(\boldsymbol{X}; \widehat{\gamma}_{l-1}, \mathcal{C})$ for $i = 1, \ldots, C$. Notice that the $\boldsymbol{X}_i, i = 1, \ldots, C$ should be approximately iid.

5: **Estimation**. Evaluate scores $\mathcal{S}_l = \{\widehat{S}_T(\boldsymbol{X}_i)\}$, $\boldsymbol{X}_i \in \mathcal{X}_l$. Sort in decreasing order $\mathcal{S}_l$ such that $\widehat{S}_T(\boldsymbol{X}_{j(1)}) \geq \widehat{S}_T(\boldsymbol{X}_{j(2)}) \geq \ldots \geq \widehat{S}_T(\boldsymbol{X}_{j(C)})$. We obtain $\widehat{\gamma}_l = \widehat{S}_T(\boldsymbol{X}_{j(C_l)})$ with $C_l = \lfloor \rho C \rfloor$. Deduct that $\widetilde{\boldsymbol{X}}_l = \boldsymbol{X}_{j(1)}$, $\widetilde{\gamma}_l = \widehat{S}_T(\boldsymbol{X}_{j(1)})$, $\widehat{\boldsymbol{X}}_{0:l} = \widetilde{\boldsymbol{X}}_l$ if $\widetilde{\gamma}_l > \widehat{\gamma}_{0:l-1}$, else $\widehat{\boldsymbol{X}}_{0:l} = \widehat{\boldsymbol{X}}_{0:l-1}$ and $\widehat{\gamma}_{0:l} = \max\{\widetilde{\gamma}_l, \widehat{\gamma}_{0:l-1}\}$.

6: **Stopping condition**. If one of the stopping condition is reached, stop the algorithm and give $\widehat{\boldsymbol{X}}_{0:l}$ as an estimator of the optimal solution. Else $l = l + 1$ and go back to step 2.

4.3 The Dedicated Gibbs Sampler

For our problem, we use a *random* Gibbs sampler $\pi_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_{l-1}) = \frac{1}{C_{l-1}} \sum_{i=1}^{C_{l-1}} \kappa_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i)$ with the transition density κ_{l-1} defined by:

$$\kappa_{l-1}\left(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i\right) = \sum_{j=1}^{6} \lambda_j \prod_{r=1}^{b_l} m_j\left(\boldsymbol{X}_i^r|\widetilde{\boldsymbol{X}}_i^{-r}\right) , \tag{15}$$

Here, $\boldsymbol{X}_i^r$ denotes the component r of a solution and $\boldsymbol{X}_i^{-r}$, all the components of $\widetilde{\boldsymbol{X}}_i$ excluding r. The λ_j are the probabilities of updating one component at a time, given that $\sum_j \lambda_j = 1$ and the m_j are the conditional pdf associated to the 6 moves defined in [7].

The random Gibbs sampler will randomly update b_l times the components of a solution $\widetilde{X}_i$. b_l varies during the simulation in this way: $b_l = b_0 + \alpha l$ where $\alpha \in \mathbb{R}_+^\star$. For the first iterations, $b_l < P$ and therefore this approach is faster than a systematic Gibbs sampler. On the contrary, when l is close to L, $b_l \geq P$. Thus, we do more updates than a systematic Gibbs sampler would do but we maintain more diversity in our solutions.

Since we do not know how to update a solution in a way that still satisfies the constraints $\mathcal{C}$, we first recursively propagate the modifications starting from the sensor/activation we have modified in the sequence of activations. Then we check its feasibility, that is, if it respects all the spatial and temporal constraints $\mathcal{C}$. We apply acceptance–rejection (for a limited amount of times) to each updated component until we find a feasible solution. Considering that the cost function S also verifies the consistency of a solution, an updated solution $\boldsymbol{X}_i$ from $\widetilde{\mathcal{X}}_{i,l-1}^{boot/clon}$ is then accepted with probability $\mathbb{1}_{\{S(\boldsymbol{X}_i) \geq \widehat{\gamma}_{l-1}\}}$.

Before we proceed further, let us introduce and recall a few notations. $\boldsymbol{s}_i \triangleq [s_{i_x}; s_{i_y}]^T$ denotes the ith sensor position (and more generally the ith sensor), P is the number of sensors in the current solution, P_{max} is the maximum number of sensors, np_i stands for the activations' number for sensor i while $t_{i,\{1,\ldots,np_i\}}$ and $\boldsymbol{\tau}_i$ respectively are the instants of activation of sensor i and the set of activation times associated with sensor i. Also denote by $t_{\boldsymbol{s}_i}$ the set up duration of the sensor $\boldsymbol{s}_i$. Remark that a sensor whose instants of activation are negative is considered as disabled. Consequently, deleting an instant of activation consists of assigning a negative value to this instant. Removing a sensor is then equivalent to deleting all of its instants of activation and ignoring it. Below are the details of the six moves.

1. **Add a sensor.** Sample a position $\boldsymbol{s}'_{P+1}$ from $\mathcal{U}(\Omega; \mathcal{C})$ for the new sensor. Then draw its first instant of activation $t'_{P+1,1} \sim \mathcal{U}([t_{\boldsymbol{s}_{P+1}}, T])$.
2. **Add an instant of activation.** First, choose a sensor randomly *i.e.* draw j uniformly in $\{1, \ldots, P\}$. If $np_j < np_{max}$ then draw $t'_{j,np_j+1} \sim \mathcal{U}([t_{j,1}, T])$.
3. **Remove a sensor.** To apply the move m_3, we apply the following steps : choose a sensor randomly, *i.e.* draw j uniformly in $\{1, \ldots, P\}$. Then delete all of its instants of activation and mark it as disabled
4. **Remove an instant of activation**. Choose a sensor randomly *i.e.* draw j uniformly in $\{1, \ldots, P\}$. We assume that $np_j > 1$. Choose an instant of activation $t_{j,k}$, *i.e.*, draw k uniformly in $\{2, \ldots, np_j\}$. Delete $t'_{j,k}$.
5. **Move a sensor.** Select a sensor $\boldsymbol{s}_j$ randomly, *i.e.* draw j uniformly in $\{1, \ldots, P\}$. Then, draw $\boldsymbol{s}'_j \sim \sum_{k=1}^2 w_k \, \mathcal{N}(\boldsymbol{s}_j, \boldsymbol{\Sigma}_k^2)$ with $\sum_{k=1}^2 w_k = 1$. Notice that the weights w_k may evolve during the optimization in order to promote one of the move versus the other. For this mixture of two Gaussian pdf the covariance of the first Gaussian defines a small move while the covariance of the second Gaussian defines a larger move.
6. **Swap two sensors.** If we assume there are at least 2 active sensors, select two sensors $\boldsymbol{s}_k$ and $\boldsymbol{s}_r$ with k uniformly drawn in $\{1, \ldots, P\}$ and r uniformly drawn in $\{1, \ldots, P\} \setminus \{k\}$. For all $k = 2, \ldots, np_k$, delete $t'_{k,j}$ and $t'_{r,j}$ for all $r = 2, \ldots, np_r$. Swap their first instant of activation: $t'_{k,1}$ and $t'_{r,1}$.

