UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Metaheuristic Algorithms for Examination Timetabling Problems

Nuno Miguel da Costa de Sousa Leite

Supervisor:	Doctor Agostinho Cláudio da Rosa
Co-Supervisor:	Doctor Fernando Manuel Fernandes Melício

Thesis approved in public session to obtain the PhD Degree in Electrical and Computer Engineering

Jury final classification: Pass with Distinction

2019

UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Metaheuristic Algorithms for Examination Timetabling Problems

Nuno Miguel da Costa de Sousa Leite

Supervisor: Doctor Agostinho Cláudio da Rosa
Co-Supervisor: Doctor Fernando Manuel Fernandes Melício

Thesis approved in public session to obtain the PhD Degree in Electrical and Computer Engineering

Jury final classification: Pass with Distinction

Funding Institutions – Fundação para a Ciência e a Tecnologia, under grants [UID/EEA/5009/2019] and [SFRH/PROTEC/67953/2010]

2019

Abstract

The timetabling problem involves the scheduling of a set of entities (e.g., lectures, exams, vehicles, or people) to a given set of resources in a limited number of time slots, while satisfying a set of constraints. Many timetabling problems found in practice are NP-complete decision problems. Hence, their approach using exact solution methods is only adequate for instances of relatively small size.

Timetabling problems are usually solved offline, well in advance of the moment in which they will be used. For example, in examination timetabling, timetables are constructed weeks before the beginning of the school period. This is done to allow students to plan the course work during the period (semester, quarter, etc.). Decision makers are usually interested in obtaining timetables with good solution quality, within a reasonable computation time limit.

In this thesis, the examination timetabling problem is solved using metaheuristic methods. The research begins with an investigation of local search approaches using the Kempe chain neighbourhood operator. A solution construction algorithm based on the saturation degree graph colouring heuristic, that can generate feasible solutions, is investigated. A study of the impact of local search intensity on the exam scheduling is made, allowing for the development of accelerated versions of the studied local search algorithms. An investigation of memetic algorithms seeks to find hybrid algorithms that achieve superior performance compared to evolutionary algorithms and local search alone. In addition, a multi-objective memetic approach is also investigated for solving multi-objective versions of the examination timetabling problem. A real examination timetabling problem (ISEL-DEETC benchmark) containing two examination epochs, emerged from practice, is investigated.

The proposed approaches were tested on the public Toronto and ITC 2007 benchmark sets and also on the proposed ISEL-DEETC benchmark set. Our techniques are able to attain competitive results and new upper bounds on the Toronto and ITC 2007 benchmark sets. Regarding the ISEL-DEETC benchmark set, the developed approach was able to attain a lower number of clash conflicts compared to the manual solution and in negligible time.

Keywords: Examination timetabling, ITC 2007 benchmark set, Local search, Memetic algorithms, Uncapacitated Toronto benchmark set.

Resumo

O problema de elaboração de horários envolve o agendamento de um conjunto de entidades (por exemplo, aulas, exames, veículos ou pessoas), para um determinado conjunto de recursos, num intervalo de tempo predefinido e satisfazendo um conjunto de restrições de vários tipos. Muitos problemas de horários encontrados na prática são problemas de decisão NP-completos. Portanto, a sua abordagem usando métodos exatos é adequada apenas para instâncias de dimensão relativamente pequena.

Nos problemas de elaboração de horários, os horários são geralmente elaborados com grande antecedência relativamente ao momento em que serão usados. Por exemplo, nos calendários de exames, os calendários são construídos semanas antes do início do período escolar, de forma a permitir que os alunos planifiquem as suas atividades durante o período (semestre, trimestre, etc.). Em geral, os decisores estão interessados em obter horários/calendários com boa qualidade de solução, dentro de um limite razoável de tempo de computação.

Na presente tese, o problema de elaboração de horários é resolvido usando métodos meta-heurísticos. A investigação começa com um estudo de abordagens de procura local usando o operador de vizinhança *Kempe Chain*. Segue-se a investigação dum algoritmo de construção de soluções viáveis, baseado na heurística de coloração de grafos *saturation degree*. É realizado um estudo do impacto da intensidade da procura local na organização de exames no calendário, permitindo o desenvolvimento de versões aceleradas dos algoritmos de procura local estudados. Da investigação de algoritmos meméticos, aplicados ao problema estudado, resultou o desenvolvimento de algoritmos híbridos que alcançam um desempenho superior em comparação com algoritmos evolutivos e procura local aplicados isoladamente. Além disso, uma abordagem memética multi-objetivo, para resolver versões multi-objetivo do problema de elaboração de horários, é também investigada. Um problema real de elaboração de calendários de exames (caso de teste ISEL-DEETC), contendo duas épocas de exame, é investigado.

As abordagens propostas foram testadas nos conjuntos de teste de referência Toronto e ITC 2007 (*International Timetabling Competition – 2007*), e também no conjunto de teste ISEL-DEETC. As técnicas desenvolvidas são capazes de alcançar resultados competitivos e novos limites superiores nos conjuntos de teste Toronto e ITC 2007. Em relação ao conjunto de teste ISEL-DEETC, a abordagem desenvolvida é capaz de atingir um número menor de conflitos em comparação com a solução manual e em tempo reduzido.

Palavras-chave: Algoritmos Meméticos, Algoritmos de Procura Local, Conjunto de Teste ITC 2007, Conjunto de Teste Toronto, Problema de Elaboração de Calendários de Exames.

Acknowledgements

I would like to thank to the people and the institutions directly involved in this work.

To professors Agostinho Rosa and Fernando Melício, the supervisors of my thesis, for their guidance and all the constructive talks we had.

To the Instituto de Sistemas e Robótica – Laboratório de Sistemas Evolutivos e Engenharia Biomédica (ISR-LaSEEB) for providing conditions to make this thesis possible. To the Portuguese Fundação para a Ciência e a Tecnologia (FCT), which partially supported this work, under projects [PEst-OE/EEI/LA0009/2011, UID/EEA/50009/2013]. To both Instituto Politécnico de Lisboa (IPL) and Instituto Superior de Engenharia de Lisboa (ISEL) for grant [SFRH/PROTEC/67953/2010], which partially supported this work from October 2010 to October 2014.

To my colleagues of Área Departamental de Engenharia de Eletrónica, Telecomunicações e de Computadores (ISEL-ADEETC) and to its board for the conditions to make this thesis possible.

To my friends and colleagues of ISEL-ADEETC, for their support and encouragement along the past years, André Lourenço, Artur Ferreira, Carlos Carvalho, David Coutinho, Filipe Freitas, Manuel Carvalho, Matilde Pato, Nuno Datia, Paulo Marques, Pedro Fazenda, Pedro Miguens, Pedro Sampaio, Rui Duarte, Tiago Dias, Vítor Fialho, and Valentim Madeira.

To my parents, I thank their unconditional love, support, dedication, and many efforts. To my sister Susana and brother Afonso, for their love, friendship and support. To my parents in law Rosa and João Simões for all the love and support.

Finally, I would like to thank my family. To my wife Magda and my children Tiago and Iara, I want to express my gratitude to them, for their unconditional love, patience, support, and encouragement in the course of this work. This thesis is dedicated to them.

[1] `http://www.onlinewebfonts.com/icon`

To the memory of my father-in-law, João Simões.

To my family, Magda, Tiago and Iara.

List of Acronyms

BB	branch and bound
CBS	conflict-based statistics
cEA	cellular evolutionary algorithm
cMA	cellular memetic algorithm
COP	Combinatorial Optimisation Problem
EA	Evolutionary Algorithm
EPP	Exam Proximity Problem
ESD	extended saturation degree
ETP	Examination Timetabling Problem
FastSA	fast simulated annealing
FastTA	fast threshold acceptance
GA	Genetic Algorithm
GD	great deluge
GRASP	greedy randomized adaptive search procedure
HC	hill climbing
HH	hyper-heuristic
HMOEA	hybrid multi-objective evolutionary algorithm
HSFLA	hybrid shuffled frog-leaping algorithm

IFS	iterative forward search
ILS	iterated local search
IP	Integer Programming
ITC 2007	Second International Timetabling Competition
LA	late-acceptance
MA	Memetic Algorithms
MOEA	Multi-Objective Evolutionary Algorithm
NSGA-II	non-dominated sorting genetic algorithm-II
SA	simulated annealing
SCE	shuffled complex evolution
SCEA	shuffled complex evolution algorithm
SD	saturation degree
SFLA	shuffled frog-leaping algorithm
SHC	stochastic hill climbing
SWO	“Squeaky Wheel” optimisation
TA	threshold acceptance
TS	tabu search
UEPP	Uncapacitated Exam Proximity problem

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Contents

1.1 The Timetabling Problem

Transportation companies, educational, health, and sport institutions, all have to solve timetabling problems several times in a year. Due to its combinatorial nature, solving timetabling problems for relatively large size institutions is a complex task. For this reason, efficient tools for constructing feasible and optimal (or near-optimal) timetables automatically are asked for by decision makers.

A significant amount of research has been conducted covering several applications of timetabling, e.g., sports timetabling (Trick, 2011), vehicle timetabling (de Oliveira & Vasconcelos, 2010), employee timetabling (Meisels & Schaerf, 2003), educational timetabling, which includes school timetabling (Avella, D'Auria, Salerno & Vasil'ev, 2007; Pillay, 2014), examination timetabling (Qu, Burke, McCollum, Merlot & Lee, 2009) and course timetabling (Abdullah, Burke & McCollum, 2005).

In timetabling, one has to schedule a set of events (lectures, exams, surgeries, sport events, trips) using a set of resources (teachers, nurses and medical doctors, referees, vehicles) over space (classrooms, examination rooms, operating rooms, sport fields), in a given period of time. For instance, in examination timetabling (Qu et al., 2009), the goal is to allocate exams and corresponding enrolled students to examination rooms over time periods. Additionally, a set of *hard* and *soft* constraints are considered. The hard constraints must be satisfied in order to have a feasible timetable; on the other hand, there is no obligation to satisfy the soft constraints, and violations of these may occur. The optimisation goal is usually the minimisation of the soft constraints violations.

Educational timetabling problem variants differ from each other, based on both the type of institution (university or school) and the type of constraints. According to Schaerf (1999), these problems are categorised into three categories, as defined in Table 1.1. In all these groups a similar characteristic is present: the produced timetables should be clash-free, i.e., students or teachers cannot attend more than one event (class or exam) at the same time. School timetabling differs from course timetabling in the sense that, in the former, students are organised in classes which receive lectures, usually in the same room; in this way, it is the teacher who moves to teach each class. In course timetabling, the students attend courses which are spread over the week; in this case, the students have to travel to attend classes. In examination timetabling, students cannot attend one or more examinations at the same time. Despite the similarity between examination and course timetabling, there are significant differences, which rely mainly on the imposed constraints. For instance, no student can attend more than a given number of examinations per day, or some examinations must be scheduled on a given order, among others.

The focus of this thesis is the *Examination Timetabling Problem* (ETP). Examples of

Table 1.1: Educational timetabling problem categories, as defined in (Schaerf, 1999).

Category	Description
School timetabling	The weekly scheduling of classes in a school, avoiding teachers having two classes at the same time, as well as the same class having two different lectures at the same time.
Course timetabling	The weekly scheduling of the lectures of a set of university courses, avoiding overlapping lectures of courses having common students.
Examination timetabling	The scheduling of the exams of a set of university courses, avoiding overlaps of exams of courses having students in common, and spreading the students' exams as much as possible.

hard constraints for the ETP include: schedule all exams (the timetable must be *complete*), do not exceed room capacity, guarantee room exclusiveness for given exams, guarantee that no students will attend more than one exam in the same time slot, guarantee exam ordering (e.g., exam A should be placed after exam B), etc. (McCollum, McMullan, Parkes, Burke & Qu, 2012). Real examination timetabling problems include the following soft constraints: avoid students being enrolled in two exams the same day and in consecutive periods, avoid students having examinations in distinct periods within a given gap (period spread constraint), allocate exams with more students enrolled at the beginning of the timetable to allow for exam grading and proofing, among others. The ETP is further classified as *uncapacitated* ETP (if the room capacity is unlimited) or *capacitated* ETP (if the room capacity is limited) (Qu et al., 2009).

The ETP can be modelled as a multi-objective problem since several objectives are considered (reflecting the interests of the various stakeholders such as students, institution decision makers, and teachers) (Burke, McCollum, McMullan & Parkes, 2008). However, due to its computational complexity, the ETP has been addressed as a single-objective problem (McCollum et al., 2012; Qu et al., 2009) or as a two-objective problem (Cheong et al., 2009; Côté, Wong & Sabourin, 2004), where the first objective is the minimisation of the soft constraint cost and the second objective is the minimisation of the timetable length. In standard benchmarks, such as the Toronto and ITC 2007 sets, there are essentially two goals, achieving feasibility, i.e. obtain a hard constraint cost of zero, and minimising the soft constraint cost.

The ETP belongs to the NP-complete class of problems (de Werra, 1985,9). It has been approached mainly by mathematical programming methods (for relatively small size instances), and by approximate methods such as Artificial Intelligence (Schaerf, 1999) methods and metaheuristics (Qu et al., 2009).

Examination timetable optimisation is mainly conducted offline. In fact, in educational institutions, timetables are constructed weeks before the beginning of the school period. This is done to allow students to plan the course work during the period (semester, quarter, etc.). Due to the ETP computational complexity, execution times of algorithms could last days in order to produce good quality timetables for a large instance. Hence, there are usually two main indicators for assessing the overall performance: 1) solution quality, and 2) algorithm time performance. These indicators are inversely proportional to each other, in the sense that if one wants to achieve better results, a longer execution time is needed in order to explore a larger set of solutions and vice versa.

1.2 Research Motivation

Decision makers are usually interested in obtaining timetables with good solution quality, within a reasonable computation time limit. However, if the used algorithm, or another algorithm, can produce a solution with better quality at the expense of more time, for instance, a couple of days more, it is often an acceptable scenario when solving timetabling problems. Algorithm performance improvement could be accomplished for example by means of effective parallelisation, i.e., converting the algorithm to its parallel form and executing it on parallel hardware.

Concerning the optimisation methods used to solve the ETP, these can be broadly divided in *exact* approaches (e.g., mathematical programming methods Woumans, Boeck, Beliën & Creemers (2016)) and *approximate* methods (e.g., heuristic algorithms). The use of exact methods to real problems of relatively large size is still a challenging task, due to the problem size and complexity imposed by the presence of a large number of constraints. An alternative to exact approaches is the use of approximate methods, such as heuristic algorithms, namely *metaheuristics* (Gendreau & Potvin, 2010; Glover & Kochenberger, 2003; Siarry, 2017) (e.g., local search and evolutionary algorithms). This class of algorithms was the subject of many research works, which showed its efficiency in solving educational timetabling problems.

Local search methods such as *simulated annealing* (SA) (Kirkpatrick, Gelatt & Vecchi, 1983), *tabu search* (TS) (Glover & Laguna, 1997), or *great deluge* (GD) (Dueck, 1993), comprise the bulk of successful methods applied to the ETP. However, the study of the effect of local search intensity, the choice of neighbourhood, and the use of feasible versus infeasible operators plus repairing, are still considered open research fields that require further research. One effective neighbourhood operator applied to the ETP is the *Kempe chain* neighbourhood (Thompson & Dowsland, 1998). In this neighbourhood, a solution exam, included in a Kempe chain, is perturbed in a feasible fashion. However, the use of the Kempe chain neighbourhood operator to state-of-the-art benchmark problems in ETP, such as the *Second International Timetabling Competition* (ITC 2007), has been the target of only a few research works (e.g., the works of Gogos, Alefragis & Housos (2012) and Rahman, Bargiela, Burke, Özcan, McCollum & McMullan (2014)).

Another aspect that has been investigated is the solution construction process for complex instances such as the ITC 2007 benchmark instances, in terms of quality and feasibility of the produced solutions (Gogos et al., 2012; Müller, 2009; Rahman et al., 2014). The use of the *saturation degree* (SD) graph colouring heuristic have proved to be one of the most efficient heuristics used in construction algorithms for the Toronto benchmark set (Cheong et al., 2009). However, its use on the ITC 2007 was the subject of few research works (Abdul Rahman, 2012; Bykov & Petrovic, 2013; Gogos et al., 2012).

Regarding the examination timetabling problem instances solved, all the benchmarks available (being the Toronto and the ITC 2007 benchmark sets the most used) encompass a single examination epoch. However, in practical timetabling problems found in school and universities, several examination epochs are carried out, typically two examination epochs: a regular one and a special one for resit exams, with different durations.

1.2.1 Hybrid Metaheuristics

In the recent years, the use of hybrid metaheuristics in the field of optimisation has grown considerably (Blum, Puchinger, Raidl & Roli, 2011; Blum & Raidl, 2016; Blum & Roli, 2003; Talbi, 2016). Hybrid algorithms have produced the best results for many optimisation problems in science and industry. Typical hybridisations involve metaheuristics, mathematical programming, constraint programming or machine learning (Talbi, 2016).

In the field of timetabling some hybrid metaheuristic algorithms were applied recently. In Cambazard, Hebrard, O'Sullivan & Papadopoulos (2012), a hybridisation of local search and constraint programming for the post enrolment-based course timetabling problem is made. In the works of Burke, Newall & Weare (1996), Alkan & Özcan (2003) and Abdullah, Turabieh, McCollum & McMullan (2010), *Memetic Algorithms* (MA) (Moscato, 1999; Neri, Cotta & Moscato, 2012) are applied to the ETP. A memetic algorithm is a hybrid algorithm where an *Evolutionary Algorithm* (EA) is combined with other components, typically local search.

Exploration and *exploitation* are two competing goals that influence the design of a metaheuristic. The exploration component is needed to ensure that the search space is sufficiently searched, in a global way, in order to provide a reliable estimate of the global optimum (Talbi, 2013). Exploitation is used to further refine the obtained solution in order to improve it. Population-based metaheuristics such as evolutionary algorithms, scatter search, particle swarm, and ant colonies are good at exploring the search space, and weak in exploiting the solutions found. On the other side, local search methods such as simulated annealing and tabu search, are powerful in terms of exploiting the local neighbourhood of a solution. The two classes of algorithms have complementary strengths and weaknesses (Talbi, 2013). Memetic algorithms were designed in order to simultaneously address the goals of exploration and exploitation in a hybrid algorithm.

1.3 Research Objective

Given the above described facts about the ETP along with the need to solve real-world examination timetabling problems efficiently, we have identified the following set of research objectives:

1. Solve the examination timetabling problem in a competitive way by exploring approaches based on single and multi-objective hybrid algorithms.
2. Formulation of a new benchmarking problem based in a real-world examination timetabling problem.
3. Application and analysis of the proposed techniques to real-world problem instances and to relevant public benchmarks.
4. Comparative analysis of the proposed approaches with state-of-the-art techniques.

The research begins with an investigation of local search approaches using the Kempe chain neighbourhood operator. A solution construction algorithm based on the saturation degree graph colouring heuristic, that can generate feasible solutions, is investigated. A study of the impact of local search intensity on the exam scheduling is made. An investigation of memetic algorithms seeks to find hybrid algorithms that achieve superior performance compared to evolutionary algorithms and local search alone. In addition, a multi-objective memetic approach is also investigated for solving multi-objective versions of the ETP. A real examination timetabling problem containing two examination epochs, emerged from practice, is investigated.

1.4 Research Contribution

The research reported in this thesis resulted in the following original research contributions:

1. Formulation of a new benchmarking problem, the ISEL–DEETC problem. The ISEL–DEETC benchmark set derives from a real-world problem found at the Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa. It is comprised of *two examination epochs* (a regular one and a special one for resit exams) with different durations, corresponding to a scenario found in practice in typical universities. The ISEL–DEETC formulation given comprises two parts: the single-epoch problem and the two-epoch problem. A first contribution is the addition of a real-world problem (the ISEL-DEETC single-epoch problem) to existing single-epoch benchmark data (Toronto, ITC 2007, etc.). A second contribution is the formulation of a new benchmark problem based on the ISEL–DEETC two-epoch problem.

2. A new construction algorithm that is able to construct feasible solutions for the ITC 2007. The algorithm is based on the *saturation degree* graph colouring heuristic.

3. Two new feasible neighbourhood operators based on the *Kempe chain* heuristic for the ITC 2007 benchmark set.

4. Development of new approaches and hybrids:

   (a) New accelerated versions of the standard *simulated annealing* and standard *threshold acceptance* local search algorithms. These algorithms use a newly proposed acceptance criterion that is able to reduce the number of evaluations carried out by the simulated annealing based algorithm, while not worsening the solution cost in a significant way.

   (b) Three new approaches based on single-objective memetic algorithms.

   (c) A multi-objective algorithm for solving examination timetabling problems.

1.5 Publications

The research results described in this dissertation have been published in peer-reviewed conferences and journals. The following is a list of papers and abstracts that resulted from the investigation undertaken.

Journal papers

- N. Leite, F. Melício, and A. C. Rosa. A Fast Simulated Annealing Algorithm for the Examination Timetabling Problem. *Expert Systems with Applications*, 2019, vol. 122, pp. 137–151.
  Reference: (Leite, Melício & Rosa, 2019).
- N. Leite, C. M. Fernandes, F. Melício, and A. C. Rosa. A Cellular Memetic Algorithm for the Examination Timetabling Problem. *Computers & Operations Research*, 2018, vol. 94, pp. 118–138.
  Reference: (Leite, Fernandes, Melício & Rosa, 2018).

Book chapters

- N. Leite, F. Melício, and A. C. Rosa. A Shuffled Complex Evolution Algorithm for the Examination Timetabling Problem. Vol. 620 of *Studies in Computational Intelligence* (IJCCI 2014 revised selected papers), Springer International Publishing, 2016, pp. 151–168.
  Reference: (Leite, Melício & Rosa, 2016a).
- N. Leite, F. Melício, and A. C. Rosa. A Hybrid Shuffled Frog-Leaping Algorithm for the University Examination Timetabling Problem. Vol. 613 of *Studies in Computational Intelligence* (IJCCI 2013 revised selected papers), Springer International Publishing, 2016, pp. 173–188.
  Reference: (Leite, Melício & Rosa, 2016c).
- N. Leite, R. F. Neves, N. Horta, F. Melício, and A. C. Rosa. Solving a Capacitated Exam Timetabling Problem Instance Using a Bi-objective NSGA-II. Vol. 577 of *Studies in Computational Intelligence* (IJCCI 2012 revised selected papers), Springer International Publishing, 2015, 115–129.
  Reference: (Leite, Neves, Horta, Melício & Rosa, 2015).

Conference papers

- N. Leite, F. Melício, and A. C. Rosa. A Shuffled Complex Evolution Based Algorithm for Examination Timetabling - Benchmarks and a New Problem Focusing Two Epochs. In Proceedings of the *6th International Conference on Evolutionary Computation Theory and Applications (IJCCI-ECTA 2014)*, 112–124.
  Reference: (Leite, Melício & Rosa, 2014).

- N. Leite, F. Melício, and A. C. Rosa. Solving the Examination Timetabling Problem with the Shuffled Frog-leaping Algorithm. In Proceedings of the *5th International Conference on Evolutionary Computation Theory and Applications (IJCCI-ECTA 2013)*, 175–180.
  Reference: (Leite, Melício & Rosa, 2013a).
- N. Leite, R. F. Neves, N. Horta, F. Melício, and A. C. Rosa. Solving an Uncapacitated Exam Timetabling Problem Instance using a Hybrid NSGA-II. In Proceedings of the *4th International Conference on Evolutionary Computation Theory and Applications (IJCCI-ECTA 2012)*, 106–115.
  Reference: (Leite, Neves, Horta, Melicio & Rosa, 2012).

Conference proceedings – abstracts

- N. Leite, F. Melício, A. C. Rosa. A Fast Threshold Acceptance Algorithm for Solving Educational Timetabling Problems. Presented at the *21st Conference of the International Federation of Operational Research Societies (IFORS 2017)*, 17–21 July 2017, Quebec, Canada (Leite, Melício & Rosa, 2017).
- N. Leite, F. Melício, and A. C. Rosa. A Hybrid Shuffled Complex Evolution Algorithm for the Examination Timetabling Problem. Presented at the *28th European Conference on Operational Research (EURO 2016)*, 4–6 July 2016, Poznań, Poland (Leite, Melício & Rosa, 2016b).

Doctoral Consortium

- N. Leite, F. Melício, and A. C. Rosa. Multiobjective Memetic Algorithms applied to University Timetabling Problems. *5th International Conference on Evolutionary Computation Theory and Applications (IJCCI-ECTA 2013)*, Doctoral Consortium.
  Reference: (Leite, Melício & Rosa, 2013b).

1.6 Outline of the Thesis

In addition to this introduction, this dissertation is divided into seven additional chapters and three appendices.

Chapter 2 describes the state of the art and related work in examination timetabling. It starts by introducing background concepts on optimisation problems in general and on the class of combinatorial optimisation problems in particular. Then, the family of heuristic algorithms is introduced. A survey of examination timetabling approaches related to our work is then given. The Toronto and ITC 2007 public benchmark sets, widely used in the timetabling community, are described at the end of this chapter.

In Chapter 3, a new benchmark timetabling problem, the ISEL–DEETC problem, is presented. This problem differs from the Toronto and ITC 2007 problems in that two examinations epochs are considered, instead of a single examination epoch.

Chapter 4 presents two local search approaches proposed for solving the examination timetabling problem. The approaches are accelerated versions of the standard *simulated annealing* and standard *threshold acceptance* local search algorithms.

In Chapters 5 and 6 three approaches based on single objective memetic algorithms are described. In the proposed algorithms, populations are organised into subsets that communicate with each other. Local search is then performed in each subset.

Chapter 7 presents a multi-objective memetic algorithm for solving the examination timetabling problem. Local search is used to intensify the search after the variation operators.

The thesis conclusions and directions of future work are provided in Chapter 8.

Appendix A describes the obtained results' statistical significance tests. Appendix B describes the algorithms' computation times. In Appendix C, the repositories containing the developed software are summarised.

Chapter 2

Solving University Examination Timetabling Problems

Contents

This chapter introduces some background concepts on optimisation using heuristic search, and overviews the state-of-the-art approaches for examination timetabling. In Section 2.1, the formulation of a generic optimisation problem is given, as well as the formulation of a *Combinatorial Optimisation Problem* (COP), from which the *Examination Timetabling Problem* (ETP) is derived. Section 2.2 describes the class of heuristic search algorithms used to solve combinatorial optimisation problems.

Section 2.3 provides a survey of some algorithmic approaches applied to solve the ETP, focusing the seminal works in the area and the recent advances. Section 2.3 ends with a summary of recent surveys concerning university timetabling.

In Section 2.4.1, the uncapacitated Toronto timetabling problem is formulated. Section 2.4.2 provides a description of the capacitated *Second International Timetabling Competition* (ITC 2007) benchmark set. A mathematical formulation of the ITC 2007 timetabling problem is given in McCollum et al. (2012).

2.1 Optimisation Problems

Let x be a solution, represented by a vector of n design variables $(x_1, x_2, \ldots, x_i, \ldots, x_n)$. Each of the design variable x_i can take values from a domain D_i (e.g., an interval $[x_i^L, x_i^U]$ if variables are continuous, or a certain discrete collection of values otherwise). The Cartesian product of these domains for each design variable is called the *decision space* D. Let $f_1, f_2, \ldots, f_m$ be a set of functions defined in D and returning real values. Under these conditions, an optimisation problem can be generically formulated by (Neri et al., 2012):

$$\begin{array}{lll} \text{Maximise / Minimise} & f_m & m = 1, 2, \ldots, M \\ \text{subject to} & g_j(x) \leq 0 & j = 1, 2, \ldots, J \\ & h_k(x) = 0 & k = 1, 2, \ldots, K \\ & x_i^L \leq x \leq x_i^U & i = 1, 2, \ldots, n \end{array} \tag{2.1}$$

where g_j and h_k are inequality and equality constraints, respectively.

From the definition above, if $m = 1$ then the problem is *single-objective*, while for $m > 1$ the problem is *multi-objective*. The presence/absence of the functions g_j and h_k make the problem more or less severely constrained. Finally, the continuous or combinatorial nature of the problem is given by the fact that D is a dense or discrete set, respectively. The formulation of the examination timetabling problems (presented in Section 2.4) approached in this thesis are derived from the general definition of (2.1).

An element $x \in D$ is said to be a *feasible solution* if all the constraints $g_j(j = 1, 2, \ldots, J)$ and $h_k(k = 1, 2, \ldots, K)$ are satisfied. In the context of single-objective problems, an *optimal solution* is a feasible solution x for which the objective function is maximised (considering a maximisation problem). If the problem is multi-objective, the set of optimal solutions comprise the set of trade-off (non-dominated) solutions that maximise the objective functions.

2.1.1 Combinatorial Optimisation Problems

The examination timetabling problem belongs to the class of NP–Complete combinatorial optimisation problems de Werra (1985,9). As mentioned above, this class of problems is characterised by discrete decision variables and a finite search space. The objective function and constraints, however, may take any form such as non-linear, non-analytic, black box, among others (Talbi, 2009).

2.2 Heuristic Search

Exact methods (such as backtracking algorithms and *branch and bound* (BB)) guarantee finding of optimal solutions of a COP. However, in examination timetabling, real problems of large size cannot be solved in a practical way using these methods, due to its combinatorial nature. Moreover, the decision maker is not usually interested in the optimal solution but instead in a feasible solution that is “close to” optimal, and thus exploring the entire state space tree may not be necessary.

In this thesis, the ETP is approached by using *heuristic algorithms*, a class of approximate methods (usually randomised algorithms) in which heuristics are used to solve an optimisation problem (Kreher & Stinson, 1999). In the context of heuristic algorithms, a heuristic is a method that performs a minor modification or sequence of modifications to a solution or partial solution, in order to obtain a different solution or partial solution. The modifications that are done involve a *neighbourhood search*. A heuristic algorithm is a method that iteratively applies one or more heuristics, following a certain design strategy.

2.2.1 Neighbourhood Function

To construct a heuristic, one first needs to define a *neighbourhood function*. Formally, a neighbourhood function is defined as

$$N : D \to \mathcal{P}(D). \tag{2.2}$$

The function in (2.2) declares a neighbourhood on D by assigning the set of direct neighbours to points in D. Note that $\mathcal{P}(D)$ denotes the set of all subsets (or power set) of D.

The *neighbourhood* of any element x in D is generally defined to be a subset of elements that are "similar" or "close to" x in some sense. The neighbourhood $N(x)$ may contain elements y that are not feasible (Kreher & Stinson, 1999).

2.2.2 Neighbourhood Search

Once a neighbourhood function is defined, one has to devise a method that tries to find a feasible solution in the neighbourhood of a feasible solution X. A *neighbourhood search*, based on a neighbourhood function N, will be an algorithm (possibly a randomised algorithm) that receives as input a feasible solution $X \in D$, and produces as output a feasible solution $Y \in N(X) \setminus \{X\}$, or *Fail* (Kreher & Stinson, 1999). Since the neighbourhood $N(X)$ may contain solutions Y that are not feasible, the neighbourhood search must guarantee that the produced solution is indeed feasible.

2.3 Examination Timetabling Algorithmic Approaches

This section provides an overview of some techniques that are related to our work.

2.3.1 Integer Programming

Mathematical programming approaches for solving the ETP were proposed in the research literature. In a recent work, Woumans et al. (2016) propose a column generation approach for solving the examination timetabling problem. The authors apply two mathematical models to solve the ETP at KU Leuven campus Brussels (Belgium), for the business engineering degree program, and apply the models to the sta83 and yor83 instances of the Toronto benchmark set from the literature. The reported results are good on smaller data sets such as the ETP at KU Leuven. The results are also good for the Toronto data set using one of the proposed models, obtaining new upper bounds on the sta83 and yor83 instances. However, on larger datasets such as the yor83 instance, the second model was not able to obtain a feasible solution within the time limit of 400 hours. As demonstrated by the conclusions of this research work, the use of exact methods to real problems of relatively large size is still a challenging task, due to the problem size and complexity imposed by the presence of a large number of constraints.

2.3.2 Graph Colouring

Welsh & Powell (1967) establish a relation between graph colouring and timetabling. In examination timetabling problems, the exams are represented by vertices in a graph, and an edge connects any two vertices having common students. This basic framework only enforces one hard constraint (the *clash* constraint), which guarantees that no student will attend two or more exams in the same period. The additional soft and hard constraints are considered separately and evaluated to obtain the solution fitness (Qu et al., 2009). The graph colouring problem consists of assigning colours to vertices, so that no adjacent vertices have the same colour. In our context, it corresponds to assigning timeslots to exams, while guaranteeing that the hard clash constraint is satisfied. Several graph colouring heuristics proposed (e.g., the *saturation degree* (SD) heuristic (Brélaz, 1979)) were applied to the ETP (e.g., as in Carter (1986)).

The application of the SD heuristic to the ETP was also studied in Cheong et al. (2009), among other graph colouring heuristics. The authors have concluded that the SD heuristic was among the two best out of five tested heuristics (*largest degree*, *color degree*, *saturation degree*, *extended saturation degree*, and *random*).

2.3.3 Local Search & Kempe Chain

The use of *simulated annealing* (SA) Kirkpatrick et al. (1983) for timetabling problems dates back to the 1990s, with the proposals of Dowsland (1990) and Abramson (1991). In a later investigation, Thompson & Dowsland (1996) use SA to solve a variant of the ETP (a multi-objective formulation of the ETP). The same authors propose in Thompson & Dowsland (1998) a SA approach to solve the ETP, comparing three neighbourhood operators (*standard* – where the neighbourhood comprises the set of solutions produced by modifying the colour of a single vertex, *Kempe chains*, and *S-Chains*) and conclude that the operator based on Kempe chains is the most effective. The algorithm was tested on eight ETP instances from different universities.

Kempe Chain Neighbourhood

In the *Kempe chain* neighbourhood (Thompson & Dowsland, 1998), a solution exam, included in a *Kempe chain*, is perturbed in a feasible fashion. As mentioned earlier, focusing only on exams and conflicts, each timetabling problem instance can be seen as a graph G where nodes are exams and edges connect exams with students in common. In a feasible timetable, a *Kempe chain* is the set of nodes that form a connected component in the subgraph of G induced by the nodes that belong to two periods Lü & Hao (2008).

The Kempe chain heuristic produces a new feasible assignment by swapping the period labels assigned to the conflicting exams, located in two distinct periods, belonging to a specified Kempe chain.

Figure 2.1 illustrates the use of the Kempe chain operator for the Toronto benchmark set. In Figure 2.1 (left), there are four Kempe chains: $K_1 = \{e_1, e_3, e_4, e_6, e_7, e_8\}$, $K_2 = \{e_2\}$, $K_3 = \{e_5\}$, and $K_4 = \{e_9\}$.

The pseudo-code of the Kempe chain move is given in Algorithm 1.

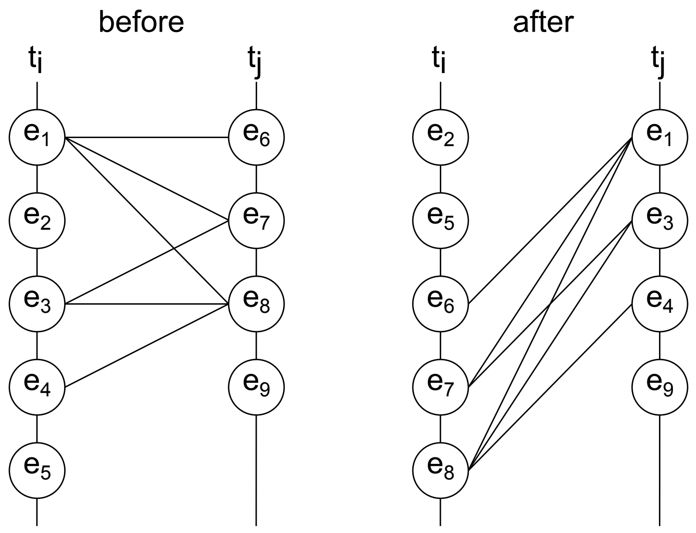


Figure 2.1: An example of the Kempe chain heuristic (Adapted from Demeester et al. (2012).) Moving exam e_1 from period t_i to period t_j while maintaining feasibility implies moving the conflicting exams (e_6, e_7, and e_8) from period t_j to period t_i. In turn, conflicting exams remaining in t_i (e_3 and e_4) also have to be moved to period t_j. In the worst case, when all exams in time slots t_i and t_j have conflicts, a swap of the exams between the two time slots is carried out.

Algorithm 1 Pseudo-code of the Kempe chain based heuristic.

```
1: function KEMPECHAIN
2:     Inputs: Two time slots $t_i$ and $t_j$ (randomly selected); let $K$ be a Kempe chain in
       the subgraph with respect to periods $t_i$ and $t_j$. Note that exams of $K$ in $t_i$ have hard
       conflicts with exams of $K$ belonging to $t_j$.
3:     The Kempe chain heuristic produces an assignment by replacing $t_i$ with $(t_i \setminus K) \cup$
       $(t_j \cap K)$ and $t_j$ with $(t_j \setminus K) \cup (t_i \cap K)$.
4:     Output: Updated time slots $t_i$ and $t_j$
5: end function
```

For the example shown in Figure 2.1, let us consider the Kempe chain K_1, given by $K_1 = \{e_1, e_3, e_4, e_6, e_7, e_8\}$. Initially, we have $t_i = \{e_1, e_2, e_3, e_4, e_5\}$ and $t_j =$

$\{e_6, e_7, e_8, e_9\}$. Then, after applying the Kempe chain heuristic (see Algorithm 1) t_i becomes $(t_i \setminus K_1) \cup (t_j \cap K_1) = \{e_2, e_5, e_6, e_7, e_8\}$ and t_j becomes $(t_j \setminus K_1) \cup (t_i \cap K_1) = \{e_1, e_3, e_4, e_9\}$. Using the Kempe chain neighbourhood, the move operator always produces feasible solutions for the Toronto data set.

In terms of implementation, Kempe chain-based operators are typically implemented in order to allow for solutions to be evaluated incrementally. In incremental (also called *delta*) evaluation (Corne, Ross & Fang, 1994), only the exam edges that are updated by the operator are evaluated, allowing for a substantial increase in the operator's performance.

Mühlenthaler (2015) investigates the structure of the course timetabling search space and establishes sufficient conditions for the connectedness of clash-free timetables under the Kempe-exchange operation. The SA metaheuristic was applied to the high school timetabling problem by Melício, Caldeira & Rosa (2000). The authors claim that any SA algorithm depends on how the structural elements are defined (i.e., solution space, generation of new solutions, cost function) and present a comparison for several parameters of the algorithm. In a later work (Melício, Caldeira & Rosa, 2004), the same authors analyse two well-known neighbourhood operators adapted to the school timetabling problem. The authors demonstrate that, for the studied problem, the double move intra-class neighbourhood always showed a better performance than the single move neighbourhood, even if the latter is improved with a heuristic method. The tests were made using real data from three different Portuguese schools. The SA metaheuristic was also applied to other educational timetabling problems (school and course timetabling) by Zhang, Liu, M'Hallah & Leung (2010), and Bellio, Ceschia, Gaspero, Schaerf & Urli (2016).

Burke & Bykov (2008) employ a variant of *hill climbing* (HC) to include a so-called *late-acceptance* (LA) strategy, which accepts a neighbouring solution evaluated several iterations before. Burke, Eckersley, McCollum, Petrovic & Qu (2010) investigate the use of hybrid variable neighbourhood approaches to university exam timetabling.

