Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Examination Timetabling Automation using Hybrid Meta-heuristics

Miguel De Brito e Nunes

(Licenciado em Engenharia Informática e de Computadores)

Trabalho de projeto realizado para obtenção do grau
de Mestre em Engenharia Informática e de Computadores

Relatório de Progresso

Orientadores:
Artur Jorge Ferreira
Nuno Miguel da Costa de Sousa Leite

Março de 2015

Contents

List of Figures

List of Tables

List of Code Listings

1

Introduction

Many people believe that AI (Artificial Intelligence) was created to imitate human behavior and the way humans think and act. Even though people are not wrong, AI was also created to solve problems that humans are unable to solve, or to solve them in a shorter time window, with a better solution. Humans may take days to find a solution, or may not find a solution that fits their needs. Optimization algorithms may deliver a very good solution in minutes, hours or days, depending on how much time the human is willing to use in order to get a better solution.

A concrete example of this scenario is the creation of timetables. Timetables can be used for educational purposes, sports scheduling, transportation timetabling, among other applications. The timetabling problem consists in scheduling a set of events (e.g., exams, people, trains) to a specified set of time slots, while respecting a predefined set of rules constraints. In some cases the search space is so limited by the constraints that one is forced to relax them in order to find a solution.

1.1 Timetabling Problems

Timetabling problems consists in scheduling classes, lectures or exams on a school or university depending on the timetabling type. These types of timetabling may include scheduling classes, lectures, exams, teachers and rooms in a predefined set of timeslots which are scheduled considering a set of rules. These rules can be, for example, a student can't be present in two classes at the same time, a student can't have two exams in the same day or even an exam must be scheduled before another.

Timetabling is divided into various types, depending on the institution type and if we're scheduling classes/lectures or exams. Timetabling is divided in tree main types:

- Examination timetabling: consists on scheduling university exams depending on different courses, avoiding the overlap of exams containing students from the same course and spreading the exams as much as possible in the timetable;
- Course timetabling: consists on scheduling lectures considering the multiple university courses, avoiding the overlap of lectures with common students;

- School timetabling: consists on scheduling all classes in a school, avoiding the need of students being present at multiple classes at the same time.

  In this project, the main focus is the examination timetabling problem.

The process of creating a timetable requires that the final solution follows a set of constraints. These sets differ depending on the timetabling type and problem specifications. Constraints are divided in two groups: *hard constraints* and *soft constraints*. Hard constraints are a set of rules which must be followed in order to get a working solution. On the other hand, soft constraints represent the views of the different interested players (e.g. institution, students, nurses, train operators) in the resulting timetable. The satisfaction of these type of constraints is not mandatory as is the case of the hard constraints. In the timetabling problem, the goal is usually to optimize a function with a weighted combination of the different soft constraints, while satisfying the set of hard constraints.

1.2 Objectives

This project's main objective is the production of a prototype application which serves as a examination timetabling generator tool. The problem at hand focus on the specifications submitted on International Timetabling Competition 2007 (ITC 2007), *First track*, which includes 12 benchmark instances. In ITC 2007's specifications, the examination timetabling problem considers a set of periods, room assignment, and the existence of constraints considering real problems.

The application's features are as follows:

- Automated generation of examination timetabling, considering the ITC 2007's specifications (*mandatory*);
- Graphical User Interface to allow the user to edit generated solutions and to optimize user's edited solutions (*optional*);
- Validation (correction and quality) of a timetable provided by the user (*mandatory*);
- The creating of an extension to ITC 2007 in order to support two or more exam periods;

This project is divided in two main phases. The first phase consists on studying some techniques and solutions for this problem emphasizing meta-heuristics like: Genetic Algorithms,Simulated Annealing, Taby Search, and some of its hybridizations. The second phase is based on developing, rating and comparison of solution problems, using ITC 2007 data e real data from the 6 courses taught in my university.

1.3 Planning

The main objective of the first part of this thesis is to familiarize me about the problem, and so keeping me up-to-date on what exists on the state-of-art considering various problems and solutions. While studying articles and papers about this subject, it was required the development of this thesis so that the first two topics (1 Introduction and 2 State-of-Art) would be complete and resumed in order to be committed as a *progress report*.

The second part's main goal is to develop solutions to ITC 2007 problem. These solutions include developing a *Loader* to load all the information to be considered while creating the timetables. One of the approaches to be developed will use a local search algorithm (e.g. Simulated Annealing) to improve the solution obtained using a graph coloring heuristic. The results of this approach will be compared to another approaches applied to ITC 2007's timetabling problem, namely the first five winners. Another approach may be implemented with the same heuristics as mentioned before, but instead of a single solution based method, it will be implemented as a population-based algorithm. In the end after comparing results and taking conclusions, a GUI to edit generated solutions may be created, as an optional feature.