5 Illustrative Example: The *Flaming Datum* Search Problem

The first result we present here concerns a scenario in which a target is running away from the position where it has just been detected. Its initial position is drawn from a Gaussian law centred on $\Omega/2$ and with a variance σ^2_{target}. Moreover, the target is supposed to be smart and reactive and therefore, while it is running away, it tries to avoid being detected another time. Considering that the search starts with a delay of t_c^{aoz} which represents the time of arrival of the hunter, we aim to maximize the chances to detect the target during the time T. We use $P_{max} = 10$ sensors that are able to ping only once. For this simulation, we use $C = 800$ solutions, $N = 70000$ trajectories, $b_0 = 2$, $b_l = b_0 + 0.2\ l$ and decide to keep 10% of elites ($\rho = 0.1$). We also let the algorithm perform up to 50 iterations. Because our algorithm is not able yet to adjust the number of sensors considering the cost of their deployment, we have chosen to work with a constant number of sensors. However, we have allowed the removal of a sensor if it is directly followed by an addition of a new sensor. We have used two of the six moves we have defined above : move a sensor and a combination of removing a sensor followed by the addition of a new sensor. The probability of each move to occur is 0.5.

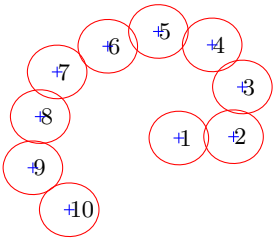


Fig. 2. Graphic of $\boldsymbol{X}^{\dagger}$: position and activation order of the 10 sensors. The red circles delimit the sensors' detection range.

In the best solution we obtain, the sensors position and activation describe a spiral. This result, illustrated in figure 2, is related to the studies of Washburn [12,18] and Son [17] for an only-spatial optimization case, *i.e.*, when the target is not able to avoid the sensor ("myopic" case). In this context, the best spatial sensor deployment designs an Archimedean spiral. Note that as the algorithm reaches the 30^{th} iteration, it has almost converged and the solution resembles a spiral. Figure 3 shows 3 steps of the simulation for the best solution found. The green crosses (+) represent non detected targets, the orange stars represent warned/avoiding targets and the red crosses (x) represent detected targets. The red circles delimit the sensors' detection range and the orange circles delimit the target avoidance zone.

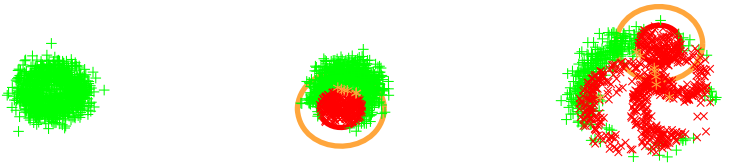


Fig. 3. 3 steps of the simulation for the best solution found. (a) Beginning of the simulation, the targets are not detected. (b) Activating of the first sensor. (c) Activation of the fourth sensor.

Continuing, we plot the optimization evolution behaviour versus iterations $\widehat{\gamma}_{0:l} = \widehat{S}_T(\widehat{\mathbf{X}}_{0:l})$ and $\mathbb{E}[\widehat{S}_T(\widehat{\mathbf{X}}_l)]$ which represents the mean score of the current population (figure 4). Both are smoothly increasing in a logarithmic way. On figure 5 we see that the support of scores pdf is large at the beginning ($l = 0$) but becomes thinner and converges towards a Dirac pdf as the optimization is conducted. Moreover, $\widehat{\gamma}_{0:l}$ increases and the standard deviation of the distribution decreases. Once the optimization is over, we observe that the detection probability reaches 0.9406 whereas when $l = 0$, the best score is below 0.25 and the mean scores $\mathbb{E}\left[\widehat{S}_T(\boldsymbol{X})\right]$ is equal to 0.0187 with a large standard deviation. This gap illustrates the efficiency of our approach, which also gives us good results with other types of scenarii [14].

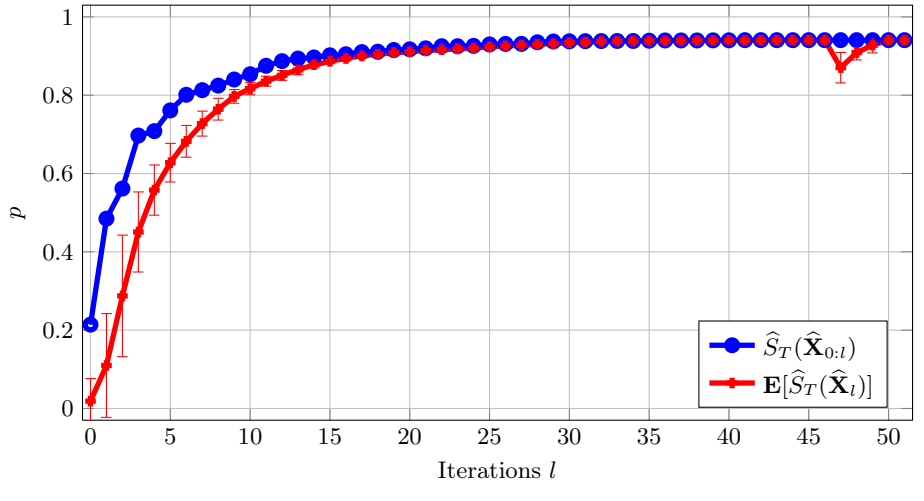


Fig. 4. In blue $\widehat{S}_T(\widehat{\mathbf{X}}_{0:l})$ and in red $\mathbb{E}[\widehat{S}_T(\widehat{\mathbf{X}}_l)]$ with $C = 800$, $N = 70000$, $L = 50$

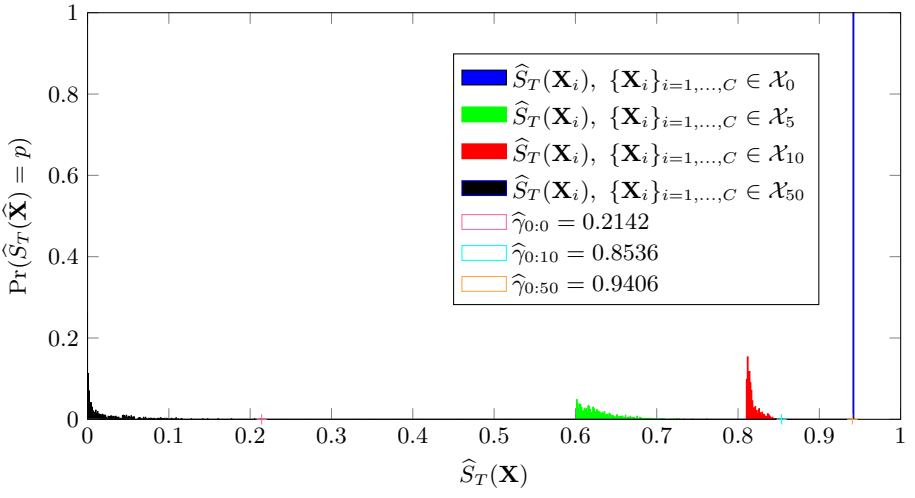


Fig. 5. Scores densities support versus iterations GSRES. In black $l = 0$, in green $l = 5$, in red $l = 10$ and in blue $l = 50$.