2.3.4 Evolutionary Algorithms

Colorni, Dorigo & Maniezzo (1991) investigate the application of a *Genetic Algorithm* (GA) to the timetable case. The application of a GA to timetabling and scheduling is also studied in Fang (1994). Wong, Côté & Gely (2002) apply a GA to solve the ETP. They present an exam timetable automation tool based on a genetic algorithm. A repair mechanism is developed in order to repair infeasible solutions that result from the application of the variation operators. The developed system was tested at the École de Technologie Supérieure of Université du Québec. Caldeira & Rosa (1997) approach the high school timetabling problem using genetic search. In their algorithm, the authors adopt a problem specific chromosome representation and use a repair algorithm after the genetic operators, in order to avoid searching through non-feasible timetables. The authors also explore different fitness functions. Fernandes, Caldeira, Melício & Rosa (1999a,9) also employ evolutionary algorithms to solve the high school timetabling problem. Pillay & Banzhaf (2010) apply a two-phase approach to the ETP problem. In the first phase, feasible timetables are produced, while improvements are made to these timetables in the second phase to reduce the soft constraint costs. Domain specific knowledge in the form of heuristics is used to guide the evolutionary process. For the variation operators, only a mutation operator was used. Beligiannis, Moschopoulos, Kaperonis & Likothanassis (2008) apply an evolutionary algorithm to the related problem of school timetabling. As in the previous work, no crossover operator is used in the algorithm. The authors state that using mutation is enough to generate new good solutions.

Soria-Alcaraz, Carpio, Puga & Sotelo-Figueroa (2012) investigate the use of a parallel model in a metaheuristic algorithm for solving the course timetabling problem. The developed parallel approach uses a so-called *Methodology of design* model for solving the timetabling problem. The parallel model implemented is the *cellular evolutionary algorithm* (cEA) (Alba & Dorronsoro, 2005). The performance of a parallel cEA is compared with a sequential *Evolutionary Algorithm* (EA) on problem instances available from the ITC 2002 and ITC 2007 International Timetable Competitions. In the solution optimisation, only the hard constraints are considered. It is shown that cEA outperforms the sequential EA.

2.3.5 Memetic Algorithms

The class of *Memetic Algorithms* (MA) Moscato (1999); Neri et al. (2012) comprise a type of metaheuristics which combine population-based algorithms with other components in the form of heuristics, approximation algorithms, local search techniques, among other ways Moscato (1999). MA were first applied to the ETP in Burke et al. (1996). Alkan & Özcan (2003) apply MA to timetabling. Burke & Landa Silva (2005) present MA design guidelines for scheduling and timetabling problems. In Abdullah et al. (2010), a tabu-based memetic approach for examination timetabling problems is presented, where the Toronto benchmark problem is solved. Fonseca & Santos (2013) apply MA to the High School Timetabling Problem.

Recent memetic approaches such as the *shuffled frog-leaping algorithm* (SFLA) Eusuff, Lansey & Pasha (2006) were applied to the ETP as in Wang, Pan & Ji (2009) (in chinese). The authors propose a Discrete SFLA (DSFLA) where solutions are encoded using a time permutation scheme suited to be manipulated by the DSFLA. As the original

SFLA is only suitable for continuous optimisation problems, a specific update operator was defined for the discrete case of ETP. The algorithm manipulates both feasible and infeasible solutions, being these last ones penalised to avoid further exploring them. The DSFLA is evaluated on four datasets of the capacitated Toronto benchmark set (Toronto variant *c* in (Qu et al., 2009)).

2.3.6 Multi-Objective Approaches

Côté et al. (2004) investigate a bi-objective evolutionary algorithm with the objectives of minimising timetable length and spacing out conflicting exams. Two local search operators (instead of recombination operators) are used to deal with hard and soft constraints. Cheong et al. (2009) also investigate a bi-objective EA with the same objectives than those used in Côté et al. (2004). A *day crossover operator* is introduced which exchanges parent chromosome *days* (represented by three timeslots) to generate the offspring. They apply their approach to the capacitated Toronto instances and the Nottingham benchmark.

2.3.7 Approaches for the ITC 2007 Benchmark Set

In the following, we review some of the approaches proposed for the ITC 2007 benchmark instances. The five finalists in this competition were:

- 1st Place - Tomáš Müller.
- 2nd Place - Christos Gogos.
- 3rd Place - Mitsunori Atsuta, Koji Nonobe, and Toshihide Ibaraki.
- 4th Place - Geoffrey De Smet.
- 5th Place - Nelishia Pillay.

Müller (2009) solved all three tracks of the ITC 2007 (see Section 2.4.2). A hybrid, two-phase, algorithm was proposed. In the first phase, the *iterative forward search* (IFS) algorithm is used to obtain feasible solutions, coupled with *conflict-based statistics* (CBS) to prevent IFS from looping. The second phase applies multiple metaheuristics, namely, HC, GD and (optionally) SA. Müller's solution generation procedure has some similarities with a previously proposed method, the *"Squeaky Wheel" optimisation* (SWO) (Joslin & Clements, 1999). In SWO, a greedy algorithm is used to construct a solution which is then analysed to find the trouble spots, i.e., those elements, that, if improved, are likely to improve the objective function score. The results of the analysis are used to generate new priorities that determine the order in which the greedy algorithm constructs the next solution.

Gogos, Alefragis & Housos (2008) also applied a two-phase algorithm. After a preprocessing stage, a feasible solution is constructed with the *greedy randomized adaptive search procedure* (GRASP) metaheuristic. In the second phase, several optimisation methods are applied in this order: HC, SA, *Integer Programming* (IP) (using the *Branch & Bound* procedure); it ends with the so-called *Shaking Stage*, where the current solution is restarted/reheated and given to the SA again. In a later work, Gogos et al. (2012) present an improved version of their algorithm. The authors claim that the improved behaviour can be attributed to the more sophisticated process flow, early detection of plateaus, added heuristics and optimised data structures that make possible the exploration of a much larger number of Kempe Chain moves.

The approach of Atsuta et al. McCollum, Schaerf, Paechter, McMullan, Lewis, Parkes, Gaspero, Qu & Burke (2010) consists of applying a constraint satisfaction problem solver adopting a hybridisation of *tabu search* (TS) and *iterated local search* (ILS).

De Smet's approach McCollum et al. (2010) employ a rule engine software (coined Drool's rule engine) for obtaining a feasible solution. Then, the TS metaheuristic is used to improve the solution.

Pillay McCollum et al. (2010) proposes a two-phase algorithm variant, using a *Developmental Approach based on Cell Biology*, whose goal is to form a well-developed organism by creating a cell and then proceeding with cell division, cell interaction, and cell migration. In this approach, each cell represents a time slot. The first phase includes the creation of the first cell, cell division and cell interaction. The second phase realizes the cell migration process.

McCollum, McMullan, Parkes, Burke & Abdullah (2009) apply an adaptive ordering heuristic for constructing solutions, followed by the application of an extended version of the *great deluge* (GD) metaheuristic.

Demeester et al. (2012) employ a *hyper-heuristic* (HH) based approach to solve three timetabling problems: the Toronto and ITC 2007 problems, as well as the KAHO Sint-Lieven (Ghent, Belgium) timetabling problem. They apply a construction and improvement approach. The construction algorithm used for the ITC 2007 benchmark set does not guarantee the generation of feasible solutions. In the case that no feasible solution is found, the algorithm still goes on with the improvement phase, and extra correcting actions are carried out in order to remove any violation of hard constraints.

In Alzaqebah & Abdullah (2014), the authors propose an adaptive artificial bee colony algorithm (Karaboga, 2005) combined with a late-acceptance hill-climbing algorithm for examination timetabling. The proposed method is applied to the Toronto and ITC 2007 benchmark sets. In a recent study (Alzaqebah & Abdullah, 2015), the same authors report a hybrid bee colony optimisation for solving examination timetabling problems.

In a recent work, (Battistutta, Schaerf & Urli, 2017) propose a single-stage procedure based on the SA approach for the ITC 2007 examination timetabling problem. In the proposed method, non-feasible solutions are included in the search space and are dealt with appropriate penalties. A statistically-principled experimental analysis is performed in order to understand the effect of parameter selection. Then, a feature-based parameter tuning is carried out. The authors conclude that their properly tuned SA approach is able to compete with state-of-the-art solvers.

2.3.8 Previous Surveys on Examination Timetabling

The examination timetabling problem is studied by the scientific community since the 1960s. The surveys from Carter & Laporte (1996a) and Schaerf (1999) provide a review of the early approaches to solve the ETP. Some surveys about educational timetabling were published recently. Qu et al. (2009) present a detailed survey of algorithmic strategies applied to the ETP. Kristiansen & Stidsen (2013), Johnes (2015), and Teoh, Wibowo & Ngadiman (2015) survey the application of operations research and metaheuristic approaches to academic scheduling problems. Pillay (2016) presents a review of hyper-heuristics for educational timetabling.

2.4 Public Benchmark Sets

In the following sections, the Toronto and ITC 2007 benchmark sets are described.

2.4.1 Uncapacitated Toronto Data Set

The uncapacitated ETP formulation provided here was adapted from Abdullah, Turabieh & McCollum (2009) and Burke, Bykov, Newall & Petrovic (2004). The following terms are defined:

- $E = \{e_1, \dots, e_n\}$ is the set of exams.
- $P = \{p_1, \dots, p_k\}$ is the set of timeslots (or periods).
- $C = (c_{ij})_{|E|\times|E|}$ (called the *conflict matrix*), is a symmetric matrix of size $|E|$ where each element, denoted by c_{ij} $(i, j \in \{1, \dots, |E|\})$, represents the number of students attending both exams i and j.

- M, is the total number of students.
- t_k ($t_k \in P$) denotes the assigned timeslot for exam e_k ($e_k \in E$).

This ETP has one soft constraint which forbids a student to take two exams in neighbour periods (periods that are one to five timeslots apart). This soft constraint is represented by the function f_c, defined in (2.3). The problem formulation is:

$$\text{minimise} \quad f_c = \frac{1}{M} \cdot \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} c_{ij} \cdot \text{prox}(i, j) \tag{2.3}$$

where

$$\text{prox}(i, j) = \begin{cases} 2^{5-|t_i - t_j|}, \; if \;\; 1 \le |t_i - t_j| \le 5 \\ 0, \text{otherwise} \end{cases} \tag{2.4}$$

subject to

$$\sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} c_{ij} \cdot \delta_{t_i, t_j} = 0, \quad \delta_{t_i, t_j} = \begin{cases} 0, \; t_i \ne t_j \\ 1, \; t_i = t_j \end{cases} . \tag{2.5}$$

Equation (2.4) gives the penalty incurred by scheduling exams e_i and e_j in timeslots t_i and t_j, respectively. The penalty weighting factor is 16, 8, 4, 2, and 1, for exams that are, one, two, three, four, and five timeslots apart, respectively. For exams that are more than five timeslots apart, the weighting factor is set to zero. Equation (2.5) represents the single hard constraint, requiring that there can be no conflicts between exams scheduled in the same time slot.

In this thesis, the Toronto benchmark set, Version I (Table 2.1), is used. This public data set is available at: `ftp://ftp.mie.utoronto.ca/pub/carter/testprob`.

2.4.2 The ITC 2007 Data Set

A recent benchmark set The Second International Timetabling Competition (ITC 2007) (2007) was also used in the experiments conducted in this thesis. The ITC 2007 set consists of three tracks, each one related to a type of educational timetabling problem:

- Track 1 – examination timetabling;
- Track 2 – post enrolment based course timetabling;
- Track 3 – curriculum based course timetabling.

The examination timetabling track is made of 12 instances with different characteristics, and contains various constraint types, similar to those encountered in practice. The

Table 2.1: Specifications of the uncapacitated Toronto benchmark set (version I). The conflict matrix density is the ratio between the number of non-zero elements in the conflict matrix and the total number of elements.

	# exams	# students	# enrolments	conflict matrix density	# time slots
car91	682	16 925	56 877	0.13	35
car92	543	18 419	55 522	0.14	32
ear83	190	1125	8109	0.27	24
hec92	81	2823	10 632	0.42	18
kfu93	461	5349	25 113	0.06	20
lse91	381	2726	10 918	0.06	18
pur93	2419	30 029	120 681	0.03	42
rye93	486	11 483	45 051	0.08	23
sta83	139	611	5751	0.14	13
tre92	261	4360	14 901	0.18	23
uta92	622	21 266	58 979	0.13	35
ute92	184	2749	11 793	0.08	10
yor83	181	941	6034	0.29	21

characteristics of the ITC 2007 instances are described in Table 2.2. The examination timetabling problem introduced in the ITC 2007 Competition extends the problem defined in the Toronto benchmark set by considering additional types of new hard and soft constraints.

Table 2.2: Specifications of the 12 instances of the ITC 2007 benchmark set.

	# students	# exams	# rooms	conflict matrix density	# time slots
Instance 1	7891	607	7	0.05	54
Instance 2	12 743	870	49	0.01	40
Instance 3	16 439	934	48	0.03	36
Instance 4	5045	273	1	0.15	21
Instance 5	9253	1018	3	0.009	42
Instance 6	7909	242	8	0.06	16
Instance 7	14 676	1096	15	0.02	80
Instance 8	7718	598	8	0.05	80
Instance 9	655	169	3	0.08	25
Instance 10	1577	214	48	0.05	32
Instance 11	16 439	934	40	0.03	26
Instance 12	1653	78	50	0.18	12

The resulting extended set of hard constraints are as follows (McCollum et al., 2012):

- *No Conflicts* – Conflicting exams cannot be scheduled in the same period.
- *Room Occupancy* – For each room and period, do not allocate more seats than those available in that room.
- *Period Utilisation* – There is no exam, assigned to a given period, that requires more time than what is available for that period.
- *Period Related* – A set of time-ordering requirements between pairs of exams must be fulfilled. In particular, for any pair (e_1, e_2) of exams, the following constraints are specified:
  - *After Constraint* – e_1 must take place strictly after e_2,
  - *Exam Coincidence* – e_1 must take place at the same time than e_2,
  - *Period Exclusion* – e_1 must not take place at the same time than e_2.
- *Room Related* – The following room requirement was specified:
  - *Room Exclusive* – Exam e_1 must be held in an exclusive room.

Additionally, all exams must be scheduled (complete solution), and the exams cannot be split between periods or rooms. A feasible solution satisfies all the hard constraints.

The problem also includes the following soft constraints:

- *Two Exams in a Row* – Its aim is to minimise the number of occurrences of two exams being scheduled consecutively for any given student.
- *Two Exams in a Day* – Its aim is to minimise the number of occurrences of two exams being scheduled the same day (while not being directly adjacent) for any given student.
- *Period Spread* – This soft constraint requires to spread the set of exams taken by each student over a fixed number of time slots.
- *Mixed Durations* – A penalty is incurred whenever there exist exams in the same room and period with different durations.
- *Front Load* – This soft constraint forces exams with the largest number of students to be carried out at the beginning of the examination session.

- *Room and Period Penalties* – This penalty for some rooms or periods is aimed at minimising their use in the timetable.

A complete description of the ITC 2007 ETP, including a mathematical formulation of the related optimisation problem, is available in McCollum et al. (2012).

Chapter 3

A New Benchmark Timetabling Problem from Practice

Contents

3.1 Introduction

The actual programme curricula seen at universities are designed to offer a great degree of diversity and flexibility to students, letting them choose a considerable number of free or optional courses. In order to make this possible with the available teachers, university employees and resources (rooms, equipment, among others), courses are being offered in multiple related programmes (e.g., Computer Science, Electrical Engineering, and Multimedia Engineering programmes). This growing number of combined courses imposes extra difficulties in solving the *Examination Timetabling Problem* (ETP). For instance, the first semester's Algebra course is usually offered simultaneously to a given set of programmes, resulting in a more constrained problem. Moreover, real problems found in practice have several examination periods or epochs, e.g. normal exam epoch, second exam epoch, and special exam epoch (taken by working students, or in other special cases). Until now, examination timetabling benchmarks introduced in the literature have only addressed a single examination epoch.

In this chapter, the two-epoch ETP is introduced and a real-world problem instance, named ISEL–DEETC data set, is described. Section 3.2 describes the ISEL–DEETC data set. In the remaining sections, three formulations of the ISEL–DEETC benchmark set are introduced. These different formulations are used in the approaches described in subsequent chapters. In Sections 3.3.2 and 3.3.3, the single and two-epoch timetabling problems are specified, respectively, both considering a single objective and uncapacitated problem. In Section 3.4, the ISEL–DEETC problem is modelled as a multi-objective and capacitated problem.

3.2 The ISEL–DEETC Data Set

The ISEL–DEETC examination timetabling problem is a real-world problem instance comprising two examination epochs, with different number of time slots allocated for each epoch. This problem has emerged in the Electrical, Telecommunications and Computer Engineering Department (DEETC) at the Instituto Superior de Engenharia de Lisboa (ISEL) – Instituto Politécnico de Lisboa. The problem instance described is the ISEL–DEETC timetable of the 2009/2010 academic year, winter semester (Leite et al., 2014,1).

The ISEL–DEETC timetable comprises five programmes: three B.Sc. programmes (named LEETC, LEIC and LERCM) and two M.Sc. programmes (named MEIC and MEET). The B.Sc. and M.Sc. programmes have six and four semesters duration, respectively. The ISEL–DEETC data set characteristics are listed in Table 3.1.

The number of exams per programme is listed in Table 3.2. To get an idea of the number of students involved in each semester, we present in Table 3.3 the number of classes proposed for the winter semester for each programme. For the B.Sc. programmes, each class of 1st and 2nd semesters has, on average, 35 students and the remaining semesters have approximately 22 students per class, on average. For the M.Sc. programmes, each class has on average 12 students.

Table 3.4 presents the courses shared among the five programmes offered in DEETC. The number of shared courses is 34 (out of 80 courses that have exams). The first five columns of Table 3.4 contain the semesters where a given course is offered. Semesters in M.Sc. programmes are numbered 7 to 10 (four semester master programme).

Table 3.1: Characteristics of the ISEL–DEETC data set.

Parameter	Value	
Number of exams		80
Number of students		1238
Number of enrolments		4548
Conflict matrix density	1st epoch	0.32
	2nd epoch	0.31
Number of time slots	1st epoch	18
	2nd epoch	12

Table 3.2: Number of exams per programme in DEETC.

LEETC	LEIC	LERCM	MEIC	MEET
32	30	29	19	25

Table 3.3: Number of classes proposed for the winter semester for each programme.

Semester	LEIC	LEETC	LERCM	MEIC	MEET
1st	5	5	3	2	2
2nd	3	3	1	–	–
3rd	3	3	2	2	2
4th	2	2	1	–	–
5th	3	3	1		
6th	–	–	–		
Total:	16	16	8	4	4

Table 3.4: Courses shared among the five programmes offered in the DEETC.

MEIC	MEET	LERCM	LEIC	LEETC	Course	Acronym
		1	1	1	Linear Algebra	ALGA
		1		1	Mathematical Analysis I	AM1
		1	1	1	Programming	Pg
			1	2	Logic and Digital Systems	LSD
		2		2	Mathematical Analysis II	AM2
		2	2	2	Object Oriented Prog.	POO
		2	2	3	Probability and Statistics	PE
			2	3	Computer Architecture	ACp
			3 and 5		Computer Graphics	CG
			3 and 5		Computation and Logic	LC
			3 and 5		Functional Programming	PF
		3	3	4	Imperative Prog. in C/C++	PICC/CPg
	7	3		5	Digital Comm. Syst.	SCDig
		4	4	4	Computer Networks	RCp
	7		4	5	Virtual Execution Systems	AVE
	8	4			Multimedia Signal Codific.	CSM
		4		5	Operating Systems	SOt
7		5			Unsupervised Learning	AA
	8	5			Database Systems	BD
	8		5	6	Internet Programming	PI
	8	5		6	Distributed Comput. Syst.	SCDist
7	7	5	5	5	Internet Networks	RI
7			5		Compilers	Cpl
	7			5	Control	Ctrl
	7			5	Radio Communications	RCom
7			5		Security Informatics	SI
	7			5	Telecommunication Systems	ST
7	7		5	5	Embedded Systems I	SE1
7	7	6			Multim. Comm. Networks	RSCM
7			6		Distributed Systems	SD
7		6			Software Engineering	ES
7 to 9	8	6	3 to 6	6	Project Management	EGP
7 to 9	8	6	3 to 6	6	Enterprise Management	OGE
7 to 9	8	6	3 to 6	6	Management Systems	SG

The ISEL–DEETC is a relatively complex problem, due mainly to two reasons: 1) high degree of course sharing in different programmes and different semesters (e.g. the LSD course is offered in the 1st and 2nd semesters of LEIC and LEETC programmes, respectively, see Table 3.4); 2) the courses of the even semesters (taken in the summer semester) are also being lectured in the winter semester, thus increasing the timetable complexity, because there are students attending courses in the even and odd semesters. The manual solution of the ISEL–DEETC timetable took one week to be constructed from scratch by a two-person team.

3.2.1 Room specification

The list of rooms used in the DEETC department is listed in Table 3.5. The room designation has the following meaning: *<Building>.<Floor number>.<Room number>*. The largest exam, ALGA, has 489 students enrolled.

The room assignment was made by an human planner.

Table 3.5: Rooms designation and capacity.

Designation	**Capacity**	**Designation**	**Capacity**
A.2.03	50	G.0.14	30
A.2.08–A.2.09	40+40	G.0.15	30
A.2.10–A.2.11	40+40	G.0.16	50
A.2.12–A.2.13	40+40	G.0.24	81
A.2.16–A.2.18	45+45	G.1.03	50
C.2.14	47	G.1.04	45
C.2.21	16	G.1.13	45
C.2.22	47	G.1.15	79
C.2.23	48	G.1.18	40
C.3.07	75	G.2.06	50
C.3.14	36	G.2.07	50
C.3.15	40	G.2.08	50
C.3.16	40	G.2.09	50
G.0.08	30	G.2.10	45
G.0.13	30	G.2.21	48
	Sum of rooms seating capacity = 1532		

3.3 Single-objective Problem Formulation

In this section, we describe the single and two epoch examination timetabling problems. In single-epoch problems, there exists a single examination epoch comprised of a fixed number of time slots. The Toronto and ITC 2007 problems belong to this category. The two-epoch problem is an extension of the single-epoch problem where two examination epochs of different lengths are considered. The ISEL–DEETC is a two-epoch problem. However, in this work we also model the ISEL–DEETC as a single-epoch problem, by considering just the first examination session.

3.3.1 Constraints

Table 3.6 describes the hard and soft constraints of the ISEL–DEETC problem.

Table 3.6: Hard and Soft constraints of the ISEL–DEETC ETP.

Constraint and Type	Explanation
H_1 (Hard)	There cannot exist students sitting for more than one exam simultaneously.
H_2 (Hard)	A minimum distance between two course's exams must be observed.
H_3 (Hard)	The total number of students in the examination room should not exceed the room capacity.
S_1 (Soft)	Exams should be spread out evenly through the timetable.

Constraints H_1 and S_1 are typical constraints found in uncapacitated problems (e.g., the Toronto specification). H_2 is a new constraint introduced in the ISEL–DEETC's two-epoch formulation. This constraint requires that a minimum number of time slots between the first and second exams of a given course exists. This exam spacing out is required to allow that enough time exists for studying between examination sessions, and for exam correction and proofing. The hard constraint H_3 applies to the capacitated problem. In this work, the ISEL–DEETC is solved both as an uncapacitated and as a capacitated problem.

3.3.2 The Single-Epoch Problem

The problem formulation presented in this section was adapted from the one proposed in Côté et al. (2004). The new formulation is equivalent to the earlier formulation introduced in Section 2.4.1 for the Toronto benchmark set, but is more suited than the previous one for the two-epoch extension elaborated in this section. The formulation is as follows. Given a set of exams $\mathcal{E} = \{e_1, e_2, \ldots, e_{|\mathcal{E}|}\}$ and a set of time slots $\mathcal{T} = \{1, 2, \ldots, |\mathcal{T}|\}$, the optimisation goal is to find the optimal timetable represented by the set h of ordered pairs (t, e) where $t \in \mathcal{T}$ and $e \in \mathcal{E}$. The obtained timetable is called *feasible* if it satisfies all hard constraints. Otherwise, the timetable is said to be *unfeasible*.

The following additional symbols are defined:

- $C = (c_{ij})_{|\mathcal{E}|\times|\mathcal{E}|}$ (*Conflict matrix*), is a symmetric matrix of size $|\mathcal{E}|$ where each element, denoted by c_{ij} $(i, j \in \{1, \ldots, |\mathcal{E}|\})$, represents the number of students attending exams e_i and e_j. The diagonal elements c_{ii} denote the total of students enrolled in exam e_i;

- N_s, is the total number of students;

- t_k $(t_k \in \mathcal{T})$ denotes the assigned time slot for exam e_k $(e_k \in \mathcal{E})$.

In practical timetabling one should allow students to have some free time between exams. In the *Exam Proximity Problem* (EPP) (Côté et al., 2004), the objective is to find a feasible timetable while minimising the number of students having to take consecutive exams. Equation (3.1) is a variant of the EPP model where q is the number of timeslots per day, $N \geq 0$ is the number of free time slots between exams and K is a constant representing the maximum timetable length.

$$\begin{aligned} &\text{minimise} \quad u_2 = \frac{1}{2} \sum_{k=1}^{|\mathcal{T}|-(N+1)} \sum_{i=1}^{|\mathcal{E}|} \sum_{j=1}^{|\mathcal{E}|} c_{ij}\varepsilon_{ik}\varepsilon_{jk+(N+1)}, \forall k \text{ where } k \bmod q \neq 0, \\ &\text{subject to} \quad \sum_{k=1}^{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{E}|} \sum_{j=1}^{|\mathcal{E}|} c_{ij}\varepsilon_{ik}\varepsilon_{jk} = 0, \quad |\mathcal{T}| \leq K. \end{aligned} \tag{3.1}$$

The above model represents an *Uncapacitated Exam Proximity problem* (UEPP) because the classroom seating capacity is not considered. The model allows the differentiation between consecutive exam periods within the same day, versus overnight. For example, setting $q = 3$ and $K = 30$ corresponds to a ten day examination session with three time slots per day. In (3.1), the overnight gap is accounted for and no penalty exists between conflicting exams scheduled in the last time slot of a day and in the first time slot of the next day, respectively.

A common proximity metric has also been defined for benchmark sets (e.g., Toronto data set), and it is also used in the ISEL–DEETC benchmark set. This proximity metric is a weighted version of (3.1) with $0 < N \leq 4$ (counting the number of students having 0–4 free time slots between exams). The single-epoch uncapacitated optimisation problem using this proximity metric is expressed using (3.1) as follows:

$$\text{minimise} \quad f = \frac{1}{N_s} \sum_{x=0}^{4} w_{x+1}\, u_2|_{q=|\mathcal{T}|,N=x} \tag{3.2}$$

where

$$u_2|_{q=|\mathcal{T}|,N=x} = \frac{1}{2}\sum_{k=1}^{|\mathcal{T}|-(x+1)}\sum_{i=1}^{|\mathcal{E}|}\sum_{j=1}^{|\mathcal{E}|} c_{ij}\varepsilon_{ik}\varepsilon_{jk+(x+1)}, \tag{3.3}$$

$\forall k$ where $k \bmod |\mathcal{T}| \neq 0$,

subject to
$$\sum_{k=1}^{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{E}|}\sum_{j=1}^{|\mathcal{E}|} c_{ij}\varepsilon_{ik}\varepsilon_{jk} = 0, \tag{3.4}$$

$$|\mathcal{T}| \leq K. \tag{3.5}$$

Equation (3.2) represents the standard soft constraint S_1 Carter, Laporte & Lee (1996) (see Table 3.6). Function f computes the average proximity cost per student as the sum of the number of students having 0–4 free time slots between exams, divided by the total number of students. Conflicting exams that are closer to each other are more penalised. The weighting factors are $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, and $w_5 = 1$ (see Section 2.4.1).

In (3.3), $u_2|_{q=|\mathcal{T}|,N=x}$ means computing the number of students having $N = x$ free timeslots between exams. The value of $q = |\mathcal{T}|$ means that the timeslots are numbered contiguously with no overnight gap. $\varepsilon_{jk} \in \{0, 1\}$ is a binary quantity with $\varepsilon_{jk} = 1$ if exam e_j is assigned to time slot k. Otherwise, $\varepsilon_{jk} = 0$.

Equation (3.4) represents the hard constraint H_1 (Table 3.6).

3.3.3 The Two-Epoch Problem

In the two-epoch problem formulation, the following terms were added to the single-epoch problem formulation:

- set of time slots of the second examination epoch, $\mathcal{L} = \{|\mathcal{T}|+1, |\mathcal{T}|+2, \ldots, |\mathcal{T}|+|\mathcal{L}|\}$;
- $C_{relax} = (c_{relax_{ij}})_{|\mathcal{E}|\times|\mathcal{E}|}$ (*Relaxed conflict matrix*), is a relaxed version of the conflict matrix C. This matrix is used in the generation of the second examination epoch. Further details are given below;
- v_k ($v_k \in \mathcal{L}$) denotes the assigned time slot for exam e_k ($e_k \in \mathcal{E}$) in the second examination epoch.

As mentioned earlier in this chapter, a new hard constraint is needed in the two-epoch formulation in order to guarantee that a minimum number of time slots between the first and second exams of a given course exists. For this purpose, the hard constraint H_2 (Table 3.6) was defined. H_2 can be specified by:

$$v_k - t_k \geq L_{min}, \tag{3.6}$$

where L_{min} is the minimum time slot distance between the first and second epoch exams of a given course. v_k and t_k denote the assigned time slots for exam e_k in the second and in the first examination epochs, respectively. For example, if the *Algebra* course ($e_k =$ Algebra) first epoch's exam is in time slot 3 (t_k = 3) and the second epoch's exam is in time slot 19 (v_k = 19), and the minimum distance is $L_{min} = 10$, then H_2 is satisfied since $19 - 3 \geq 10$; on the other hand, if t_k = 17 for the same v_k value, then H_2 is not satisfied since $19 - 17 < 10$. Examples of two-epoch timetables can be seen in Chapter 5, in Tables 5.10 and 5.11.

Some details concerning the conflict matrix (C_{relax}) used in the generation of the second epoch timetable are now given. In the ISEL–DEETC data set, the second epoch is more constrained than the first epoch because it has fewer time slots (the second epoch has 12 time slots, whereas the first epoch has 18 time slots). In order to be possible to generate feasible initial solutions for the second epoch, some entries (with very few students enrolled) were set to zero, resulting in a relaxed conflict matrix. It is to be noted that, in general, this pre-processing is only performed if needed, depending on the data set characteristics.

3.3.4 Optimisation Problem

We now describe the two-epoch optimisation problem and the related algorithmic steps. In short, the devised algorithm for solving the two-epoch case divides the problem into two related single-epoch problems. Then, it proceeds to generate the second epoch timetable and ends with the generation of the first epoch timetable. In detail, the algorithm executes the following steps:

1. Solve the two-epoch problem by dividing it into two single-epoch problems. Solve the second epoch problem independently by considering the single-epoch formulation presented in Section 3.3.2, but using the relaxed conflict matrix, C_{relax}.

2. Then, solve the first epoch problem using a variation (denoted as *two-epoch exam proximity problem*) of the formulation given in Section 3.3.2, in order to be able to spread out the first epoch exams from conflicting exams in the second epoch. In addition, constraint H_2 is also included in the extended formulation. It is noted that

the generation of the first epoch timetable depends on the timetable of the second epoch. Thus, these two sub-timetables are passed as a parameter to the optimisation algorithm.

3. At the end of the optimisation step, join the two timetables forming the two-epoch timetable that satisfies both constraints H_1 and H_2 (uncapacitated problem). The cost of the two-epoch timetable is the sum of the costs of the individual timetables.

The Two-Epoch Exam Proximity Problem

The UEPP model presented earlier in equations (3.1) and (3.2)–(3.5) is now extended in order to relate the first epoch exams with the second epoch exams. The UEPP model shown in (3.1) is augmented with a third parameter, M:

$$\begin{aligned}
\text{minimise} \quad & u_{2_{extended}} = \frac{1}{2} \sum_{k=1}^{|\mathcal{T}|-(N+1)+M} \sum_{i=1}^{|\mathcal{E}|} \sum_{j=1}^{|\mathcal{E}|} c_{ij} \varepsilon_{ik} \varepsilon_{jk+(N+1)}, \\
& \forall k \text{ where } k \bmod q \neq 0, \\
\text{subject to} \quad & \sum_{k=1}^{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{E}|} \sum_{j=1}^{|\mathcal{E}|} c_{ij} \varepsilon_{ik} \varepsilon_{jk} = 0, \quad |\mathcal{T}| \leq K. && (3.7)
\end{aligned}$$

Using (3.7), the extended UEPP is as follows:

$$\text{minimise} \quad f_{epoch1} = \frac{1}{N_s} \sum_{x=0}^{4} w_{x+1}\, u_{2_{extended}}\big|_{q=|\mathcal{T}|+1, N=x, M=x+1} \tag{3.8}$$

$$\text{where} \quad u_{2_{extended}}\big|_{q=|\mathcal{T}|+1, N=x, M=x+1} = \frac{1}{2} \sum_{k=1}^{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{E}|} \sum_{j=1}^{|\mathcal{E}|} c_{ij} \varepsilon_{ik} \varepsilon_{jk+(x+1)}, \tag{3.9}$$

$$\forall k \text{ where } k \bmod |\mathcal{T}| + 1 \neq 0,$$

$$\text{subject to} \quad \sum_{k=1}^{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{E}|} \sum_{j=1}^{|\mathcal{E}|} c_{ij} \varepsilon_{ik} \varepsilon_{jk} = 0, \tag{3.10}$$

$$|\mathcal{T}| \leq K, \tag{3.11}$$

$$\sum_{k=1}^{|\mathcal{E}|} \sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^{|\mathcal{L}|} \lambda_{ki} \cdot \sigma_{kj} \cdot \text{dist}(j-i, L_{min}) = |\mathcal{E}|, \tag{3.12}$$

$$\text{where} \quad \text{dist}(l, m) = \begin{cases} 1, & l \geq m \\ 0, & l < m \end{cases} \tag{3.13}$$

Equation (3.8) represents the extended proximity cost function, that computes the average number of conflicts per student of the first epoch timetable taking into account the exams already scheduled in the second epoch. Parameter M added to $u_{2_{extended}}$ in (3.9) is used to modify the first sum, allowing for the comparison between first and second epoch exams.

Equations (3.12) and (3.13) specify the H_2 constraint. $\lambda_{ki} \in \{0, 1\}$ is a binary quantity with $\lambda_{ki} = 1$ if exam e_k is assigned to time slot i in the *first epoch*. Otherwise, $\lambda_{ki} = 0$. Similarly, $\sigma_{kj} \in \{0, 1\}$ is a binary quantity with $\sigma_{kj} = 1$ if exam e_k is assigned to time slot j in the *second epoch*. Otherwise, $\sigma_{kj} = 0$.

Figure 3.1 illustrates how the proximity cost function represented by (3.8) is computed.

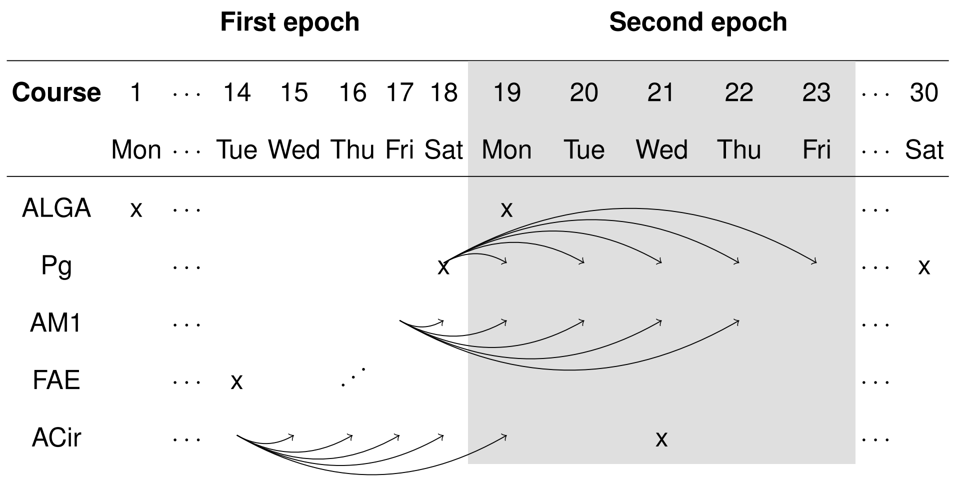

First epoch
Second epoch
Course 1 ··· 14 15 16 17 18 19 20 21 22 23 ··· 30
Mon ··· Tue Wed Thu Fri Sat Mon Tue Wed Thu Fri ··· Sat
ALGA x ··· x ···
Pg ··· x ··· x
AM1 ··· ···
FAE ··· x ···
ACir ··· x ···

Figure 3.1: Exam proximity cost computation for the two-epoch problem. When computing the first epoch proximity cost, an extended version of the chromosome is considered, having more five periods from the second epoch, marked in greyscale. The goal of using this extended model is to spread away first epoch exams that conflict with exams in the second epoch in a neighbourhood of zero to four time slots away.

3.4 Multi-objective Problem Formulation

In this section, the ETP is modelled as a multi-objective problem and considering a single examination epoch. We consider an instance of the ETP that was first formulated in Burke et al. (1996). In their formulation, if a student is scheduled to take two exams in any one day there should be a free period between the two exams. Violation of this constraint is referred to as a *clash*.

The ISEL–DEETC is solved in Chapter 7 using a multi-objective evolutionary algorithm, considering both capacitated and uncapacitated problems. The capacitated multi-objective ETP formulation used in this thesis was first formulated by Cheong et al. (2009), and is as follows:

$$\text{minimise} \quad f_1 = \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} \sum_{p=1}^{|P|-1} a_{ip}\, a_{j(p+1)}\, c_{ij} \tag{3.14}$$

$$f_2 = |P| \tag{3.15}$$

$$\text{subject to} \quad \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} \sum_{p=1}^{|P|} a_{ip}\, a_{jp}\, c_{ij} = 0, \tag{3.16}$$

$$\sum_{i=1}^{|E|} a_{ip}\, s_i \le S, \quad \forall p \in P, \tag{3.17}$$

$$\sum_{p=1}^{|P|} a_{ip} = 1, \quad \forall i \in \{1, \ldots |E|\}, \tag{3.18}$$

where:

- $E = \{e_1, e_2, \ldots, e_{|E|}\}$ is the set of exams to be scheduled,
- $P = \{1, 2, \ldots, |P|\}$ is the set of periods,
- S is the total seating capacity in a given period,
- a_{ip} is one if exam e_i is allocated to period p, and is zero otherwise,
- c_{ij} is the number of students registered for exams e_i and e_j. Matrix c is termed the *Conflict matrix*,
- s_i is the number of students registered for exam e_i.

Equations (3.14) and (3.15) are the two objectives of minimising the number of clashes and timetable length, respectively. The constraint specified by (3.16) is the hard constraint that no student is to be scheduled to take two exams in the same period. Equation (3.17) states the capacity constraint that the total number of students sitting in the same room and in the same timeslot, for all exams scheduled at that timeslot, must be less than or equal to the total seating capacity S. In (3.18), a constraint is specified indicating that every exam can only be scheduled once in any timetable.

In the uncapacitated problem variant, the constraint specified by (3.17) is not considered.

Chapter 4

Local Search Algorithms for Examination Timetabling

Contents

This chapter describes the local search approaches developed for solving the *Examination Timetabling Problem* (ETP). Section 4.1 presents two approaches, based on the standard *simulated annealing* (SA) Kirkpatrick et al. (1983) and *threshold acceptance* (TA) Dueck & Scheuer (1990) metaheuristics, developed for the ETP. The application of the proposed algorithms is shown for the public Toronto and ITC 2007 benchmark sets. The impact of the local search cooling schedule on the optimisation is analysed in Section 4.2. An accelerated version of the SA, named *fast simulated annealing* (FastSA), is proposed in Section 4.3. Section 4.4 reports the experimental results on the ITC 2007 benchmark set and compare the FastSA's results with the ones in the literature. Finally, concluding remarks are provided in Section 4.5.