The GANTT diagram presented on 1.1 shows a detailed planning for the development of all the main topics of this project until the end of the year including both phases mentioned above. The colors presented on the diagram are green, blue and yellow. Green and blue designate the first and second phase, respectively. The yellow color designates the optional topics.

The planning shown on the GANTT diagram may not be accurate. Some topics and features may take less or more time than specified. The developing order of certain features may change depending on the necessity of these to implement others already in development.

1.4 Document Organization

This document starts by explaining the problems of timetabling and its types in the Introduction, following by the objectives of this project. After the Introduction, the State-of-the-Art is introduced specifying the timetabling problem focusing on examination timetabling and existing approaches including ITC 2007s. In the existing approaches, some of the most used heuristics are mentioned and explained. It follows by explaining the implementation, organization and modeling of the solutions, including the loader and the used heuristic methods. This implementation topic ends by explaining the planning for the implementation of all this project's features and which features takes more time and effort than others, by demonstrating a Gantt chart [2]. This thesis ends by explaining the conclusions taken after implementing all the project's features and comparing the results with others approaches.

Figure 1.1: GANTT Diagram for project planning.

2

State of the Art

In this section, we review the state of the art of the problem at hand. We start by describing why timetabling is a rather complex problem, some possible approaches on trying to solve it and some of the solutions already taken, specifically for the ITC 2007 benchmarks.

2.1 Timetabling Problem

When solving timetabling problems, it is possible to generate one of multiple types of solutions which are *feasible*, *non feasible*, *optimal* or *sub-optimal*. A feasible solution solves all the mandatory problem constraints, in contrary to non feasible solutions. An optimal solution is the best feasible solution possible considering the problem and its optimal solution value. A problem may have multiple optimal solutions. For last, non-optimal solutions are feasible solutions that can't reach the optimal solution value and so are not as good compared to an optimal solutions.

Timetabling automation is a subject that has been a target of research for about 50 years. The timetabling problem may be formulated as a search or an optimization problem [3]. As a search problem, the goal consists on finding a feasible solution that satisfies all the hard constraints, while ignoring the soft constraints. On the contrary, posing the timetabling problem as an optimization problem, one seeks to minimize (considering a minimization problem) the violations of soft constraints while satisfying the hard constraints. Typically, the optimization is done after using a search procedure for finding an initial feasible solution.

The basic examination timetabling problem, where only the clash (conflict) hard constraint is observed, reduces to the graph coloring problem [4]. This is a well studied hard problem. Deciding whether a solution exists in the graph coloring problem is a NP-complete problem [5]. Considering the graph coloring as an optimization problem, it is proven that the task of finding the optimal solution is a NP-Hard problem [5]. Graph Coloring problems are explained further in Section 2.2

2.2 Existing Approaches

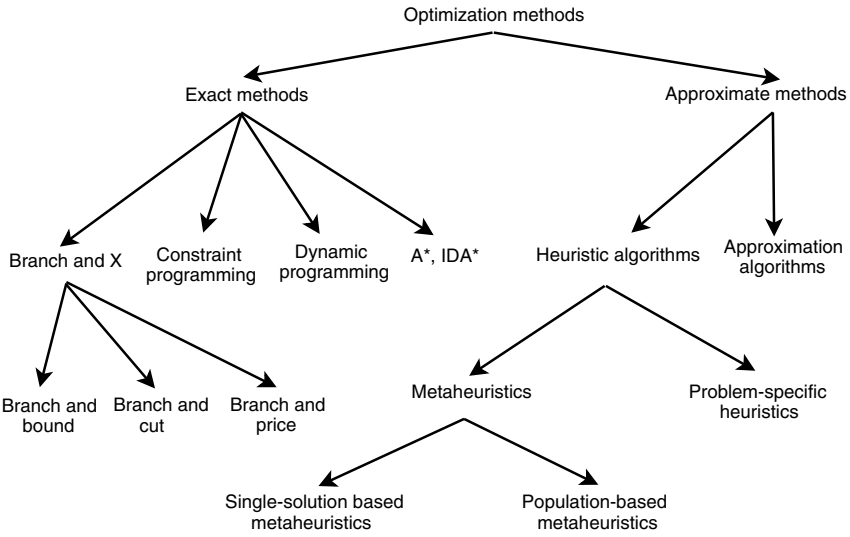

Figure 2.1: Types of algorithms adapted from [1].