6 Conclusion and Prospects

In this work, we have presented an approach based on the rare events simulation framework and the generalized splitting algorithm. We have shown that this method is very similar to non-parametric genetic algorithms. This method has been applied to a strongly constrained optimization problem and tested with the flaming datum scenario.

The next step is to take the cost of each solution into account in order to extend our algorithm to the multi-objective case. To achieve this, we shall develop a method based on a Pareto-ranking algorithm. Although using a Choquet integral [10] may also be a good choice, it requires much more information from the decision maker. Furthermore, the Pareto-ranking method [2] has already been used with evolutionary algorithm with success.

Acknowledgement. This work was partially supported by DGA (Direction générale de l'armement). All four authors gratefully acknowledge Emile Vasta (DGA/Techniques navales) for his friendly support and interest in this work.

References

1. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evol. Comput. 1, 1–23 (1993), http://dx.doi.org/10.1162/evco.1993.1.1.1

2. Bekker, J., Aldrich, C.: The cross-entropy method in multi-objective optimisation: An assessment. European Journal of Operational Research 211(1), 112–121 (2011)
3. de Boer, P.T., Kroese, D., Mannor, S., Rubinstein, R.: A tutorial on the cross-entropy method. Annals of Operations Research 134(1), 19–67 (2005), http://dx.doi.org/10.1007/s10479-005-5724-z
4. Botev, Z., Kroese, D.: An efficient algorithm for rare-event probability estimation, combinatorial optimization, and counting. Methodology and Computing in Applied Probability 10(4), 471–505 (2008), http://dx.doi.org/10.1007/s11009-008-9073-7
5. Boubezoul, A., Paris, S., Ouladsine, M.: Application of the cross entropy method to the GLVQ algorithm. Pattern Recogn. 41, 3173–3178 (2008), http://portal.acm.org/citation.cfm?id=1385702.1385950
6. Bouzarkouna, Z., Auger, A., Ding, D.Y.: Investigating the Local-Meta-Model CMA-ES for Large Population Sizes. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A.I., Goh, C.-K., Merelo, J.J., Neri, F., Preuß, M., Togelius, J., Yannakakis, G.N. (eds.) EvoApplicatons 2010, Part I. LNCS, vol. 6024, pp. 402–411. Springer, Heidelberg (2010), http://hal.archives-ouvertes.fr/hal-00450238/en/
7. Chouchane, M., Paris, S., Le Gland, F., Ouladsine, M.: Splitting method for spatio-temporal search efforts planning (May 2011), http://arxiv.org/abs/1105.3351v1
8. Dell, R.F., Eagle, J.N., Alves Martins, G.H., Garnier Santos, A.: Using multiple searchers in constrained-path, moving-target search problems. Naval Research Logistics 43(4), 463–480 (1996)
9. Dianonis, P., Holmes, S.: Three examples of Monte-Carlo Markov chains. Discrete Probability and Algorithms, 43–56 (1994)
10. Grabisch, M.: L'utilisation de l'int'egrale de Choquet en aide multicritère à la décision. Newsletter of the European Working Group "Multicriteria Aid for Decision" 3(14), 5–10 (2006)
11. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, pp. 312–317. Morgan Kaufmann (1996)
12. Hohzaki, R., Washburn, A.: The diesel submarine flaming datum problem. Military Operations Research 6(4), 19–30 (2001)
13. Akbari, R., Ziarati, K.: A multilevel evolutionary algorithm for optimizing numerical functions. International Journal of Industrial Engineering Computations 2, 419–430 (2011)
14. Rodrigues, C., Michelon, P., Quadri, D.: Un modèle bi-niveau pour le problème de la recherche d'une cible dynamique. MajecSTIC 2009 (2009)
15. Rong Li, X., Jilkov, V.P.: Survey of maneuvering target tracking. Part i. Dynamic models. IEEE Transactions on Aerospace and Electronic Systems 39(4), 1333–1364 (2003), http://dx.doi.org/10.1109/TAES.2003.1261132
16. Rubinstein, R.Y.: The Gibbs cloner for combinatorial optimization, counting and sampling. Methodology and Computing in Applied Probability 11(4), 491–549 (2009)
17. Son, B.: Tracking Spacing for an Archimedes Spiral Search by a Maritime Patrol Aircraft (MPA) in Anti-submarine Warfare (ASW) Operations. Master's thesis, Naval Postgraduate School (December 2007)
18. Washburn, A.: Search and Detection. INFORMS (2002)
19. Washburn, A.: Branch and bound methods for a search problem. Naval Research Logistics 45(3), 243–257 (1998)

The Vehicle Routing Problem with Backhauls: A Multi-objective Evolutionary Approach

Abel Garcia-Najera

Departamento de Ingeniería Eléctrica, División de Ciencias Básicas e Ingeniería
Universidad Autónoma Metropolitana – Iztapalapa
Av. San Rafael Atlixco No.186, Col.Vicentina, C.P. 09340, México, D. F., México
gana@xanum.uam.mx

Abstract. In the Vehicle Routing Problem with Backhauls there are linehaul customers, who demand products, and backhaul customers, who supply products, and there is a fleet of vehicles available for servicing customers. The problem consists in finding a set of routes with the minimum cost, such that all customers are serviced. A generalization of this problem considers the collection from the backhaul customers optional. If the number of vehicles, the cost, and the uncollected demand are assumed to be equally important objectives, the problem can be tackled as a multi-objective optimization problem. In this paper, we solve these as multi-objective problems with an adapted previously proposed evolutionary algorithm and evaluate its performance with proper tools.

Keywords: Multi-objective optimization, vehicle routing problem.

1 Introduction

Since the Vehicle Routing Problem (VRP) was introduced more than 50 years ago, it has been a subject of extensive research and has become one of the most studied combinatorial optimization problems. The VRP's main objective is to obtain the lowest-cost set of routes to deliver demand to customers. Several variants of the problem exist because of the diversity of operating rules and constraints encountered in real-life applications. Thus the VRP should perhaps be viewed as a class of problems [9].

One of these variants is the VRP *with Backhauls* (VRPB) [8], which considers delivery and collection points. *Linehaul* customers are sites which have a demand of goods, thus a delivery has to be made from the depot. *Backhaul* customers are points where a quantity of goods has to be collected and taken to the depot. A practical example of this customer partition is that of the grocery industry, where supermarkets and shops are the linehaul customers and grocery suppliers are the backhaul customers [12].