4.1 Local Search Approaches

This section describes the proposed approaches based on the SA metaheuristic. In Section 4.1.1, the standard SA is presented, whereas in Section 4.1.2 a description of the standard TA is provided. Sections 4.1.3 and 4.1.4 describe the algorithm components for the Toronto benchmark (uncapacitated problem) and ITC 2007 benchmark (capacitated problem) sets.

4.1.1 SA Algorithm

The SA metaheuristic is detailed next. Let f be the objective function to be minimised on a set X of feasible solutions, and denote by $N(s)$ the neighbourhood for each solution s in X. Let L be the state-space graph induced by X and by the definition of $N(s)$. The SA is an iterative algorithm that starts from an initial solution and attempts to reach an optimal solution by moving step by step in L. In each step, a neighbour solution s' of the current solution s is generated; if the move improves the cost function then the algorithm moves s to the neighbour solution s'; otherwise, the solution s' is selected with a probability that depends on the current temperature and the amount of degradation of the objective function (Talbi, 2009). The temperature is reduced gradually according to a defined cooling schedule. Algorithm 2 describes the template of the SA algorithm (Kirkpatrick et al., 1983).

The temperature T is updated by simulating the exponentially-decreasing temperature over time, as used in the metal annealing process. This is achieved by the function $T(t)$:

$$T(t) = T_{max} \cdot \exp(-r \cdot t), \tag{4.1}$$

Algorithm 2 Template of the *simulated annealing* algorithm.

```
Input: Cooling schedule.
1:  $s \leftarrow s_0$                                  ▷ Generation of the initial solution
2:  $T \leftarrow T_{max}$                              ▷ Starting temperature
3:  repeat
4:      repeat                                          ▷ At a fixed temperature
5:          Generate a random neighbour $s' \in N(s)$
6:          $\Delta E \leftarrow f(s') - f(s)$
7:          if $\Delta E \leq 0$ then $s \leftarrow s'$     ▷ Accept the neighbour solution
8:          else Accept $s'$ with a probability $e^{\frac{-\Delta E}{T \cdot f(s)}}$
9:          end if
10:     until Equilibrium condition                     ▷ e.g., a given number of iterations executed at each temperature $T$
11:     $T \leftarrow g(T)$                             ▷ Temperature update
12: until Stopping criteria satisfied                   ▷ e.g., $T < T_{min}$
13: Output: Best solution found.
```

where r is the temperature decreasing rate and T_{max} is the initial temperature (T_{max} should have a large value as compared to r).

4.1.2 TA Algorithm

The TA is similar to the SA metaheuristic but, instead of using a probabilistic acceptance criterion, a deterministic one is used. After generating the neighbour solution, s', of the current solution s, the TA moves to the neighbour solution s', even if $f(s') > f(s)$, as long as $f(s') - f(s)$ is less than or equal to the current threshold Q. The threshold is reduced gradually in a similar way as in the SA, according to a predefined cooling schedule. The steps of the TA metaheuristic are described in Algorithm 3.

The threshold Q is updated using (4.1) but replacing T_{max} with Q_{max} (the initial threshold).

4.1.3 Application to the Toronto Benchmark Set

Solution Representation and Fitness Function

The adopted solution representation for the Toronto benchmark set is illustrated in Figure 4.1. Each solution is represented by an array of periods where each period contains the list of the scheduled exams in that period. For example, the exams allocated in time slot t_1 are e_2, e_{14}, e_{10}, e_3, and e_{16}, whereas the exams allocated in time slot t_2 are e_1, e_{11},

Algorithm 3 Template of the *threshold acceptance* algorithm.

```
Input: Threshold annealing $Q_{max}$
 1: $s \leftarrow s_0$                                   ▷ Generation of the initial solution
 2: $Q \leftarrow Q_{max}$                               ▷ Starting threshold
 3: repeat
 4:     repeat                                           ▷ At a fixed threshold
 5:         Generate a random neighbour $s' \in N(s)$
 6:         $E \leftarrow f(s') - f(s)$
 7:         if $E \leq Q$ then $s \leftarrow s'$
 8:         end if                                       ▷ Accept the neighbour solution
 9:     until Equilibrium condition                      ▷ e.g., a given number of itera-
                                                           tions executed at each thresh-
                                                           old $Q$
10:     $Q \leftarrow g(Q)$                              ▷ Threshold update
11: until Stopping criteria satisfied                    ▷ e.g., $Q \leq Q_{min}$
12: Output: Best solution found.
```

t_1	t_2	$\cdots$	t_T
e_2	e_1		e_8
e_{14}	e_{11}		e_{12}
e_{10}	e_4	$\cdots$	e_{15}
e_3			e_{17}
e_{16}			

Figure 4.1: Solution representation for the Toronto benchmark set (uncapacitated problem).

and e_4 (see Figure 4.1). The exams allocated to a given period must satisfy the hard clash constraint.

In terms of software implementation, the solution is encoded as an array, say V, of column vectors, having the period index as the first index (column) and the exam index as the second index (row). V_{ji} is for period j and exam i and is 1 if exam i is assigned to that period, otherwise it is 0. This implementation provides for nuclear operations (exam allocation to period, exam's period change) to be performed in constant time. Also, there is no need to manage the exam list size in each period as the exam list is always of fixed size (a column vector). This comes at the expense of more memory being used for the exams indices.

The fitness function used is given by Equation (2.3) in Section 2.4.1.

Construction of Initial Feasible Timetables

The initial solution for the Toronto instance is constructed using the *saturation degree* (SD) graph colouring heuristic (Brélaz, 1979). Initially, all exams are eligible for insertion in the timetable. An exam is randomly selected and assigned to a random period. The remaining exams are sorted in non-decreasing order of the number of available periods. Then, one of the exams with the lowest number of available periods is removed from the exams' queue and assigned to a random period. The process is repeated until all exams are inserted into the timetable. If, at a given time, a feasible time slot cannot be found for a candidate exam, the process fails to succeed and no attempt is done to repair the timetable. The process is simply repeated from the beginning until a feasible solution is found, fulfilling the timetable fixed length constraint. According to the tests conducted, less than five cycles (on average) are needed to find a feasible solution. As a consequence, no further investigation on a repair mechanism was carried out.

Neighbourhood Operator

The Kempe chain neighbourhood was used for the Toronto benchmark set, as described in Section 2.3.3. In the implemented operator, the solution is evaluated in an incremental way, in order to improve the operator's performance.

4.1.4 Application to the ITC 2007 Benchmark Set

Solution Representation and Fitness Function

	t_1	t_2	$\cdots$	t_T
r_1	e_3	e_6		e_9, e_{22}
r_2		e_2, e_5		e_{19}, e_{10}
.	.	.		.
.	.	.	$\cdots$	.
.	.	.		.
r_{R-1}	.	.		.
r_R		e_4		e_7

Figure 4.2: Solution representation for the ITC 2007 benchmark set (capacitated problem).

Since the ITC benchmark set is capacitated, we have to consider examination rooms in the solution. The generic representation of a solution in this case is given in Figure 4.2.

A given solution encodes a complete and feasible timetable. In the figure, for example, exam e_3 is allocated in time slot t_1 and room r_1, and exams e_2 and e_5 are allocated in time slot t_2 and room r_2. Time slots may have different lengths, in order to be possible to schedule examinations with different durations.

In terms of software implementation, the solution was implemented differently than what is shown in Figure 4.2. The solution is encoded as a matrix, say A, having in the row index the exam index, and in the column index the period number. A_{ij} is for exam i and period j and contains the room index if it is assigned to that period, otherwise it is -1. This implementation provides for nuclear operations (exam allocation to period and room, room change, period change) to be performed in constant time.

As mentioned in Section 2.4.2, the fitness function is the same as the one used in McCollum et al. (2012).

Construction of Initial Feasible Timetables

For the ITC 2007 case, the initial solution is constructed using a variant of the SD graph colouring heuristic. In our approach, for performance reasons, the number of available periods for each exam is computed by considering only the *No Conflicts* (or clash) hard constraint, instead of considering all hard constraints. The remaining hard constraints are satisfied when the exam is scheduled.

In order to schedule the more constrained exams first, we initially sort the exams by their number of *After* hard constraints. The exams that must take place before others (involved in an *After* constraint) have fewer available periods, and are thus more difficult to schedule. The other types of period related hard constraints (*Exam Coincidence* and *Period Exclusion*) are not considered in the initial exam sorting.

We start by setting the initial priority (number of available periods) of each exam equal to the total number of periods. Then, all the exams that must be scheduled before another exam have their number of feasible periods decremented by one. This is done for each *After* hard constraint in which they are involved. For instance, if an exam e_i must be scheduled *before* exam e_j (an *After* constraint exists between e_j and e_i), then the number of available periods for e_i is equal to the total number of periods minus one.

The construction algorithm proceeds in two phases:

Phase 1 Extract an exam from the exam priority queue and, if all hard constraints are met, schedule the exam in the selected period and room, and update the priorities (number of available periods) of exams in the queue, considering only the *No Conflicts* hard constraint. Then, repeat the same process for the next maximum priority exam in the queue. If a given exam cannot be scheduled, then go to Phase 2.

Phase 2 If a selected exam cannot be assigned due to violations of hard constraints, the conflicting exams are unscheduled and the selected exam is scheduled. The conflicting exams are again inserted in the exam priority queue. The priority of each exam in the queue is recomputed according to the SD heuristic. Then, go to Phase 1. The construction phase ends when all exams are scheduled.

In order to prevent repetitive assignments of variables (exams) to the same values (period and room), the *conflict-based statistics* (CBS) Müller (2009) data structure is used. CBS stores hard conflicts which have occurred during the search, together with their frequency, and the assignments that caused them. In the (period, room) value selection, the value which corresponds to the lowest sum over weighted hard conflicts is chosen. Each hard conflict is weighted by its frequency, i.e., by the number of times the conflicting variable was unassigned due to the selected assignment Müller (2009).

Neighbourhood Operators

In solving the ITC 2007 timetabling problem instance, two neighbourhood operators, based on the Kempe chain neighbourhood, were implemented:

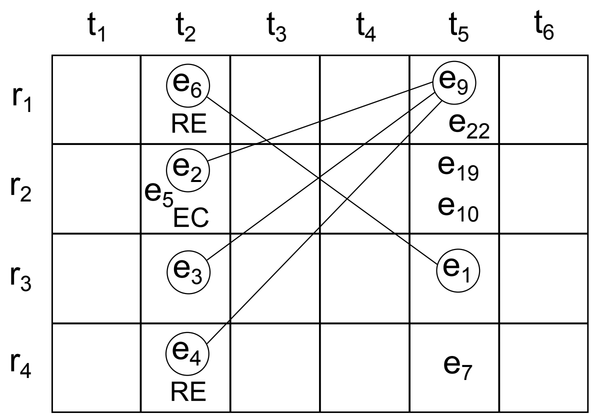
Slot-Room move – A random exam is scheduled in a different period and room, both chosen randomly.

Room move – An exam, chosen randomly, is scheduled in a different room for the same period. The destination room is chosen in a random fashion.

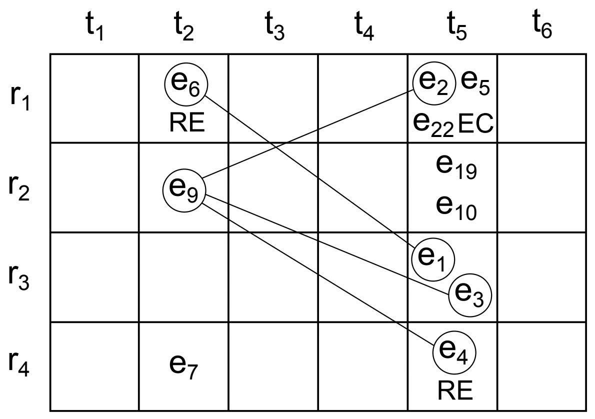
Figure 4.3 illustrates the use of the *Slot-Room* move operator on an example ITC 2007 solution. In this example, exam e_2 is to be moved from time slot t_2 and room r_2 to time slot t_5 and room r_1. The Kempe chain involving e_2 is given by $K_1 = \{e_2, e_3, e_4, e_5, e_7, e_9\}$ (Algorithm 1). Time slots t_i and t_j in Algorithm 1 are set to t_2 and t_5, respectively, with initial contents given by $t_2 = \{e_2, e_3, e_4, e_5, e_6\}$ and $t_5 = \{e_1, e_7, e_9, e_{10}, e_{19}, e_{22}\}$. Then, t_2 is updated with $(t_2 \setminus K_1) \cup (t_5 \cap K_1) = \{e_6, e_7, e_9\}$ and t_5 is updated with $(t_5 \setminus K_1) \cup (t_2 \cap K_1) = \{e_1, e_2, e_3, e_4, e_5, e_{10}, e_{19}, e_{22}\}$.

The *Room* and *Slot-Room* move operators may yield infeasible neighbour solutions for the ITC 2007 data set. If it is not possible to apply the move, then the move is set to be infeasible and ignored by the local search neighbourhood move operator. In this case, the solution is not evaluated and the local search move acceptance rule (lines 6–9 in Algorithm 2) is skipped. Hence, the generated infeasible neighbour is not accepted.

In the *Slot-Room* move, the *After* and *Period Utilisation* hard constraints are only tested after the move is completed. These hard constraints are checked for the exams



(a) before moving exam e_2 from time slot t_2 and room r_2 to time slot t_5 and room r_1



(b) after moving exam e_2

Figure 4.3: An example of the Kempe chain heuristic. There are two Kempe chains in the figure, with lines connecting conflicting exams. In the first Kempe chain, a move of exam e_2 from time slot t_2 and room r_2 to time slot t_5 and room r_1 requires repair moves to maintain feasibility. Exam e_9 has to be moved to time slot t_2 and room r_2, and exam e_5 (due to the *Exam Coincidence* (EC) constraint), e_3, and e_4, have to be moved to time slot t_5 and rooms r_1, r_3, and r_4, respectively. Consequently, exam e_7 has to be moved from time slot t_5 and room r_4 to time slot t_2 and room r_4 due to the *Room Exclusion* (RE) constraint.

scheduled in the affected time slots t_i and t_j, and the neighbour solution is set to be infeasible if they are violated. The presented Kempe chain operators also implemented solution's incremental evaluation.

A final remark about the implemented operators is given. Although the *Slot-Room* move also changes an exam's room, the *Room move* operator was included for two reasons. First, for completeness, i.e., it is desirable to have a set of operators that could move an exam to any place (period and room), and the *Slot-Room* move always changes the period and room. Hence, there is a need for an operator that only changes the exam room. The second reason was justified by the need to have an operator that could be used to minimise the room specific soft constraints (*Mixed Durations* and *Room Penalty*) violations.

4.2 Effect of the Local Search Cooling Schedule

In this section, we study the impact of the cooling schedule on the algorithm performance. Using a more intensive local search, one can achieve better results than using a light local search Dowsland & Thompson (2012). But some relevant questions can be asked: what is the effect of local search on the organisation of exams in the timetable? How are the exams moved as the temperature varies in SA based algorithms? The study undertaken here provides some answers to those questions. In addition, some important properties of the local search operator are identified. These properties served as the basis for the development of an accelerated version of the local search algorithm (described in Section 4.3).

The presented study involves both the Toronto and ITC 2007 benchmark sets and the SA and TA algorithms. The section's text is organised as follows. In Section 4.2.1 the parameter settings used in the experiments are described. Sections 4.2.2 and Section 4.2.3 present the studies made for the Toronto and ITC 2007 benchmark sets, respectively.

4.2.1 Parameter Settings

Table 4.1: Parameter settings. Legend: T_{max} – initial temperature/threshold, r – decreasing rate, k – # iterations at each temperature/threshold, and T_{min} – final temperature/threshold.

Data set	Algorithm	Cooling schedule					
		T_{max}	r	k	T_{min}	Type	# Evaluations
Toronto	TA	0.1	1×10^{-3}	5	2×10^{-5}	light	42 590
		0.1	1×10^{-6}	5	2×10^{-5}	intensive	42 585 970
ITC 2007	SA	0.1	1×10^{-4}	5	1×10^{-7}	light	690 780
		0.1	1×10^{-5}	5	1×10^{-7}	intensive	6 907 760

The parameter settings are specified in Table 4.1. The TA metaheuristic was used when solving the Toronto instances, whereas the SA was used in the ITC 2007 case. Two different cooling schedules were used in both algorithms: a *light* and an *intensive* cooling schedules.

4.2.2 Toronto Data Set

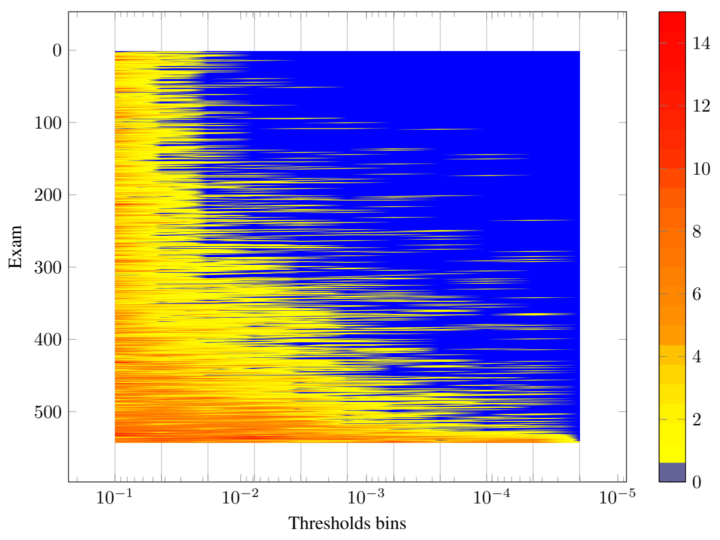
Figures 4.4, 4.5, and 4.6 illustrate the evolution of accepted exam movements (using the Kempe chain neighbourhood) as the TA threshold varies. In these figures, the x-axis value range is divided into ten *threshold bins*. A *threshold bin* corresponds to a threshold interval and is represented by vertical grid lines in the figures. The threshold bins were calculated in order to have 1/10 of the total number of evaluations in each bin. The colour codes for each bin, and for each exam on the y-axis, represent the number of accepted movements for the corresponding exam. The total number of exams' accepted moves per bin is generally lower than the number of evaluations performed in that bin, because not all exams are part of an accepted move in each local search move. Some exams do not move because the fitness difference violates the TA acceptance criterion, or are not selected by the neighbourhood operator. The exams are ordered decreasingly by the number of conflicts they have with other events (*largest degree* heuristic).

It can be observed that the more difficult exams (the ones with lower indexes) only move (i.e. the move is accepted by the TA) when higher thresholds are applied, being gradually fixed at their definitive place. As can be seen from the top and bottom plots of Figures 4.4 and 4.5, it is essential that the difficult exams be well placed, in an optimal or near optimal position, for the easier exams to be placed in an optimal or near optimal position. If a light cooling schedule is used, the algorithm cannot find the best places for the difficult exams and so the easier exams are also placed in a suboptimal way. Figure 4.6 provides a detailed view of the movement count of the top-100 high degree exams represented in Figure 4.5. In Figure 4.6, we can observe that the use of the intensive cooling (Figure 4.6(b)) allows the top-100 high degree exams to move more, as compared to the light cooling schedule (Figure 4.6(a)). This is supported by the numbers shown in the colour bar (right of the figure) which are much larger for the intensive cooling (Figure 4.6(b)) than for the light cooling schedule (Figure 4.6(a)).

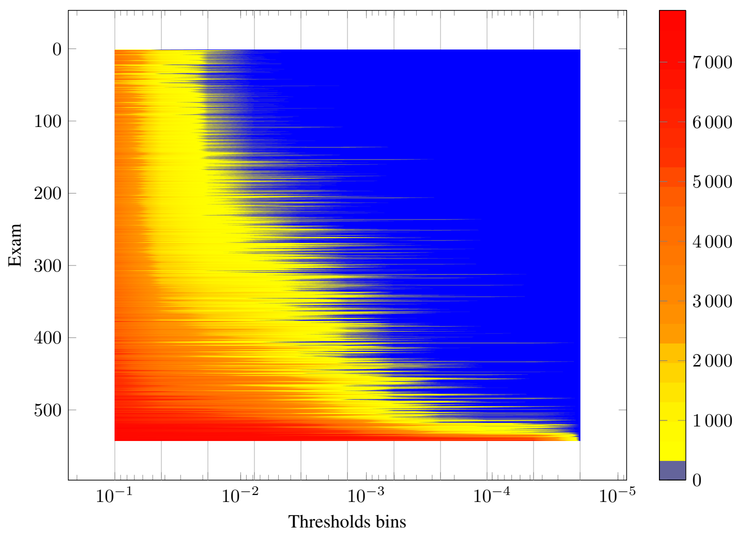
4.2.3 ITC 2007 Data Set

We now repeat the experiments made before but now considering the SA metaheuristic and the ITC 2007 data set. Figures 4.7 and 4.8 illustrate the evolution of the number of accepted exam movements as the SA temperature varies, for two example ITC 2007 instances.

We can observe the same behaviour as for the TA/Toronto configuration set: the use of the intensive cooling schedule yields better solutions as a greater number of evaluations is made.



(a) car92 instance optimised with the *light* cooling schedule. Obtained cost: 4.57

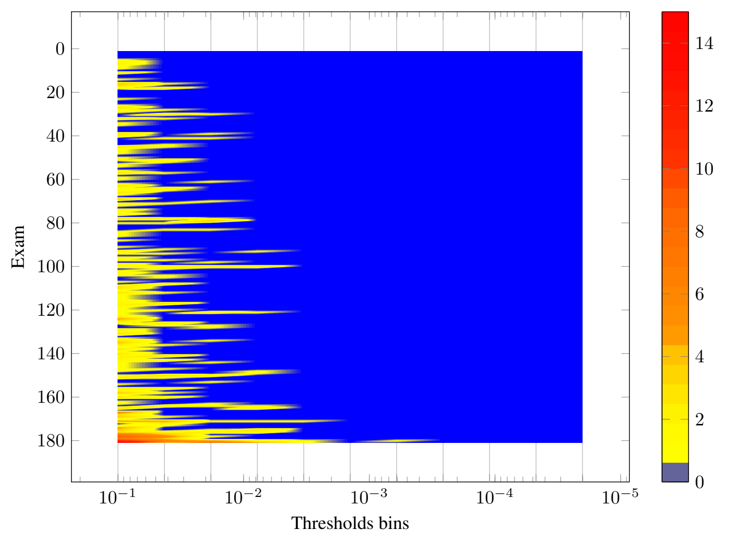


(b) car92 instance optimised with the *intensive* cooling schedule. Obtained cost: 3.77

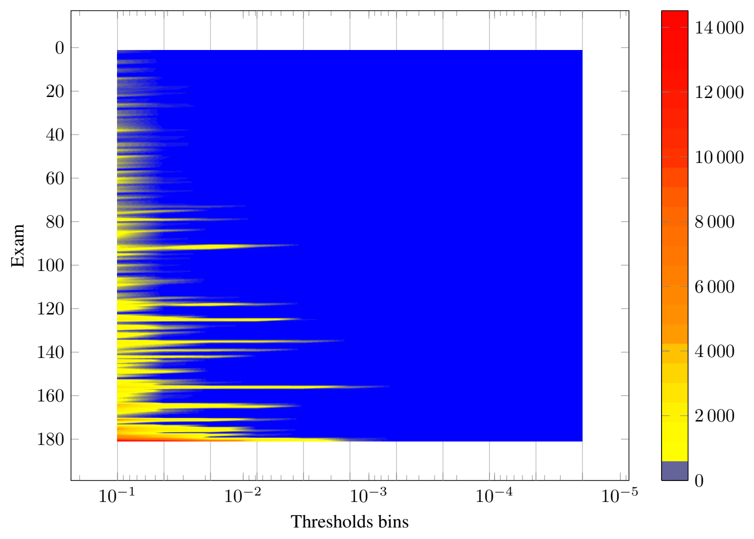
Figure 4.4: Study of the effect of applying *light* and *intensive* cooling schedules in the TA algorithm. Each figure above shows the evolution of the number of accepted exam moves, for each exam in the y-axis and for each threshold bin in the x-axis. The exams in the y-axis are ordered decreasingly by the number of conflicts they have with other events (*largest degree* heuristic). The x-axis value range is divided into ten threshold bins, each with an equal number of evaluations. The long ticks along the x-axis represent the bins' limits.

4.3 Accelerated SA for the ETP

In this section, the FastSA search method for solving the ETP is explained.



(a) yor83 instance optimised with the *light* cooling schedule. Obtained cost: 39.41

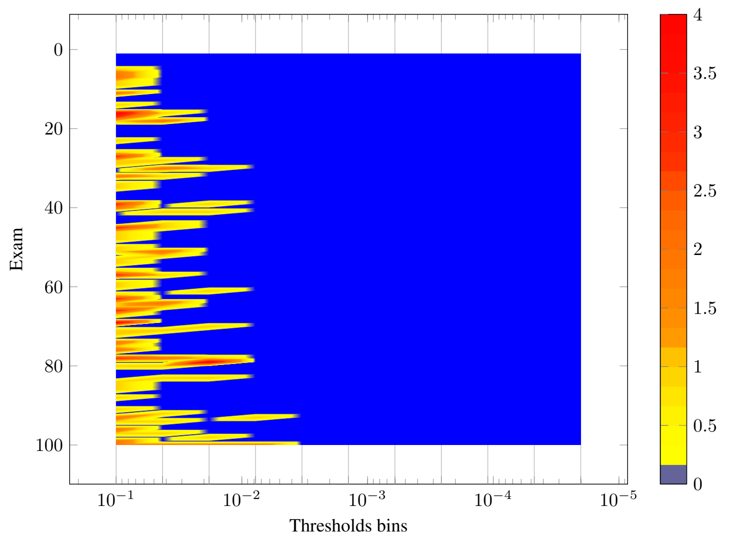


(b) yor83 instance optimised with the *intensive* cooling schedule. Obtained cost: 35.35

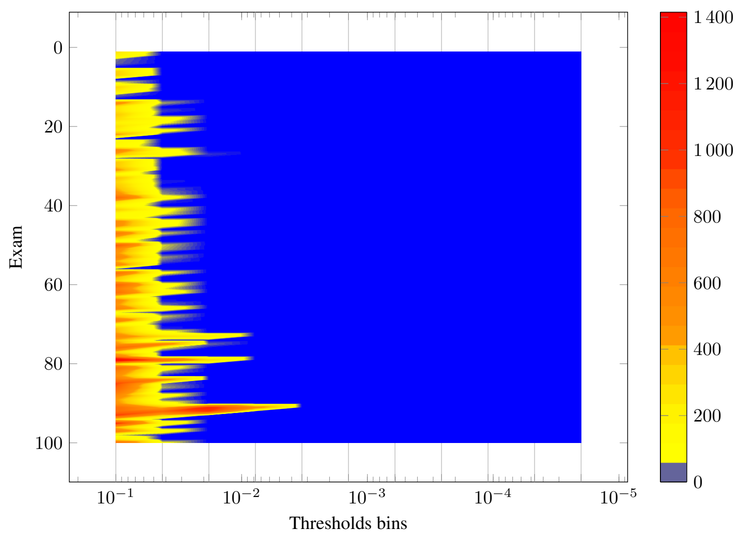
Figure 4.5: Study of the effect of applying *light* and *intensive* cooling schedules in the TA algorithm (continued).

4.3.1 FastSA Search Method for the ETP

As it can be observed from Figures 4.4–4.8, several of the more difficult exams are going to be fixed (or have few moves) as lower temperatures are reached. In the algorithm, if it were decided not to move (and not evaluate the corresponding movement) an exam having zero or few movements in the previous temperature bin, it would produce a faster SA algorithm. The faster execution comes at the expense of a degradation of the results, if there



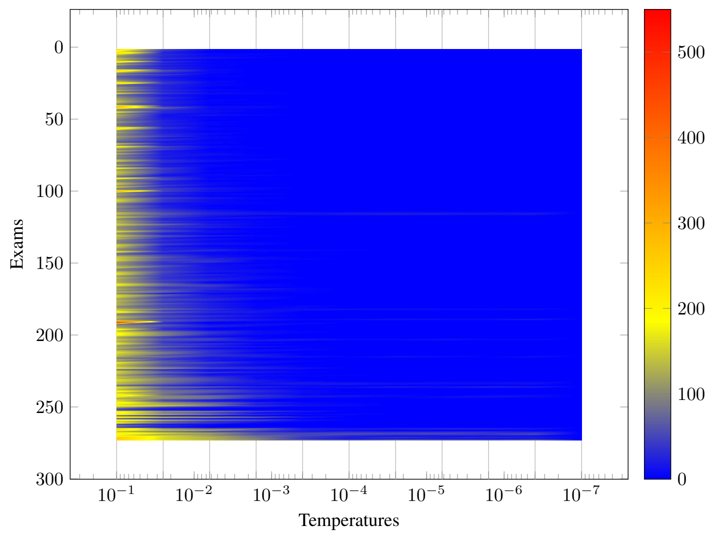
(a) yor83 instance optimised with the *light* cooling schedule. The number of accepted exams' movements for the 100 most difficult exams of Figure 4.5(a) is shown.



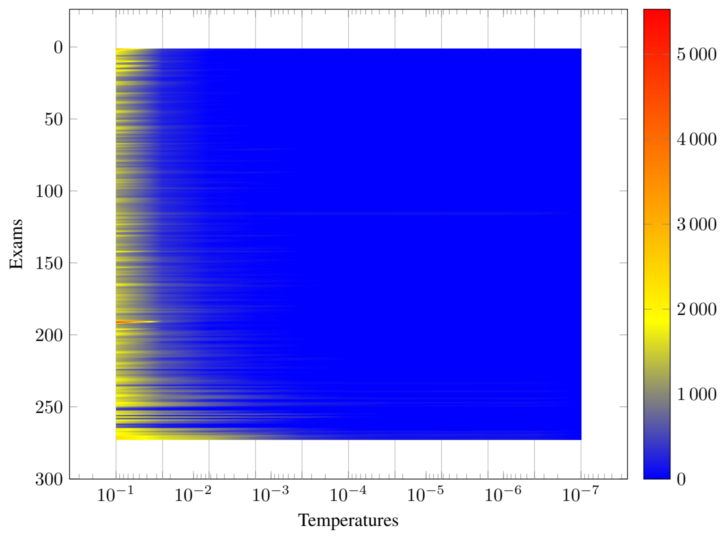
(b) yor83 instance optimised with the *intensive* cooling schedule. The number of accepted exams' movements for the 100 most difficult exams of Figure 4.5(b) is shown.

Figure 4.6: Study of the effect of applying *light* and *intensive* cooling schedules in the TA algorithm for the yor83 instance. The figures show the number of accepted exams' movements for the 100 most difficult exams of Figures 4.5(a) and 4.5(b), respectively.

exist bins with exam movements followed or interleaved with bins with no movements, because the algorithm stops evaluating moves for a given exam when a bin with zero moves is found. This faster SA variant was implemented in this work, and was named FastSA. The template of the FastSA algorithm is given in Algorithm 4.

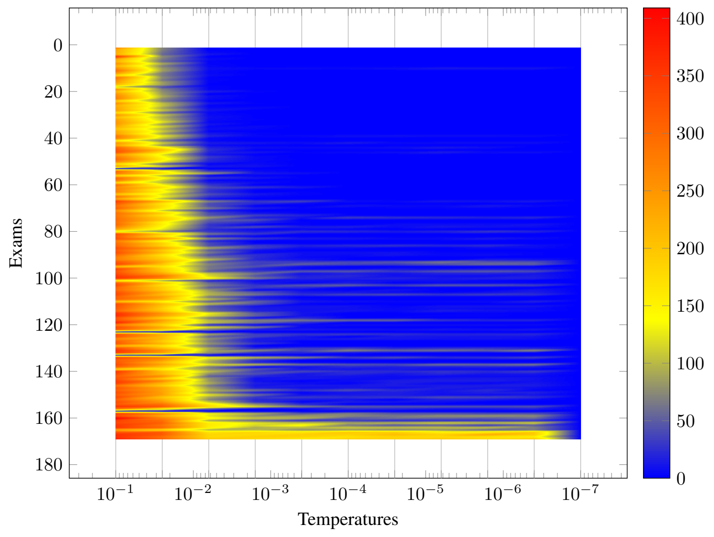


(a) dataset4 instance optimised with the *light* cooling schedule. Obtained cost: 14 058

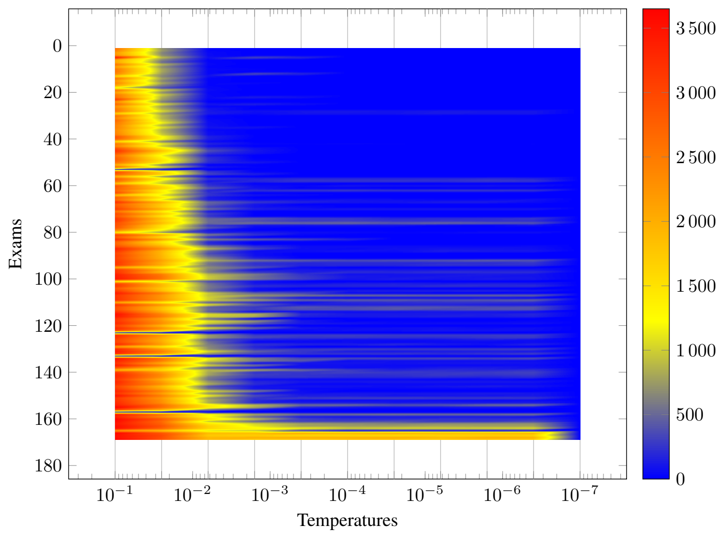


(b) dataset4 instance optimised with the *intensive* cooling schedule. Obtained cost: 12 528

Figure 4.7: Study of the effect of applying *light* and *intensive* cooling schedules in the SA algorithm, for ITC 2007 instances 4 and 9. Each figure shows the evolution of the number of accepted exam moves, for each exam in the y-axis, per each temperature bin. The exams in the y-axis are sorted by decreasing order of number of conflicts with other exams (*largest degree* heuristic). The marked x-axis long ticks represent the temperature intervals.



(a) dataset9 instance optimised with the *light* cooling schedule. Obtained cost: 1243



(b) dataset9 instance optimised with the *intensive* cooling schedule. Obtained cost: 1024

Figure 4.8: (continuation).

Algorithm 4 Template of the *fast simulated annealing* algorithm.

```
Input: Cooling schedule.
1: $prevBin \leftarrow$ nil    ▷ Array containing # accepted moves in the previous bin, initially nil
2: $currBin \leftarrow$ createArray(numExams)    ▷ Array containing # accepted moves in the current bin, initialised with zeros
3: $s \leftarrow s_0$    ▷ Generation of the initial solution
4: $T \leftarrow T_{max}$    ▷ Starting temperature
5: repeat
6:     repeat    ▷ At a fixed temperature
7:         Generate a random neighbour $s' \in N(s)$
8:         $examToMove \leftarrow$ selectExamFromSolution($s'$)
9:         if $prevBin =$ nil or ($prevBin <>$ nil and $prevBin[examToMove] > 0$) then    ▷ Evaluate neighbour solution
10:            $\Delta E \leftarrow f(s') - f(s)$
11:            if $\Delta E \leq 0$ then $s \leftarrow s'$    ▷ Accept the neighbour solution
12:            else Accept $s'$ with a probability $e^{\frac{-\Delta E}{T \cdot f(s)}}$
13:            end if
14:            if $s'$ was accepted then
15:                $currBin[examToMove] \leftarrow currBin[examToMove] + 1$
16:            end if
17:        end if
18:    until Equilibrium condition    ▷ e.g., a given number of iterations executed at each temperature $T$
19:    $T \leftarrow g(T)$    ▷ temperature update
20:    $prevBin \leftarrow$ updateBin($T, currBin$)    ▷ If $T$ is in the next bin, make $prevBin \leftarrow currBin$ and zero $currBin$
21: until Stopping criteria satisfied    ▷ e.g., $T < T_{min}$
22: Output: Best solution found.
```

The lines marked in bold are lines that were added to the original SA algorithm (Algorithm 2). The FastSA keeps a record of all successful neighbourhood movements taken in a given bin. When performing a neighbourhood movement, a random exam is selected, and the algorithm first checks if there were any movements, for the exam to be moved, in the previous bin. If there was no previous movement for the candidate exam, then the current movement is not performed and not evaluated. With this strategy, the FastSA could attain a reduced number of evaluations compared to the original SA. The cost degradation attained by the FastSA compared to the SA is not significant, as shown in the experimental evaluation carried out in Section 4.4.

Example 4.3.1 illustrates a simple example of the bin structure used by the FastSA.

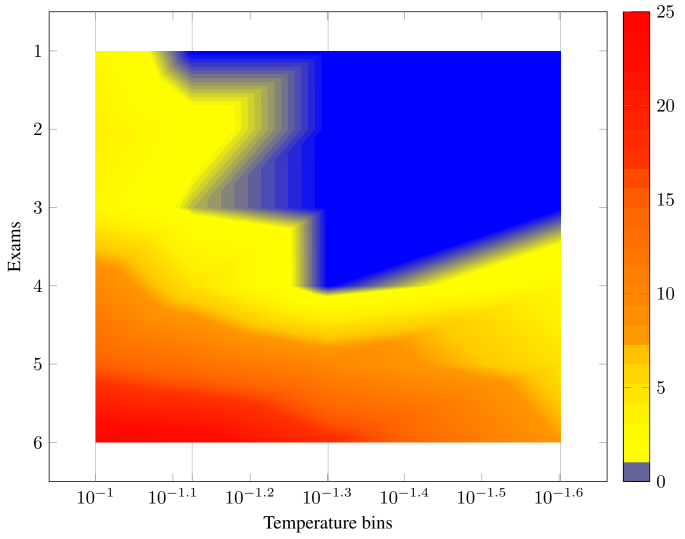
Example 4.3.1 *This example uses synthetic data to show the bin structure used by the*

FastSA. Six examinations and four bins were considered. Figure 4.9(a) illustrates the number of times each exam is selected in each bin. Figure 4.9(b) give a graphical representation of the same data using a colour map.

When the FastSA is run using this data, the following behaviour is registered. Because exam e_1 was never selected in bin b_2, e_1 remains fixed (a move involving this exam is not evaluated) from bin b_2 on, even if it had a positive number of selections in the bins b_3 and b_4. Using the same reasoning, exams e_2, e_3, and e_4, remain fixed from bin b_3 on. Note the case of exam e_4 which was selected in bin b_4 (marked in shaded grey). As e_4 has been fixed already (in the previous bin), this exam remains fixed until the end, and all further selections are not evaluated, yielding a faster algorithm.

	b_1	b_2	b_3	b_4
e_1	3	0	0	0
e_2	4	2	0	0
e_3	3	1	0	0
e_4	10	5	0	3
e_5	15	13	10	5
e_6	25	24	18	8

(a)



(b)

Figure 4.9: Different representations of the bin structure used by the FastSA. Six examinations and four bins were considered: (a) number of times each exam is selected in each bin, (b) graphical representation of the data in (a) using a colour map.

4.4 Experimental Results and Discussion

This section presents the experimental simulations conducted to test the proposed method in addressing the examination timetabling problem. Section 4.4.1 describes the parameter settings of the algorithm. Section 4.4.2 presents a comparison between three algorithms: two FastSA variants, named FastSA100 and FastSA80, and the standard SA. The FastSA100 variant is further compared with the state-of-the-art approaches in Section 4.4.3.