The Figure 2.1 represents the organization of Optimization methods. These methods are divided into *Exact methods* and *Approximate methods*.

Timetabling solution approaches are usually divided in the following categories [6]: *exact algorithms* (Branch-and-Bound, Dynamic Programming), *graph based sequential techniques*, *local search based techniques* (Tabu Search, Simulated Annealing, Great Deluge (é?)), *population based algorithms* (Evolutionary Algorithms, Memetic algorithms, Ant algorithms, Artificial immune algorithms), *Multi-criteria techniques*, *Hyper-heuristics*, *Decomposition/clustering techniques* and *Hybrid algorithms*, which combine features of several algorithms, comprise the state-of-the-art. Due to its complexity, approaching the examination timetabling problem using exact method approaches can only be done for small size instances. Real problem instances found in practice are usually of large size, making the use of exact methods impracticable. Heuristic solution algorithms have been usually employed to solve this problem.

Real problem instances are usually solved by applying algorithms which use both *heuristics* and *meta-heuristics*. Heuristic algorithms are problem-dependent, meaning that these are adapted to a specific problem in which take advantage of its details. Heuristics are used to generate a feasible solution, focusing on solving all hard constraints only. Meta-heuristics on the other hand are problem-independent. These are used to, given the feasible solution obtained using heuristic algorithms, generate a better solution focusing on solving as many soft constraints as possible.

Most of the existing Meta-heuristic algorithms belong to one of the three categories: One-Stage algorithms, Two-Stage algorithms and algorithms that allow relaxations [7].

- The One-Stage algorithm is used to get an initial optimal solution, which the goal is to satisfy both hard and soft constraints at the same time. Approaches using this stage are not very common because it's hard to get proper solutions in a reasonable amount of time trying to simultaneously satisfy both types of constraints;
- The Two-Stage algorithms are the most used types of approaches. This category is divided in two phases: the first phase consists in all soft constraints being "discarded" and focusing on solving hard constraints to obtain a feasible solution; the second phase is an attempt to find the best solution, trying to solve the largest number of soft constraints as possible, given the solution of the first phase.

2.2.1 Exact methods

Approximation algorithms like Heuristics and meta-heuristics proceed to enumerate partially the search space and, for that reason, they can't guarantee finding the optimal solution. On the other side, exact approaches perform a complete enumeration of the search space and thus guarantee that the encountered solution is optimal. A negative aspect is the time taken to find the solution. If the decision problem is very difficult (e.g. NP-Complete), in practical scenarios, if the size of the instances is large, this approach may not be applied due to the long execution time.

Programming Based Technique

The Constraint Programming Based Technique (CPBT) allows direct programming with constraints which gives ease and flexibility in solving problems like timetabling. Two important features about this technique are backtracking and logical variables that facilitate searching for an optimal solution at the expense of time. Constraint programming is different from other types of programming, as in these types it is specified the steps that need to be executed, but in constraint programming it is specified the properties (hard constraints) of the solution or properties that should not be in the solution [6].

Integer Programming

The Integer Programming (IP) is a mathematical programming technique in which the optimization problem to be solved must be formulated as an Integer Linear Problem, that is, the objective function and the constraints must be linear, and all problem variables are integer valued. If there are some variables that are continuous and other are integer, then the problem is called Mixed-Integer Linear Programming (MILP). Schaerf [3] surveys some approaches using the MILP technique to school, course, and examination timetabling.

2.2.2 Graph Coloring Based Techniques

Timetabling problems can be reduced to a graph coloring problem. The usual approaches use graph coloring heuristics in the first phase of the two-stage algorithms. Graph Coloring

itself is not an heuristic or meta-heuristic but a method that designates a problem and its variants.

Graph Coloring Problem

The Graph Coloring (GC) problems consists about assigning colors to an element type of the graph which corresponds to a constraint. The simplest sub-type is the *vertex coloring*, which the main goal is to, given a number of vertices and edges, color the vertices so that no adjacent vertices have the same color. In this algorithm, it's best to find a solution with the lowest number of colors as possible. In examination timetable problem, a basic approach could be to represent the exams as vertices and the hard constraints as edges (considering this is a search algorithm, it is good to use optimization algorithms to deal with soft constraints) so that exams with the same color, can be assign to the same timeslot. After coloring, it proceeds to assign the exams into timeslots considering the colors of the solution [6].