An instance of the VRPB can be formally defined as follows [12]. There is a set $\mathcal{V} = \{0, \ldots, N_L, N_L + 1, \ldots, N_L + N_B\}$ of $N + 1$ vertices, $N = N_L + N_B$, representing the geographical location of the depot and N customers. Customers are represented by the vertices in subset $\mathcal{V}' = \mathcal{V} \setminus \{0\} = \{1, \ldots, N_L,$

J.-K. Hao and M. Middendorf (Eds.): EvoCOP 2012, LNCS 7245, pp. 255–266, 2012.

$N_L + 1, \ldots, N_L + N_B\}$. Subset $\mathcal{V}_L = \{1, \ldots, N_L\}$ corresponds to linehaul customers, while subset $\mathcal{V}_B = \{N_L + 1, \ldots, N_L + N_B\}$ represents the backhaul customers. Each customer $i \in \mathcal{V}'$ is geographically located at coordinates (x_i, y_i). Each customer $i \in \mathcal{V}_L$ has a demand of goods $d_i > 0$ to be delivered, while each customer $i \in \mathcal{V}_B$ has a supply $s_i > 0$ to be collected. There is a homogeneous fleet of K vehicles available to deliver and collect goods to and from customers, departing from and arriving at the depot, and having capacity $Q \geq \max \{ \max \{d_i : i = 1, \ldots, N_L\}, \max \{s_i : i = N_L + 1, \ldots, N_L + N_B\}\}$.

Hence one aims at finding a set of exactly K routes while minimizing the total cost, subject to the following constraints: (i) each vehicle services exactly one route, (ii) each customer is visited exactly once by one vehicle, (iii) a route is not allowed to consist entirely of backhaul customers, (iv) backhaul customers in a route, if any, must be served after the linehaul customers, and (v) for each route, the total load associated with linehaul and backhaul customers must not exceed the vehicle capacity Q. The fourth constraint is justified by the fact that many vehicles are rear-loaded and rearrangement of the loads on the trucks at the delivery points is not deemed feasible [8]. The constraint is also justified by the fact that the linehaul customers frequently prefer early deliveries, while backhaul customers prefer late collection [11].

An interesting generalization of this problem is the VRP *with Selective Backhauls* (VRPSB) [1], where each customer $i \in \mathcal{V}_B$ has an associated profit p_i, consequently

$$P = \textstyle\sum_{i \in \mathcal{V}_B} p_i \tag{1}$$

is the total possible profit. The VRPSB consists in determining a set of K routes with minimum net cost (cost minus collected profit), and the second constraint above is relaxed so that visiting backhaul customers is optional. If we do not restrict the cardinality of the set of routes to K, the study of the VRPSB will allow us, in fact, to consider this problem as a multi-objective problem.

Let $\mathcal{R} = \{r_1, \ldots, r_K\}$ be the designed set of routes and $r_k = \langle u(1,k), \ldots, u(N_k,k)\rangle$ the sequence of N_k customers serviced in the k-th route, where $u(i,k)$ is the i-th customer to be visited in that route, and $u(0,k) = u(N_k + 1, k) = 0$ is the depot. Then, the cost C_k and the profit P_k of that route are

$$C_k = \textstyle\sum_{i=0}^{N_k} c_{u(i,k)u(i+1,k)} \ , \tag{2}$$

$$P_k = \textstyle\sum_{i=1}^{N_k} p_{u(i,k)} \ , \tag{3}$$

where $p_{u(i,k)} = 0$ if $u(i,k) \in \mathcal{V}_L$. Having defined the VRPB and VRPSB, there are three objective functions that we shall concentrate on minimizing in this paper, namely the number of routes (f_1), the total cost (f_2), and the total uncollected profit (f_3):

$$f_1(\mathcal{R}) = |\mathcal{R}| = K \ , \tag{4}$$

$$f_2(\mathcal{R}) = \textstyle\sum_{k=1}^{K} C_k \ , \tag{5}$$

$$f_3(\mathcal{R}) = P \ - \ \textstyle\sum_{k=1}^{K} P_k \ , \tag{6}$$

subject to the constraints explained earlier.

Many approaches for solving VRPB have been proposed, from which a considerable number have presented heuristic methods. For example, Toth and Vigo [12] proposed a cluster-first route-second heuristic which may also be used for solving problems with asymmetric cost matrix. The approach exploits the information of the normally infeasible VRPB solutions associated with a lower bound. The bound used is a Lagrangian relaxation previously proposed by the authors. The final set of feasible routes is built through a modified Traveling Salesman Problem heuristic, and inter-route and intra-route arc exchanges. Ropke and Pisinger [11] surveyed the models of the backhaul constraints and a unified model that is capable of handling a number of VRP variants. The unified model can be seen as a Rich Pickup and Delivery Problem with Time Windows, which can be solved through an improved version of the large neighborhood search heuristic previously proposed by the authors.

Metaheuristic approaches have also been used for solving the problem. For example, Osman and Wassan [10] proposed two route-construction heuristics to generate initial solutions that are improved by a reactive Tabu Search. The reactive concept is used in a way to trigger the switch between different neighborhood structures for the intensification and diversification phases of the search. Brandão [2] presented a tabu search algorithm that starts from pseudo-lower bounds. Three different procedures were applied to generate the initial solution. The tabu search sequentially applies three phases using neighborhoods defined by insert, swap and interchange moves. An intra-route repair operator is applied if the precedence constraint is violated. Finally, Gajpal and Abad [5] used a multi-ant colony system for solving the VRPB: the first colony is used to assign customers to vehicles and the second is used to construct a route for a vehicle given the assigned customers, i.e. to solve the underlying Traveling Salesman Problem. After routes are constructed, they apply three local search procedures.

In all these studies, VRPB has been solved considering the minimization of the total cost only, since the number of routes is fixed to K. As far as we are concerned, there are no previous studies regarding the solution of the VRPB under multiple objectives. Furthermore, to the best of our knowledge, VRPSB has not been subject of study.

The remainder of this paper is organized as follows: The next section reviews the main concepts of multi-objective optimization and explains the performance metric that is used here to compare algorithms. Our proposed approach for solving the VRPB and VRPSB as multi-objective problems is described in Section 3. Then, Section 4 presents and evaluates the results from our algorithm. Finally, we present our conclusions in Section 5.

2 Multi-objective Combinatorial Optimization

Any multi-objective combinatorial optimization problem can, without loss of generality, be defined as a minimization problem of the form

$$\text{minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_F(\mathbf{x})) \tag{7}$$

subject to the constraints

$$g_i(\mathbf{x}) \leq 0 \qquad \forall\, i = 1, \ldots, p\ , \tag{8}$$

$$h_j(\mathbf{x}) = 0 \qquad \forall\, j = 1, \ldots, q\ , \tag{9}$$

where $\mathbf{x} \in \mathcal{X}$ is a solution to the problem, $\mathcal{X}$ is the solution space, and $f_i : \mathcal{X} \to \mathbb{R}$, for $i = 1, \ldots, F$, are F objective functions. The constraint functions $g_i, h_j : \mathcal{X} \to \mathbb{R}$ in (8) and (9) restrict $\mathbf{x}$ to a feasible region $\mathcal{X}' \subseteq \mathcal{X}$.