4.4.1 Settings

The developed algorithms were programmed in the C++ language using the ParadisEO framework (Talbi, 2009). The hardware and software specifications are: Intel Core i7-2630QM, CPU @ 2.00 GHz $\times$ 8, with 8 GB RAM; OS: Ubuntu 16.10, 64 bit; Compiler used: GCC v. 4.8.2. The algorithm execution time was set to 276 seconds, as measured by the provided benchmarking tool from the ITC 2007 site (The Second International Timetabling Competition (ITC 2007), 2007). Table 4.2 summarises the cooling schedule parameters used by the tested algorithms. The parameter values were chosen using as reference the FastSA100 algorithm (described in the next section). Specifically, the cooling schedule values were adapted empirically for each dataset instance, while guaranteeing that the FastSA100 execution time was within the ITC 2007 time limit constraint. The neighbourhood operators, the *Room* and *Slot-Room* moves, were selected with equal probability. To obtain our simulation results, each algorithm was run ten times on each instance with different random seeds. In the experiments, all the statistical tests were performed with a 95% confidence level. The developed source code, the resulting solution files for each instance/run, and the produced statistics, are publicly available on the following Git repository: `https://github.com/nunocsleite/FastSA-ETP-ITC2007`.

Table 4.2: Cooling schedules used to solve the different ITC 2007 instances. The value presented in the bottom row, rightmost column, corresponds to the sum of the # evaluations. The cooling schedule values were adapted empirically for each dataset instance using as reference the FastSA100 algorithm.

Inst.	T_{max}	$rate$	k	T_{min}	# evaluations
1	0.01	0.000 003	5	1.00×10^{-6}	15 350 571
2	0.01	0.000 004	5	1.00×10^{-6}	11 512 931
3	0.01	0.000 005	4	1.00×10^{-6}	7 368 277
4	0.1	0.000 005	6	1.00×10^{-6}	13 815 517
5	0.01	0.000 007	6	1.00×10^{-6}	7 894 579
6	0.002	0.000 003	5	1.00×10^{-6}	12 668 176
7	0.001	0.000 003	5	1.00×10^{-6}	11 512 931
8	0.001	0.000 001 5	5	1.00×10^{-6}	23 025 856
9	0.01	0.000 001	5	1.00×10^{-6}	46 051 706
10	0.01	0.000 001	5	5.00×10^{-6}	38 004 516
11	0.01	0.000 009	5	1.00×10^{-6}	5 116 861
12	0.01	0.000 001 3	5	1.00×10^{-6}	35 424 391
				$\sum$	227 746 312

4.4.2 Comparison between SA and FastSA Approaches

In this section, three algorithms, described as follows, were compared:

1. SA – The original SA metaheuristic as described in Algorithm 2.

2. FastSA100 – The basic FastSA algorithm as described in Algorithm 4. In this approach, the exam movements in each bin are registered, and if, for a selected exam, the number of movements in the previous bin is zero, that exam is *fixed* until the end of the optimisation phase. This means that the current and future movements of this exam are not considered and thus not evaluated.

3. FastSA80 – In this variant, the behaviour is similar to the FastSA100 but the selected exam is fixed only if two conditions hold: the number of movements in the previous bin is zero *and the exam belongs to the set comprising 80% of the exams with the highest degree*. Observing again Figures 4.7 and 4.8, we can see that the highest degree exams are those having the lower indices (top indices in the y-axis); on the other side, the lowest degree exams are the ones having the higher indices (the bottom ones in the y-axis). For the mentioned set of largest degree exams, the behaviour is the same as described for the FastSA100; for the remaining 20% of the exams, the method behaves like the original SA, that is, the selected exam is never fixed and its move is always evaluated. The reasoning behind this variant is the following: because the lowest degree exams have a low number of conflicts with other exams, their moves imply a small change in the objective function; this, in turn, imply that they are likely to move more often than the higher degree exams, because small changes in the objective function are often accepted by the SA acceptance criterion, even at low temperatures. Hence, in the scenario that a lowest degree exam has no accepted moves in a previous bin, it is likely that it will have accepted moves in the future. With the FastSA80, the moves of the set comprising 20% of the exams with the smallest degree are always evaluated, despite having some bins with zero counts.

Tables 4.3, 4.4, and 4.5, show, respectively, the obtained costs, the corresponding number of evaluations done, and the corresponding execution times in seconds for the FastSA100, FastSA80, and SA algorithms, for the twelve instances of the ITC 2007 benchmark set. For each algorithm the minimum and average values, and the standard deviation of ten runs are shown.

Table 4.3: Solution costs obtained on the ITC 2007 benchmark set by the FastSA100, FastSA80, and SA algorithms. For the Fast SA variants, the percentage of evaluations not done (marked in column '% less evals') is shown.

Inst-	FastSA100				FastSA80				SA		
ance	% less evals	f_{min}	f_{avg}	σ	% less evals	f_{min}	f_{avg}	σ	f_{min}	f_{avg}	σ
1	**34**	4872	5054.7	111.3	30	4916	5026.9	73.9	**4855**	**4953.9**	100.5
2	**1**	395	408.5	7.1	**1**	400	413.0	10.3	**395**	**407.5**	8.3
3	**5**	**9607**	**9945.6**	258.6	**5**	9749	10 141.7	239.6	9737	10 079.9	277.3
4	**41**	12 076	12 825.8	407.7	38	**12 073**	**12 667.2**	420.0	12 268	12 783.5	407.8
5	**6**	3058	**3378.2**	194.6	**6**	**3052**	3431.6	345.7	3188	3666.1	512.1
6	**14**	**25 515**	**25 960.5**	213.8	12	25 790	26 166.0	292.1	25 870	26 139.5	180.1
7	**7**	4291	4533.9	120.6	**7**	4306	4446.8	104.5	**4182**	**4426.6**	159.3
8	**25**	**7226**	7532.2	168.8	24	7471	7569.9	97.6	7287	**7515.7**	147.4
9	28	983	1025.8	33.1	**29**	**970**	1027.8	33.3	991	**1017.3**	23.8
10	**8**	13 400	13 662.3	214.5	**8**	13 412	13 656.9	145.2	**13 351**	**13 511.7**	104.0
11	**9**	30 090	**31 520.8**	1027.9	**9**	29 860	31 886.8	1375.8	**29 800**	31 930.9	1156.8
12	19	5137	5200.6	46.8	**20**	5143	5191.4	56.2	**5109**	**5179.1**	51.8
Avg.:	**17**				16						

Table 4.4: Number of evaluations performed by the proposed algorithms. The acronym ‘LE’ stands for ‘Less evaluations’. The column ‘% LE’ reports the percentage of the number of evaluations not performed, on average, by the respective FastSA compared to the SA, and computed as $(1 - (FastSA_{e_{avg}}/SA_{\#eval})) \cdot 100$, rounded to the nearest integer. The row marked with ‘# LE:’ reports the difference between the SA and respective FastSA f_{avg} # evaluations’ sums.

Inst-	FastSA100				FastSA80				SA
ance	% LE	e_{min}	e_{avg}	σ	% LE	e_{min}	e_{avg}	σ	# eval.
1	**34**	**9 980 873**	**10 198 009.6**	178 424.4	30	10 384 549	10 701 045.7	171 364.4	15 350 571
2	**1**	11 340 700	11 395 324.5	38 536.8	**1**	**11 318 777**	**11 377 261.4**	45 800.7	11 512 931
3	**5**	**6 866 754**	**6 969 666.3**	58 551.8	**5**	6 911 407	7 003 992.1	38 556.1	7 368 277
4	**41**	**7 857 028**	**8 148 037.6**	195 561.7	38	8 302 759	8 543 151.7	177 125.8	13 815 517
5	**6**	**7 237 700**	**7 409 559.1**	91 615.2	**6**	7 292 084	7 422 571.9	87 851.4	7 894 579
6	**14**	**10 424 248**	**10 883 293.4**	296 014.2	12	10 865 907	11 179 400.3	293 235.7	12 668 176
7	**7**	**10 599 677**	**10 657 882.6**	52 611.8	**7**	10 607 738	10 678 176.9	42 537.6	11 512 931
8	**25**	**16 879 540**	**17 275 934.8**	303 030.0	24	17 227 232	17 573 194.9	251 960.2	23 025 856
9	28	**31 718 395**	33 005 694.9	1 003 862.4	**29**	31 737 041	**32 872 225.8**	1 123 363.1	46 051 706
10	**8**	**34 223 507**	**34 915 395.0**	353 838.2	**8**	34 449 437	34 960 059.1	389 610.7	38 004 516
11	**9**	4 606 347	4 649 939.4	31 193.4	**9**	**4 581 774**	**4 648 839.5**	43 663.4	5 116 861
12	19	26 917 968	28 700 008.5	908 071.6	**20**	**26 504 157**	**28 276 194.8**	1 104 050.8	35 424 391
$\sum$		**178 652 737**	**184 208 745.7**			180 182 862	185 236 114.1		227 746 312
# LE:			43 537 566.3				42 510 197.9		
Avg.:	**17**				16				

Table 4.5: Execution times (in seconds) on the ITC 2007 benchmark set for the FastSA100, FastSA80, and SA algorithms.

Inst-	FastSA100			FastSA80			SA		
ance	t_{min}	t_{avg}	σ	t_{min}	t_{avg}	σ	t_{min}	t_{avg}	σ
1	**224**	**234.70**	6.80	232	236.30	2.58	369	377.20	6.11
2	227	233.70	4.32	**225**	**229.70**	3.53	249	258.00	8.03
3	193	197.90	4.33	**190**	**197.60**	5.23	206	219.95	9.06
4	**236**	**244.90**	4.36	245	255.70	5.60	428	442.60	8.85
5	**198**	211.00	7.13	204	**209.00**	2.62	236	257.70	14.84
6	**172**	192.41	10.82	174	**188.90**	9.00	217	226.20	5.47
7	228	234.40	3.92	**217**	**229.18**	8.33	260	277.80	15.13
8	227	234.80	4.71	**221**	**232.10**	6.08	305	314.26	10.04
9	**217**	238.90	11.64	226	**236.70**	10.04	327	342.50	11.65
10	**232**	242.70	5.60	234	**240.40**	3.92	253	258.20	2.53
11	208	**209.90**	1.97	**204**	210.40	5.21	231	238.80	5.51
12	**210**	220.70	6.06	214	**219.20**	6.09	269	274.20	4.24

Discussion

Analysing the results in terms of solution cost (Table 4.3), it can be observed that the FastSA100 has results that compete with the SA, obtaining the best average fitness in four of twelve instances, while requiring less evaluations. In terms of the performed number of evaluations (Table 4.4), the FastSA100 and FastSA80 execute, respectively, 17% and 16% less evaluations, on average, compared to the SA algorithm.

For the FastSA100, on more than half of the datasets, the number of neighbours not evaluated varies from 10% to 41% while not degrading the fitness value significantly, which is a significant number. From Table 4.5 results, we can conclude that the presented FastSA approaches are globally faster than the SA approach.

Concerning the execution times, we can observe that a lower average execution time on a given instance time does not necessary mean a lower average number of evaluations. If we compare, for example, instance 6 solved by FastSA100 and FastSA80 (Tables 4.3, 4.4, and 4.5), we observe that the FastSA100 performs 14% less evaluations, whereas FastSA80 performs 12% less evaluations. Despite executing a lower number of evaluations, the FastSA100 obtains a better cost and spends more time. This behaviour is justified by the several factors influencing the execution of the algorithm, namely: the stochastic nature of the construction algorithm and of the FastSA algorithm itself, and the existence of two move operators (room and slot-room moves), with different execution times. The slot-room move is more complex than the room operator thus taking more time to execute. It should be noted that the times of the SA algorithm are not within the time limit imposed by ITC 2007. Using as criteria the average percentage of the number of evaluations not performed, and also the execution time within the ITC 2007 rules, the FastSA100 is the best method. In the next section, the FastSA100 is compared with the state-of-the-art approaches.

4.4.3 Comparison with State-of-the-art Approaches

In this section we perform a comparison between the FastSA100 approach and other state-of-the-art methodologies. In the following tables, the compared approaches are identified by an acronym of the first author's name and year of the publication.

The compared authors are: Gog08 – Gogos et al. (2008) (Gogos et al., 2008), Ats08 – Atsuta et al. (2008) (McCollum et al., 2010), Sme08 – De Smet (2008) (McCollum et al., 2010), Pil08 – Pillay (2008) (McCollum et al., 2010), Mul09 – Müller (2009) (Müller, 2009), Col09 – McCollum et al. (2009) (McCollum et al., 2009), Dem12 – Demeester et al. (2012) (Demeester et al., 2012), Gog12 – Gogos et al. (2012) (Gogos et al., 2012), Byk13 – Bykov and Petrovic (2013) (Bykov & Petrovic, 2013), Ham13 – Hamilton-Bryce et al. (2013) (Hamilton-Bryce, McMullan & McCollum, 2013), Alz14 – Alzaqebah and Abdullah (2014) (Alzaqebah & Abdullah, 2014), Alz15 – Alzaqebah and Abdullah (2015) (Alzaqebah & Abdullah, 2015), and Bat17 – Battistuta et al. (2017) (Battistutta et al., 2017). Table 4.6 presents a comparison between the ITC 2007's five finalists and the FastSA, considering the solutions' minimum fitness. In Table 4.7, the average results obtained by state-of-the-art approaches and by the FastSA are compared. The last column reports the standard deviation of the FastSA over ten runs. The results included in Table 4.7 are from approaches whose evaluation is compliant with the ITC 2007 rules, as mentioned in the original papers.

Statistical Analysis

We now present a statistical analysis of the obtained average results (Table 4.7) for the complete ITC 2007 benchmark set (12 instances). We assessed the statistical significance of our results using Friedman's test, as suggested by García & Herrera (2008). The results of the statistical tests were produced using the Java tool made available by the same authors.

Table A.1 summarises the rank obtained by the Friedman test. The *p*-value computed by the Friedman test is 3.93×10^{-6}, which is below the significance interval of 95% ($\alpha = 0.05$), confirming that there is a significant difference among the observed results. Byk13 is the best performing algorithm.

Table 4.6: Minimum fitness obtained by the ITC 2007 finalists' approaches and by the FastSA. The best solutions are indicated in bold. "–" indicates that the corresponding instance is not tested or a feasible solution was not obtained.

Instance	Müller, (2009)	Gogos et al. (2008)	Atsuta et al. (2008)	De Smet, (2008)	Pillay, (2008)	FastSA
1	**4370**	5905	8006	6670	12 035	4872
2	400	1008	3470	623	2886	**395**
3	10 049	13 771	17 669	–	15 917	**9607**
4	18 141	18 674	22 559	–	23 582	**12 076**
5	**2988**	4139	4638	3847	6860	3058
6	26 585	27 640	29 155	27 815	32 250	**25 515**
7	**4213**	6572	10 473	5420	17 666	4291
8	7742	10 521	14 317	–	15 592	**7226**
9	1030	1159	1737	1288	2055	**983**
10	16 682	–	15 085	14 778	17 724	**13 400**
11	34 129	43 888	–	–	40 535	**30 090**
12	5535	–	5264	–	6310	**5137**

Table A.2 shows the adjusted p-values obtained by the post-hoc Holm and Hochberg's tests considering Byk13 as the control algorithm. These tests confirm that Byk13 is better than all algorithms except Bat17, with $\alpha = 0.05$ (3/4 algorithms).

Table A.3 present adjusted p-values for the same studied scenario, obtained by Nemenyi, Holm, Shaffer, and Bergmann procedures, and showing pairwise comparisons. There exist significant differences for the cases ($i = 1, \dots, 4$) using all procedures.

Discussion

Analysing the results, we observe that the FastSA is able to compete with the ITC 2007 finalists (Table 4.6), being superior on all instances except three. With respect to average results (Table 4.7), the FastSA is also able to compete with the state-of-the-art approaches obtaining the best result on one instance.

The employed statistical tests on the average results support our observations: the FastSA is ranked in third position among five algorithms (Table A.1), where Byk13 is considered the best approach; Table A.2, containing the adjusted p-values obtained by Holm's and Hochberg's tests comparing the control algorithm with the remaining algorithms ($1 \times N$ comparison), allows to ascertain that Byk13 is better than all methods except Bat17. Finally, analysing the results of the multiple algorithm comparison ($N \times N$) from Table A.3, we can confirm the same conclusions of Table A.2 and also ascertain that Bat17 is better than Ham13. From these tables, we can say that only Byk13 is better than

Table 4.7: Comparison of the average results obtained by the FastSA algorithm with state-of-the-art approaches for the ITC 2007 benchmark set. The best solutions are indicated in bold. "–" indicates that the corresponding instance was not tested or a feasible solution was not obtained.

Instance	Col09	Dem12	Gog12	Byk13	Ham13	Alz14	Alz15	Bat17	**FastSA**	
1	4799	6330.20	5032	4008	5469	5517.30	5227.81	**3926.96**	5054.70	(111.34)
2	425	612.50	**404**	**404**	450	537.90	457.55	407.72	408.50	(7.09)
3	9251	23 580.00	9484	**8012**	10 444	10 324.90	10 421.64	8849.46	9945.60	(258.60)
4	15 821	–	19 607	13 312	20 241	16 589.10	16 108.27	15 617.82	**12 825.80**	(407.72)
5	3072	5323.00	3158	**2582**	3185	3631.90	3443.72	2849.00	3378.20	(194.58)
6	25 935	28 578.13	26 310	**25 448**	26 150	26 275.00	26 247.27	26 081.35	25 960.50	(213.79)
7	4187	6250.00	4352	3893	4568	4592.40	4415.00	**3661.64**	4533.90	(120.61)
8	7599	9260.90	8098	**6944**	8081	8328.80	8225.81	7729.46	7532.20	(168.82)
9	1071	1255.90	–	**949**	1061	–	–	991.57	1025.80	(33.06)
10	14 552	16 113.33	–	**12 985**	15 294	–	–	13 999.56	13 662.30	(214.54)
11	29 358	–	–	**25 194**	44 820	–	–	27 781.50	31 520.80	(1 027.87)
12	5699	5829.14	–	**5181**	5464	–	–	5550.20	5200.60	(46.84)

FastSA. These results have statistical significance.

4.5 Conclusion

In this work, a novel approach for solving examination timetabling problems is described. The proposed metaheuristic algorithm uses a graph colouring heuristic for solution initialisation and the SA metaheuristic for solution optimisation. Two optimisation algorithms were proposed. The first one is the original SA algorithm. The second one, named *FastSA*, is based on the SA algorithm but uses a modified acceptance criterion, which fixes the selected exam as soon as the exam's number of accepted moves in the previous bin is zero.

The developed SA and FastSA approaches were tested on the public ITC 2007 data set. Compared to the SA, the FastSA uses 17% less evaluations, on average. In terms of solution cost, the FastSA is competitive with the SA algorithm attaining the best average value in four out of twelve instances. Compared with the state-of-the-art approaches, the FastSA improves on one out of twelve instances, and ranks third out of five algorithms. An algorithm comparison was carried out confirming that only one algorithm, Byk13, is better than the FastSA. These results have statistical significance.

Future work will focus on three research lines. The first will encompass the use of the FastSA metaheuristic to the remaining ITC 2007 tracks (course timetabling and post-enrolment course timetabling tracks), and also to other optimisation problems. The second one will involve the study and design of other neighbourhood movements and its combination with the presented ones. The third research direction will involve the study and application of the proposed framework to other simulated annealing variants such as the threshold acceptance algorithm.

Chapter 5

Shuffled Complex Evolution Algorithms for Examination Timetabling

Contents

In this chapter, two approaches based on the *shuffled complex evolution* (SCE) algorithm are proposed for solving the *Examination Timetabling Problem* (ETP). The chapter starts by introducing, in Section 5.1, the SCE algorithm and the related *shuffled frog-leaping algorithm* (SFLA).

In Section 5.2, the proposed SFLA memetic algorithm for the ETP is presented. Section 5.3 addresses the proposed SCE memetic algorithm for the ETP. The chapter ends with the experimental evaluation of the solution methods on the Toronto and ISEL–DEETC benchmark sets, with a discussion of the obtained results.

5.1 Shuffled Complex Evolution Based Algorithms

The SCE is a population-based global optimisation algorithm proposed by Duan, Gupta & Sorooshian (1993). A related algorithm, the SFLA, was proposed later in 2003 by Eusuff et al. (2006); Eusuff & Lansey (2003). In SCE and SFLA, global search is managed as a process of natural evolution. The sample points form a population that is partitioned in distinct groups called *complexes* and *memeplexes* in SCE and SFLA, respectively. Each of the complexes (memeplexes) evolve independently, by searching the space into different directions. After completing a certain number of generations the complexes are combined, and new complexes are formed through the process of *shuffling*. These procedures enhance survivability by a sharing of information about the search space, constructed independently by each complex (Duan et al., 1993). Figure 5.1 illustrates the SCE and SFLA algorithms steps.

The approaches proposed in this chapter are based on the SFLA, which is described in more detail in the next section.

5.1.1 Shuffled Frog-Leaping Algorithm

In SFLA, the set of complexes represent a population of F frogs, denoted $U(i)$, $i = 1, \dots, F$, with identical structure, but different adaptation to the environment. The F frogs are divided in m substructures (complexes/memeplexes), where they “search for food” (they are optimised, in the algorithm sense) and meanwhile, exchange information (exchange memes) with other frogs, trying to reach the food localisation (global optimum). Each memeplex is comprised of n frogs, so that $F = mn$. After searching locally in their memeplex, the frogs are ranked and shuffled in order to go, eventually, to a different memeplex and exchange their memes with the frogs located there. The ranking consists in sorting the frogs in descending order of performance.

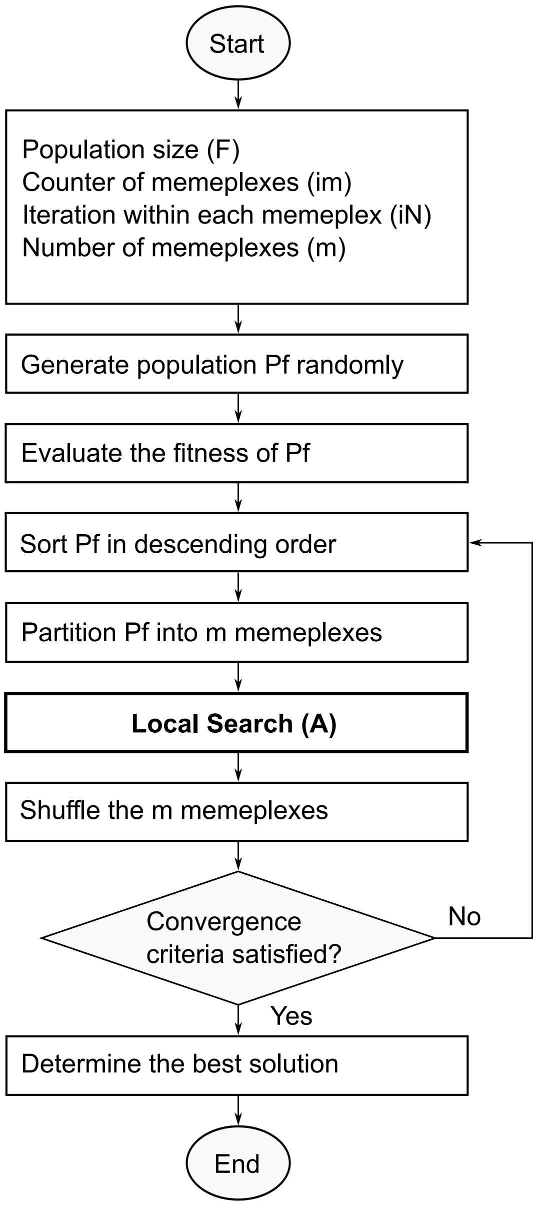

Start
Population size (F)
Counter of memeplexes (im)
Iteration within each memeplex (iN)
Number of memeplexes (m)
Generate population Pf randomly
Evaluate the fitness of Pf
Sort Pf in descending order
Partition Pf into m memeplexes
Local Search (A)
Shuffle the m memeplexes
Convergence criteria satisfied?
No
Yes
Determine the best solution
End

Figure 5.1: SCE and SFLA algorithms steps. (Adapted from (Amiri et al., 2009).)

The partition of frogs is done as follows. The first frog (the frog with the best fitness) in the sorted list is allocated to memeplex 1; the second frog is allocated to memeplex 2, and so on, so that frog m will go to memeplex m; then, the frog in the $m + 1$ position will go to memeplex 1, the frog in the $m + 2$ position frog will go to memeplex 2, and the process continues in this fashion for the remaining frogs.

In the original SFLA, in order to prevent the algorithm getting stuck in a local optimum, a *submemeplex* of size $q < n$ is constructed in each memeplex (Eusuff et al., 2006).

The individual frogs in the memeplex are selected to form a submemeplex according to their fitness. The selection strategy is to give higher weights to frogs that have higher performance values and less weight to those with lower performance values.

Local Search Step

The SFLA's local search is now detailed. This process is known as *Frog-Leaping local search* and is illustrated in Figure 5.2.

In each submemeplex, the P_b and P_w vectors denote, respectively, the *best* and the *worst* frog. A *global best* frog is also maintained in the algorithm, denoted as P_g. At the end of each iteration of the Frog-Leaping local search, the worst frog in the submemeplex is updated according to the following rule (Eusuff et al., 2006):

$$S = \begin{cases} \min\left\{\text{int}\left[rand * (P_b - P_w)\right], S_{max}\right\}, & \text{for a positive step} \\ \max\left\{\text{int}\left[rand * (P_b - P_w)\right], -S_{max}\right\}, & \text{for a negative step} \end{cases} \tag{5.1}$$

$$U(q) = P_w + S \tag{5.2}$$

where S denotes the update step size, $rand$ represents a random number between $(0, 1)$ and S_{max} is defined as the maximum step size that any frog can take. $U(q)$ is the frog placed in the last position in the memeplex (the previous worst frog) and that will be replaced according to (5.2). The resulting frog could be worse, equal or better than the previous worst frog. After each loop, the memeplex is reordered. The idea of this step is to update the worst frog position towards the direction of the best frog in the memeplex.

5.2 Hybrid SFLA for the ETP

In this section, we describe a memetic algorithm for solving the ETP that uses the SFLA working principles. The proposed hybrid heuristic algorithm, named *hybrid shuffled frog-leaping algorithm* (HSFLA), incorporates features from the standard SFLA and *simulated annealing* (SA) (Kirkpatrick et al., 1983) metaheuristics. The algorithm flow is illustrated in Figure 5.3. It starts by generating a population of feasible solutions which is then optimised by the HSFLA. The SA metaheuristic has the following known features:

- SA local search can lead to near optimal solutions if a slow annealing process is conducted, at the cost of a longer execution time.

- The quality of the optimised solution depends not only on the SA parameters but

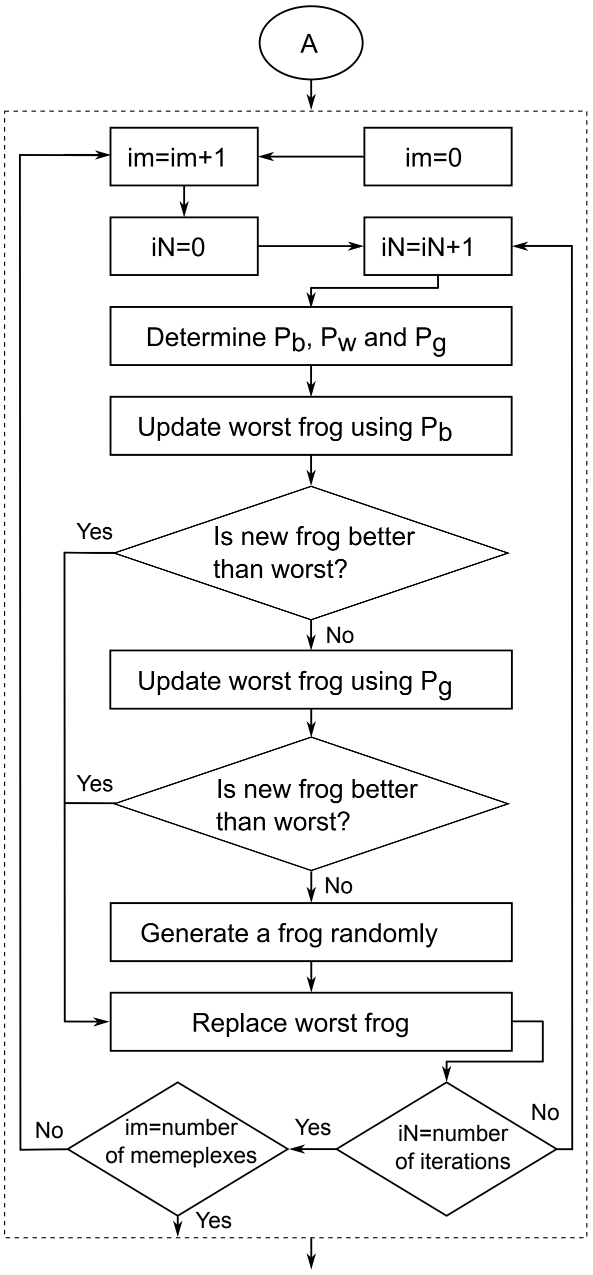

A
im=im+1
im=0
iN=0
iN=iN+1
Determine P_b, P_w and P_g
Update worst frog using P_b
Yes
Is new frog better than worst?
No
Update worst frog using P_g
Yes
Is new frog better than worst?
No
Generate a frog randomly
Replace worst frog
No
im=number of memeplexes
Yes
iN=number of iterations
No
Yes

Figure 5.2: Shuffled Frog-Leaping Algorithm local search. (Adapted from (Amiri et al., 2009).)

also on the initial solution. If the initial solution is not very optimised, the improvement attained could be considerable; on the other way, when we rerun SA on an optimised solution, we could obtain a worse solution or a better solution, but in the last case the improvement is marginal.

The HSFLA was designed taking these points into consideration. It works like a multi-

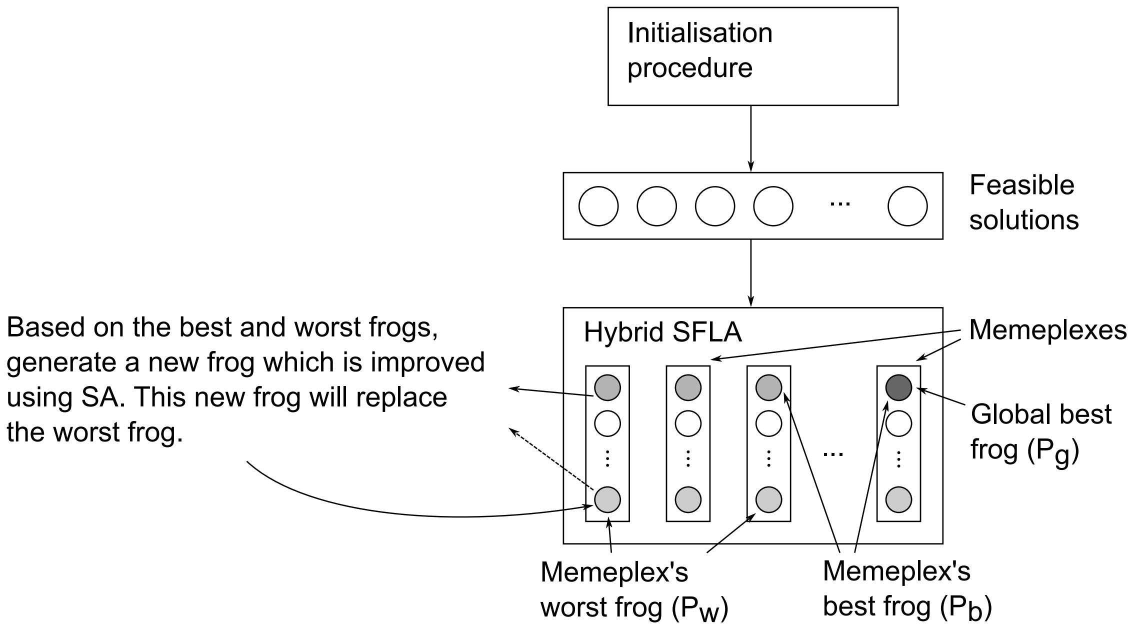

Initialisation procedure
Feasible solutions
Hybrid SFLA
Memeplexes
Based on the best and worst frogs, generate a new frog which is improved using SA. This new frog will replace the worst frog.
Global best frog (P_g)
Memeplex's worst frog (P_w)
Memeplex's best frog (P_b)

Figure 5.3: Flow of the hybrid heuristic algorithm. A set of feasible solutions is obtained using the Saturation Degree graph colouring heuristic. Next, these solutions are organised in memeplexes and optimised using the hybrid SFLA. In the SFLA local search, the memeplex's worst frog is replaced by a new frog which results from the combination of the memeplex's best and worst frogs followed by application of the SA metaheuristic.

start SA optimising different initial solutions. It maintains elitism by keeping the global best frog. After shuffling the memeplexes, the SA is executed again on solutions of a given memeplex, and the process is repeated for all the memeplexes for a given number of time loops (Figure 5.1).

In the next section, we describe the HSFLA's application to the Toronto benchmark set. The following algorithm components are described: i) solution representation, ii) initialisation procedure, iii) neighbourhood structure, and iv) SFLA's worst frog improvement and random frog generation.

5.2.1 Application to the Toronto Benchmark Set

Solution Representation

Each individual frog (solution) is represented by an array of dimension equal to the number of time slots, where each position contains an array of exams scheduled at that time slot. The adopted representation is the same as the one described in Section 4.1.3 and illustrated in Figure 4.1. In our method, only feasible solutions are manipulated as all the operators produce feasible timetables. The fitness of a solution is the value of the proximity cost function to be minimised, which is a measure of the soft constraints violations (given by Equation (2.3) in Section 2.4.1).

Initialisation Procedure

The initial frog population is created using a construction algorithm that is based on the *saturation degree* (SD) graph colouring heuristic as described in Section 4.1.3.

Neighbourhood Structure

Local search methods like SA start from an initial solution and explore other candidate solutions in the neighbourhood. The neighbourhood comprises a set of solutions that are reached from the initial one by applying a move. The search progresses by moving to a candidate solution (which, in the case of SA, may or may not improve the previous solution) and repeating the process until a given stopping criterion is met (see Section 4.1). In our approach we use two neighbourhoods, *PeriodSwap* and *KempeChain*, published in the literature. The neighbourhoods are denoted N_1 and N_2, respectively, and are defined as follows.

Neighbourhood N_1: exchange exams in time slot t_i with exams in time slot t_j, where t_i and t_j are two randomly chosen time slots. This neighbourhood was introduced in (Burke & Bykov, 2008). It maintains the solution feasibility since all exams in a time slot are swapped.

Neighbourhood N_2: perturb, in a feasible fashion, an exam included in a *Kempe chain* (see Section 4.1.3).

Neighbourhoods N_1 and N_2 are applied in the HSFLA's worst frog improvement step, described in detail next.

Worst Frog Improvement and Random Frog Generation

In the original SFLA, each solution (frog) is a vector, and the worst performance frog within each submemeplex is updated towards the direction of the best frog, according to Equations (5.1) and (5.2) (see Figure 5.2). In our adaptation of the SFLA for the ETP, we update the worst frog by applying three operators, which are: crossover, mutation, and local search based on the SA metaheuristic. These are specified in Algorithm 5 which describes the worst frog improvement procedure.

In Step 1 of Algorithm 5, we combine the worst and best frogs in order to produce the new candidate frog. In Step 2, we apply a mutation operator using neighbourhood

Algorithm 5 Worst frog improvement procedure.

```
Input: $P_b$, $P_w$: Memeplex's best and worst frogs.
Output: $P_w^{'}$: New candidate frog. It will replace $P_w$ if it is better.
1: Set $P_w^{'} = P_b$.
2: **Step 1:** With probability $pc$ make $P_w^{'} = crossover(P_b, P_w)$.
3: **Step 2:** With probability $pm$ make $P_w^{'} = mutation_{N_1}(P_w^{'})$.
4: **Step 3:** With probability $pi$ make $P_w^{'} = SA_{N_2}(P_w^{'})$.
```

N_1 to the solution $P_w^{'}$ obtained in Step 1. In Step 3, we apply the SA metaheuristic with neighbourhood N_2 to the solution $P_w^{'}$ obtained in Step 2. The implemented SA is described in Algorithm 2. Each step is executed with a given probability, so in the event that no operator is applied, the new candidate frog $P_w^{'}$ is equal to P_b.

Executing the steps of the SFLA local search (Figure 5.2), the new frog is going to replace the worst frog if it is better than this last one. Otherwise, the procedure is repeated but substituting P_b by the global best frog, P_g. If the new frog doesn't still improve over the worst frog $U(q)$, then a random solution is generated as the new $U(q)$, replacing the worst frog. To generate a random frog, we use the construction method described above.

The crossover operator in Step 1 is described next.

Algorithm 6 Crossover operator for the Toronto benchmark set.

```
1: function CROSSOVEROPERATORTORONTO($P_1$,$P_2$)    ▷ '$P_1$' and '$P_2$' are the parent solutions
2:     $O_1 \leftarrow P_1$                ▷ Initialise offspring $O_1$ as a copy of $P_1$
3:     k ← 0
4:     while k < 3 do
5:         Select random time slots $t_i$ and $t_j$, from $P_1$ and $P_2$, respectively.
6:         Insert (if possible) exams from time slot $t_j$ from $P_2$ into time slot $t_i$ of $O_1$. Exams that generate hard constraint violations or that are already found in time slot $t_i$, are not inserted. The duplicated exams in the other time slots of $O_1$ are removed (Figure 5.4).
7:         k ← k+1
8:     end while
9:     Output: Offspring individual $O_1$
10: end function
```

Recombination Operator

The proposed recombination (crossover) operator, used in the case of the Toronto benchmark set, was adapted from the one reported in Abdullah et al. (2010); Sabar, Ayob, Kendall & Qu (2012). The steps of the crossover operator are detailed in Algorithm 6.

t_1	t_2	t_3	t_4	t_5
e_2	e_1	e_9	e_6	e_8
e_{14}	e_{11}	e_{20}	e_{13}	e_{12}
e_{10}	e_4	e_5	e_7	e_{15}
e_3		e_{18}		e_{17}
e_{16}				e_{19}

(a) Solution P_1

t_1	t_2	t_3	t_4	t_5
e_{15}	e_9	e_6	e_5	e_8
e_{20}	e_2	e_1	e_{18}	e_{14}
	e_{12}	e_{17}	e_4	e_{11}
	e_{10}	e_{13}	e_{16}	e_3
	e_7			e_{19}

(b) Solution P_2

t_1	t_2	t_3	t_4	t_5
e_2	e_1	e_9	e_6	e_8
~~e_{14}~~	e_{11}	e_{20}	e_{13}	e_{12}
e_{10}	e_4	e_5	e_7	e_{15}
~~e_3~~	~~e_8~~	e_{18}		e_{17}
e_{16}	e_{14}			~~e_{19}~~
	~~e_{11}~~			
	e_3			
	e_{19}			

(c) Offspring O_1

Figure 5.4: Crossover between P_1 and P_2. The resulting solution O_1 in (c) is the result of combining solution P_1 (a) with solution P_2 (b). Initially, O_1 is set to P_1. The operator inserts exams chosen from a random time slot of P_2 (time slot t_5 shown dark shaded in (b)) in a random time slot t_2 of O_1 (shown dark shaded in (a) and (c)). Let the exams e_8 and e_1 be conflicting exams. When inserting exams from time slot t_5 of P_2 in time slot t_2 of O_1 (shown light gray in (c)), some exams could be infeasible (the case of e_8 in (c), given the above assumption) or may already be found in that time slot (for example, e_{11} in (c)). These exams are not inserted. The duplicated exams in the other time slots are removed.

This crossover operator generates feasible offspring solutions, so no special repair procedure is needed. Figure 5.4 illustrates an excerpt of the crossover operation for generating one of the two offspring, and using just a single crossover point (instead of using three points, as it is done in the proposed operator).