Let $\mathcal{X}$ be the domain of solutions to a multi-objective optimization problem. We say that solution $\mathbf{x} \in \mathcal{X}$ *weakly dominates* (or *covers*) solution $\mathbf{y} \in \mathcal{X}$, written as $\mathbf{x} \preceq \mathbf{y}$, if $\mathbf{x}$ is at least as good as $\mathbf{y}$. Solution $\mathbf{x}$ *dominates* solution $\mathbf{y}$, written as $\mathbf{x} \prec \mathbf{y}$, if and only if $\mathbf{x} \preceq \mathbf{y}$ and $\mathbf{x}$ is strictly better than $\mathbf{y}$ in at least one objective. Consequently, one says that a solution $\mathbf{x} \in \mathcal{S} \subseteq \mathcal{X}$ is *non-dominated* with respect to $\mathcal{S}$ if there is no solution $\mathbf{y} \in \mathcal{S}$ such that $\mathbf{y} \prec \mathbf{x}$.

A solution $\mathbf{x} \in \mathcal{X}$ is said to be *Pareto optimal* if it is non-dominated with respect to $\mathcal{X}$, and the *Pareto optimal set* is defined as $\mathcal{P}_s = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{x} \text{ is Pareto optimal}\}$. Finally, the *Pareto front* is defined as $\mathcal{P}_f = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^F \mid \mathbf{x} \in \mathcal{P}_s\}$.

For multi-objective problems, we have to compare whole sets of solutions in order to evaluate performance, since the task of approximating the Pareto optimal set involves: minimizing the distance of the generated solutions (the *approximation set*) to the Pareto optimal set, and maximizing the diversity of the achieved approximation set, because it is desirable to avoid identical solutions in the resulting set [13].

One indicator that is commonly used for optimizer comparison is the *hypervolume* metric $\mathsf{M}_\mathsf{H}(\mathcal{A}, \mathbf{z})$, which concerns the size of the objective space defined by the approximation set $\mathcal{A}$, which is limited by setting a suitable reference point $\mathbf{z}$. When using this metric to compare the performance of two or more algorithms, the one providing solutions with the largest delimited hypervolume is regarded to be the best. Formally, for a two-dimensional objective space $(f_1(\mathbf{x}), f_2(\mathbf{x}))$, each solution $\mathbf{x_i} \in \mathcal{A}$ delimits a rectangle defined by its coordinates $(f_1(\mathbf{x_i}), f_2(\mathbf{x_i}))$ and the reference point $\mathbf{z} = (z_1, z_2)$, and the size of the union of all such rectangles is used as the measure. This concept can be extended to any number of dimensions F to give the general hypervolume metric [13]:

$$\mathsf{M}_\mathsf{H}(\mathcal{A}, \mathbf{z}) = \lambda\left(\bigcup_{\mathbf{x_i} \in \mathcal{A}} \{[f_1(\mathbf{x_i}), z_1] \times \cdots \times [f_F(\mathbf{x_i}), z_F]\}\right) \tag{10}$$

where $\lambda(\cdot)$ is the standard Lebesgue measure [4].

3 Multi-objective EA for Solving VRPB and VRPSB

We explain in this section the adaptation of a previously proposed Multi-objective Evolutionary Algorithm (MOEA) for solving both VRPB and VRPSB as multi-objective problems. The description below emphasizes the modifications to the original approach, which are the initial population, the mutation process and the solutions repair, and we ask the interested reader to refer to the work of Garcia-Najera and Bullinaria [7] for more details in the unchanged procedures.

Solution encoding. Since solutions to VRPB and VRPSB are lists of routes, which are themselves lists of customers, the proposed encoding is a list of lists. A solution encoding simply lists the list of customer in the order they are serviced.

Initial population. As is standard practice, the initial population is chosen randomly with the aim of covering the entire search space. Thus the MOEA starts with a set of *popSize* solutions, each being a randomly generated route constructed as follows. Since in VRPSB the uncollected profit, i.e. f_3 in (6), is to be minimized, we need solutions with different numbers of backhaul customers. Hence, we first choose a random number n_B in the interval $[0, |\mathcal{V}_B|]$ (in the case of the VRPB, $n_B = |\mathcal{V}_B|$). Then, n_B backhaul customers are randomly selected and a permutation of these customers is generated. Then, since the problem requires solutions with exactly K routes, these are created serving exactly one backhaul customer. Thus, K customers are taken from the previous permutation in the order they appear. The remaining backhaul customers are assigned to the first route. If a backhaul customer cannot be assigned to the first route due to the capacity constraint, it and the remaining backhaul customers are assigned to the second route, and so on. If after assigning backhaul customers to the K-th route there are still unassigned backhaul customers, new routes are created. Afterwards, a random permutation is generated with the linehaul customers identifiers. One linehaul customer is allocated to each of the previously generated routes. The remaining linehaul customers are assigned to the existing routes using the same procedure explained above.

Fitness assignment. Fitness is assigned by means of the non-dominance sorting criterion [3], where the population is divided into non-dominated fronts, and their depth specifies the fitness of the individuals belonging to them. In this case, the lower the front, the fitter the solution.

Solution similarity measure. For multi-objective algorithms, it is important that the final population contains solutions that represent the full Pareto front, rather than just a small portion of it. Diversity here not only refers to the number of distinct solutions in the population, but also to how different they are. To accomplish this, we use the similarity measure proposed by Garcia-Najera and Bullinaria [6], which considers each solution $\mathcal{R}$ as the union of the set of arcs $(u(i,k), u(i+1,k))$, and the similarity of two solutions equals the ratio between the number of arcs that are common to both solutions and the total number of arcs used by them. For the purposes of the proposed MOEA, a measure of how similar a given solution is to the rest of the population is also required. If P is the population of solutions, the similarity $S_{\mathcal{R}}$ of solution $\mathcal{R} \in \mathsf{P}$ with the rest of the solutions in P will be given by the average similarity of $\mathcal{R}$ with every other solution $\mathcal{Q} \in \mathsf{P}$. Consequently, one minus the average solution similarity defines the diversity Δ of the population P.

Parent selection. The standard tournament method size T_{size} is used in our algorithm for parent selection. A crucial difference is that, in addition to using fitness to select good parents, it also uses the similarity measure to maintain

population diversity. The first of two parents is chosen on the basis of fitness and the second on the basis of similarity.

Recombination. The MOEA here is designed to randomly select and preserve routes from both parents. First, a random number of routes is chosen from the first parent and copied into the offspring. Then all those routes from the second parent which are not in conflict with customers already copied from the first, are copied into the offspring. Finally, if there remain any unassigned customers, these are allocated, in the order they appear in the second parent, to the route where the lowest cost is achieved. If there is no way to insert such a remaining customer without violating a constraint, a new route is created. The recombination procedure is carried out with probability γ.

Mutation. Changes to a solution of any VRP variant are limited to the customer sequence and the customer assignment to routes. Having this in mind, mutation involves three basic functions:

`selectRoute.` Stochastically selects a route according to the ratio of the cost to the number of customers, i.e. routes with a larger cost and fewer customers are more likely to be selected.

`selectCustomer.` Stochastically selects one customer from a specific route according to the average length of its inbound and outbound arcs, i.e. customers with longer associated costs are more likely to be chosen. A special case exists for the first and last customers in a route, where only the outbound and inbound arcs, respectively, are taken into account.

`insertCustomers.` Tries to insert, one at a time, a set of customers into a specific route where the lowest cost is obtained.

These functions are used by the following mutation operators:

Reallocation. Takes a number of customers from a given route and allocates them to another. First, `selectCustomer` is used to choose two customers from the route. These are removed from the route, along with all those customers in between them. Then, `insertCustomers` attempts to reallocate the removed customers into any of the existing routes.

Exchange. Swaps customers between two given routes. This operator uses `selectCustomer` to choose two customers from each route. The sequences of customers between them are then removed from their route, and `insertCustomers` attempts to reallocate them into the other route. If one or more customers cannot be inserted into the other route, the original routes are preserved.

Modify_*n_B.* Randomly chooses whether to insert or remove one backhaul customer, except when $n_B = 0$ or $n_B = |\mathcal{V}_B|$, in which case one backhaul customer is added and removed, respectively. If a customer is to be removed, one route is selected with `selectRoute` and from this, one backhaul customer is chosen `selectCustomer`. If a customer is to be inserted into the solution, one of the backhaul customers which are not present in the solution is randomly chosen.

Then, `insertCustomers` is called to insert that customer into the route where the lowest cost is obtained.

Reposition. Uses `selectCustomer` and `insertCustomers` respectively to select one customer from a specific route and to reinsert it into the same route.

The mutation process proceeds as follows: `selectRoute` is used to chose two routes. If they are the same route, the *Reallocation* operation is performed and then *Modify_*n_B is called, otherwise the *Exchange* operator is executed. Then `selectRoute` selects another route and the *Reposition* operator is carried out.

Solution Repair. The result of the recombination and mutation processes could be an infeasible solution, specifically, one solution which is formed by one or more infeasible routes serving backhaul customers only. If such a solution exist, it is submitted to a repair process. This process consists, first, in identifying those infeasible routes. Then, one linehaul customer is randomly chosen and removed from the feasible routes and taking care that, after its removal, that route is still feasible. Finally, this customer is inserted in one of the infeasible routes. This process is repeated as many times as the number of infeasible routes.

Survival. In order to select individuals to form the next generation, the offspring and parent populations are combined and individual fitness is determined as described earlier. Those solutions having the highest fitness, i.e. falling in the outermost fronts, are taken to survive and form the next generation. When the population size is exceeded in the last selected front, similarity is computed for the solutions in that front, and the least similar are chosen.

Repetition. The whole process described above is repeated for a fixed number *noGen* of generations or until the diversity of solutions in the Pareto approximation has not changed during a number *noChange* of generations.

4 Experimental Study

The main purpose of our study is to solve the multi-objective VRPB and VRPSB and evaluate our MOEA performance, first, by comparing our results with those from previous approaches, and second, by means of using the hypervolume and diversity metrics and compare it with NSGA-II [3]. To accomplish this, both algorithms were set, first, to optimize the total cost and the number of routes simultaneously for solving the VRPB, and then, additionally to both objectives, to optimize the total profit for solving the VRPSB.

We tested our algorithm on the benchmark set of Goetschalckx [8] that includes 68 instances of sizes $N = 25$ to 200, which are grouped into 14 categories. We ran our algorithm and NSGA-II 30 times for each problem instance. The parameters of our algorithm were set to values that have proved to work well in preliminary testing: $popSize = 100$, $noGen = 5000$, $noChange = 300$, $T_{size} = 5$, $\gamma = 1.0$, and $\mu = 0.9$. For a fair comparison, all parameter values, processes and operators set in our MOEA were also set in NSGA-II, with the exception of the diversity mechanism due to NSGA-II using its crowding distance and MOEA its solution similarity measure.

Table 1. Best-known results, averaged over instance category, and results from MOEA set to minimize the number of routes and total cost for solving VRPB

IC	#I	N_L	N_B	Best-known	Best	%Gap	#BK	Avg. Best	Std. Dev.	%Gap	Gen.	Time
A	4	25	20	182299.75	182301.65	0.00	4	182788.56	33067.35	0.27	435.00	7.65
B	3	30	20	202163.33	202166.75	0.00	3	202194.12	35031.21	0.02	431.11	8.94
C	4	40	20	214794.00	215072.40	0.13	4	217134.64	26190.28	1.09	450.83	13.80
D	4	38	30	271137.36	271137.36	0.00	4	271828.03	57992.27	0.25	565.00	17.78
E	3	45	30	219266.53	219267.30	0.00	3	221062.99	18179.07	0.82	558.89	25.83
F	4	60	30	250841.75	252354.91	0.60	1	256015.18	17719.40	2.06	571.67	30.26
G	6	57	45	241493.50	241493.86	0.00	6	245810.83	34314.80	1.79	753.33	35.15
H	6	68	45	252537.30	252561.63	0.01	5	257861.93	7330.34	2.11	672.78	40.44
I	5	90	45	310381.86	311032.83	0.21	3	318160.98	22667.54	2.51	804.00	78.06
J	4	95	75	305293.75	306185.30	0.29	1	314142.74	24236.86	2.90	1337.50	134.40
K	4	113	75	367710.89	368185.22	0.13	0	379343.21	20166.37	3.16	1454.17	215.82
L	5	150	75	398800.66	404593.50	1.45	0	418893.39	15338.23	5.04	1366.00	334.93
M	4	125	100	379836.39	383665.75	1.01	0	393012.72	26469.49	3.47	2009.17	284.75
N	6	150	100	392087.87	399218.85	1.82	0	410687.50	16309.52	4.74	1764.45	373.02
O	6	200	100		469946.24			485325.71	19885.53		1827.22	834.99
Average				284903.21	286374.09	0.40		292066.91		2.16	940.99	114.34

4.1 Comparison with Previous Approaches

The first series of experiments were conducted in order to optimize the bi-objective VRPB. For every benchmark instance, the solution with K routes and the lowest cost was taken after each of the 30 repetitions. Then, the mean lowest cost was computed for each instance and these were averaged over instance category. The percentage gap and the average percentage gap between our best results and the best-known results were calculated. Table 1 presents these results. This table shows, in the first four columns, the name of the instance category, the number of instances, and the number of linehaul and backhaul customers. The next column presents the average of the best-known results, taken from the studies reviewed earlier, for the instances in that category. The best result from MOEA, the percentage gap between this and the best-known result, and the number of instances for which MOEA found the best-known result are presented, respectively, in the following three columns. The next three columns show the average of the best result obtained by MOEA after the 30 repetitions, its corresponding standard deviation, and the per cent gap between the average best result and the best-known. The last two columns are the average number of generations when MOEA stopped and the average execution time in seconds.