The generic crossover illustrated in Figure 5.4 is applied in the HSFLA, in Step 1 of Algorithm 5, using the rule: with probability pc make $P_w^{'} = crossover(P_b, P_w)$. Thus, P_1, and P_2 correspond to P_b and P_w, respectively, and $P_w^{'}$ corresponds to O_1 (Figure 5.4)

5.3 Hybrid SCE for the ETP

In this section we describe the hybrid SCE algorithm for solving the ETP. The proposed memetic algorithm, named *shuffled complex evolution algorithm* (SCEA), integrates in its local search step the *great deluge* (GD) metaheuristic. The SCEA main steps are illustrated in Figure 5.1, whereas the SCEA local search step is illustrated in Figure 5.5. The main loop of SCEA is identical to the SCE and SFLA main loop, where complexes are formed by creating random initial solutions that span the search space. Here, instead of points, the solutions correspond to complete and feasible timetables.

5.3.1 SCEA Local Search Step

Like the HSFLA presented in the previous section, the local search step of the SCEA, presented in Figure 5.5, is based in the SFLA's Frog-Leaping local search, but adapted in order to operate with ETP solutions. Like the SFLA, we maintain the best and worst solutions of the memeplex, denoted respectively as P_b and P_w, and elitism is achieved by maintaining the best global solution, denoted as P_g. The local search step starts by selecting, randomly, and according with crossover probability cp, two parent solutions, P_1 and P_2, for recombination in order to produce a new offspring. P_1 must be different from the complex's best solution. The solution P_1 is recombined with solution P_2 using the crossover operation specified in Algorithm 6 (and illustrated in Figure 5.4), with the exception that only a single iteration of the loop in Algorithm 6 is performed. The resulting offspring replaces the parent P_1 (Figure 5.5). After this, the complex is sorted in order of increasing objective function value.

After the crossover, a solution in the complex is selected for improvement according to an improvement probability, pi. The solution is improved by employing the local search meta-heuristic GD (Dueck, 1993). The template of GD is presented in Algorithm 7.

The selection of the solution to improve is made on the group of the top t best solutions. The exploitation using GD is done on a clone of the original solution, selected from this group. If the optimised solution is better than the original, then it will replace the complex's worst solution. This updating step in conjunction with the crossover operator guarantees a reasonable diversity, in an implicit fashion.

The GD (Algorithm 7) was integrated in the SCEA in the following fashion. We use as the initial solution s_0 the chosen solution to improve. The level $LEVEL$ is set to the fitness value of this initial solution s_0. The search stops when the water level is equal to the solution fitness.

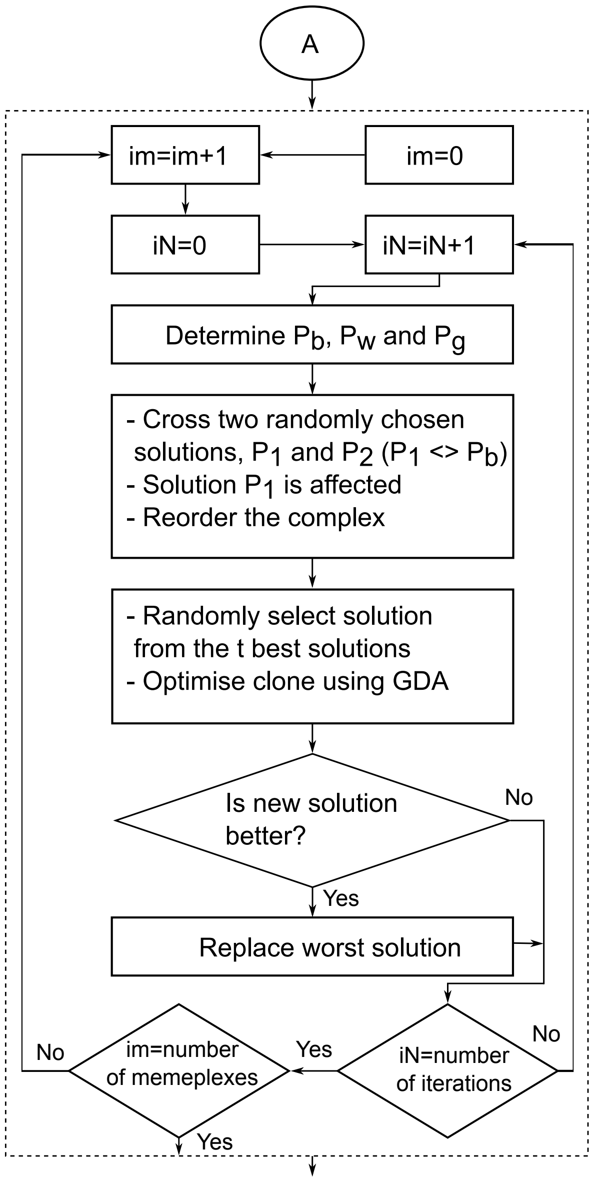


Figure 5.5: Shuffled Complex Evolution Algorithm local search step.

5.3.2 Application to the Toronto and ISEL–DEETC Benchmarks

Solution Representation and Construction

The solution representation and construction of the initial feasible solutions is the same as for the HSFLA, and described in Section 5.2.1.

Algorithm 7 Template of the Great Deluge Algorithm.

```
1: Input:
2:     – $s_0$                                              ▷ Initial solution
3:     – Initial water level $LEVEL$
4:     – Rain speed $UP$                                    ▷ $UP > 0$
5: $s \leftarrow s_0$;                                      ▷ Generation of the initial solution
6: repeat
7:     Generate a random neighbour $s'$
8:     If $f(s') < LEVEL$ Then $s \leftarrow s'$            ▷ Accept the neighbour solution
9:     $LEVEL \leftarrow LEVEL - UP$                        ▷ update the water level
10: until Stopping criteria satisfied
11: Output: Best solution found.
```

Neighbourhood Structure

In the local search with GD we employ the Kempe chain neighbourhood (Demeester et al., 2012), which corresponds to Neighbourhood N_2 of HSFLA presented in Section 5.2.1.

Two-epoch Feasibility

In order to be able to execute the SCEA on the ISEL–DEETC two-epoch problem, a different version of the initialisation procedure, crossover and neighbourhood operators had to be implemented. This was done in order for the algorithm to manage the hard constraint H_2 mentioned in Section 3.3.3. This modified version is executed in the first epoch generation, while the original version is executed in the second epoch generation.

5.4 Experimental Results

5.4.1 Hybrid SFLA

The performance of the HSFLA was evaluated using the Toronto benchmark set (see Section 2.4.1). The algorithm was programmed in the C++ language and was based on the ParadisEO framework Cahon, Melab & Talbi (2004). The hardware and software specifications are: Intel Core i7-2630QM, CPU @ 2.00 GHz $\times$ 8, with 8 GB RAM; OS: Ubuntu 12.04, 32 bit; Compiler used: GCC v. 4.6.3. The parameters of HSFLA are: Population size $F = 50$, Memeplex count $m = 10$, Memeplex and Submemeplex size $n = q = 5$ (no submemeplexes were defined), and Number of time loops (convergence criterion) $L = 3$. The SA parameters are: $T_{max} = 0.1$, $r = 0.00001$, $k = 5$, and $T_{min} = 0.0000001$. For this cooling schedule the number of evaluations done in each SA is 6 907 760. The crossover, mutation and improvement probabilities, respectively, pc, pm, and pi, were set to 0.1. The parameter values were chosen empirically, in a way to achieve a reasonable balance between global and local exploration, and also establish a satisfactory compromise between solution quality and execution time. To obtain our computational results, the HSFLA was run five times on each instance with different random seeds.

Comparative Results and Discussion

Tables 5.1, 5.2 and 5.3 show the best results of the HSFLA on the Toronto benchmark set as well as a selection of the best results available in the literature. In the last two rows of each table, the *TP* and *TP (11)* indicate the total penalty for the 13 instances and the total penalty except the pur93 and rye92 instances, respectively.

Tables 5.4 and 5.5 compare HSFLA with the top seven best algorithms. For the HSFLA we present the lowest penalty value f_{min}, the average penalty value f_{ave}, and the standard deviation σ over five independent runs. For the reference algorithms we present the best and average (where available) results. The authors analysed in Tables 5.4 and 5.5 mention computation times that are within several minutes – 1 hour, to several hours (12 hours maximum). Demeester et al. (2012) mention a maximum of 12 hours of computation time for all instances.

Table 5.6 compares the computation times of HSFLA and Demeester et al.'s algorithms. For the largest instance, pur93, the stopping criterion was the completion of a single run of the SA metaheuristic.

The best results obtained by HSFLA are comparable with the ones produced by state-of-the-art algorithms, and HSFLA is able to produce some of the best average results. We also observe that HSFLA obtains the lowest sum of average cost on the *TP* and *TP (11)* quantities, for the Toronto set. For the larger instances, HSFLA attains longer computation times. However, good solutions are obtained soon after the first SA execution.

Further studies should focus on the HSFLA parameters optimisation in order to reduce computation time while not degrading the performance significantly. Future investigations will rely on the use of other competitive metaheuristics, e.g., *tabu search* (TS) and GD. In terms of implementation, incremental evaluation of the neighbourhood operator should be carried out.

As future research, we intend to apply our solution method to the instances of the first Track (Examination Timetabling) of the *Second International Timetabling Competition* (ITC 2007), which contains more hard and soft constraints.

Table 5.1: HSFLA results and comparison with a selection of the best algorithms from literature. Values in bold represent the best results reported. "–" indicates that the corresponding instance was not tested or a feasible solution was not obtained.

Data set	Car96	Bur02	Mer03	Bur04	Bur04a	Ken05
car91	7.10	4.65	5.10	5.00	4.80	5.37
car92	6.20	4.10	4.30	4.30	4.20	4.67
ear83	36.40	37.05	35.10	36.20	35.40	40.18
hec92	10.80	11.54	10.60	11.60	10.80	11.86
kfu93	14.00	13.90	13.50	15.00	13.70	15.84
lse91	10.50	10.82	10.50	11.00	10.40	–
pur93	3.90	–	–	–	4.80	–
rye92	7.30	–	8.40	–	8.90	–
sta83	161.50	168.73	157.30	161.90	159.10	157.38
tre92	9.60	8.35	8.40	8.40	8.30	8.39
uta92	3.50	3.20	3.50	3.40	3.40	–
ute92	25.80	25.83	25.10	27.40	25.70	27.60
yor83	41.70	37.28	37.40	40.80	36.70	–
TP (11)	327.10	325.45	310.80	325.00	312.50	
TP	338.30				326.20	

Table 5.2: HSFLA results and comparison with a selection of the best algorithms from literature (continuation).

Data Set	Yan05	Bur06	Bur08	Car08	Abd09	Sab09
car91	4.50	4.42	4.58	6.60	4.42	4.79
car92	3.93	**3.74**	3.81	6.00	3.76	3.90
ear83	33.71	32.76	32.65	**29.30**	32.12	34.69
hec92	10.83	10.15	10.06	**9.20**	9.73	10.66
kfu93	13.82	12.96	12.81	13.80	**12.62**	13.00
lse91	10.35	9.83	9.86	**9.60**	10.03	10.00
pur93	–	–	4.53	**3.70**	–	–
rye92	8.53	–	7.93	**6.80**	–	10.97
sta83	158.35	157.03	157.03	158.20	156.94	157.04
tre92	7.92	7.75	7.72	9.40	7.86	7.87
uta92	3.14	3.06	3.16	3.50	**2.99**	3.10
ute92	25.39	24.82	24.79	**24.40**	24.90	25.94
yor83	36.35	34.84	34.78	36.20	34.95	36.18
TP (11)	308.29	301.36	301.25	306.20	**300.32**	307.17
TP			**313.71**	316.70		

Table 5.3: HSFLA results and comparison with a selection of the best algorithms from literature (continuation).

Data Set	Bur10	Abd10	Tur11	Dem12	Abd13	**HSFLA**
car91	4.90	**4.35**	4.81	4.52	4.76	4.59
car92	4.10	3.82	4.11	3.78	3.94	3.86
ear83	33.20	33.76	36.10	32.49	33.61	32.72
hec92	10.30	10.29	10.95	10.03	10.56	10.08
kfu93	13.20	12.86	13.21	12.90	13.44	12.87
lse91	10.40	10.23	10.20	10.04	10.87	9.85
pur93	–	–	–	5.67	–	4.47
rye92	–	–	–	8.05	8.81	8.00
sta83	**156.90**	**156.90**	159.74	157.03	157.09	157.03
tre92	8.30	8.21	8.00	**7.69**	7.94	7.78
uta92	3.30	3.22	3.32	3.13	3.27	3.15
ute92	24.90	25.41	26.17	24.77	25.36	24.76
yor83	36.30	36.35	36.23	**34.64**	35.74	34.85
TP (11)	305.80	305.40	312.84	301.02	306.58	301.54
TP				314.74		314.01

Table 5.4: HSFLA results and comparison with the best algorithms from literature. Values in bold represent the best results reported. "–" indicates that the corresponding instance was not tested or a feasible solution was not obtained.

Data set	HSFLA			Bur06	Car08	Abd09	
	f_{min}	f_{ave}	σ	f_{min}	f_{min}	f_{min}	f_{ave}
car91	4.59	**4.62**	0.03	4.42	6.60	4.42	4.81
car92	3.86	3.87	0.01	**3.74**	6.00	3.76	3.95
ear83	32.72	32.80	0.07	32.76	**29.30**	32.12	33.69
hec92	10.08	10.10	0.01	10.15	**9.20**	9.73	10.10
kfu93	12.87	**12.91**	0.03	12.96	13.80	**12.62**	12.97
lse91	9.85	**9.90**	0.06	9.83	**9.60**	10.03	10.34
pur93	4.47	**4.49**	0.03	–	**3.70**	–	–
rye92	8.00	**8.03**	0.03	–	**6.80**	–	–
sta83	157.03	**157.03**	0.00	157.03	158.20	156.94	157.30
tre92	7.78	7.84	0.05	7.75	9.40	7.86	8.20
uta92	3.15	3.18	0.02	3.06	3.50	**2.99**	3.32
ute92	24.76	**24.80**	0.02	24.82	**24.40**	24.90	25.41
yor83	34.85	35.00	0.09	34.84	36.20	34.95	36.27
TP (11)	301.54	**302.05**		301.36	306.20	**300.32**	306.36
TP	314.01	**314.57**			316.70		

Table 5.5: HSFLA results and comparison with the best algorithms from literature (continuation).

Data set	Abd10	Bur10	Dem12		Bur08	
	f_{min}	f_{min}	f_{min}	f_{ave}	f_{min}	f_{ave}
car91	**4.35**	4.90	4.52	4.64	4.58	4.68
car92	3.82	4.10	3.78	**3.86**	3.81	3.92
ear83	33.76	33.20	32.49	**32.69**	32.65	32.91
hec92	10.29	10.30	10.03	**10.06**	10.06	10.22
kfu93	12.86	13.20	12.90	13.24	12.81	13.02
lse91	10.23	10.40	10.04	10.21	9.86	10.14
pur93	–	–	5.67	5.75	4.53	4.71
rye92	–	–	8.05	8.20	7.93	8.06
sta83	**156.90**	**156.90**	157.03	157.05	157.03	157.05
tre92	8.21	8.30	**7.69**	**7.79**	7.72	7.89
uta92	3.22	3.30	3.13	**3.17**	3.16	3.26
ute92	25.41	24.90	24.77	24.88	24.79	24.82
yor83	36.35	36.30	**34.64**	**34.83**	34.78	35.16
TP (11)	305.40	305.80	301.02	302.42	301.25	303.07
TP			314.74	316.37	**313.71**	315.84

5.4.2 SCEA

The performance of the SCEA was evaluated using the Toronto benchmark set (Section 2.4.1) and the ISEL–DEETC benchmark set (Section 3.2). The algorithm was programmed in C++ using the ParadisEO framework (Talbi, 2009). The hardware and software specifications are: Intel Core i7-2630QM, CPU @ 2.00 GHz $\times$ 8, with 8 GB RAM; OS: Ubuntu 14.04, 64 bit; Compiler used: GCC v. 4.8.2. The parameters of SCEA are: Population size $F = 24$, Memeplex count $m = 3$, Memeplex size $n = 8$ (no sub-memeplexes were defined), and Number of time loops (convergence criterion) $L = 100\,000\,000$. The number of best solutions to consider for selection on a complex is given by $t = n/4 = 2$. The GD algorithm parameter, UP, was set to: $UP = 1{\times}10^{-7}$. The crossover and improvement probabilities, cp and ip, were set equal to 0.2 and 1.0, respectively. The parameter values were chosen empirically. To obtain the simulation results, the SCEA was run five times on each instance with different random seeds. The running time of the algorithm was limited to 24 hours to all instances except for the Toronto's pur93 instance. For this larger instance, the running time was limited to 48 hours.

Table 5.6: Minimum and average fitness and standard deviation comparison. For the pur93 instance, the HSFLA was stopped after executing one local search with SA.

Data	HSFLA				Demeester et al. (2012)			
set	Execution time	f_{min}	f_{ave}	σ	Stopping criterion	f_{min}	f_{ave}	σ
car91	27 h	4.59	**4.62**	0.03	4 h	4.68	4.75	0.05
					12 h	4.52	4.64	0.05
car92	14 h	3.86	3.87	0.01	4 h	3.84	3.94	0.05
					12 h	3.78	**3.86**	0.06
ear83	6 h	32.72	32.80	0.07	2 h	32.82	33.02	0.16
					12 h	32.49	**32.69**	0.13
hec92	1 h	10.08	10.10	0.01	1 h	10.09	10.20	0.13
					12 h	10.03	**10.06**	0.03
kfu93	17 h	12.87	**12.91**	0.03	2 h	13.06	13.45	0.31
					12 h	12.90	13.24	0.20
lse91	10 h	9.85	**9.90**	0.06	2 h	10.06	10.38	0.19
					12 h	10.04	10.21	0.13
pur93	15 h	4.47	**4.49**	0.03	4 h	6.45	6.57	0.07
					12 h	5.67	5.75	0.05
rye92	17 h	8.00	**8.03**	0.03	4 h	8.18	8.31	0.10
					12 h	8.05	8.20	0.12
sta83	4 h	157.03	**157.03**	0.00	1 h	157.03	157.05	0.01
tre92	8 h	7.78	7.84	0.05	2 h	7.73	7.91	0.06
					12 h	**7.69**	**7.79**	0.07
uta92	30 h	3.15	3.18	0.02	2 h	3.32	3.37	0.03
					12 h	3.13	**3.17**	0.03
ute92	3 h	24.76	**24.80**	0.02	2 h	24.83	24.99	0.24
					12 h	24.77	24.88	0.17
yor83	5 h	34.85	35.00	0.09	2 h	34.79	35.06	0.25
					12 h	**34.64**	**34.83**	0.14
	Total			σ_{ave}	Total			σ_{ave}
	157 h	314.01	**314.57**	**0.0346**	32 h	316.88	319.00	0.13
					145 h	314.74	316.37	0.0916

Results on the Toronto Benchmark Set

Tables 5.7 and 5.8 show the best results of SCEA on the Toronto benchmark set as well as a selection of the best results available in the literature. The results of the approaches of Car96 and Bur06 were validated in Qu et al. (2009). In the last two rows of each table, the *TP* and *TP (11)* indicate, the total penalty for the 13 instances and the total penalty

Table 5.7: Simulation results of SCEA and comparison with selection of best algorithms from literature. Values in bold represent the best results reported. "–" indicates that the corresponding instance was not tested or a feasible solution was not obtained.

Data set	Car96	Bur06	Abd09		Bur10
	f_{min}	f_{min}	f_{min}	f_{ave}	f_{min}
car91	7.10	4.42	4.42	4.81	4.90
car92	6.20	**3.74**	3.76	3.95	4.10
ear83	36.40	32.76	**32.12**	33.69	33.20
hec92	10.80	10.15	**9.73**	10.10	10.30
kfu93	14.00	12.96	**12.62**	**12.97**	13.20
lse91	10.50	**9.83**	10.03	10.34	10.40
pur93	**3.90**	–	–	–	–
rye92	**7.30**	–	–	–	–
sta83	161.50	157.03	156.94	157.30	**156.90**
tre92	9.60	7.75	7.86	8.20	8.30
uta92	3.50	3.06	**2.99**	3.32	3.30
ute92	25.80	24.82	24.90	25.41	24.90
yor83	41.70	34.84	34.95	36.27	36.30
TP (11)	327.10	301.36	**300.32**	306.36	305.80
TP	338.30	–	–	–	–

Table 5.8: Simulation results of SCEA (continuation).

Data set	Abd10	Dem12		**SCEA**		
	f_{min}	f_{min}	f_{ave}	f_{min}	f_{ave}	σ
car91	**4.35**	4.52	4.64	4.41	**4.45**	0.03
car92	3.82	3.78	3.86	3.75	**3.77**	0.01
ear83	33.76	32.49	**32.69**	32.62	**32.69**	0.07
hec92	10.29	10.03	**10.06**	10.03	**10.06**	0.03
kfu93	12.86	12.90	13.24	12.88	13.00	0.13
lse91	10.23	10.04	10.21	9.85	**9.93**	0.12
pur93	–	5.67	5.75	4.10	**4.17**	0.05
rye92	–	8.05	8.20	7.98	**8.06**	0.06
sta83	**156.90**	157.03	157.05	157.03	**157.03**	0.00
tre92	8.21	**7.69**	**7.79**	7.75	7.80	0.05
uta92	3.22	3.13	3.17	3.08	**3.15**	0.05
ute92	25.41	**24.77**	24.88	24.78	**24.81**	0.02
yor83	36.35	34.64	34.83	**34.44**	**34.73**	0.17
TP (11)	305.40	301.02	302.42	300.62	**301.42**	
TP	–	314.74	316.37	**312.70**	**313.65**	

except the pur93 and rye92 instances, respectively. For the SCEA we present the lowest penalty value f_{min}, the average penalty value f_{ave}, and the standard deviation σ over five independent runs. For the reference algorithms we present the best and average (where available) results. The computation times for the analysed algorithms range from several minutes – 1 hour, to several hours (12 hours maximum).

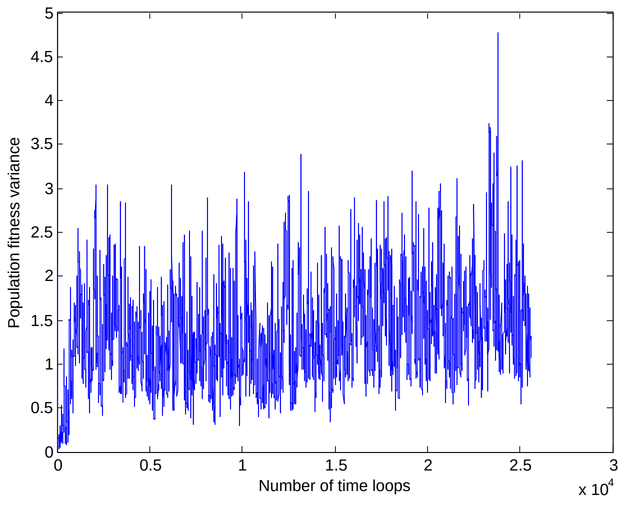
The best results obtained by SCEA are competitive with the ones produced by state-of-the-art algorithms. It attains a new lower bound on the yor83 instance. We also observe that SCEA obtains the lowest sum of average cost on the *TP* and *TP (11)* quantities, and the lowest sum of best costs on the *TP* quantity, for the Toronto instances. This demonstrates that SCEA can optimise very different instances with good efficiency. A negative aspect of SCEA is the time taken compared with other algorithms. The time taken is a reflex of the exploration high diversity and the use of a low value for the decreasing rate UP. A low UP value is needed in order for the GD be able to explore the best exam movements. If the UP value is higher, the optimisation is faster but with worse results, because the initial, larger-conflicting exams, are scheduled into sub-optimal time slots, and thus the remaining exams, as the GD algorithm's level decreases, are scheduled into sub-optimal time slots as well.

The SCEA exploration diversity is now analysed. Figure 5.6 illustrates the SCEA evolution on the yor83 instance. We can observe, in Figure 5.6(a), that the mean value of the population fitness variance along time remains near 1.5, guaranteeing a diverse set of solutions in the population. Figure 5.6(b) illustrates the best solution fitness evolution along time. We can observe a fast decrease in the solution fitness value in the first time loops, attaining a stationary value from that point on. Despite this, the algorithm is still able to improve the cost further.

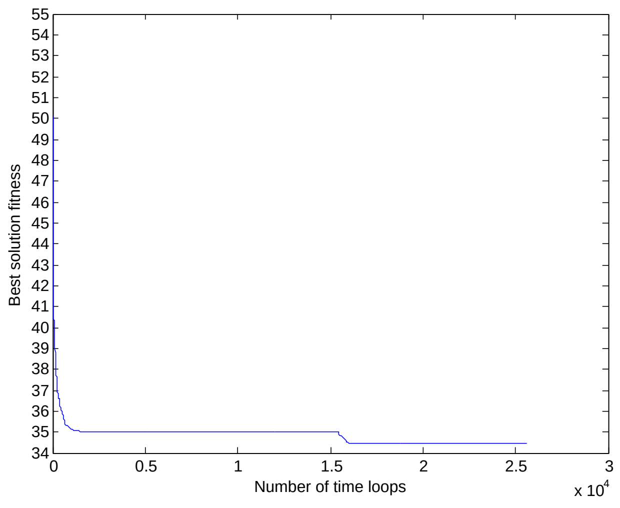
Results on the ISEL–DEETC Benchmark Set

For the ISEL–DEETC, we compare the automatic solution with a manual solution available from the ISEL academic services. The L_{min} parameter (Section 3.3.3) was set to $L_{min} = 10$. With this L_{min} value, the first and second epoch examinations of a given course are 10 time slots apart.

Table 5.9 presents the costs for the manual and automatic solutions produced by the SCEA. We mention that while the timetables were optimised using the fitness function specified in Equations (3.8) and (3.2), for the first and second epochs, respectively, in the collected results the fitness function defined by (3.14) was used instead. This latter function is also used in the approach presented in Chapter 7 and considers conflicts from two consecutive time slots, excluding conflicts on Saturdays. This measure is used in



(a)



(b)

Figure 5.6: SCEA evolution on yor83 instance (a) Population fitness variance evolution along time; (b) Best solution fitness evolution along time.

order to be able to compare SCEA with the method developed in Chapter 7.

The SCEA could generate timetables with lower cost compared with the manual solution. We note that SCEA optimised the merged timetable comprising the five ISEL–DEETC timetables and not the individual timetables, so, in some cases, the course programmes' timetables have worse cost (e.g., MEIC, in the second epoch). The results produced in the first epoch are comparable with the results obtained by the method developed in Chapter 7, in which the ISEL–DEETC single-epoch problem instance is solved using a multi-objective evolutionary algorithm.

Tables 5.10 and 5.11 illustrate the manual and automatic solutions for the most diffi-

Table 5.9: Number of clashes for the manual and automatic solutions of the ISEL–DEETC benchmark set. For the automatic solution, the best cost out of five runs is presented.

Timetable	**Manual sol.**		**Automatic sol.**	
	1st ep.	2nd ep.	1st ep.	2nd ep.
LEETC	287	647	238	550
LEIC	197	442	171	418
LERCM	114	208	125	195
MEIC	33	63	23	66
MEET	50	144	23	124
Combined	549	1163	447	1060
Sum	1712		1507	

cult timetable, the LEETC timetable, respectively.

Conclusions

We presented a memetic algorithm that combines features from the SCE and GD metaheuristics. The experimental evaluation of the SCEA shows that it is competitive with state-of-the-art methods. In the set comprising the 13 instances of the Toronto benchmark data, it attains the lowest cost on one instance, and the lowest sum of best and average cost with a low standard deviation. The algorithm main disadvantage is the time taken on the larger instances.

Further studies will address the diversity management in order to accelerate the algorithm while maintaining a satisfactory diversity. As future research, we intend to apply the SCEA to the instances of the first track (Examination Timetabling) of the ITC 2007, which contains more hard and soft constraints.

Table 5.10: Manual solution for the LEETC examination timetable. The courses marked in bold face are shared with other programs. The number of clashes of this timetable is 287 and 647 for the first and second epochs, respectively.

Course	First epoch																		Second epoch											
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	Mo	Tu	Wd	Tr	Fr	Sa	Mo	Tu	Wd	Tr	Fr	Sa	Mo	Tu	Wd	Tr	Fr	Sa	Mo	Tu	Wd	Tr	Fr	Sa	Mo	Tu	Wd	Tr	Fr	Sa
ALGA			x																		x									
Pg							x																x							
AM1												x																x		
FAE																	x													x
ACir														x											x					
POO								x																x						
AM2											x											x								
LSD																		x											x	
E1															x												x			
MAT					x															x										
PE													x													x				
ACp																	x													x
EA									x														x							
E2	x																		x											
SS						x															x									
RCp											x															x				
PICC/CPg															x													x		
PR			x																	x										
FT							x																	x						
SEAD1																		x											x	
ST														x													x			
RCom					x														x											
RI										x												x								
SE1													x													x				
AVE								x																x						
SCDig												x													x					
SOt																	x													x
PI												x											x							
SCDist															x														x	
EGP	x																			x										
OGE	x																			x										
SG	x																			x										

Table 5.11: Automatic solution for the LEETC examination timetable. The courses marked in bold face are shared with other programs. The number of clashes of this timetable is 238 and 550 for the first and second epochs, respectively. As can be observed, all first epoch examinations respect the minimum distance ($L_{min} = 10$) to the corresponding exam time slot in the second epoch.

Course	First epoch																		Second epoch											
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	Mo	Tu	Wd	Tr	Fr	Sa	Mo	Tu	Wd	Tr	Fr	Sa	Mo	Tu	Wd	Tr	Fr	Sa	Mo	Tu	Wd	Tr	Fr	Sa	Mo	Tu	Wd	Tr	Fr	Sa
ALGA															x															x
Pg	x																		x											
AM1					x																						x			
FAE								x																x						
ACir											x										x									
POO		x																		x										
AM2				x																						x				
LSD							x																		x					
E1													x										x							
MAT																	x												x	
PE										x												x								
ACp								x																x						
EA						x													x											
E2												x																x		
SS																x														x
RCp				x																		x								
PICC/CPg														x													x			
PR									x																				x	
FT	x																			x										
SEAD1											x														x					
ST			x																		x									
RCom																	x											x		
RI													x													x				
SE1										x												x								
AVE		x																		x										
SCDig					x																		x							
SOt							x																	x						
PI							x												x											
SCDist															x												x			
EGP												x																	x	
OGE												x																	x	
SG												x																	x	

Chapter 6

A Cellular Memetic Algorithm for Examination Timetabling

Contents

This chapter describes the developed approach based on the *cellular memetic algorithm* (cMA) for the *Examination Timetabling Problem* (ETP). The proposed memetic algorithm comprises the hybridisation of the *cellular evolutionary algorithm* (cEA) with the *threshold acceptance* (TA) algorithm. Section 6.1 describes the cMA structure. Section 6.2 describes the cMA components (chromosome representation, construction of initial solutions, crossover, mutation and local search operator) for the Toronto and the *Second International Timetabling Competition* (ITC 2007) benchmark sets, respectively. Section 6.3 is devoted to the experimental evaluation of cMA. Final remarks are given in Section 6.4.

6.1 Cellular Memetic Algorithm

The cellular evolutionary algorithm used to solve the ETP relies on the cellular model. In this model, the populations are organised in a special structure defined as a connected graph, in which each vertex is a solution that communicates with its neighbours Alba & Dorronsoro (2005). More specifically, individuals are set in a toroidal mesh and are only allowed to recombine with the closest neighbours (Figure 6.1). The population structure in cellular evolutionary algorithms, in addition to the inherent parallelism, promotes the population genetic diversity Alba & Dorronsoro (2005), thus alleviating the premature convergence observed in non-cellular evolutionary algorithms.

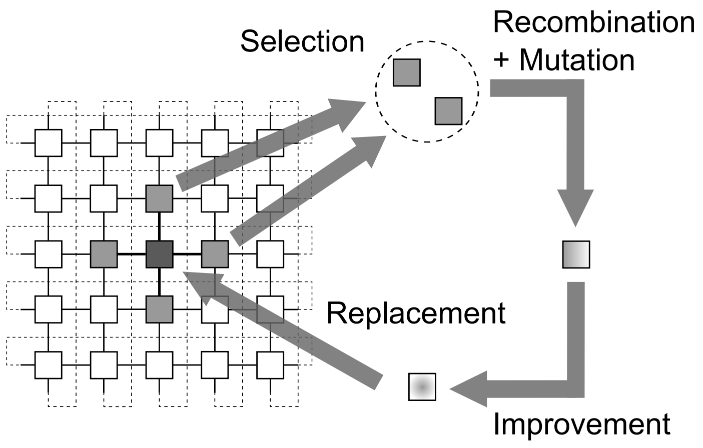


Figure 6.1: The cellular model for evolutionary algorithms. The figure illustrates the application of the L5 neighbourhood type, also known as the von Neumann neighbourhood.

Algorithm 8 describes the proposed cellular memetic algorithm. The cMA uses a tournament selection method of size two (binary tournament), as suggested in Alba &

Dorronsoro (2008). The binary tournament has the property that any member of the population, besides the absolute worst, has a chance of getting to the next generation, but better ones have a better chance. Thus, the binary tournament has a lower selective pressure compared to other types of tournament.

Algorithm 8 Pseudo-code of the canonical cMA.

```
 1: procedure CMA
 2:     Pop ← GenerateInitialPopulation
 3:     Evaluation(Pop)
 4:     while stopping condition is not true do
 5:         AuxPop ← ∅
 6:         for indiv = 1 to |Pop| do
 7:             neighs ← GetNeighbours(indiv, Pop)
 8:             parents ← Selection(neighs)
 9:             offspring ← Recombination(parents)
10:             offspring ← Mutation(offspring)
11:             offspring ← LocalSearch(offspring)
12:             Evaluation(offspring)
13:             NewIndiv ← Best(indiv, offspring, Pop)
14:             AuxPop ← AuxPop ∪ {NewIndiv}
15:         end for
16:         Pop ← AuxPop
17:     end while
18: end procedure
```

We now explain how the offspring are generated. For each generation (see Algorithm 8), we obtain the neighbours of the individual i ($i = 1, \ldots, \mu$, where μ is the population size). In our implementation, we use the L5 neighbourhood (comprising the immediate neighbours located in the north, south, west, and east positions). With this set of neighbours, we perform a binary tournament, i.e., we choose randomly two parents from the neighbourhood set and return the one with the best fitness as the first offspring. The second offspring is simply individual i. Then, we perform recombination (crossover) with probability P_c, mutation with probability P_m and local search with probability P_{ls} over these two offspring, and the best of the two is selected and compared with the original individual i. The best between the offspring and individual i is finally kept for the next generation.

There are two possible implementations of the cEA (Alba & Dorronsoro, 2008): *synchronous* and *asynchronous*. If the cycle is applied to all the individuals simultaneously, the cEA is said to be synchronous, since the individuals in the next generation are all created concurrently. Otherwise, the cEA is said to be asynchronous. In this work, the synchronous algorithm was implemented.

6.2 Application to Benchmark Sets

In this section the cMA components (chromosome representation, construction of initial solutions, crossover, mutation and local search operator) for the Toronto and ITC 2007 benchmark sets are presented. Most of cMA's components are common to the ones introduced in Chapter 4 (local search approaches) and in Chapter 5 (single-objective memetic approaches).

6.2.1 Toronto Data Set

Chromosome Representation and Fitness Function

The adopted chromosome representation for the Toronto benchmark set is described in Section 4.1.3.

Construction of Initial Feasible Timetables

The initial solution population for the Toronto benchmark set is constructed by applying an initialisation procedure, that is based on the *saturation degree* (SD) graph colouring heuristic, on each solution. The devised procedure is described in Section 4.1.3.

Neighbourhood Operator

The Kempe chain neighbourhood was used for the Toronto benchmark set, as described in Section 4.1.3.

Recombination and Mutation Operators

The proposed recombination (crossover) operator, used in the case of the Toronto benchmark set, is described in Section 5.2.1.

Concerning the mutation operator, a randomly selected exam is moved to a different feasible period using a Kempe chain-based neighbourhood operator, described in Section 4.1.3.

6.2.2 ITC 2007 Data Set

Excepting the variation operators, all cMA components for the ITC 2007 case are the same as the ones described previously in Chapter 4.

Chromosome Representation and Fitness Function

The adopted chromosome representation for the ITC 2007 benchmark set is presented in Section 4.1.4.

Construction of Initial Feasible Timetables

For the ITC 2007 case, the initial solution population is constructed also using an initialisation procedure based on the SD graph colouring heuristic. However, the construction algorithm is more complex than that presented earlier for the Toronto set, due to the increased complexity of the ITC 2007 hard constraints. The devised algorithm is similar to the algorithm described in Section 4.1.4 for the *fast simulated annealing* (FastSA). The difference relies in the algorithms's Phase 2:

Phase 1 (*The same steps as the FastSA's Phase 1 construction algorithm are executed.*)

Phase 2 If a selected exam cannot be assigned due to violations of hard constraints, the conflicting exams are unscheduled and the selected exam is scheduled. The conflicting exams are again inserted in the exam priority queue, with priority equal to *zero* (maximum priority). Then, we take the next exam from the queue and we restart Phase 2. The construction phase ends when all exams are scheduled.

The initialisation algorithm of cMA was developed before the one used in FastSA. In cMA, for simplicity, the priority of each unassigned exam in Phase 2 of the algorithm is set to zero. In the FastSA algorithm, the same priority is computed as the number of available periods in the timetable (SD heuristic). In addition, it is necessary to also update the priority of the other exams remaining in the queue, in order to take into account the exams removed from the timetable.

Recombination and Mutation Operators

Due to its implementation complexity, no recombination operator was implemented for the ITC 2007 benchmark set. Concerning the mutation operator, a randomly selected exam is moved to a different feasible period and room using the *Slot-Room* move, described in Section 4.1.4.

6.3 Experimental Results and Discussion

This section presents the experiments conducted to test the proposed method on the examination timetabling problem. Section 6.3.1 describes the parameter settings of the algorithm. Section 6.3.2 presents a comparison between cEA (cellular evolutionary algorithm without local search), TA (local search only), and cMA (hybrid algorithm). The cMA comparison with the state-of-the-art approaches is done in Sections 6.3.3 and 6.3.4 for the Toronto and ITC 2007 benchmark sets, respectively.

6.3.1 Settings

The algorithm was programmed in the C++ language and was based on the ParadisEO framework Cahon et al. (2004). The experiments were conducted on an Intel Core i7-2630QM (CPU @ 2.00 GHz with 8 GB RAM) PC running Ubuntu 14.04 LTS – 64 bit OS. The Linux kernel used was 3.13.0-49-generic. The compiler used was the GCC v. 4.8.2. In the experiments, all the statistical tests were performed with a 95% confidence level. For each algorithm configuration, ten executions were made on each Toronto and ITC 2007 problem instance. We assessed the statistical significance of our results using the Friedman test, as suggested in García & Herrera (2008). The results of the statistical tests were produced using the Java tool in García & Herrera (2008).

Table 6.1: Parameter settings. Legend: G – Grid size, N – Neighbourhood type, P_c – crossover probability, P_m – mutation probability, P_{ls} – local search probability, Q_{max} – initial threshold, r – decreasing rate, k – # iterations at each threshold, and Q_{min} – final threshold. The parameter values in the second and third rows that differ from the first row are underlined.

Data set	G	N	P_c	P_m	P_{ls}	TA Cooling schedule			
						Q_{max}	r	k	Q_{min}
Toronto	4×4	L5	0.4	0.1	0.1	0.1	$1{\times}10^{-3}$	5	$2{\times}10^{-5}$
						0.1	$\underline{1{\times}10^{-6}}$	5	$2{\times}10^{-5}$
ITC 2007	4×4	L5	$\underline{0}$	0.1	0.1	$\underline{10}$	$1{\times}10^{-3}$	5	$\underline{2{\times}10^{-4}}$

The parameter settings are specified in Table 6.1. Two different cooling schedules were used in the TA algorithm when solving the Toronto instances: a *light* cooling schedule, with a decreasing rate $r = 1{\times}10^{-3}$, and an *intensive* cooling schedule, with a decreasing rate $r = 1{\times}10^{-6}$. With the light cooling schedule, each TA execution computes 42 590 solution fitness evaluations, whereas 42 585 970 evaluations are computed by each TA execution with the intensive cooling schedule. The light cooling schedule was mostly used in the experiments presented in Section 6.3.2, whereas the intensive cooling schedule was used to generate the final timetables for the Toronto data set.