We can observe that MOEA was able to find the best-known solution for 34 out of 62 instances. Our best results for instances of size $N \geq 125$ are slightly higher than 1.0% above the best-known. In fact, the average percentage gap between the best results from MOEA and the best-known results is 0.4%. Regarding the average of the best results obtained after 30 repetitions, we can say that the average of the results for the instances in four categories is less than 1.0% above the best-known results, and, on average, the gap is 2.16% above the best-known. We can also observe that MOEA performed nearly 941 generations on average, which took approximately 115 seconds.

We would like to point out that, as far as we are concerned, no previous study has considered the solution of instances size $N = 200$, which means that the results presented in Table 1 are, up to now, the best-known. Furthermore, we would like to remind the reader that the main purpose of this study is the multi-objective performance analysis of MOEA while solving VRPB and VRPSB, and that the investigation presented above was carried out in order to know if MOEA is able to find the best-known solutions to VRPB instances, which is true for many of the Goetschalckx instances.

4.2 Multi-objective Performance

In order to compute the hypervolume covered by a non-dominated set, we require an appropriate reference point. For each instance, there is a solution, though not feasible, with maximal objective function evaluation, which is formed by N routes, i.e. one customer allocated to each route. This solution has a maximal cost C_{max}, which is the double of the cost from the depot to every customer. Thus, our reference point is $\mathbf{z} = (N, C_{max})$. To calculate the hypervolume, for each instance and repetition, we took the non-dominated set and computed the hypervolume covered by those solutions. Then, we applied a two-sample, two-tailed, unequal variance t-test to the two vectors of 30 hypervolume values, from MOEA and NSGA-II, to test if there is a difference at the 5% significance level. We also computed the diversity Δ of the non-dominated solutions from MOEA ans NSGA-II as described in Section 3 and the results were also submitted to a t-test. These results, averaged over instance category, are shown in Table 2. This Table is divided into five main columns. The first presents the instance category and the number of instances comprising that category. The next corresponds to the hypervolume performance metric, showing, for each algorithm, the mean hypervolume, the number of instances for which there is significant difference, and the algorithm which solutions delimit a significantly larger hypervolume (M = MOEA, N = NSGA-II). The following main column corresponds to the population diversity measure, which is structured in the same manner as that of the hypervolume. The last two show the average number of generations each algorithm ran and the average execution time, respectively, for each algorithm.

In this case, we observe that the difference in the hypervolume covered by the non-dominated solutions from both algorithms is statistically significant in only three instances, one in each of the categories D, I, and N, where non-dominated solutions found by MOEA cover a slightly larger hypervolume than those from NSGA-II. The hypervolume for the remaining 65 instances does not present a significant difference. Considering the diversity measure, there is a significant difference in only one of the instances in category O, where solutions found by NSGA-II are significantly more diverse. Regarding the average number of generations and execution time, we see that MOEA ran, on average, for nearly 18% more generations, which correspond to approximately 20% more time.

Overall, considering the previous analysis, we can say that, for the bi-objective optimization of the VRPB, both algorithms perform equally well, since, for the vast majority of the benchmark instances, there is no significant difference

Table 2. Mean results of the performance metrics for the non-dominated solutions found by MOEA and NSGA-II while solving VRPB as a bi-objective problem

Inst.		Hypervolume				Diversity				Generations		Time (s)	
C	#	MOEA	NSGA	tt	Alg.	MOEA	NSGA	tt	Alg.	MOEA	NSGA	MOEA	NSGA
A	4	0.5488	0.5484	0		0.8960	0.8955	0		435.00	427.50	7.65	7.34
B	3	0.5776	0.5774	0		0.8478	0.8464	0		431.11	416.67	8.94	8.86
C	4	0.6602	0.6600	0		0.5638	0.5709	0		450.83	447.50	13.80	13.51
D	4	0.5353	0.5353	1	M	0.9800	0.9733	0		565.00	497.50	17.78	13.71
E	3	0.6680	0.6676	0		1.0000	1.0000	0		558.89	517.78	25.83	17.72
F	4	0.7265	0.7259	0		0.6401	0.6212	0		571.67	545.00	30.26	28.66
G	6	0.7203	0.7199	0		0.9513	0.9326	0		753.33	641.11	35.15	28.51
H	6	0.7611	0.7612	0		0.8991	0.8890	0		672.78	618.33	40.44	37.25
I	5	0.7650	0.7644	1	M	0.8375	0.8396	0		804.00	717.33	78.06	79.05
J	4	0.7628	0.7628	0		0.9833	0.9800	0		1337.50	1102.50	134.40	114.31
K	4	0.7816	0.7812	0		0.9884	0.9896	0		1454.17	1115.00	215.82	157.06
L	5	0.8201	0.8198	0		0.8259	0.8132	0		1366.00	1150.00	334.93	296.58
M	4	0.7853	0.7848	0		0.9283	0.9369	0		2009.17	1499.17	284.75	258.74
N	6	0.8156	0.8151	1	M	0.9895	0.9640	0		1764.45	1488.89	373.02	347.22
O	6	0.8430	0.8426	0		0.8134	0.8597	1	N	1827.22	1533.33	834.99	630.31
Avg.		0.7181	0.7178			0.8763	0.8741			1000.07	847.84	162.39	135.92

between the hypervolume and diversity of the non-dominated solutions found by MOEA and NSGA-II.

We then ran both algorithms for solving the tri-objective VRPSB, minimizing the number of routes, the total cost and the uncollected profit. Since the VRPB benchmark instances we are using do not provide profit information, we consider the backhaul customer supply as unitary profit. Thus, for each instance, we have a maximal profit P defined in (1). Hence, the reference point to be considered for computing the hypervolume performance metric is now $\mathbf{z} = (N, C_{max}, P)$.

After every run of each algorithm, we recorded the non-dominated set and computed the hypervolume metric and diversity measure, and submitted these values to a t-test as we did in the bi-objective case. Results are presented in Table 3, which has the same structure as Table 2. Regarding the hypervolume performance metric, we can see that there is a significant difference between the hypervolume covered by the solutions from MOEA and that covered by the solutions from NSGA-II in 21 instances, being significantly larger that covered by solutions from MOEA in eight instances and in 13 that covered by solutions from NSGA-II. Let us now concentrate on solution diversity. In this respect, we observe that solutions from MOEA present a significantly higher diversity than those from NSGA-II in 49 instances and there is no instance for which the solution diversity from NSGA-II is significantly larger than that from MOEA. These results suggest that, despite both algorithms finding solutions that do not delimit a significantly different hypervolume to more than half of the instances, MOEA does find solutions significantly more diverse than those found by NSGA-II for the majority of the instances. This may be due to MOEA's parent selection, since one of the parent is chosen according to the similarity measure. We can also see that, for instances I, K, L, M, N and O, both algorithms performed 5000 generations, which is the maximum number of generations (*noGen*). This means that there were no 300 consecutive generations (*noChange*) in which the

Table 3. Mean results of the performance metrics for the non-dominated solutions found by MOEA and NSGA-II while solving VRPSB as a tri-objective problem.

Inst.		Hypervolume				Diversity				Generations		Time (s)	
C	#	MOEA	NSGA	tt	Alg.	MOEA	NSGA	tt	Alg.	MOEA	NSGA	MOEA	NSGA
A	4	0.5670	0.5660	0		0.5360	0.5209	0		1798.33	1714.17	41.51	32.25
B	3	0.6092	0.6101	1	M	0.4870	0.4599	2	M	925.56	928.89	29.30	24.20
C	4	0.6865	0.6581	1	M	0.5701	0.5404	2	M	3767.50	3986.67	238.13	170.95
D	4	0.5515	0.5519	0		0.3785	0.3446	3	M	1835.00	1761.67	71.25	55.23
E	3	0.6569	0.6800	1	M	0.4909	0.4658	2	M	1813.33	1787.78	129.34	84.90
F	4	0.7506	0.7652	3	N	0.5569	0.4559	4	M	4632.50	5000.00	598.49	472.86
G	6	0.7266	0.7244	0		0.3918	0.3768	0		2891.11	2892.22	228.08	185.23
H	6	0.7838	0.7568	3	M	0.4481	0.3958	5	M	4142.22	4577.22	617.13	451.72
I	5	0.7933	0.7847	1	M	0.4997	0.3816	5	M	5000.00	5000.00	1097.37	739.08
J	4	0.7617	0.7713	1	N	0.3449	0.3175	1	M	4216.67	4127.50	669.00	555.01
K	4	0.7941	0.7977	1	N	0.4084	0.2286	4	M	5000.00	5000.00	1130.00	1037.00
L	5	0.8116	0.8324	2	N	0.4449	0.2650	5	M	5000.00	5000.00	2057.49	2030.21
M	4	0.7799	0.7895	1	M	0.3770	0.3055	4	M	5000.00	5000.00	1230.79	1046.73
N	6	0.7832	0.8117	1	N	0.3009	0.1961	6	M	5000.00	5000.00	1705.96	1734.13
O	6	0.8169	0.8536	5	N	0.4473	0.2228	6	M	5000.00	5000.00	3602.91	2985.93
Avg.		0.7248	0.7302			0.4455	0.3652			3734.81	3785.07	896.45	773.69

solution diversity remained unchanged. On average, NSGA-II ran slightly more generations than MOEA. Finally, we can observe that MOEA's execution time was always higher than that of NSGA-II, except for instance category N. On average, MOEA takes approximately 16% more time to finish than NSGA-II.

5 Conclusions

We have analyzed the performance of a recently proposed Multi-objective Evolutionary Algorithm (MOEA) which has been modified for solving the bi-objective VRPB and the tri-objective VRPSB. This MOEA introduced a similarity-based selection to enhance solution diversity. We have proposed modified initial solution generation and mutation procedures and tested the adapted MOEA on 68 well-known VRPB benchmark instances.

Since VRPB concentrates on minimizing the total cost given a fixed number of vehicles, we compare the best solutions from MOEA with those from previous approaches. The analysis made here demonstrated that, despite the average of the best results from MOEA is approximately 2% above the best-known, MOEA was able to find the best-known solutions to many of the instances, specifically, to instances in which size $N \leq 100$.

Considering the bi-objective VRPB, we compared the performance of MOEA and NSGA-II by means of using the hypervolume performance metric and the diversity measure over the non-dominated solutions found by the algorithms. After applying a t-test over the hypervolume and diversity values, we did not find a significant difference, hence we concluded that both MOEA and NSGA-II performed equally well on the bi-objective problem.

Regarding the tri-objective VRPSB, we use the same tools for comparing the non-dominated solutions found by each algorithm. The t-test applied on

the hypervolume values showed us that there was a significant difference in 21 instances, in eight of which MOEA solutions have a higher hypervolume and in 13 NSGA-II solutions cover a larger hypervolume. The t-test applied on the diversity values demonstrated the significantly higher solution diversity on the non-dominated sets found by MOEA. These results suggest that, despite both algorithms finding solutions which do not cover significantly different hypervolumes for more than half of the instances, the non-dominated solutions found by MOEA do have a significantly higher diversity.

We have demonstrated that considering solution similarity in the optimization process result in an improved performance of MOEA, leading to finding best-known solutions and, in many cases, this performance is comparable to the popular and successful NSGA-II.

We now look forward to take advantage of this similarity and diversity information, for example, testing MOEA on stochastic and dynamic variants of the VRP. We are also focusing on reducing the execution time of MOEA in order to have a clear benefit over other approaches.

References

1. Baldacci, R., Bartolini, E., Laporte, G.: Some applications of the generalized vehicle routing problem. J. Oper. Res. Soc. 61(7), 1072–1077 (2010)
2. Brandão, J.: A new tabu search algorithm for the vehicle routing problem with backhauls. Eur. J. Oper. Res. 173(2), 540–555 (2006)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE T. on Evolut. Comput. 6(2), 182–197 (2002)
4. Franks, J.: A (Terse) Introduction to Lebesgue Integration. AMS (2009)
5. Gajpal, Y., Abad, P.L.: Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. Eur. J. Oper. Res. 196(1), 102–117 (2009)
6. Garcia-Najera, A.: Preserving population diversity for the multi-objective vehicle routing problem with time windows. In: Rothlauf, F. (ed.) Genetic and Evolutionary Computation Conference 2009, pp. 2689–2692. ACM (2009)
7. Garcia-Najera, A., Bullinaria, J.A.: An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Comput. Oper. Res. 38(1), 287–300 (2011)
8. Goetschalckx, M., Jacobs-Blecha, C.: The vehicle routing problem with backhauls. Eur. J. Oper. Res. 42(1), 39–51 (1989)
9. Laporte, G.: Fifty years of vehicle routing. Transport. Sci. 43(4), 408–416 (2009)
10. Osman, I.H., Wassan, N.A.: A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. J. Sched. 5(4), 263–285 (2002)
11. Ropke, S., Pisinger, D.: A unified heuristic for a large class of vehicle routing problems with backhauls. Eur. J. Oper. Res. 171(3), 750–775 (2006)
12. Toth, P., Vigo, D.: A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. Eur. J. Oper. Res. 113(3), 528–543 (1999)
13. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V 1998. LNCS, vol. 1498, pp. 292–304. Springer, Heidelberg (1998)

Author Index