The parameter settings for the ITC 2007 instances were the same as the ones determined empirically for the Toronto set, with two differences: only the mutation operator was used (no crossover operator was implemented, as mentioned in Section 6.2.2) and the cooling schedule was specifically tuned for the ITC 2007 case (particularly, the threshold used, see Table 6.1). With this cooling schedule, a total of 54 100 evaluations were computed by each TA execution. All executions of cMA were time limited. For the Toronto benchmark set, the algorithm was stopped after 24 hours of computation for the smaller, and after 48 hours for the largest instances (car92, car91, uta92, and pur93). These limits were found to be adequate given our problem type.

For the ITC 2007 benchmark set, the algorithm was stopped after 276 seconds of computation. The ITC 2007 rules enforce a limit on the computation time of the candidate algorithms. This limit depends on the particular machine used and was determined by running a benchmarking tool on our machine (available from the ITC 2007 site).

All obtained solutions were validated using Qu et al.'s validator tool Qu et al. (2009) (for the Toronto benchmark set) and the ITC 2007's online validator tool (for the ITC 2007 benchmark set).

For each examination timetabling benchmark set, the source code, the resulting solution files for each instance/run, and the produced statistics, are publicly available in the following Git repositories:

- Toronto solver – `https://github.com/nunocsleite/cMA-ETP-Toronto`;
- ITC 2007 solver – `https://github.com/nunocsleite/cMA-ETP-ITC2007`.

cMA Parameter Values Selection Criterion

All parameter values were selected by following commonly used guidelines presented in the literature, and then by performing empirical tuning studies, taking into account a reasonable balance between solution quality and computation time. With respect to the cEA parameters *Grid size* $G = 4 \times 4$ and *neighbourhood pattern* N = L5, it is reported in Alba & Dorronsoro (2005) that the square and rectangular grids show better performance than narrow grids on some combinatorial optimisation problems, while the use of smaller local neighbourhoods, such as L5, promotes a lower selection intensity, thus increasing diversity in the population. Due to the relatively high computing cost for generating feasible solutions, especially in the case of the more constrained ITC 2007 data sets, we chose a relatively small population of 16 individuals. The use of local search served to intensify the search around promising regions.

Tested values for the mutation probability P_m and the crossover probability P_c were selected from the sets $\{0.01, 0.05, 0.1\}$ and $\{0.4, 0.6, 0.8\}$, respectively. It was found that $P_m = 0.1$ and $P_c = 0.4$ led to the best results. For the local search probability, a low value was chosen in order not to degrade diversity in the population and also to prevent the algorithm from slowing down too much.

The cooling schedule values were chosen empirically in order to initially accept a reasonable number of non-improving solutions and to have a reasonable exploitation intensity. The justification is given in the next section.

Cooling Schedule Selection

The impact of the local search cooling schedule on the optimisation is analysed in Section 4.2. It is known that the number of non-improving solutions is directly proportional to the number of accepted exam moves, as a percentage of the accepted exams are non-improving moves. Thus, we have selected an initial threshold value that allows for the great majority of exams to be moved several times, in order to better explore the search space. Graphically, we know that this effect is achieved by having the first threshold bin practically filled in Figures 4.4 and 4.5. After several experiments we have determined that the values $Q_{max} = 0.1$ and $Q_{max} = 10$ (see Table 6.1), for the Toronto and ITC 2007 benchmark sets, respectively, were reasonable values. The remaining parameters of the cooling schedule (r, k, and Q_{min}) were set in order to have two distinct rates (parameter r), one light and one intensive, a low number of iterations at a fixed threshold (parameter k) since the intensification is mainly controlled by the rate parameter, and a sufficiently low threshold (Q_{min} parameter), that is, a value from which non-improving moves are practically never accepted.

6.3.2 Comparison between cEA, TA, and cMA

This section presents three series of experiments, which are:

i. Study of the hybridisation in cMA. The cMA evolution is studied for two example instances of the Toronto data set.

ii. Comparison between cEA, TA, and cMA, for the Toronto and ITC 2007 benchmark sets.

iii. Sensitivity study. cMA and cEA using larger populations are studied for the complete Toronto data set.

Study of the Combined Effect of the cMA

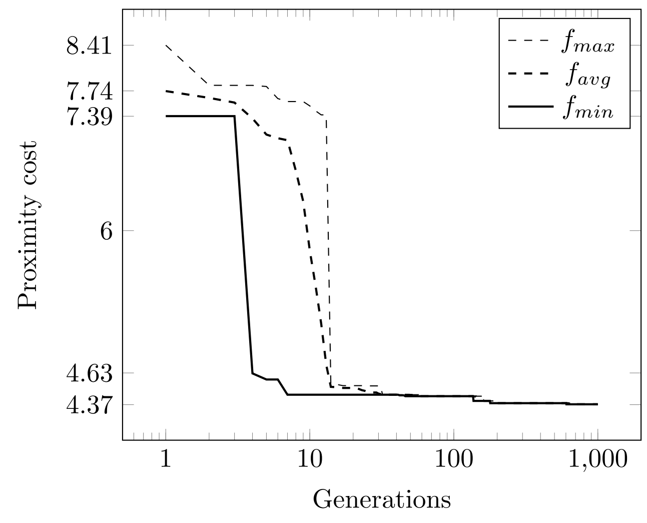
In this section, an investigation of the hybrid cMA is carried out. It is shown empirically that cMA improves over TA alone. Two Toronto instances were used to illustrate the operation of the hybrid algorithm: car92 (a large size instance) and yor83 (a small size instance). The *light* cooling schedule was used in the experiments. The other parameter values reported in Table 6.1 were kept. Figure 6.2 illustrates the evolution of the timeslot proximity cost (see Equation (2.3)) in the population, more precisely the maximum, average and minimum values. In this experiment, instead of limiting the computation time, the number of generations was set based on two criteria: (a) the number of generations should be larger than what is required for all individuals to reach the same baseline fitness; in this way, it is possible to observe the evolution from that point on, and (b) the number of generations should be set to a value that allows the memetic algorithm to reach a stationary phase. After preliminary tests, the number of generations was set to 1000.

As observed in Figure 6.2, the evolution of the minimum cost in the population improves with the number of generations. Due to the local search operator, a relatively good minimum cost is achieved in the first generations, but this cost is further improved due to the combined effect of crossover, mutation and local search. Indeed, crossover and mutation are able to produce different starting points in the solution space that are optimised by the local search operator. With the local search probability $P_{ls} = 0.1$, at least ten generations are needed in order for all the solutions to approach the same fitness baseline; from this point on, the algorithm is able to improve the best fitness in the population beyond this baseline, even if the population diversity is very low. The same conclusion was reached by running similar experiments on the other instances of the Toronto benchmark set.

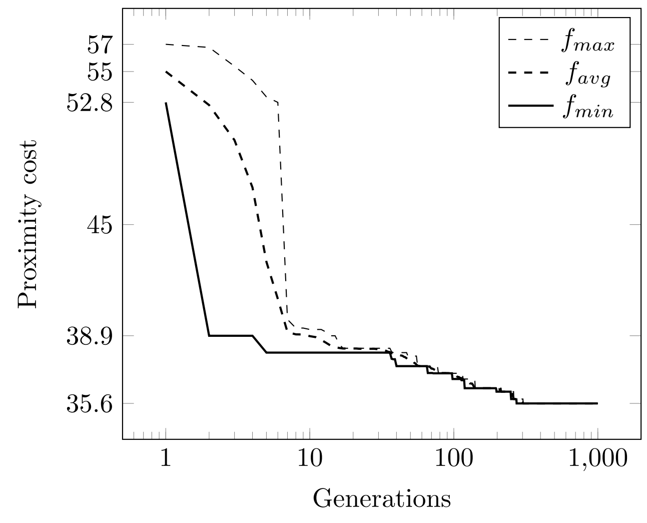
Comparison of cEA, TA, and cMA, on the Toronto and ITC 2007 Sets

In this section, the results of cEA, TA and cMA, for both the Toronto and ITC 2007 benchmark sets, are presented and compared. Statistical significance of the results is also presented. The settings used in the experiments are the same as those found in Table 6.1. The statistical analysis was carried out according to the methodology suggested in García & Herrera (2008). For the statistical analysis, the Friedman test was first applied, followed by the Holm and Hochberg's tests as post-hoc methods (if significant differences are detected) to obtain the adjusted p-values for each comparison between the control algorithm (the best-performing one) and the other algorithms.

Table 6.2 presents the results of cEA, TA, and cMA on the Toronto benchmark set. cEA and cMA were executed for a fixed time limit (Max time), determined empirically as the time required for the algorithm to reach a stationary state. Then f_{min} is the best solution value (minimum penalty) over ten executions, f_{avg} is the average and f_{max} the worst solution value, while σ is the standard deviation. Table A.4 in Appendix A sum-



(a)



(b)

Figure 6.2: Evolution of the population maximum, average and minimum proximity cost for (a) car92 and (b) yor83 instances using the *light* cooling schedule. The number of generations was set to 1000.

Table 6.2: Results of cEA, TA, and cMA on the Toronto benchmark set. The best values are shown in bold. The cEA and the cMA were run for a fixed time limit.

	cEA						TA				cMA				
Inst.	Max time	f_{min}	f_{max}	f_{avg}	σ	Avg. time (s)	f_{min}	f_{max}	f_{avg}	σ	Max time	f_{min}	f_{max}	f_{avg}	σ
car91	20 s	7.39	7.81	7.65	0.12	**9.00**	5.65	6.05	5.84	0.13	1 h 10 min	**5.31**	**5.57**	**5.46**	0.07
car92	20 s	6.00	6.62	6.27	0.21	**6.80**	4.57	4.81	4.65	0.07	1 h 10 min	**4.27**	**4.45**	**4.37**	0.05
ear83	10 s	44.17	49.08	46.27	1.50	**1.60**	35.25	38.71	37.61	1.00	35 min	**33.21**	**34.46**	**33.81**	0.38
hec92	3 s	12.42	14.03	13.34	0.51	**0.60**	10.27	11.38	10.91	0.27	20 min	**10.11**	**10.37**	**10.20**	0.09
kfu93	20 s	19.87	21.05	20.64	0.42	**2.00**	13.79	14.97	14.40	0.37	40 min	**13.34**	**13.54**	**13.42**	0.06
lse91	10 s	15.14	18.19	16.65	0.83	**1.90**	10.63	12.06	11.45	0.38	25 min	**10.22**	**10.82**	**10.45**	0.22
pur93	3 min	8.88	9.11	8.97	0.06	**32.50**	6.38	6.58	6.46	0.06	5 h	**6.17**	**6.30**	**6.24**	0.05
rye92	30 s	14.60	16.07	15.62	0.50	**3.60**	8.95	9.59	9.25	0.21	30 min	**8.65**	**8.79**	**8.72**	0.05
sta83	5 s	158.12	161.34	159.42	1.14	**0.50**	157.03	157.43	157.16	0.11	1 min 20 s	**157.03**	**157.03**	**157.03**	0
tre92	10 s	10.48	11.22	10.94	0.23	**2.20**	8.70	9.21	8.89	0.16	30 min	**8.30**	**8.43**	**8.36**	0.04
uta92	30 s	4.78	4.98	4.87	0.08	**8.10**	3.69	3.88	3.78	0.07	1 h 15 min	**3.59**	**3.70**	**3.64**	0.03
ute92	10 s	32.50	34.49	33.44	0.62	**1.10**	25.01	26.24	25.41	0.45	17 min	**24.84**	**24.93**	**24.87**	0.03
yor83	10 s	43.93	46.09	45.35	0.60	**2.00**	37.46	40.02	39.08	0.78	30 min	**35.49**	**37.33**	**36.38**	0.49

marises the ranks obtained by the Friedman test. The p-value computed by the Friedman test is 2.26×10^{-6}, which is below the significance interval of 95 % ($\alpha = 0.05$). This value indicates that there is a significant difference among the observed results. Table A.5 shows the adjusted p-values for the post-hoc Holm and Hochberg's tests on the cMA algorithm, revealing significant differences when using cMA as the control algorithm. Both procedures confirm that cMA is better than TA and cEA with $\alpha = 0.05$.

Table 6.3 presents the results of cMA on the Toronto benchmark set using the light and intensive cooling schedules. Table A.6 summarises the ranks obtained by the Friedman test, with a p-value of 8.74×10^{-4} (which is below the significance interval of 95%).

Table A.7 shows the adjusted p-values obtained by the post-hoc Holm and Hochberg's tests on the cMA-intensive algorithm. These procedures reveal significant differences when using cMA-intensive as the control algorithm, confirming that cMA-intensive is better than cMA-light with $\alpha = 0.05$.

Table 6.4 presents the results of cEA, TA, and cMA on the ITC 2007 benchmark set. cEA and cMA were executed for a fixed time limit which corresponds to the ITC 2007 time limit. Table A.8 summarises the ranks obtained with the Friedman test. The p-value computed by the Friedman test is 8.84×10^{-5}, which is below the significance interval of 95% ($\alpha = 0.05$). Table A.9 shows the adjusted p-values obtained by the post-hoc Holm and Hochberg's tests on the cMA algorithm. Holm and Hochberg's procedures reveal significant differences when using cMA as the control algorithm, confirming that cMA is better than TA and cEA with $\alpha = 0.05$.

Parameter Sensitivity Study

In this section, two experiments were conducted with the objective of testing different parameter sets. In the first experiment, the performance of cMA with a larger population (8×8 instead of 4×4), is studied for the Toronto data set. This experiment complements the study undertaken earlier in this chapter, regarding the hybrid cMA. In the second experiment, the performance of cEA is studied, also with a larger population (50×50). Both studies are done using the Toronto data set.

In Table 6.5, results are reported for cMA using a squared cell grid of dimension 8×8 to structure the population. The *light* cooling schedule was used, while the other parameters remained the same. The execution time limit was fixed a priori for each instance, as specified in Table 6.5. Based on preliminary experiments, when the specified time limits are reached, the algorithm stagnates and only a few improvements occur from this point on.

In Table 6.6, the cEA results are reported, using a 50×50 squared cell grid in order

Table 6.3: Comparison of two cMA approaches on the Toronto benchmark set using a *light* and an *intensive* cooling schedule. The best results are shown in bold.

	Light cooling schedule					Intensive cooling schedule				
Inst.	Max time	f_{min}	f_{max}	f_{avg}	σ	Max time	f_{min}	f_{max}	f_{avg}	σ
car91	**1 h 10 min**	5.31	5.57	5.46	0.07	48 h	**4.31**	**4.42**	**4.39**	0.03
car92	**1 h 10 min**	4.27	4.45	4.37	0.05	48 h	**3.68**	**3.75**	**3.72**	0.02
ear83	**35 min**	33.21	34.46	33.81	0.38	24 h	**32.48**	**32.76**	**32.61**	0.08
hec92	**20 min**	10.11	10.37	10.20	0.09	24 h	**10.03**	**10.07**	**10.05**	0.01
kfu93	**40 min**	13.34	13.54	13.42	0.06	24 h	**12.81**	**12.85**	**12.83**	0.01
lse91	**25 min**	10.22	10.82	10.45	0.22	24 h	**9.78**	**9.84**	**9.81**	0.02
pur93	**5 h**	6.17	6.30	6.24	0.05	48 h	**4.14**	**4.21**	**4.18**	0.02
rye92	**30 min**	8.65	8.79	8.72	0.05	24 h	**7.89**	**7.97**	**7.93**	0.03
sta83	**1 min 20 s**	**157.03**	**157.03**	**157.03**	0	24 h	**157.03**	**157.03**	**157.03**	0
tre92	**30 min**	8.30	8.43	8.36	0.04	24 h	**7.66**	**7.75**	**7.70**	0.03
uta92	**1 h 15 min**	3.59	3.70	3.64	0.03	48 h	**3.01**	**3.05**	**3.04**	0.01
ute92	**17 min**	24.84	24.93	24.87	0.03	24 h	**24.80**	**24.85**	**24.83**	0.02
yor83	**30 min**	35.49	37.33	36.38	0.49	24 h	**34.45**	**34.74**	**34.63**	0.08

Table 6.4: Results of cEA, TA, and cMA on the ITC 2007 benchmark set. The best values are shown in bold. The time limit for cEA and cMA corresponds to the ITC 2007 time limit.

	cEA						TA				cMA				
Inst.	Max time	f_{min}	f_{max}	f_{avg}	σ	Avg. time (s)	f_{min}	f_{max}	f_{avg}	σ	Max time	f_{min}	f_{max}	f_{avg}	σ
1		7996	8698	8320.90	209.74	**1.30**	8234	8912	8663.40	228.41		**6207**	**6617**	**6478.20**	121.97
2		811	1139	942.10	90.05	**1.10**	814	1014	908.40	76.04		**535**	**604**	**572.90**	17.66
3		17 281	19 139	18 238.10	575.60	**1.90**	17 778	21 186	19 117.00	1184.32		**13 022**	**14 180**	**13 680.50**	423.68
4		17 092	19 856	18 619.80	991.90	**1.80**	16 653	21 115	18 409.40	1536.55		**14 302**	**16 423**	**15 493.70**	673.28
5		5853	10 498	6913.60	1332.09	**2.40**	5712	7955	6644.30	773.44		**3829**	**4351**	**4155.60**	170.25
6	276 s	27 905	29 665	28 539.50	673.78	**0.90**	28 245	29 905	29 055.00	489.26	276 s	**26 710**	**27 230**	**26 873.00**	183.38
7		10 086	10 960	10 664.20	299.37	**1.40**	8287	8938	8611.90	231.15		**5508**	**6064**	**5844.40**	178.62
8		11 519	11 974	11 711.80	138.98	**0.80**	11 470	12 071	11 791.40	185.99		**8716**	**9106**	**8942.30**	113.00
9		1338	1552	1395.30	66.31	**0.40**	1369	1608	1459.80	90.54		**1030**	**1150**	**1080.30**	42.72
10		15 678	19 793	16 562.00	1241.51	**0.20**	15 806	22 646	18 842.70	2641.72		**13 894**	**14 625**	**14 208.70**	211.16
11		48 931	57 581	52 332.50	2843.47	**3.00**	51 649	59 939	56 152.10	3141.53		**39 783**	**48 061**	**43 585.80**	2779.85
12		5484	5932	5668.20	150.37	**0.50**	5450	5899	5660.50	153.97		**5142**	**5360**	**5249.00**	81.08

Table 6.5: cMA results on the Toronto benchmark set using a squared cell grid of dimension 8×8 to structure the population. The crossover, mutation and local search probabilities were set to 0.4, 0.1, and 0.1, respectively; the *light* cooling schedule was used. In the last row, the *total penalty* – TP (sum of individual costs) for the complete set of instances is indicated.

	Initial solution (f_{min})		Optimised solution			
Instance	Cost	Time limit	f_{min}	f_{avg}	σ	Time limit
car91	5.58	4 min.	5.30	5.36	0.04	7h00
car92	4.51	3 min.	4.30	4.32	0.02	7h00
ear83	36.74	1 min.	32.82	33.14	0.28	3h00
hec92	10.52	10 sec.	10.06	10.12	0.05	30 min.
kfu93	13.73	35 sec.	13.22	13.32	0.07	1h30
lse91	11.12	1 min.	10.22	10.32	0.07	2h00
pur93	6.27	10 min.	6.11	6.17	0.04	5h30
rye92	8.81	1 min.	8.54	8.59	0.04	2h00
sta83	157.03	10 sec.	157.03	157.03	0.00	5 min.
tre92	8.53	40 sec.	8.19	8.24	0.04	2h00
uta92	3.70	2 min.	3.58	3.62	0.02	3h00
ute92	25.07	15 sec.	24.81	24.85	0.02	1h30
yor83	37.23	20 sec.	34.85	35.32	0.47	5h30
TP	–	–	319.03	320.42	–	–

to cope with the absence of local search, and to guarantee a reasonable diversity in the population. cEA was able to attain results comparable to the cMA's initial solution in Table 6.5, but required a much longer time. A similar behaviour was verified experimentally on the ITC 2007 benchmark set.

Discussion

In this subsection, a discussion of the results presented in the previous three subsections is carried out. From the experimental results reported previously, the following key conclusions are drawn:

- *algorithm's components comparison* – from Table 6.2 (cEA vs. TA vs. cMA on the Toronto set) we can observe that cEA obtains poor results when compared to TA and cMA; this behaviour can be attributed to the use of a small population and to the power of the variation operators (crossover and mutation). With a small population, cEA tends to stagnate faster.

Table 6.6: cEA results on the Toronto benchmark set using a squared cell grid of dimension 50 × 50 to structure the population. The other cEA parameters from Table 6.1 (N, P_c, P_m) were kept unchanged. In the last row, the *total penalty* – TP (sum of individual costs) for the complete set of instances is indicated.

Instance	f_{min}	f_{avg}	σ	Time limit
car91	5.38	5.53	0.08	2 h 30 min
car92	4.29	4.46	0.08	2 h 30 min
ear83	35.07	36.26	0.72	10 min
hec92	10.49	10.87	0.20	2 min
kfu93	13.67	14.03	0.28	40 min
lse91	10.83	11.21	0.29	30 min
pur93	6.09	6.33	0.16	4 h
rye92	8.93	9.21	0.22	50 min
sta83	157.03	157.08	0.04	2 min
tre92	8.55	8.76	0.12	30 min
uta92	3.57	3.65	0.06	2 h
ute92	25.08	25.43	0.21	5 min
yor83	37.37	38.40	0.56	10 min
TP	326.35	331.22	–	–

From Table 6.4 (cEA vs. TA vs. cMA on the ITC 2007 set) we can observe that cEA obtains results that are competitive with TA but inferior to cMA; cEA did not stagnate as fast on the ITC 2007 set as it did on the Toronto set due to the power of the mutation operator and the nature of the ITC 2007 instances themselves. Even if both mutation operators are based on the Kempe chain neighbourhood, they are different and the operator used in the ITC 2007 is able to explore more efficiently the solution space. This may be due to the variability of the ITC 2007 operator which moves both exams and rooms.

From the analysis made earlier in this section, cMA is better than the cEA and the TA algorithms for both the Toronto and ITC 2007 benchmark sets. Furthermore, the reported results are statistically significant;

- *light vs. intensive cooling schedule* – Table 6.3 shows that for some instances, e.g., hec92 and yor83 from the Toronto set, the light cooling schedule has a very satisfactory performance. For other instances, however, e.g., car91, car92, pur93, or uta92, a more intensive cooling schedule is needed in order to reach the best known solutions. This choice is justified by the analysis made in Section 4.2. We also demonstrate that the use of the intensive cooling schedule provides improvements over the light cooling schedule. The reported results are statistically significant;

- *use of larger population size* – when comparing cEA (4 × 4) in Table 6.2 with cEA (50 × 50) in Table 6.6, we can observe that the latter obtains better results (at the expense of more computation time). The same happens when we compare cMA (4 × 4) in Table 6.2 with cMA (8 × 8) in Table 6.5. Thus, the use of a larger population is beneficial. In the case of cEA, a large number of solutions needs to be maintained in order to have satisfactory results. This is problematic for the ITC 2007 set, due to the large time required to construct a single solution. Hence, cEA is not of practical use on this benchmark set.

6.3.3 Comparison with State-of-the-Art Approaches – Toronto Data Set

In this section, results for the complete Toronto benchmark set are presented and analysed. The parameter values are those shown in Table 6.1, and the *intensive* cooling schedule was used. The existing solutions that are compared against our approach include:

Car96 (Carter et al. (1996)) The authors propose several graph heuristics with clique initialisation and backtracking.

Yan05 (Yang & Petrovic (2005)) A *hyperheuristic* (HH) coupled with *case-based reasoning* is used to choose graph heuristics for constructing a feasible initial solution. A GD algorithm is then employed to improve the solution. HH are generic optimisation methods that explore the search space of heuristics instead of searching for direct solutions.

Ele07 (Eley (2006)) Ant Colony algorithm Dorigo & Stützle (2010).

Car08 (Caramia, Dell'Olmo & Italiano (2008)) Local search-based approach.

Bur08 (Burke & Bykov (2008)) Hill-climbing with late-acceptance strategy (see Section 2.3).

Bur10 (Burke et al. (2010)) Hybrid variable neighbourhood (see Section 2.3).

Pil10 (Pillay & Banzhaf (2010)) Genetic algorithm (see Section 2.3).

Dem12 (Demeester et al. (2012)) A HH is used to solve the Toronto and ITC 2007 benchmark sets, as well as a problem instance from KAHO Sint-Lieven (Ghent, Belgium).

Abd13 (Abdullah & Alzaqebah (2013)) Hybrid bee algorithm.

Lei14 (Leite et al. (2016a)) This is our previous algorithm (Section 5.3). Both cMA and Lei14 are memetic algorithms Neri et al. (2012). The algorithms have in common the use of structured populations: in the Lei14 approach, the *Shuffled Complex Evolution* (SCE) algorithm is used to organise the population into *complexes* Leite et al. (2016a) which are sub-populations or islands where the local search is carried out. After the local search step is executed in each complex (island), the solutions are redistributed among the complexes in order to propagate the best solutions to the other complexes, implementing an exploration step. The local search used in the Lei14 algorithm is a variation of the *great deluge* (GD) algorithm. The cEA also uses a structured population in which each cell corresponds to a complex. Each cell is only allowed to communicate with close neighbours, whereas the complexes in the SCE are groups of solutions that only communicate within that complex. The two algorithms adopt the decentralised model Alba & Dorronsoro (2008).

Fon15 (Fong, Asmuni & McCollum (2015)) A hybrid swarm-based approach to university timetabling is proposed. Both the examination timetabling (tested on the Toronto benchmark set) and the course timetabling (tested on the Socha benchmark set) problems are addressed.

Alz15 (Alzaqebah & Abdullah (2015)) In this study, a hybrid *bee colony optimisation* (BCO) algorithm is introduced. Two hybridisations are presented: (1) BCO hybridised with *hill-climbing* using the *late acceptance* strategy, (2) BCO hybridised with SA. It is shown that the first hybrid attains the best results on the tested benchmarks (Toronto and ITC 2007).

The above algorithms were chosen using the following criteria: (i) best fitness value on some instances, (ii) solution validation, and (iii) approach heterogeneity. Pillay and Banzhaf (identified as Pil10) were the only authors to publish the solutions referred to in their paper. The results published by the authors Car96, Yan05, and Ele07, were validated in Qu et al. (2009). Table 6.7 presents a comparison between cMA and the selection of the best algorithms from the literature as listed above. In the columns that refer to the analysed algorithms, the minimum solution value is shown for each considered algorithm. For cMA, the minimum, average and standard deviation are shown over ten runs. In the last two rows of Table 6.7, the *total penalty* (TP) and the *total penalty excluding the pur93 and rye92 instances* (TP (11)) are presented. For each approach, the TP and TP (11) values are computed by summing up the individual costs obtained by the algorithm on all instances (TP) and on 11 instances (TP (11)).

Table 6.7: Results of cMA and comparison with a selection of the best algorithms from the literature. The best results are shown in bold. "–" indicates that the corresponding instance was not tested or no feasible solution was obtained.

Instance	Car96	Yan05	Ele07	Car08	Bur08	Bur10	Pil10	Dem12	Abd13	Lei14	Fon15	Alz15	**cMA**		
	f_{min}												f_{min}	f_{avg}	σ
car91	7.10	4.50	5.20	6.60	4.58	4.90	4.92	4.52	4.76	4.41	4.79	4.38	**4.31**	4.39	0.03
car92	6.20	3.93	4.30	6.00	3.81	4.10	4.22	3.78	3.94	3.75	3.89	3.88	**3.68**	3.72	0.02
ear83	36.40	33.71	36.80	**29.30**	32.65	33.20	35.87	32.49	33.61	32.62	33.43	33.34	32.48	32.61	0.08
hec92	10.80	10.83	11.10	**9.20**	10.06	10.30	11.50	10.03	10.56	10.03	10.49	10.39	10.03	10.05	0.01
kfu93	14.00	13.82	14.50	13.80	**12.81**	13.20	14.37	12.90	13.44	12.88	13.72	13.23	**12.81**	12.83	0.01
lse91	10.50	10.35	11.30	**9.60**	9.86	10.40	10.89	10.04	10.87	9.85	10.29	10.52	9.78	9.81	0.02
pur93	3.90	–	4.60	**3.70**	4.53	–	4.65	5.67	–	4.10	–	–	4.14	4.18	0.02
rye92	7.30	8.53	9.80	**6.80**	7.93	–	9.30	8.05	8.81	7.98	–	8.92	7.89	7.93	0.03
sta83	161.50	158.35	157.30	158.20	157.03	**156.90**	157.81	157.03	157.09	157.03	157.07	157.06	157.03	157.03	0.00
tre92	9.60	7.92	8.60	9.40	7.72	8.30	8.38	7.69	7.94	7.75	7.86	7.89	**7.66**	7.70	0.03
uta92	3.50	3.14	3.50	3.50	3.16	3.30	3.35	3.13	3.27	3.08	3.10	3.13	**3.01**	3.04	0.01
ute92	25.80	25.39	26.40	**24.40**	24.79	24.90	27.24	24.77	25.36	24.78	25.33	25.12	24.80	24.83	0.02
yor83	41.70	36.53	39.40	36.20	34.78	36.30	39.33	34.64	35.74	**34.44**	36.12	35.49	34.45	34.63	0.08
TP (11)	327.10	308.47	318.40	306.20	301.25	305.80	317.88	301.02	306.58	300.62	306.09	304.43	**300.04**	300.65	–
TP	338.30	–	332.80	316.70	313.71	–	331.83	314.74	–	312.70	–	–	**312.07**	312.76	–

From Table 6.7, one can verify that cMA attains competitive costs on the Toronto benchmark set. It attains the lowest TP and TP (11) values, proving the cMA efficiency.

Average Relative Deviation to the Best Solution – Cost and Time Analysis

In this section, the average relative deviation of cMA to the best known solution is studied. The time performance of the analysed algorithms is also detailed.

Table 6.8 presents the previous best known solutions and the cost of the best solutions obtained by cMA. Table 6.9 presents the algorithms' rankings based on their average and best performance, for different *number of instances solved* (NIS). NIS is the number of instances for which a feasible and optimised solution was obtained by all algorithms that are compared.

The rankings are computed as in Demeester et al. (2012). The algorithm that produces the best solution on a given instance gets the lowest value, while the worst algorithm gets the highest value. The same procedure is applied to the average values obtained after 10 runs on each instance. The final overall ranking is based on the average of the rankings over all instances. The algorithm with the overall lowest rank can be considered the best performing algorithm. cMA is in first position in all rankings shown in Table 6.9.

For each algorithm, we calculate the relative deviation $\mathrm{RD} = 100{\cdot}(\mathrm{MSF}-\mathrm{BKS})/\mathrm{BKS}$ for each instance, where BKS is the *best known solution* and MSF is the *minimum solution found*. BKS values were extracted from Table 6.8. ARD denotes the average relative deviation over all instances.

Table 6.10 shows the NIS and the ARD for the cost and computation time for all algorithms that are compared in Table 6.7. In order to present the information in a concise way, we have included the highest NIS value available for each compared approach (except cMA); for comparing cMA with the other techniques, we have included NIS values equal to 11, 12, and 13. The running times of these algorithms are presented in Table B.1 in Appendix B. The two rightmost columns (cMA improvement) present the ARD reduction in cost and time obtained by cMA with respect to each competitor.

As reported in Table 6.10, the ARD for cMA is only 3.95% for the complete Toronto benchmark set. We can also see under cMA improvement that the ARD reduction is significant, yielding a large improvement in solution quality with respect to all other algorithms. However, this improvement comes at a computational cost, as indicated by the negative values in Table 6.10.

Table 6.8: Previous best known solutions with corresponding authors and comparison with cMA's best solutions. Values in bold are equal or improved solutions produced by cMA.

Inst.	Previous best known solution	Authors	**cMA**	
			f_{min}	Integer cost
car91	4.38	Alzaqebah and Abdullah, 2015	**4.311 078**	72 965
car92	3.75	Leite et al., 2014	**3.682 556**	67 829
ear83	29.30	Caramia et al., 2008	32.483 556	36 544
hec92	9.20	Caramia et al., 2008	10.033 652	28 325
kfu93	12.81	Burke and Bykov, 2008	**12.806 693**	68 503
lse91	9.60	Caramia et al., 2008	9.782 832	26 668
pur93	3.70	Caramia et al., 2008	4.144 594	124 458
rye92	6.80	Caramia et al., 2008	7.887 834	90 576
sta83	156.90	Burke et al., 2010	157.032 733	95 947
tre92	7.69	Demeester et al., 2012	**7.657 339**	33 386
uta92	3.08	Leite et al., 2014	**3.013 214**	64 079
ute92	24.40	Caramia et al., 2008	24.803 929	68 186
yor83	34.44	Leite et al., 2014	34.454 835	32 422

Statistical Analysis

We now present a statistical analysis of the obtained average results for the Toronto benchmark set with NIS $=$ 11, which involves a larger number of algorithms. Table A.10 summarises the ranks obtained by the Friedman test for this case. The p-value computed with the Friedman test is 6.71×10^{-11}, which is below the significance interval of 95% ($\alpha = 0.05$), confirming that there is a significant difference among the observed results and that cMA is the best performing algorithm. Table A.11 shows the adjusted p-values obtained by the post-hoc Holm and Hochberg's tests when considering cMA as the control algorithm. Holm and Hochberg's tests indicate that cMA is better than all algorithms except Bur08, Dem12, and Lei14, with $\alpha = 0.05$.

Discussion

The algorithms' ARD (Table 6.10) and rankings (Tables 6.9 and A.10) provide two different points of view for the tested algorithms. We can see that cMA is the best performing algorithm, but also the slowest. Lei14 is in second position. Regarding the analysis with NIS $=$ 11, Tables 6.9 and 6.10 give the same ordering for all algorithms (cMA, Fon15 and Bur10). When considering NIS $=$ 13, the relative order of Bur08 and Dem12 is reversed in the rankings table. Also, Car96 appears before Pil10 and Ele07 in Table 6.9, but

Table 6.9: Ranking of the analysed algorithms according to their average and best performance on the complete Toronto data set (NIS = 13) and on the Toronto data set but excluding the pur93 and rye92 instances (NIS = 11). The lower the rank, the better the algorithm's performance.

NIS							
11				13			
Min		Avg		Min		Avg	
Alg.	Rk	Alg.	Rk	Alg.	Rk	Alg.	Rk
cMA	2.4	cMA	1.3	cMA	2.5	cMA	1.3
Lei14	2.9	Lei14	2.1	Lei14	2.9	Lei14	2.1
Dem12	3.3	Dem12	3.2	Dem12	3.5	Dem12	3.4
Bur08	4.1	Bur08	3.7	Bur08	3.7	Bur08	3.5
Alz15	6.0	Alz15	5.1	Car08	4.0	Pil10	5.2
Fon15	6.7	Fon15	6.2	Car96	6.2	Ele07	5.5
Bur10	7.0	Abd13	6.6	Pil10	6.3		
Car08	7.4	Pil10	8.1	Ele07	6.8		
Abd13	8.2	Ele07	8.7				
Yan05	8.5						
Pil10	11.1						
Car96	11.7						
Ele07	11.7						

is last in Table 6.10. Dem12 has a slightly better rank than Bur08 (Dem12 obtains eight better results compared to Bur08), but Bur08 has a lower TP penalty and also a lower ARD. Car96 has also two good results on instances pur93 and rye92 which improve its ranking. Regarding computation times (see columns 'ARD (%) – Time' and 'cMA improvement – Time' in Table 6.10), it is worth mentioning that computation times were extracted from the authors' papers and correspond to different programming languages and hardware, with unknown levels of optimisation. We can observe that the evolutionary approaches (cMA, Lei14, Fon15, Alz15, Pil10, and Ele07) are among the slowest algorithms, although some of them attain top positions with regard to solution quality (cMA – 1st, Lei14 – 2nd, Fon15 – 4th, Alz15 – 5th). Local search and graph based algorithms (Bur08, Car08, and Car96) are the fastest algorithms. The fastest one, Car96, has a time ARD of 217.01% (about four times slower than the best obtained times, on average), but also has the largest cost ARD; cMA, on the other hand has a time ARD of 432 002.24% (about 8640.44 times slower than the best obtained times, on average), but also has the smallest cost ARD. cMA takes more time to execute but this additional computation time warrants the improvement in the results by providing the lowest cost ARD. The extra

Table 6.10: Cost and time average relative deviations.

Algorithm	NIS	ARD (%)		cMA improvement	
		Cost	Time	Cost	Time
Lei14	13	4.69	384 773.99	0.74	−47 248.25
Bur08	13	6.21	2487.30	2.26	−429 534.94
Fon15	11	6.70	83 749.29	4.56	−422 999.94
Alz15	12	7.58	8279.84	4.28	−457 824.88
Bur10	11	7.93	7738.26	5.79	−499 010.97
Dem12	13	8.54	138 450.84	4.58	−293 571.40
Yan05	12	8.66	5354.06	5.36	−460 750.66
Abd13	12	9.57	–	6.28	–
Car08	13	12.99	638.92	9.03	−431 383.32
Pil10	13	16.26	6720.17	12.31	−425 302.07
Ele07	13	18.04	30 635.71	14.08	−401 386.53
Car96	13	21.35	217.01	17.40	−431 805.23
	13	3.95	432 022.24	–	–
cMA	12	3.29	466 104.72	–	–
	11	2.14	506 749.23	–	–

computation time is attributed to both the use of a population based algorithm and an intensive local search cooling schedule (the effect of using the intensive cooling schedule is analysed in Section 4.2). In general, single solution metaheuristics such as simulated annealing, threshold acceptance, tabu search, etc., are faster than population-based algorithms. The improvement in the results is attributed to the simultaneous use of a hybrid algorithm (cMA) and the use of an intensive cooling schedule in the threshold acceptance algorithm.

6.3.4 Comparison with State-of-the-Art Methods – ITC 2007 Data Set

Table 6.11 presents the ITC 2007 results of the five finalists. The competition winner was Müller (2009). Table 6.12 presents the comparison of cMA with state-of-the-art approaches, while fulfilling the ITC 2007 rules. The reported comparison includes the recent approaches of Col09 (McCollum et al., 2009), Dem12 (Demeester et al., 2012), Gog12 (Gogos et al., 2012), Alz14 (Alzaqebah & Abdullah, 2014), and Alz15 (Alzaqebah & Abdullah, 2015). cMA is able to obtain competitive results compared to the other algorithms. We can observe that cMA attains the best performance on the smaller instances (instances 4, 9, 10 and 12). More iterations are needed for the other instances, as

Table 6.11: ITC 2007 results. The best solutions are indicated in bold. "–" indicates that the corresponding instance was not tested or a feasible solution was not obtained. The column 'f_{min}' indicates the minimum penalty value per instance obtained among the compared algorithms.

Inst.	Mul08	Gog07	Ats07	Sme07	Pil07	f_{min}
1	**4370**	5905	8006	6670	12 035	4370
2	**400**	1008	3470	623	2886	400
3	**10 049**	13 771	17 669	–	15 917	10 049
4	**18 141**	18 674	22 559	–	23 582	18 141
5	**2988**	4139	4638	3847	6860	2988
6	**26 585**	27 640	29 155	27 815	32 250	26 585
7	**4213**	6572	10 473	5420	17 666	4213
8	**7742**	10 521	14 317	–	15 592	7742
9	**1030**	1159	1737	1288	2055	1030
10	16 682	–	15 085	**14 778**	17 724	14 778
11	**34 129**	43 888	–	–	40 535	34 129
12	5535	–	**5264**	–	6310	5264

well as a more intensive cooling schedule, thus requiring a longer execution time.

Average Relative Deviation to the Best Solution and Statistical Analysis

In this section, a further study of the cMA performance is carried out. Namely, the average relative deviation to the best known solution is studied, followed by the statistical analysis of the average case. No runtime analysis is done for the ITC 2007 case because all algorithms are run under the same time conditions, as specified by the ITC 2007 rules (see Appendix B).

Table 6.13 reports the BKS for the analysed algorithms, which is compared with the cMA results.

Table 6.14 presents the algorithms' ranking for different NIS. The ranking procedure is the same as the one used in Section 6.3.3. Table 6.15 reports the average relative deviation to the BKS for the analysed algorithms.

We now present a statistical analysis of the obtained average results for the ITC 2007 benchmark set with NIS $= 7$, which involves a larger number of algorithms. Table A.12 summarises the ranks obtained by the Friedman test in this case. The p-value computed with the Friedman test is 6.13×10^{-5}, which is below the significance interval of 95% ($\alpha = 0.05$), confirming that there is a significant difference among the observed results. Gog12 is the best performing algorithm. Table A.13 shows the adjusted p-values obtained by the post-hoc Holm and Hochberg's tests considering Gog12 as the control algorithm.

Table 6.12: Comparison of the best results obtained by cMA with the best solutions from the literature for the ITC 2007 benchmark set. The best solutions are indicated in bold. "–" indicates that the corresponding instance was not tested or a feasible solution was not obtained.

Inst.	ITC 2007 best	Col09	Dem12	Gog12	Alz14	Alz15	**cMA**		
	f_{min}						f_{min}	f_{avg}	σ
1	**4370**	4633	6060	4775	5328	5154	6207	6478.20	121.97
2	400	405	515	**385**	512	420	535	572.90	17.66
3	10 049	9064	23 580	**8996**	10 178	10 182	13 022	13 680.50	423.68
4	18 141	15 663	–	16 204	16 465	15 716	**14 302**	15 493.70	673.28
5	2988	3042	4855	**2929**	3624	3350	3829	4155.60	170.25
6	26 585	25 880	27 605	**25 740**	26 240	26 160	26 710	26 873.00	183.38
7	4213	**4037**	6065	4087	4562	4271	5508	5844.40	178.62
8	7742	**7461**	9038	7777	8043	7922	8716	8942.30	113.00
9	**1030**	1071	1184	–	–	–	**1030**	1080.30	42.72
10	14 778	14 374	15 561	–	–	–	**13 894**	14 208.70	211.16
11	34 129	**29 180**	–	–	–	–	39 783	43 693.56	2926.24
12	5264	5693	5483	–	–	–	**5142**	5249.00	81.08

Table 6.13: Previous best known solutions with corresponding authors and comparison with cMA's best solutions. Values in bold represent equal or new best solutions.

Inst.	Previous best known solution	Authors	**cMA** f_{min}
1	4370	Müller, 2009	6207
2	385	Gogos et al., 2012	535
3	8996	Gogos et al., 2012	13 022
4	15 663	McCollum et al., 2009	**14 302**
5	2929	Gogos et al., 2012	3829
6	25 740	Gogos et al., 2012	26 710
7	4037	McCollum et al., 2009	5508
8	7461	McCollum et al., 2009	8716
9	1030	Müller, 2009	**1030**
10	14 374	McCollum et al., 2009	**13 894**
11	29 180	McCollum et al., 2009	39 783
12	5264	Atsuta et al., 2007	**5142**

Table 6.14: Ranking of the analysed algorithms according to their average and best performance on the ITC 2007 benchmark set for NIS = 7 (instances 4, and 9 to 12 were excluded), NIS = 8 (instances 9 to 12 were excluded), and NIS = 12 (complete data set). The lower the rank, the better the algorithm's performance.

NIS									
7				8				12	
Min		Avg		Min		Avg		Min	
Alg.	Rk	Alg.	Rk	Alg.	Rk	Alg.	Rk	Alg.	Rk
Gog12	1.7	Gog12	1.3	Col09	2.0	Gog12	1.6	Col09	1.8
Col09	2.0	Alz15	2.0	Gog12	2.0	Alz15	2.0	Mul09	1.8
Mul09	2.6	Alz14	2.7	Mul09	3.0	Alz14	2.8	cMA	2.4
Alz15	4.0	cMA	4.1	Alz15	3.9	cMA	3.6	Pil07	4.0
Alz14	4.7	Dem12	4.9	Alz14	4.8				
cMA	6.4			cMA	5.5				
Gog07	7.4			Gog07	6.9				
Dem12	7.6			Ats07	8.3				
Ats07	9.0			Pil07	8.8				
Pil07	9.6								

Holm and Hochberg's tests confirm that Gog12 is better than Dem12 and cMA with $\alpha = 0.05$.

Table A.14 presents adjusted p-values as obtained with the procedures of Nemenyi, Holm, Shaffer and Bergmann, which are used to compare pairs of algorithms. The dif-

Table 6.15: Average relative deviation to the BKS. Gog07 and Dem12 have both NIS = 10 (but they solved different instances), thus two corresponding ARD values for cMA were computed for comparison purposes. The NIS values are indicated as 10 and 10*.

Algorithm	NIS	ARD (%)	cMA improvement
Gog12	8	3.51	−23.18
Col09	12	3.67	−17.15
Mul09	12	8.38	−12.44
Alz15	8	9.76	−16.93
Alz14	8	16.21	−10.48
Sme07	7	31.36	9.66
Dem12	10*	41.25	19.90
Gog07	10	49.60	24.62
Ats07	11	131.01	111.61
Pil07	12	146.80	125.98
	12	20.82	–
cMA	11	19.41	–
	10	24.98	–
	10*	21.35	–
	8	26.69	–
	7	21.70	–

ference is significant with all procedures for the pairs $i = 1, 2, 3$ and with Bergmann's procedure only for the pairs $i = 4, 5$. These results are analysed in the next section.

Discussion

The key finding of these results are as follows. From the obtained ARD values and statistical ranking, cMA is positioned in the middle with an ARD between 19% and 27% from the BKS, for NIS = 11 and NIS = 8, respectively. The other competitors with better ARD are Gog12 (3.51), Col09 (3.67), Mul09 (8.38), Alz15 (9.76) and Alz14 (16.21). Thus, the best algorithm is Gog12. From this statistical analysis, we can conclude, with a significance level of 5%, that Gog12 is better than Dem12 and cMA, and that Alz15 is better than Dem12; under Bergmann's procedure, we can also conclude that Alz14 is better than Dem12 and that Alz15 is better than cMA. The statistical analysis confirms some of the results obtained with the ranking (Table 6.14) and ARD (Table 6.15). Thus, cMA can solve the small instances effectively, but additional time is required for the larger instances, due to the use of an evolutionary algorithm.

6.4 Conclusions

In this work, a novel evolutionary approach for solving timetabling problems is described. The proposed algorithm is a cellular evolutionary algorithm (cMA) that promotes the population diversity by means of a smooth update of solutions through the population. The algorithm was augmented with variation and local search operators that maintain solution feasibility and is controlled by threshold acceptance. Incremental evaluation was implemented in order to improve the algorithm's execution time. A study of the impact of simulated annealing-based methods on the examination timetabling problem was carried out. This study shows that a low threshold decreasing rate is needed to obtain the best results, by assigning the difficult exams to better suited time slots, thus allowing the easy exams to be placed in good time slots as well.

The performance of cMA was evaluated on the uncapacitated Toronto and capacitated ITC 2007 benchmark sets, and compared with state-of-the-art approaches. On the Toronto set, the presented solution method improves on four out of thirteen instances, and attains the lowest average relative deviation to the best known solutions. In order to achieve these results for the Toronto set, cMA requires more time than its competitors, about two to four times more than the slowest approaches on the largest instances (car91, car92, pur93, and uta92).

On the ITC 2007 set, our approach improves on three out of twelve instances, for the same execution time than the competitors. In this benchmark set, cMA is in the mid positions with respect to the average relative deviation to the best known solutions.

Future work will focus on two research lines. The first one will extend the use of cMA to the remaining ITC 2007 tracks (course timetabling and post-enrolment course timetabling tracks). The second line of research will focus on a parallel implementation of cMA.

Chapter 7

A Multi-objective Evolutionary Algorithm for Examination Timetabling

Contents

In this chapter we propose a novel hybrid *Multi-Objective Evolutionary Algorithm* (MOEA) and show its application on a real-world *Examination Timetabling Problem* (ETP) instance. The considered problem instance is the examination timetable of the Electrical, Telecommunications and Computer Engineering Department (DEETC) at the Instituto Superior de Engenharia de Lisboa (ISEL) – Instituto Politécnico de Lisboa. This real-world benchmark set is named ISEL–DEETC and is described in Chapter 3. The proposed MOEA is based on the Elitist *non-dominated sorting genetic algorithm-II* (NSGA-II) (Deb, Pratap, Agarwal & Meyarivan, 2002). The NSGA-II procedure is one of the popularly used MOEA which attempts to find multiple Pareto-optimal solutions in a multi-objective optimisation problem. Like the works of Cheong et al. (2009); Côté et al. (2004); Mumford (2010); Wong, Côté & Sabourin (2004), we also consider two objectives: one that maximises each student free time between exams, and a second objective that considers the minimisation of the timetable length.

The chapter is organised as follows. Section 7.1 presents the algorithmic flow of the proposed MOEA. Section 7.2 presents simulation results and analysis of the proposed algorithm. Finally, conclusions and future work are presented in Section 7.3.

7.1 Hybrid Multi-objective Evolutionary Algorithm

As mentioned in the introduction, we solve the ISEL–DEETC ETP using a hybrid MOEA based on the NSGA-II algorithm. The NSGA-II has the following features: (1) it uses an elitist principle, (2) it uses an explicit diversity preserving mechanism, and (3) it emphasizes non-dominated solutions. The basic NSGA-II was further transformed to include a step where a local search procedure is performed. The general steps of the hybrid algorithm (named *hybrid multi-objective evolutionary algorithm* (HMOEA)) are enumerated in Figure 7.1. In the following subsections, each block of the HMOEA is described in detail.

7.1.1 Chromosome Encoding

The timetabling problem approached in this chapter is specified in Section 3.4. In order to optimise for the second objective (see Equation (3.15)), each timetable is represented by a variable-length chromosome as proposed by Cheong et al. (2009), and illustrated in Figure 7.2. A chromosome encodes a complete and feasible timetable, and contain the periods and exams scheduled in each period. Well-formed timetables should have a number of periods belonging to a valid interval, initially given by the timetable planner.

```
Procedure HMOEA
$P^{(t)}$: parent population at iteration $t$
$Q^{(t)}$: offspring population at iteration $t$
$R^{(t)}$: combined population at iteration $t$
$L$: local search operator
OUTPUT
$N^{(t)}$: archive of non-dominated timetables

Initialise $P^0$ and $Q^0$ of size $N$ with random timetables
For each iteration $t \leftarrow 0, 1, \ldots, I_{max} - 1$ do
    (Step 1) Form the combined population, $R^{(t)} = P^{(t)} \bigcup Q^{(t)}$, of size $2N$.
    (Step 2) Classify $R^{(t)}$ into different non-domination classes.
    (Step 3) If $t \geq 1$, use local search procedure $L$ to improve elements of $R^{(t)}$.
    (Step 4) Form the new population $P^{(t+1)}$ with solutions of different
             non-dominated fronts, sequentially, and use the crowding sort procedure
             to choose the solutions of the last front that can be accommodated.
    (Step 5) $N^{(t+1)} \leftarrow NonDominated(P^{(t+1)})$. If $t = I_{max} - 1$ then Stop.
    (Step 6) Create offspring population $Q^{(t+1)}$ from $P^{(t+1)}$ by using the crowded
             tournament selection, crossover and mutation operators.
    (Step 7) Repair infeasible timetables.
```

Figure 7.1: Hybrid NSGA-II procedure.

However, the operation of crossover and mutation can produce invalid timetables, because of the extra periods added to the timetable as a result of these operations. Thus, a repairing scheme must be applied in order to repair infeasible timetables. The adopted scheme is explained in detail in Section 7.1.4.

7.1.2 Population Initialisation

It is known that the basic examination timetabling problem, in which one minimises the number of slots considering the single hard constraint of not having students with overlapping exams, is equivalent to the graph colouring problem Côté et al. (2004). As such, several heuristics of graph colouring have been applied to the ETP. These heuristics influence the order in which exams are inserted in the timetable. In this work, we use the following two heuristics, in the initialisation and mutation processes:

- *saturation degree* (SD): Exams with the fewest valid periods, in terms of satisfying the hard constraints, remaining in the timetable are reinserted first.
- *extended saturation degree* (ESD): Exams with the fewest valid periods, in terms of

Exams	Periods: 1	2	3	4	5	6	7	8	9	10	11	12	...
e_1	1												
e_2					1								
e_3			1										
e_4													1
e_5											1		
e_6	1												
...								1					

Figure 7.2: Variable length chromosome representation. A chromosome encodes a complete and feasible timetable.

satisfying both hard and soft constraints, remaining in the timetable are reinserted first.

The ESD heuristic is used in the population initialisation procedure, while the SD heuristic is used in the reinsertion process of the mutation operator (detailed in Section 7.1.3). These two procedures are similar to the procedures applied in Cheong et al. (2009). The use of the SD heuristic in the initialisation process has been experimented but with worse results than the ESD heuristic.

In the initialisation process, a timetable with a random (valid) length is generated for each chromosome. Then, the unscheduled exams are ordered according to the ESD heuristic and a candidate exam is selected randomly being then scheduled into a randomly chosen period (chosen from the set of periods with available capacity while respecting the feasibility constraint). If no such period exists, a new period is added to the end of the timetable to accommodate the exam. In the ESD heuristic used, a candidate exam can be scheduled in a period if it does not violate feasibility and if the number of clashes is bellow or equal to 70. This process is repeated until all exams have been scheduled.

7.1.3 Selection, Crossover and Mutation

The offspring population is created from the parent population by using the crowded tournament selection operator Deb et al. (2002). This operator compares two solutions and returns as the winner of the tournament the one which has a better rank, or if the solutions have the same rank, the one who has a better crowding distance (the one which is more far apart from their direct neighbours).

The crossover and mutation operations were adapted from the ones introduced in the work of Cheong et al. (2009). In the crossover operator, termed *Day-exchange* crossover, the best days, selected based on the crossover rate, are exchanged between chromosomes. The best day of a chromosome consist of the day (a period, in our case) which has the lowest number of clashes per student. This operation is illustrated in Figure 7.3. To ensure feasibility after the crossover operation, the duplicated exams are deleted. Notice that, as mentioned before, the result of inserting a new period in a chromosome can yield a timetable with a number of periods larger than the valid upper limit. If this is the case, a repair scheme is applied in order to compact the timetable.

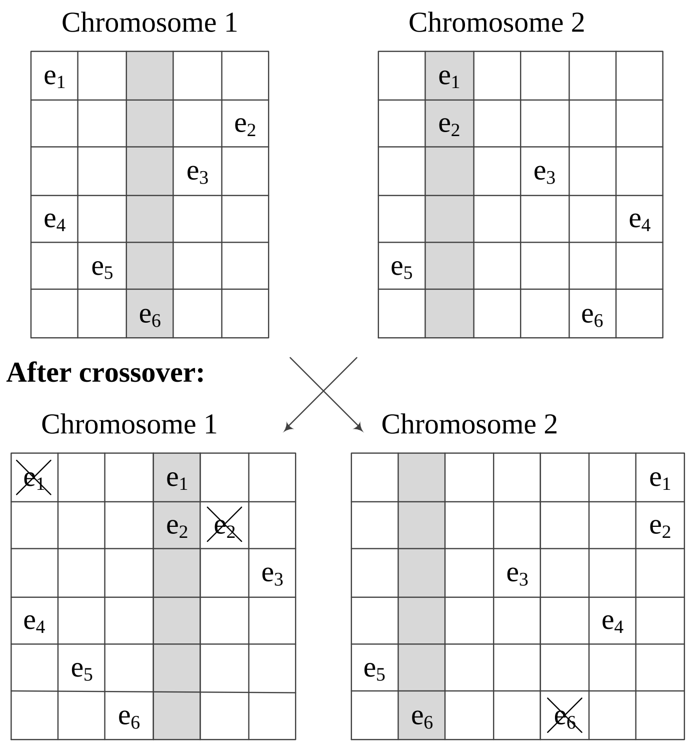


Figure 7.3: Illustration of the *Day-exchange* crossover based in Cheong et al. (2009). The shaded periods represent the chromosomes *best days*. These are exchanged between chromosomes, being inserted into randomly chosen periods. Duplicated exams are then removed.

The mutation operator removes a number of exams, selected based on the reinsertion rate, and reinserts them into other randomly selected periods while maintaining feasibility. We use the SD graph colouring heuristic to reorder the exams, prior to reinserting them. As in the case of the crossover operator, the mutation operator could also add extra periods to the timetable, for the exams that could not be rescheduled without violating the hard constraints.

7.1.4 Repairing Scheme

The repair scheme adopted was based in the period control operator of Cheong et al. (2009), consisting of the following two operations: (1) *Period expansion*, used when the timetable has a number of periods below the lower limit, and (2) *Period packing*, used when the timetable has a number of periods above the upper limit.

In the period expansion operation, empty periods are first added to the end of the timetable such that the timetable length is equal to a random number within the period range. A *clash list*, comprising all exams involved in at least one clash, is maintained. Then, all the exams in the clash list are swept in a random order and rescheduled into a random period without causing any clashes while maintaining feasibility. Exams which could not be moved are left intact.

The period packing operation proceeds as follows: first, the period with the smallest number of students is selected; then the operation searches in order of available period capacity, starting from the smallest, for a period which can contain exams from the former while maintaining feasibility and without causing any clashes. The operation stops when the timetable length is reduced to a random number in the desired range or when it goes one cycle through all periods without rescheduling any exam.

7.1.5 Ranking Computation

The non-dominated sorting procedure employed in NSGA-II uses the evaluation of the two objective functions to rank the solutions. We adopt a simple penalisation scheme in order to penalize solutions with an invalid number of periods. The penalisation is enforced according to the following pseudo-code:

```
If timetable length > max length Then
    $f_1^{Pen} = f_1 + \alpha_1(\text{timetable length} - \text{max length})$
    $f_2^{Pen} = f_2 + \alpha_2(\text{timetable length} - \text{max length})$
Else If timetable length < min length Then
    $f_1^{Pen} = f_1 + \alpha_1(\text{min length} - \text{timetable length})$
    $f_2^{Pen} = f_2 + \alpha_2(\text{min length} - \text{timetable length}).$
```

We set $\alpha_1 = 1000$ and $\alpha_2 = 10$ to introduce a high penalisation on the number of clashes and number of periods, respectively.

7.1.6 Local Exploitation

The local exploitation step was based in Cheong et al. (2009). It employs a local search procedure to improve locally some elements of the population. First, $2N/4$ groups of fours are formed by randomly selecting into each group elements of the population $R^{(t)}$. Then, tournaments between elements of each group are taken. The chromosome which has the lower rank (the one who belongs to the front with lower number) wins the tournament and is then scheduled for the improvement step. With this procedure, about $N/2$ of the chromosomes of population $R^{(t)}$ are selected for improvement. Also, it is guaranteed that at least one element of the non-dominated front is selected for improvement.

The selected chromosomes are improved locally using a short iteration *stochastic hill climbing* (SHC) procedure, with objective function f_1 = *minimization of the number of clashes*. We set a low temperature T in the SHC. In this way, our SHC works like a standard *hill climbing* (HC) but with only one neighbour, instead of evaluating a whole neighbourhood of solutions.

The random neighbour is selected according to the following operation. First, a clash list for the selected chromosome is built. Then, the neighbour chromosome is the one which results from applying the best move of a randomly chosen exam in the clash list into a feasible period. The best move is the one that leads to the highest decrease in the number of clashes.

7.1.7 Room Assignment Algorithm

The period seating capacity was set to $S = 600$. For room assignment, we use the algorithm of Lotfi & Cerveny, described in Carter & Laporte (1996b).

7.2 Experimental Evaluation

In this section, we present the results of the HMOEA on the capacitated ISEL–DEETC data set (specified in Section 3.4).

7.2.1 Settings

Table 7.1 presents the algorithmic parameters used in the experiments. The algorithm was programmed in the Matlab language (version 7.9 (R2009b)), and run on a Windows 7.0, i7-2630QM, 2.0 GHz, 8 GB RAM, computer.

Table 7.1: HMOEA parameters.

Parameter	Value
Population size	40
Number of iterations	I_{max} =125
Crossover probability	1.0
Mutation probability	0.2
Reinsertion rate	0.02
SHC # iterations	$t_{max} = 5$
SHC temperature	$T = 0.0001$
Seating capacity	600

7.2.2 Results on the ISEL–DEETC Benchmark Set

First, we present the results on the performance of HMOEA, and then compare it with the available manual solution. In the experiment made, the initial period range was set to the interval $[14, 22]$, that is, four periods below and above the number of periods set in the manual solution.

The performance of the HMOEA in terms of the evolution of the non-dominated front is illustrated in Figure 7.4. We can observe that the algorithm converges rapidly as in iteration 25 it has already a complete first front that is a good approximation of the final Pareto front. After that iteration, the individual solutions are further optimised but to a lesser extent. The running time for this experience (the best result out of five runs) was 411.75 seconds or $\approx$ 7 minutes.

In Table 7.2 we compare the number of clashes per programme obtained by the manual and automatic (considering the obtained solution with 18 periods) procedures. As we can conclude from this table, the automatic solution gives improved results on the number of clashes in all the individual programme timetables, which corresponds to a lower number of clashes in the optimised merged timetable.

Tables 7.3 and 7.4 present the timetables, including room assignment, for the most difficult programme: the LEETC programme. Table 7.3 shows the manual solution for the LEETC examination timetable. The number of clashes of this timetable is 287. Table 7.4 shows the automatic solution for the LEETC examination timetable. The number of clashes of this timetable is 229. In the presented tables, the courses marked in bold face are shared with other programmes, as shown in Table 3.4.

We can see that, qualitatively, the timetable produced by the automatic procedure has a reasonable layout as the exams within the same semester are well distributed. Concerning room assignment, the implemented algorithm doesn't take into account room localisation, so there are exams scheduled at multiple rooms localised far away from each other, which

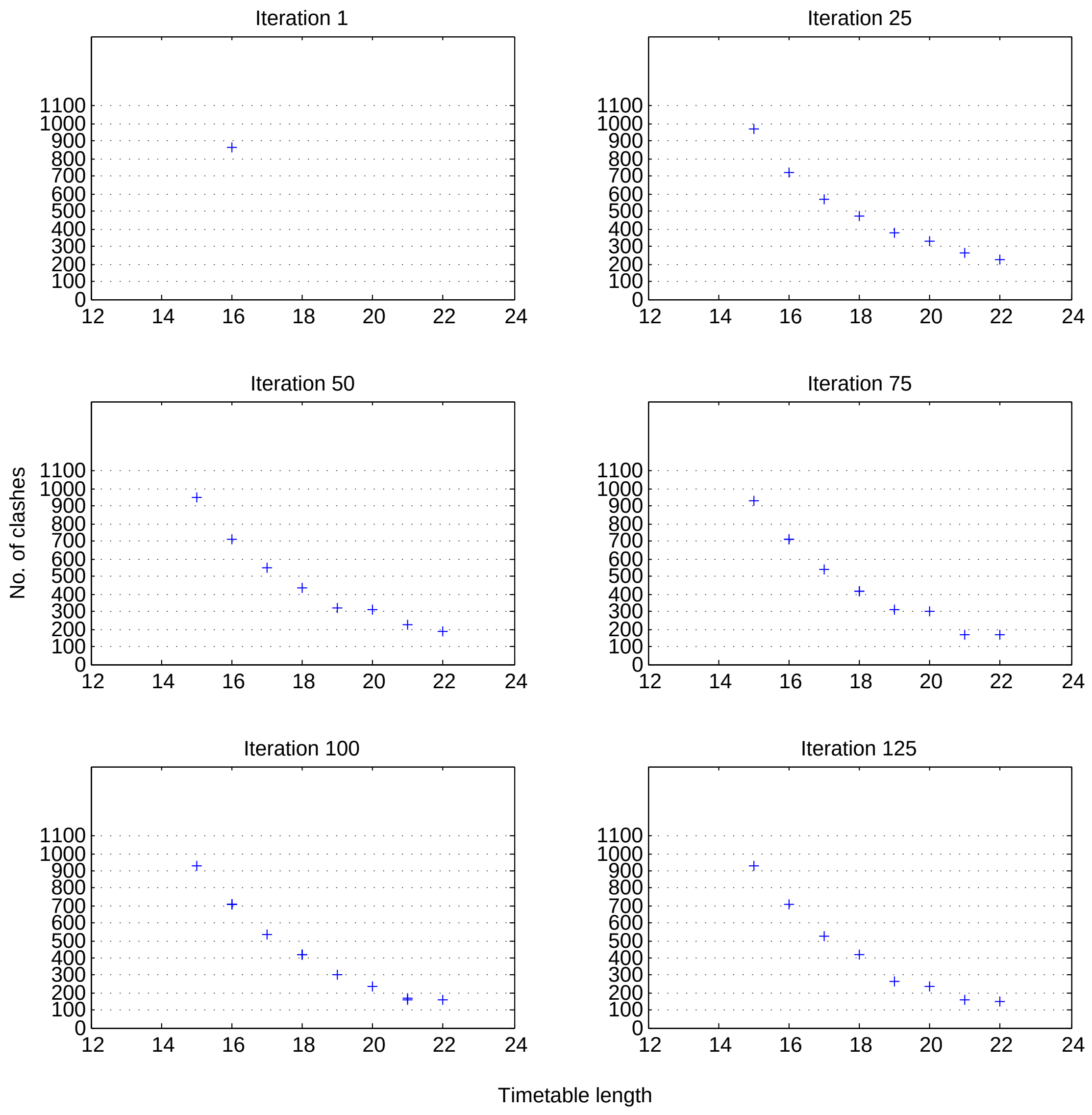


Figure 7.4: Evolution of the Pareto front.

Table 7.2: Timetable number of clashes per programme for the manual and automatic solutions with 18 periods.

	LEETC	LEIC	LERCM	MEIC	MEET	**Combined**
Manual sol.	287	197	114	33	50	**549**
Automatic sol.	229	183	64	2	18	**417**

can be a problem if there are a small number of invigilators. Another aspect observed is the number of rooms assigned for the larger exams (first semester). In the manual solution, when the human planner allocated the exams' rooms, it had available information of a better approximation (indicated by teachers) of the real number of students that were

Table 7.3: Manual solution for the LEETC examination timetable.

Sem.	Room	Course	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1st	A.2.08-A.2.09-A.2.12--A.2.13-A.2.16-A.2.18	**ALGA**			x															
	A.2.12-A.2.13--A.2.16-A.2.18	**Pg**							x											
	A.2.03-A.2.08--A.2.09-A.2.10	**AM1**												x						
	G.0.24-G.0.08-G.0.13	FAE																	x	
	G.0.24-G.0.08--G.0.13-G.0.14	ACir														x				
2nd	C.3.07-C.3.14--C.3.15-C.3.16	**POO**								x										
	G.0.24-G.0.08	**AM2**											x							
	A.2.12-A.2.13--A.2.16-A.2.18	**LSD**																		x
	G.0.24	E1															x			
	G.0.24	MAT					x													
3rd	G.0.24-G.0.08-G.0.13	**PE**													x					
	G.2.09-G.2.10-G.2.21	**ACp**																	x	
	G.0.24-G.1.15	EA									x									
	G.1.15	E2	x																	
	G.1.15	SS						x												
4th	G.2.06-G.2.07-G.2.08	**RCp**											x							
	A.2.12-A.2.13--A.2.16-A.2.18	**PICC/ CPg**															x			
	G.1.15	PR			x															
	G.2.06-G.2.07	FT							x											
	G.0.08-G.0.13	SEAD1																		x
5th	G.1.03-G.1.04	**ST**														x				
	G.2.06-G.2.07	**RCom**					x													
	G.0.24-G.1.15	**RI**										x								
	G.2.06-G.2.07	**SE1**													x					
	G.0.24-G.1.15	**AVE**								x										
	G.2.06-G.2.07	**SCDig**												x						
	A.2.08-A.2.09	**SOt**																	x	
6th	G.0.24	**PI**												x						
	G.1.03-G.1.04	**SCDist**															x			
	G.0.24	**EGP**	x																	
	G.0.13	**OGE**	x																	
	A.2.10-A.2.11	**SG**	x																	

Table 7.4: Automatic solution for the LEETC examination timetable.

Sem.	Room	Course	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	A.2.16-A.2.18-G.0.24-A.2.08-A.2.09-A.2.10-A.2.11-A.2.12-A.2.13-G.1.15	**ALGA**									x									
	A.2.16-A.2.18-G.0.24-A.2.08-A.2.09-A.2.10-A.2.11-A.2.12-A.2.13-G.1.15	**Pg**													x					
1st	A.2.16-A.2.18-G.0.24-A.2.08-A.2.09-G.2.21	**AM1**															x			
	A.2.16-A.2.18-G.0.24-C.2.21	FAE																	x	
	A.2.16-A.2.18-G.0.24-C.3.14	ACir					x													
	A.2.16-A.2.18-A.2.08-A.2.09	**POO**																		x
	C.3.07-C.3.15	**AM2**												x						
2nd	A.2.16-A.2.18-G.0.24-C.3.07	**LSD**	x																	
	G.1.15	E1			x															
	C.3.07	MAT										x								
	G.1.15-G.0.24	**PE**							x											
	G.0.24-C.2.21	**ACp**		x																
3rd	C.3.07-G.1.15	EA						x												
	C.3.07	E2																x		
	C.3.07	SS				x														
	C.3.14-C.3.07-C.3.15	**RCp**											x							
	G.1.18-G.1.15-G.1.13	**PICC/CPg**								x										
4th	G.1.15	PR																	x	
	G.0.15-G.1.03	FT													x					
	G.0.15-G.0.14	SEAD1	x																	
	G.1.18-G.2.06	**ST**					x													
	C.3.14-C.3.15	**RCom**										x								
	C.3.14-C.3.07	**RI**			x															
5th	C.2.21-C.3.07	**SE1**							x											
	G.1.15-G.2.21	**AVE**																		x
	G.1.18-G.2.10	**SCDig**															x			
	A.2.10-A.2.11	**SOt**														x				
	C.3.07	**PI**														x				
	G.2.06-G.2.07	**SCDist**								x										
6th	G.0.24	**EGP**												x						
	G.0.08	**OGE**	x																	
	A.2.08-A.2.09	**SG**												x						

examined. In this way, the allocated room capacity is shorter than those in the automatic solution, that rely solely on enrolment information.

7.2.3 Discussion

The instance considered (ISEL–DEETC) comprises five programmes with high degree of course sharing between programmes, which difficults the manual construction of the timetable. In the manual approach, there are actually five timetables optimised concurrently, one for each programme. The automatic algorithm solves this instance by optimising the combined timetable. With the application of the proposed hybrid MOEA, the present instance was solved effectively, with lower number of clash conflicts compared to the manual solution and in negligible time. Moreover, in experiences made, we obtained lower number of clashes than the actual results, but the optimisation in each timetable was even worse balanced, as some timetables were more optimised than others. This is explained by the intrinsic difficulty in optimising each timetable, e.g. the LEETC is more difficult to optimise than the the LERCM timetable, because it has a greater number of shared courses and more students registered on those courses.

7.3 Conclusions and Future Work

Several improvements can be made to the algorithm. Firstly, in order to prevent for the algorithm to optimise in an unbalanced way, we could consider adding has an objective a measure of programme balance, in order to guide the algorithm to prefer solutions where the number of clashes is minimised *and* the balance in programmes is achieved. Secondly, we could update the room assignment algorithm for assigning exam rooms to nearby locations. Finally, in order to evaluate the performance of HMOEA, we intend to run the algorithm in the set of ETP benchmarks available – the Toronto and Nottingham benchmarks (Qu et al., 2009), and the newer datasets that were proposed in the *Second International Timetabling Competition* (ITC 2007) (McCollum et al., 2012) – and compare with other approaches.

Chapter 8

Conclusions and Future Work

Contents

This chapter concludes the main text of this thesis. Section 8.1 gives a summary of the developed work. Some final remarks are also pointed in this section. Section 8.2 points directions of future work.

8.1 Research Summary

In the research undertaken, the examination timetabling problem is solved using novel hybrid metaheuristics. Additionally, a new benchmark problem is introduced in this thesis, the ISEL–DEETC benchmark.

The proposed methods comprise local search metaheuristics and single and multi-objective memetic algorithms. Five methods, divided in three categories, were proposed:

- Local Search:
  - Accelerated *simulated annealing* based approaches (*fast simulated annealing* (FastSA) and *fast threshold acceptance* (FastTA) algorithms, see Chapter 4). These techniques were applied to the ITC 2007 benchmark set;
- Single-objective Memetic Algorithms:
  - *Shuffled frog-leaping* based algorithm hybridised with *simulated annealing*. The algorithm, named *hybrid shuffled frog-leaping algorithm* (HSFLA), was applied to the Toronto benchmark set (see Chapter 5);
  - *Shuffled complex evolution* based algorithm hybridised with *great deluge*. The algorithm, named *shuffled complex evolution algorithm* (SCEA), was applied to the ISEL–DEETC (2-epochs) and Toronto benchmark sets (see Chapter 5);
  - *Cellular evolutionary algorithm* hybridised with *threshold acceptance*. This algorithm is named *cellular memetic algorithm* (cMA) and was tested on the Toronto and ITC 2007 benchmark sets (see Chapter 6).
- Multi-objective Memetic Algorithm:
  - *Multi-objective evolutionary algorithm* hybridised with *stochastic hill climbing*. The algorithm, named *hybrid multi-objective evolutionary algorithm* (HMOEA), was applied to the ISEL–DEETC (1-epoch) benchmark set (see Chapter 7).

Some final remarks and conclusions on the conducted research are given next.

In Chapter 3, the ISEL–DEETC benchmark set is introduced, with the following key results:

- The ISEL–DEETC benchmark set is a real-world data set comprising five programmes with 80 courses, having a high degree of course sharing between programmes.
- It comprises an extension of the standard *Examination Timetabling Problem* (ETP) in that two epochs are considered instead of a single one. An additional hard constraint is introduced in the ISEL–DEETC instance, to guarantee a minimum distance between the examinations of the first and second epoch for all courses. The introduction of the two-epoch problem is a novel contribution in the field of examination timetabling.

From the research on Local Search methods, reported in Chapter 4, the following remarks summarise our key contributions and findings:

- The proposed metaheuristic uses a novel construction algorithm based on the saturation degree graph colouring heuristic. *Conflict-based statistics* is employed to mitigate the looping effect on the algorithm. The devised construction algorithm always constructs feasible solutions but it is relatively slow in some ITC 2007 instances, namely on instance 11 of that data set.
- An accelerated version of the SA algorithm is proposed. This new algorithm, named FastSA, uses a modified acceptance criterion, which fixes the selected exam as soon as the exam's number of accepted moves in the previous bin is zero. On the experimental evaluation carried out with the ITC 2007 benchmark set, we conclude that FastSA uses 17% less evaluations, on average, than the SA, while attaining competitive results compared with the SA. In a comparison with the techniques from the literature, the FastSA improves on one out of twelve instances, and ranks third out of five algorithms.

Regarding the techniques HSFLA, SCEA and cMA, presented in Chapters 5 and 6, respectively, and their key results, we draw the following conclusions:

- We conclude that the use of memetic algorithms, where hybridisation of evolutionary algorithms and local search is employed, improves the results of the evolutionary algorithm and local search alone. The experimental results obtained by HSFLA, SCEA and cMA show that the proposed techniques are competitive with state-of-the-art methods on the studied benchmark sets. HSFLA is able to produce the best average results on seven instances of the Toronto benchmark set. SCEA and cMA attain one and four new best results, respectively, on the Toronto benchmark set. SCEA was also tested on the ISEL–DEETC benchmark set, considering both the single and the two-epoch problems. On both variants, SCEA attains improved results compared to the manual solution made by the human planner.

One negative aspect of the proposed memetic algorithms is the larger computation time taken compared with local search approaches. If time constraints apply, as is the case of the ITC 2007, we concluded that memetic algorithms are suitable to solve small and medium size instances. On the largest instances, more time is required by the algorithm in order to attain optimisation results competitive with the ones obtained by state-of-the-art algorithms.

- The memetic algorithms presented use structured populations which contribute to improve the population diversity. The SFLA and SCEA use the parallel island model for evolutionary algorithms, whereas the cMA uses the parallel cellular model. The parallel model allows for the algorithms to be implemented in parallel form and run on parallel hardware, thus reducing the algorithm's computation time. In this thesis, he have only implemented sequential versions of the algorithms and their parallelisation is a topic of future research.

With respect to HMOEA, described in Chapter 7, we conclude that:

- The use of a multi-objective approach to timetabling is of major importance in practice due to the inherent multi-objective nature of this problem. Several objectives, representing the various stakeholders' views, are asked for. Some of these include:
  - the students' view – who ask for examinations to be spread as much as possible;
  - the teachers and invigilators's view – who ask for exams to be spread as much as possible in order to give time for marking and proofing. Another relevant aspect is concerned with examinations rooms: if a given exam has several rooms, teachers and invigilators usually ask for these rooms to be located nearby;
  - the manager view – who may want to know different solutions, e.g. timetables with distinct timeslot duration and distinct costs, in order to decide which one adapts better to the institution's objectives in terms of quality of service, for example.
- HMOEA was able to find a set of trade-off solutions with different number of time slots, of the ISEL–DEETC benchmark set, considering the single-epoch problem.

For this problem, and considering an examination epoch with 18 time slots (three weeks including saturdays), the automatic solution attains a lower cost compared to the manual solution created by the human planner, and in negligible time.

Finally, all implemented source code, produced results and statistics, and the ISEL–DEETC benchmark set, are published in GitHub repositories (see Appendix C).

The software was implemented using the ParadisEO object-oriented framework (Cahon et al., 2004). The published software was developed using an object-oriented design in order to facilitate future extensions and is fully commented in order to facilitate its understanding.

8.2 Thesis Future Work

Future work will focus on four research lines. The first one will extend the use of the developed algorithms to the remaining ITC 2007 tracks (course timetabling and post-enrolment course timetabling tracks).

The second line of research will focus on the development of other neighbourhood operators. One hypothesis is to implement the operators proposed in Müller (2009):

- Room Change;
- Period Change;
- Period and Room Change;
- Room Swap;
- Period Swap.

These operators, due to its simplicity are faster than the Kempe chain based neighbourhood operator. It would be interesting to assess the effectiveness of a hybrid operator that mixes these five operators and the Kempe chain operator.

The third line of research will focus on adapting some of the proposed algorithms in order to deal with non-feasible solutions by searching in the search space comprising both feasible and infeasible solutions. The investigation will involve the design of new construction algorithms, neighbourhood operators, and new evaluation functions which will include some penalisation term, in order to guide the algorithm to search in the feasible space of solutions.

The last line of research will involve the design and implementation of parallel versions of the developed algorithms.

Appendices

Appendix A

Statistical Significance Tests

This appendix contains tables that summarise the average rankings (Friedman test) and respective post-hoc tests, for multiple algorithm comparison performed on the Toronto and ITC 2007 benchmark sets. All statistical tests were performed with a significance level of 5%.

A.1 FastSA

Table A.1: Average ranking of Friedman's test for the compared methods, considering the complete data set (12 instances). The average values of the distributions were used in the ranking computation. The p-value computed by the Friedman test is 3.93×10^{-6}.

Algorithm	Ranking
Byk13	1.25
Bat17	2.50
FastSA	3.08
Col09	3.50
Ham13	4.67

Table A.2: Adjusted p-values (Friedman) produced by running Holm's and Hochberg's tests (Byk13 is the control algorithm) for the algorithms compared in Table A.1. Values marked in boldface represent statistically significant values.

Byk13 vs.	Unadjusted p	p_{Holm}	p_{Hoch}
Ham13	$1.20\cdot10^{-7}$	$\mathbf{4.81\cdot10^{-7}}$	$\mathbf{4.81\cdot10^{-7}}$
Col09	0.000 491	**0.001 47**	**0.001 47**
FastSA	0.004 51	**0.009 02**	**0.009 02**
Bat17	0.0528	0.0528	0.0528

Table A.3: Adjusted p-values (Friedman) of multiple algorithm comparison obtained by Nemenyi (Neme), Holm, Shaffer (Shaf), and Bergmann (Berg) procedures. Values marked in boldface represent statistically significant values.

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	Byk13 vs. Ham13	$1.20 \cdot 10^{-7}$	$\mathbf{1.20 \cdot 10^{-6}}$	$\mathbf{1.20 \cdot 10^{-6}}$	$\mathbf{1.20 \cdot 10^{-6}}$	$\mathbf{1.20 \cdot 10^{-6}}$
2	Col09 vs. Byk13	0.000 491	**0.004 91**	**0.004 42**	**0.002 95**	**0.002 95**
3	Ham13 vs. Bat17	0.000 789	**0.007 89**	**0.006 31**	**0.004 73**	**0.004 73**
4	Byk13 vs. FastSA	0.004 51	**0.0451**	**0.0316**	**0.0271**	**0.0180**
5	Ham13 vs. FastSA	0.0142	0.142	0.0850	0.0850	0.0567
6	Byk13 vs. Bat17	0.0528	0.528	0.264	0.211	0.106
7	Col09 vs. Ham13	0.0707	0.707	0.283	0.283	0.141
8	Col09 vs. Bat17	0.121	1.21	0.364	0.364	0.364
9	Bat17 vs. FastSA	0.366	3.66	0.732	0.732	0.366
10	Col09 vs. FastSA	0.519	5.19	0.732	0.732	0.519

A.2 cMA

Table A.4: Average ranking (Friedman test) for the algorithms cEA, TA, and cMA, on the Toronto set, for *number of instances solved* (NIS) equal to 13. The median values of the distributions were used in the ranking computation. The p-value computed by the Friedman test is 2.26×10^{-6}.

Algorithm	Ranking
cMA	1.00
TA	2.00
cEA	3.00

Table A.5: Adjusted p-values (Friedman) produced by running Holm and Hochberg's tests (cMA is the control algorithm) for the algorithms compared in Table A.4. Values marked in boldface represent statistically significant values.

cMA	Unadjusted p	p_{Holm}	p_{Hoch}
cEA	$3.41 \cdot 10^{-7}$	$\mathbf{6.83 \cdot 10^{-7}}$	$\mathbf{6.83 \cdot 10^{-7}}$
TA	$1.08 \cdot 10^{-2}$	$\mathbf{1.08 \cdot 10^{-2}}$	$\mathbf{1.08 \cdot 10^{-2}}$

Table A.6: Average ranking (Friedman test) for the algorithms cMA – light cooling schedule and cMA – intensive cooling schedule, on the Toronto set, for NIS $= 13$. The median values of the distributions were used in the ranking computation. The p-value computed by the Friedman test is 8.74×10^{-4}.

Algorithm	Ranking
cMA-intensive	1.04
cMA-light	1.96

Table A.7: Adjusted p-values (Friedman) produced by running Holm and Hochberg's tests (cMA-intensive is the control algorithm) for the algorithms compared in Table A.6. Values marked in boldface represent statistically significant values.

cMA-intensive vs.	Unadjusted p	p_{Holm}	p_{Hoch}
cMA-light	$8.74{\cdot}10^{-4}$	$\mathbf{8.74{\cdot}10^{-4}}$	$\mathbf{8.74{\cdot}10^{-4}}$

Table A.8: Average ranking (Friedman test) for the algorithms cEA, TA, and cMA, on the ITC 2007 set, for NIS $= 12$. The median values of the distributions were used in the ranking computation. The p-value computed by the Friedman test is 8.84×10^{-5}.

Algorithm	Ranking
cMA	1.00
cEA	2.33
TA	2.67

Table A.9: Adjusted p-values (Friedman) produced by running Holm and Hochberg's tests (cMA is the control algorithm) for the algorithms compared in Table A.8. Values marked in boldface represent statistically significant values.

cMA vs.	Unadjusted p	p_{Holm}	p_{Hoch}
TA	$4.46{\cdot}10^{-5}$	$\mathbf{8.91{\cdot}10^{-5}}$	$\mathbf{8.91{\cdot}10^{-5}}$
cEA	$1.09{\cdot}10^{-3}$	$\mathbf{1.09{\cdot}10^{-3}}$	$\mathbf{1.09{\cdot}10^{-3}}$

Table A.10: Average ranking (Friedman test) for the cMA and state-of-the-art approaches on the Toronto set, with NIS = 11. The average values of the distributions were used in the ranking computation. The p-value computed by the Friedman test is 6.71×10^{-11}.

Algorithm	Ranking
cMA	1.23
Lei14	2.05
Dem12	3.23
Bur08	3.77
Alz15	5.09
Fon15	6.18
Abd13	6.64
Pil10	8.09
Ele07	8.73

Table A.11: Adjusted p-values (Friedman) on the Toronto benchmark set, for NIS = 11, produced by running Holm and Hochberg's tests (cMA is the control algorithm) for the algorithms compared in Table A.10. Values marked in boldface represent statistically significant values.

cMA vs.	Unadjusted p	p_{Holm}	p_{Hoch}
Ele07	$1.34{\cdot}10^{-10}$	$\mathbf{1.07{\cdot}10^{-9}}$	$\mathbf{1.07{\cdot}10^{-9}}$
Pil10	$4.16{\cdot}10^{-9}$	$\mathbf{2.91{\cdot}10^{-8}}$	$\mathbf{2.91{\cdot}10^{-8}}$
Abd13	$3.62{\cdot}10^{-6}$	$\mathbf{2.17{\cdot}10^{-5}}$	$\mathbf{2.17{\cdot}10^{-5}}$
Fon15	$2.21{\cdot}10^{-5}$	$\mathbf{1.10{\cdot}10^{-4}}$	$\mathbf{1.10{\cdot}10^{-4}}$
Alz15	$9.38{\cdot}10^{-4}$	$\mathbf{3.75{\cdot}10^{-3}}$	$\mathbf{3.75{\cdot}10^{-3}}$
Bur08	0.0293	0.0878	0.0878
Dem12	0.0868	0.174	0.174
Lei14	0.484	0.484	0.484

Table A.12: Average ranking (Friedman test) for the cMA and state-of-the-art approaches on the ITC 2007 set, with NIS = 7. The average values of the distributions were used in the ranking computation. The p-value computed by the Friedman test is 6.13×10^{-5}.

Algorithm	Ranking
Gog12	1.29
Alz15	2.00
Alz14	2.71
cMA	4.14
Dem12	4.86

Table A.13: Adjusted p-values (Friedman) on the ITC 2007 benchmark set, for NIS = 7, produced by running Holm and Hochberg's tests (Gog12 is the control algorithm) for the algorithms compared in Table A.12. Values marked in boldface represent statistically significant values.

Gog12 vs.	Unadjusted p	p_{Holm}	p_{Hoch}
Dem12	$2.38{\cdot}10^{-5}$	$\mathbf{9.52{\cdot}10^{-5}}$	$\mathbf{9.52{\cdot}10^{-5}}$
cMA	$7.23{\cdot}10^{-4}$	$\mathbf{2.17{\cdot}10^{-3}}$	$\mathbf{2.17{\cdot}10^{-3}}$
Alz14	0.0910	0.182	0.182
Alz15	0.398	0.398	0.398

Table A.14: Adjusted p-values (Friedman) on the ITC 2007 benchmark set, for NIS = 7, obtained by Nemenyi (Neme), Holm, Shaffer (Shaf), and Bergmann (Berg) procedures. Values marked in boldface represent statistically significant values.

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	Dem12 vs. Gog12	$2.38{\cdot}10^{-5}$	$\mathbf{2.38{\cdot}10^{-4}}$	$\mathbf{2.38{\cdot}10^{-4}}$	$\mathbf{2.38{\cdot}10^{-4}}$	$\mathbf{2.38{\cdot}10^{-4}}$
2	Dem12 vs. Alz15	$7.23{\cdot}10^{-4}$	$\mathbf{7.23{\cdot}10^{-3}}$	$\mathbf{6.51{\cdot}10^{-3}}$	$\mathbf{4.34{\cdot}10^{-3}}$	$\mathbf{4.34{\cdot}10^{-3}}$
3	Gog12 vs. cMA	$7.23{\cdot}10^{-4}$	$\mathbf{7.23{\cdot}10^{-3}}$	$\mathbf{6.51{\cdot}10^{-3}}$	$\mathbf{4.34{\cdot}10^{-3}}$	$\mathbf{4.34{\cdot}10^{-3}}$
4	Dem12 vs. Alz14	0.0112	0.112	0.0786	0.0674	**0.0449**
5	Alz15 vs. cMA	0.0112	0.112	0.0786	0.0674	**0.0449**
6	Gog12 vs. Alz14	0.0910	0.910	0.455	0.364	0.364
7	Alz14 vs. cMA	0.0910	0.910	0.455	0.364	0.364
8	Dem12 vs. cMA	0.398	3.98	1.19	1.19	0.796
9	Alz14 vs. Alz15	0.398	3.98	1.19	1.19	0.796
10	Gog12 vs. Alz15	0.398	3.98	1.19	1.19	0.796

Appendix B

Running times

In this section the algorithms' running times for the Toronto and ITC 2007 benchmark sets are reported. Table B.1 presents the running times of the algorithms compared to cMA for the Toronto set. The numbers were taken from the corresponding papers, since the codes were not available. Thus, running times refer to different platforms and sources codes, with unknown levels of optimisation.

For the ITC 2007 benchmark set, a fixed time limit is enforced on the execution time of the candidate algorithms, in accordance with the rules of the ITC 2007 competition. This limit is determined by running on the host machine a benchmarking tool that is available from the ITC 2007 site. Hence, all the compared algorithms are executed under similar time limits, based on the hardware characteristics of the machine running the program. For our computer, the time limit is 276 seconds.

Table B.1: Running times for the Toronto benchmark set. In the column marked with (*), the authors report execution times ranging from 1–2 minutes (small instances) to 30–90 minutes for larger instances. In the column marked with (**), the authors did not report the algorithm execution time. In the column marked with (***), the authors report execution times ranging from 10 minutes to 160 minutes.

Instance	Car96	Yan05	Ele07	Car08	Bur08	Bur10	Pil10	Dem12	Abd13	Lei14	Fon15	Alz15	cMA
car91	20.7 s	46.22 min	< 270 min	2.87 min	11.08 min	*	1 h 38 min	12 h	**	24 h	9 h 12 min	***	48 h
car92	47 s	25.3 min	< 270 min	18.67 min	10.07 min		1 h 11 min	12 h		24 h	8 h 41 min		48 h
ear83	24.7 s	18.92 min	< 270 min	5 min	7.5 min		12 min 31 s	12 h		24 h	6 h 31 min		24 h
hec92	7.4 s	18.23 min	0.3 min	0.8 min	9.83 min		7 min 28 s	12 h		24 h	3 h 51 min		24 h
kfu93	2 min	35.88 min	< 270 min	5.33 min	14.7 min		52 min 48 s	12 h		24 h	6 h 5 min		24 h
lse91	48 s	28.82 min	< 270 min	7.43 min	10.68 min		47 min 43 s	12 h		24 h	4 h 14 min		24 h
pur93	6 h 2 min 9 s	–	270 min	200.7 min	12.45 min		31 h 36 min	12 h		48 h	–		48 h
rye92	8.45 min	23.03 min	< 270 min	7.53 min	15.02 min		1 h 12 min	12 h		24 h	–		24 h
sta83	5.7 s	12.33 min	< 270 min	0.5 min	9.78 min		7 min 49 s	1 h		24 h	2 h 58 min		24 h
tre92	1.79 min	30.27 min	< 270 min	7.17 min	10.13 min		18 min 41 s	12 h		24 h	6 h 54 min		24 h
uta92	11.07 min	31.5 min	< 270 min	34.1 min	13.42 min		1 h 39 s	12 h		24 h	9 h 48 min		48 h
ute92	9.1 s	12.43 min	< 270 min	0.33 min	8.8 min		11 min 3 s	12 h		24 h	3 h 30 min		24 h
yor83	4.52 min	26.52 min	< 270 min	7.53 min	8.37 min		9 min 12 s	12 h		24 h	7 h		24 h

Appendix C

Developed Software

For each examination timetabling benchmark set, the source code, the resulting solution files for each instance/run, and the produced statistics, are publicly available in the Git [1] repositories presented in Table C.1.

Table C.1: Git repositories for the developed algorithms.

Algorithm	Data set	Language	Repository
FastSA	ITC 2007	C++	`https://github.com/nunocsleite/FastSA-ETP-ITC2007`
SFLA	Toronto	C++	`https://github.com/nunocsleite/SFLA-ETP-Toronto`
SCEA	DEETC	C++	
	Toronto	C++	`https://github.com/nunocsleite/SCEA-ETP-DEETC-Toronto`
cMA	Toronto	C++	`https://github.com/nunocsleite/cMA-ETP-Toronto`
	ITC 2007	C++	`https://github.com/nunocsleite/cMA-ETP-ITC2007`
HMOEA	DEETC	Matlab	`https://github.com/nunocsleite/NSGA-II---ISEL-DEETC-timetable`

[1] Git is a free and open source distributed version control system designed to handle from small to very large projects with speed and efficiency.

References

Abdul Rahman, S. (2012). *Search methodologies for examination timetabling*. Ph.D. thesis University of Nottingham. URL: `http://eprints.nottingham.ac.uk/12709/1/Thesis.pdf`.

Abdullah, S., & Alzaqebah, M. (2013). A hybrid self-adaptive bees algorithm for examination timetabling problems. *Appl. Soft Comput.*, *13*, 3608–3620. doi:10.1016/j.asoc.2013.04.010.

Abdullah, S., Burke, E. K., & McCollum, B. (2005). An investigation of variable neighbourhood search for university course timetabling. In G. Kendall, L. Lei, & M. Pinedo (Eds.), *In proceedings of the 2nd Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2005), 18 -21 July 2005, New York, USA* (pp. 413–427).

Abdullah, S., Turabieh, H., & McCollum, B. (2009). A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems. In M. J. Blesa, C. Blum, L. D. Gaspero, A. Roli, M. Sampels, & A. Schaerf (Eds.), *Hybrid Metaheuristics* (pp. 60–72). Springer volume 5818 of *Lecture Notes in Computer Science*. doi:10.1007/978-3-642-04918-7_5.

Abdullah, S., Turabieh, H., McCollum, B., & McMullan, P. (2010). A tabu-based memetic approach for examination timetabling problems. In J. Yu, S. Greco, P. Lingras, G. Wang, & A. Skowron (Eds.), *RSKT* (pp. 574–581). Springer volume 6401 of *Lecture Notes in Computer Science*. doi:10.1007/978-3-642-16248-0_78.

Abramson, D. (1991). Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Management Science*, *37*, 98–113. URL: `https://doi.org/10.1287/mnsc.37.1.98`. doi:10.1287/mnsc.37.1.98. `arXiv:https://doi.org/10.1287/mnsc.37.1.98`.

Alba, E., & Dorronsoro, B. (2005). The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Trans. Evolutionary Computation*, *9*, 126–142.

doi:10.1109/TEVC.2005.843751.

Alba, E., & Dorronsoro, B. (2008). *Cellular Genetic Algorithms*. (1st ed.). Springer Publishing Company, Incorporated.

Alkan, A., & Özcan, E. (2003). Memetic algorithms for timetabling. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on* (pp. 1796–1802 Vol.3). volume 3. doi:10.1109/CEC.2003.1299890.

Alzaqebah, M., & Abdullah, S. (2014). An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling. *J. Scheduling*, *17*, 249–262. doi:10.1007/s10951-013-0352-y.

Alzaqebah, M., & Abdullah, S. (2015). Hybrid bee colony optimization for examination timetabling problems. *Computers & Operations Research*, *54*, 142 – 154. doi:10.1016/j.cor.2014.09.005.

Amiri, B., Fathian, M., & Maroosi, A. (2009). Application of shuffled frog-leaping algorithm on clustering. *The International Journal of Advanced Manufacturing Technology*, *45*, 199–209. URL: `http://dx.doi.org/10.1007/s00170-009-1958-2`. doi:10.1007/s00170-009-1958-2.

Avella, P., D'Auria, B., Salerno, S., & Vasil'ev, I. (2007). A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, *13*, 543–556. URL: `http://dx.doi.org/10.1007/s10732-007-9025-3`. doi:10.1007/s10732-007-9025-3.

Battistutta, M., Schaerf, A., & Urli, T. (2017). Feature-based tuning of single-stage simulated annealing for examination timetabling. *Annals of Operations Research*, *252*, 239–254. URL: `https://doi.org/10.1007/s10479-015-2061-8`. doi:10.1007/s10479-015-2061-8.

Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., & Likothanassis, S. D. (2008). Applying evolutionary computation to the school timetabling problem: The greek case. *Computers & Operations Research*, *35*, 1265 – 1280. doi:10.1016/j.cor.2006.08.010.

Bellio, R., Ceschia, S., Gaspero, L. D., Schaerf, A., & Urli, T. (2016). Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *Computers & Operations Research*, *65*, 83 – 92. URL: `http://www.sciencedirect.com/science/article/pii/S0305054815001690`. doi:https://doi.org/10.1016/j.cor.2015.07.002.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, *11*, 4135–4151.

Blum, C., & Raidl, G. R. (2016). *Hybrid Metaheuristics: Powerful Tools for Optimization*.

Springer.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, *35*, 268–308.

Brélaz, D. (1979). New methods to color the vertices of a graph. *Commun. ACM*, *22*, 251–256. doi:10.1145/359094.359101.

Burke, E., Bykov, Y., Newall, J., & Petrovic, S. (2004). A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, *36*, 509–528. doi:10.1080/07408170490438410. arXiv:https://doi.org/10.1080/07408170490438410.

Burke, E., Eckersley, A., McCollum, B., Petrovic, S., & Qu, R. (2010). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, *206*, 46 – 53. doi:10.1016/j.ejor.2010.01.044.

Burke, E., & Landa Silva, J. (2005). The Design of Memetic Algorithms for Scheduling and Timetabling Problems. In W. Hart, J. Smith, & N. Krasnogor (Eds.), *Recent Advances in Memetic Algorithms* (pp. 289–311). Springer Berlin / Heidelberg volume 166 of *Studies in Fuzziness and Soft Computing*. URL: http://dx.doi.org/10.1007/3-540-32363-5_13.

Burke, E., Newall, J. P., & Weare, R. F. (1996). A Memetic Algorithm for University Exam Timetabling. In E. Burke, & P. Ross (Eds.), *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling, PATAT 1995* (pp. 241–250). Springer Berlin / Heidelberg volume 1153 of *Lecture Notes in Computer Science*.

Burke, E. K., & Bykov, Y. (2008). A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems. In *PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*.

Burke, E. K., McCollum, B., McMullan, P., & Parkes, A. J. (2008). Multi-objective aspects of the examination timetabling competition track. In *PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*.

Bykov, Y., & Petrovic, S. (2013). An initial study of a novel step counting hill climbing heuristic applied to timetabling problems. In *Proceedings of the 6th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2013)* (pp. 691–693).

Cahon, S., Melab, N., & Talbi, E.-G. (2004). ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. *Journal of Heuristics*, *10*, 357–380. doi:10.1023/B:HEUR.0000026900.92269.ec.

Caldeira, J., & Rosa, A. C. (1997). School timetabling using genetic search. In *PATAT '97 Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling*.

Cambazard, H., Hebrard, E., O'Sullivan, B., & Papadopoulos, A. (2012). Local search and constraint programming for the post enrolment-based course timetabling problem. *Annals of Operations Research*, *194*, 111–135. URL: `https://doi.org/10.1007/s10479-010-0737-7`. doi:10.1007/s10479-010-0737-7.

Caramia, M., Dell'Olmo, P., & Italiano, G. F. (2008). Novel local-search-based approaches to university examination timetabling. *INFORMS Journal on Computing*, *20*, 86–99. doi:10.1287/ijoc.1070.0220.

Carter, M., & Laporte, G. (1996a). Recent Developments in Practical Examination Timetabling. In E. Burke, & P. Ross (Eds.), *Practice and Theory of Automated Timetabling* (pp. 1–21). Springer Berlin / Heidelberg volume 1153 of *Lecture Notes in Computer Science*. URL: `http://dx.doi.org/10.1007/3-540-61794-9_49`.

Carter, M., Laporte, G., & Lee, S. Y. (1996). Examination Timetabling: Algorithmic Strategies and Applications. *Journal of the Operational Research Society*, *47*, 373–383. doi:10.2307/3010580.

Carter, M. W. (1986). A survey of practical applications of examination timetabling algorithms. *Oper. Res.*, *34*, 193–202. doi:10.1287/opre.34.2.193.

Carter, M. W., & Laporte, G. (1996b). Recent developments in practical examination timetabling. In E. Burke, & P. Ross (Eds.), *Practice and Theory of Automated Timetabling* (pp. 1–21). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/3-540-61794-9_49.

Cheong, C., Tan, K., & Veeravalli, B. (2009). A Multi-objective Evolutionary Algorithm for Examination Timetabling. *Journal of Scheduling*, *12*, 121–146. doi:10.1007/s10951-008-0085-5.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Genetic algorithms and highly constrained problems: The time-table case. In H.-P. Schwefel, & R. Männer (Eds.), *Parallel Problem Solving from Nature* (pp. 55–59). Springer Berlin Heidelberg volume 496 of *Lecture Notes in Computer Science*. doi:10.1007/BFb0029731.

Corne, D., Ross, P., & Fang, H.-L. (1994). Fast practical evolutionary timetabling. In T. Fogarty (Ed.), *Evolutionary Computing* (pp. 250–263). Springer Berlin Heidelberg volume 865 of *Lecture Notes in Computer Science*. doi:10.1007/3-540-58483-8_19.

Côté, P., Wong, T., & Sabourin, R. (2004). Application of a Hybrid Multi-Objective Evolutionary Algorithm to the Uncapacitated Exam Proximity Problem. In *Pro-*

ceedings of the 5th International Conference on Practice and Theory of Automated Timetabling (PATAT 2004) (pp. 151–167).

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197. URL: `http://dx.doi.org/10.1109/4235.996017`. doi:10.1109/4235.996017.

Demeester, P., Bilgin, B., Causmaecker, P. D., & Berghe, G. V. (2012). A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice. *J. Scheduling*, *15*, 83–103. doi:10.1007/s10951-011-0258-5.

Dorigo, M., & Stützle, T. (2010). Ant colony optimization: Overview and recent advances. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 227–263). Springer US volume 146 of *International Series in Operations Research & Management Science*. doi:10.1007/978-1-4419-1665-5_8.

Dowsland, K. A. (1990). A timetabling problem in which clashes are inevitable. *Journal of the Operational Research Society*, *41*, 907–918. URL: `https://doi.org/10.1057/jors.1990.143`. doi:10.1057/jors.1990.143.

Dowsland, K. A., & Thompson, J. M. (2012). Simulated annealing. In G. Rozenberg, T. Bäck, & J. Kok (Eds.), *Handbook of Natural Computing* (pp. 1623–1655). Springer Berlin Heidelberg. doi:10.1007/978-3-540-92910-9_49.

Duan, Q., Gupta, V., & Sorooshian, S. (1993). Shuffled complex evolution approach for effective and efficient global minimization. *Journal of Optimization Theory and Applications*, *76*, 501–521. doi:10.1007/BF00939380.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, *104*, 86 – 92. doi:10.1006/jcph.1993.1010.

Dueck, G., & Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, *90*, 161 – 175. doi:10.1016/0021-9991(90)90201-B.

Eley, M. (2006). Ant algorithms for the exam timetabling problem. In E. K. Burke, & H. Rudová (Eds.), *PATAT* (pp. 364–382). Springer volume 3867 of *Lecture Notes in Computer Science*. doi:10.1007/978-3-540-77345-0_23.

Eusuff, M., Lansey, K., & Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization*, *38*, 129–154. URL: `http://dx.doi.org/10.1080/03052150500384759`. doi:10.1080/03052150500384759.

Eusuff, M. M., & Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management*, *129*, 210–225.

Fang, H.-L. (1994). *Genetic algorithms in timetabling and scheduling*. Ph.D. thesis Department of Artificial Intelligence. University of Edinburgh Edinburgh, UK.

Fernandes, C., Caldeira, J. P., Melício, F., & Rosa, A. (1999a). Evolutionary algorithm for school timetabling. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2* GECCO'99 (pp. 1777–1777). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL: `http://dl.acm.org/citation.cfm?id=2934046.2934192`.

Fernandes, C. M., Caldeira, J. P., Melício, F., & Rosa, A. C. (1999b). High school weekly timetabling by evolutionary algorithms. In B. R. Bryant, G. B. Lamont, H. Haddad, & J. H. Carroll (Eds.), *Proceedings of the 1999 ACM Symposium on Applied Computing, SAC'99, San Antonio, Texas, USA* (pp. 344–350). ACM. doi:10.1145/298151.298379.

Fong, C. W., Asmuni, H., & McCollum, B. (2015). A hybrid swarm-based approach to university timetabling. *IEEE Trans. Evolutionary Computation*, *19*, 870–884. doi:10.1109/TEVC.2015.2411741.

Fonseca, G. H. G., & Santos, H. G. (2013). Memetic algorithms for the high school timetabling problem. In *2013 IEEE Congress on Evolutionary Computation* (pp. 666–672). doi:10.1109/CEC.2013.6557632.

García, S., & Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, *9*, 2677–2694.

Gendreau, M., & Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. (2nd ed.). Springer Publishing Company, Incorporated.

Glover, F., & Laguna, M. (1997). *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers.

Glover, F. W., & Kochenberger, G. A. (2003). *Handbook of Metaheuristics*. Springer.

Gogos, C., Alefragis, P., & Housos, E. (2008). A multi-staged algorithmic process for the solution of the examination timetabling problem. In *Proceedings of the 7th international conference on practice and theory of automated timetabling*. University of Montreal, Canada.

Gogos, C., Alefragis, P., & Housos, E. (2012). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals OR*, *194*, 203–221. doi:10.1007/s10479-010-0712-3.

Hamilton-Bryce, R., McMullan, P., & McCollum, B. (2013). Directing selection within an extended great deluge optimization algorithm. In *Proceedings of the 6th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2013)* (pp. 499–508).

Johnes, J. (2015). Operational research in education. *European Journal of Operational Research*, *243*, 683 – 696. doi:10.1016/j.ejor.2014.10.043.

Joslin, D. E., & Clements, D. P. (1999). "squeaky wheel" optimization. *J. Artif. Int. Res.*, *10*, 353–373. URL: `http://dl.acm.org/citation.cfm?id=1622859.1622871`.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical Report Technical Report-TR06, Erciyes university, engineering faculty, computer engineering department.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Kreher, D. L., & Stinson, D. R. (1999). *Combinatorial Algorithms: Generation, Enumeration, and Search*. CRC Press.

Kristiansen, S., & Stidsen, T. R. (2013). *A Comprehensive Study of Educational Timetabling, a Survey*. Technical Report Department of Management Engineering, Technical University of Denmark.

Leite, N., Fernandes, C. M., Melício, F., & Rosa, A. C. (2018). A cellular memetic algorithm for the examination timetabling problem. *Computers & Operations Research*, *94*, 118 – 138. doi:10.1016/j.cor.2018.02.009.

Leite, N., Melício, F., & Rosa, A. (2013a). Solving the examination timetabling problem with the shuffled frog-leaping algorithm. In *Proceedings of the 5th International Joint Conference on Computational Intelligence* (pp. 175–180).

Leite, N., Melício, F., & Rosa, A. C. (2014). A Shuffled Complex Evolution Based Algorithm for Examination Timetabling - Benchmarks and a New Problem Focusing Two Epochs. In *Proceedings of the International Conference on Evolutionary Computation Theory and Applications (ECTA)* (pp. 112–124). SciTePress. URL: `http://dx.doi.org/10.5220/0005164801120124`. doi:10.5220/0005164801120124.

Leite, N., Melício, F., & Rosa, A. C. (2016a). A Shuffled Complex Evolution Algorithm for the Examination Timetabling Problem. In J. J. Merelo, A. Rosa, M. J. Cadenas, A. Dourado, K. Madani, & J. Filipe (Eds.), *Computational Intelligence* Studies in Computational Intelligence (pp. 151–168). Cham: Springer International Publishing. doi:10.1007/978-3-319-26393-9_10.

Leite, N., Melício, F., & Rosa, A. C. (2016b). A hybrid shuffled complex evolution algorithm for the examination timetabling problem. URL: https://www.euro-online.org/media_site/reports/EURO28_AB.pdf accessed on May 2018.

Leite, N., Melício, F., & Rosa, A. C. (2017). A fast threshold acceptance algorithm for solving educational timetabling problems. URL: https://www.euro-online.org/conf/admin/tmp/program-ifors2017.pdf accessed on May 2018.

Leite, N., Melício, F., & Rosa, A. C. (2019). A fast simulated annealing algorithm for the examination timetabling problem. *Expert Syst. Appl.*, *122*, 137–151. doi:10.1016/j.eswa.2018.12.048.

Leite, N., Melício, F., & Rosa, A. (2013b). Multiobjective memetic algorithms applied to university timetabling problems. In *Doctoral Consortium (IJCCI 2013)* (pp. 29–37). INSTICC SciTePress.

Leite, N., Melício, F., & Rosa, A. C. (2016c). A hybrid shuffled frog-leaping algorithm for the university examination timetabling problem. In K. Madani, A. Dourado, A. Rosa, J. Filipe, & J. Kacprzyk (Eds.), *Computational Intelligence* (pp. 173–188). Springer International Publishing volume 613 of *Studies in Computational Intelligence*. URL: http://dx.doi.org/10.1007/978-3-319-23392-5_10. doi:10.1007/978-3-319-23392-5_10.

Leite, N., Neves, R., Horta, N., Melício, F., & Rosa, A. C. (2015). Solving a capacitated exam timetabling problem instance using a bi-objective nsga-ii. In K. Madani, A. D. Correia, A. Rosa, & J. Filipe (Eds.), *Computational Intelligence* (pp. 115–129). Springer International Publishing volume 577 of *Studies in Computational Intelligence*. URL: http://dx.doi.org/10.1007/978-3-319-11271-8_8. doi:10.1007/978-3-319-11271-8_8.

Leite, N., Neves, R. F., Horta, N., Melicio, F., & Rosa, A. C. (2012). Solving an Uncapacitated Exam Timetabling Problem Instance using a Hybrid NSGA-II. In A. C. Rosa, A. D. Correia, K. Madani, J. Filipe, & J. Kacprzyk (Eds.), *IJCCI* (pp. 106–115). SciTePress.

Lü, Z., & Hao, J.-K. (2008). Solving the course timetabling problem with a hybrid heuristic algorithm. In D. Dochev, M. Pistore, & P. Traverso (Eds.), *Artificial Intelligence: Methodology, Systems, and Applications* (pp. 262–273). Berlin, Heidelberg: Springer Berlin Heidelberg.

McCollum, B., McMullan, P., Parkes, A., Burke, E., & Abdullah, S. (2009). An extended great deluge approach to the examination timetabling problem. *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009)*, (pp. 424–434).

McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., & Qu, R. (2012). A New

Model for Automated Examination Timetabling. *Annals of Operations Research*, *194*, 291–315. doi:10.1007/s10479-011-0997-x.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., & Burke, E. K. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, *22*, 120–130. doi:10.1287/ijoc.1090.0320. arXiv:https://doi.org/10.1287/ijoc.1090.0320.

Meisels, A., & Schaerf, A. (2003). Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence*, *39*, 41–59. URL: http://dx.doi.org/10.1023/A%3A1024460714760. doi:10.1023/A:1024460714760.

Melício, F., Caldeira, J. P., & Rosa, A. (2004). Two neighbourhood approaches to the timetabling problem. *Proceedings of the practice and theory of automated timetabling (PATAT'04)*, (pp. 267–282).

Melício, F., Caldeira, P., & Rosa, A. (2000). Solving the timetabling problem with simulated annealing. In J. Filipe (Ed.), *Enterprise Information Systems* (pp. 171–178). Dordrecht: Springer Netherlands. doi:10.1007/978-94-015-9518-6_18.

Moscato, P. (1999). Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, & K. V. Price (Eds.), *New Ideas in Optimization* chapter 14. (pp. 219–234). Maidenhead, UK, England: McGraw-Hill Ltd., UK. URL: http://dl.acm.org/citation.cfm?id=329055.329078.

Mühlenthaler, M. (2015). *Fairness in Academic Course Timetabling* volume 678 of *Lecture Notes in Economics and Mathematical Systems*. Springer International Publishing. doi:10.1007/978-3-319-12799-6.

Müller, T. (2009). ITC2007 solver description: a hybrid approach. *Annals of Operations Research*, *172*, 429–446. doi:10.1007/s10479-009-0644-y.

Mumford, C. (2010). A Multiobjective Framework for Heavily Constrained Examination Timetabling Problems. *Annals of Operations Research*, *180*, 3–31. URL: http://dx.doi.org/10.1007/s10479-008-0490-3.

Neri, F., Cotta, C., & Moscato, P. (Eds.) (2012). *Handbook of Memetic Algorithms* volume 379 of *Studies in Computational Intelligence*. Springer. doi:10.1007/978-3-642-23247-3.

de Oliveira, H. C. B., & Vasconcelos, G. C. (2010). A hybrid search method for the vehicle routing problem with time windows. *Annals OR*, *180*, 125–144. URL: http://dx.doi.org/10.1007/s10479-008-0487-y. doi:10.1007/s10479-008-0487-y.

Pillay, N. (2014). A survey of school timetabling research. *Annals OR*, *218*, 261–293. URL: http://dx.doi.org/10.1007/s10479-013-1321-8.

doi:10.1007/s10479-013-1321-8.

Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. *Annals OR*, *239*, 3–38. doi:10.1007/s10479-014-1688-1.

Pillay, N., & Banzhaf, W. (2010). An informed genetic algorithm for the examination timetabling problem. *Appl. Soft Comput.*, *10*, 457–467. doi:10.1016/j.asoc.2009.08.011.

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T., & Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. *J. of Scheduling*, *12*, 55–89. doi:10.1007/s10951-008-0077-5.

Rahman, S. A., Bargiela, A., Burke, E. K., Özcan, E., McCollum, B., & McMullan, P. (2014). Adaptive linear combination of heuristic orderings in constructing examination timetables. *European Journal of Operational Research*, *232*, 287 – 297. URL: `http://www.sciencedirect.com/science/article/pii/S0377221713005596`. doi:http://dx.doi.org/10.1016/j.ejor.2013.06.052.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2012). A honey-bee mating optimization algorithm for educational timetabling problems. *European Journal of Operational Research*, *216*, 533 – 543. doi:10.1016/j.ejor.2011.08.006.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, *13*, 87–127. doi:10.1023/A:1006576209967.

Siarry, P. (2017). *Metaheuristics*. Springer.

Soria-Alcaraz, J. A., Carpio, M., Puga, H. J., & Sotelo-Figueroa, M. (2012). Application of a parallel computational approach in the design methodology for the course timetabling problem. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)* (pp. 31–44).

Talbi, E. (2013). A unified taxonomy of hybrid metaheuristics with mathematical programming, constraint programming and machine learning. In E. Talbi (Ed.), *Hybrid Metaheuristics* (pp. 3–76). Springer volume 434 of *Studies in Computational Intelligence*. URL: `https://doi.org/10.1007/978-3-642-30671-6_1`. doi:10.1007/978-3-642-30671-6_1.

Talbi, E.-G. (2009). *Metaheuristics - From Design to Implementation*. Wiley.

Talbi, E.-G. (2016). Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Annals of Operations Research*, *240*, 171–215. URL: `https://doi.org/10.1007/s10479-015-2034-y`. doi:10.1007/s10479-015-2034-y.

Teoh, C., Wibowo, A., & Ngadiman, M. (2015). Review of state of the art for metaheuristic techniques in academic scheduling problems. *Artificial Intelligence Review*, *44*, 1–21. doi:10.1007/s10462-013-9399-6.

The Second International Timetabling Competition (ITC 2007) (2007). Home page of ITC 2007. URL: `http://www.cs.qub.ac.uk/itc2007/` accessed 29 November 2017.

Thompson, J. M., & Dowsland, K. A. (1996). Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research*, *63*, 105–128. URL: `https://doi.org/10.1007/BF02601641`. doi:10.1007/BF02601641.

Thompson, J. M., & Dowsland, K. A. (1998). A robust simulated annealing based examination timetabling system. *Computers & OR*, *25*, 637–648. doi:10.1016/S0305-0548(97)00101-9.

Trick, M. A. (2011). Sports scheduling. In P. van Hentenryck, & M. Milano (Eds.), *Hybrid Optimization* (pp. 489–508). Springer New York volume 45 of *Springer Optimization and Its Applications*. URL: `http://dx.doi.org/10.1007/978-1-4419-1644-0_15`. doi:10.1007/978-1-4419-1644-0_15.

Wang, Y.-m., Pan, Q.-k., & Ji, J.-z. (2009). Discrete shuffled frog leaping algorithm for examination timetabling problem. *Computer Engineering and Applications*, *45*, 40. URL: `http://cea.ceaj.org/EN/abstract/article_21440.shtml`. doi:10.3778/j.issn.1002-8331.2009.36.013.

Welsh, D. J. A., & Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, *10*, 85–86. doi:10.1093/comjnl/10.1.85. `arXiv:http://comjnl.oxfordjournals.org/content/10/1/85.full.pdf+html`.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, *19*, 151 – 162. doi:10.1016/0377-2217(85)90167-5.

de Werra, D. (1997). The combinatorics of timetabling. *European Journal of Operational Research*, *96*, 504 – 513. doi:10.1016/S0377-2217(96)00111-7.

Wong, T., Côté, P., & Gely, P. (2002). Final exam timetabling: a practical approach. In *IEEE CCECE2002. Canadian Conference on Electrical and Computer Engineering. Conference Proceedings (Cat. No.02CH37373)* (pp. 726–731). volume 2. doi:10.1109/CCECE.2002.1013031.

Wong, T., Côté, P., & Sabourin, R. (2004). A Hybrid MOEA for the Capacitated Exam Proximity Problem. In *Congress on Evolutionary Computation* (pp. 1495 – 1501). volume 2. doi:10.1109/CEC.2004.1331073.

Woumans, G., Boeck, L. D., Beliën, J., & Creemers, S. (2016). A column generation approach for solving the examination-timetabling problem. *European Journal of Operational Research*, *253*, 178 – 194. doi:10.1016/j.ejor.2016.01.046.

Yang, Y., & Petrovic, S. (2005). A novel similarity measure for heuristic selection in examination timetabling. In E. Burke, & M. Trick (Eds.), *Practice and Theory of Automated Timetabling V* (pp. 247–269). Springer Berlin Heidelberg volume 3616 of *Lecture Notes in Computer Science*. doi:10.1007/11593577_15.

Zhang, D., Liu, Y., M'Hallah, R., & Leung, S. C. (2010). A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, *203*, 550 – 558. URL: `http://www.sciencedirect.com/science/article/pii/S0377221709006055`. doi:https://doi.org/10.1016/j.ejor.2009.09.014.