Graph Coloring heuristics like *Saturation Degree Ordering* are very commonly used to get the initial solutions. Others like *First Fit* and other *Degree Based Ordering* techniques (*Largest Degree Ordering*, *Incidence Degree Ordering*) are also heuristic techniques for coloring graphs [8].

2.2.3 Meta-heuristics

Meta-heuristics, as mentioned above, usually provide solutions for optimization problems. In timetabling problems, meta-heuristic algorithms are used to optimize the feasible solutions provided by heuristics, like the GC. Meta-heuristics are divided in two main sub-types, which are *Single-solution meta-heuristics* and *Population-based meta-heuristics* [1].

Single-solution meta-heuristics

Single-solution meta-heuristics' main goal is to modify and optimize one single solution, maintaining the search focused in local regions. This type of meta-heuristic is therefore exploitation oriented. Some examples of this type are *Simulated Annealing (SA)*, *Variable-Neighborhood Search*, *Tabu Search (TS)*, *Guided Local Search* [1].

Population-based meta-heuristics

Population-based meta-heuristics' main goal is to modify and optimize multiple candidate solutions, maintaining the search focused in the whole space. This type of meta-heuristic is therefore exploration oriented. Some examples of this type are *Particle Swarm*, *Evolutionary Algorithms*, *Genetic Algorithms* [1].

2.2.4 ITC 2007 Examination timetabling problem: some approaches

In this section the significant techniques applied to the ITC 2007 - Examination timetabling track are described. This timetabling problem comprises 12 instances of different degree of complexity. Through the available website, competitors could submit their solutions for the given benchmark instances. Submitted solutions were evaluated as follows. First, it is checked if the solution is feasible and a so-called distance to the feasibility is computed. If it is feasible, the solution is further evaluated based on the fitness function, which measures the soft constraints total penalty. Then, competitors' solutions are ranked based on the distance to feasibility and solution's fitness value. The competitor with lower distance to feasibility value is the winner. In the case of a tie, the competitor's solution with the lowest fitness value wins. A solution is considered feasible if the value of distance to feasibility is zero. The set of hard constraints is the following:

- no student must be elected to be present in more than one exam at the same time;
- the number of students in a class must not exceed the room's limit capacity;
- exam's length must not surpass the length of the assigned timeslot;
- exams ordering hard constraints must be followed; e.g., $Exam_1$ must be scheduled after $Exam_2$;
- room assignments hard constraints must be followed; e.g., $Exam_1$ must be scheduled in $Room_1$.

It is also necessary to compute the fitness value of the solution and so consider the soft constraints that were not obeyed. The soft constraints are listed below:

- two exams in a row - a student should not be assigned to be in two adjacent exams in the same day;
- two exams in a day - a student should not be assigned to be in two non adjacent exams in the same day;
- period spread - reduce the number of times a student is assigned to be in two exams that are N timeslots apart;
- mixed durations - reduce the number of exams with different durations that occur in a room and period;
- larger exams constraints - reduce the number of large exams that occur later in the timetable;
- room penalty - avoid assigning exams to rooms with penalty;
- period penalty - avoid assigning exams to periods with penalty.

To get a detailed explanation on how to compute the values of fitness and distance to feasibility based on the weight of each constraint, please check ITC 2007's website [9]

In this thesis, we review some of the winners approaches. The winners list of the ITC 2007 competition is as follows:

- 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

Tomáš Müller's approach

Tomáš Müller's approach [10] was actually used to solve all three problems established by the ITC 2007 competition. He was able to win two of them and be finalist on the third. For solving the problems, he opted for an hybrid approach, organized in a two-phase algorithm. In the first phase, Tomáš used Iterative Forward Search (IFS) algorithm [11] to obtain feasible solutions and Conflict-based Statistics [12] to prevent IFS from looping. The second phase consists in using multiple optimization algorithms. These algorithms are applied using this order: Hill Climbing (HC) [13], Great Deluge (GD) [14] and optionally SA [15].

Christos Gogos' approach

Gogos was able to reach second place in Examination Timetabling track, right after Muller. Gogos' approach [16], like Müller's, is a two-phase approach but with a pre-processing stage. In the first phase, is starts using a Pre-processing stage, dealing with hidden dependencies between exams, prior to searching for a solution. After the pre-processing stage, a construction stage takes place, using a meta-heuristic called *greedy randomized adaptive search procedure*. In the second phase, optimization methods are applied in this order: HC, SA, IP that uses Branch and Bound, finishing with the Shaking Stage. Shaking Stage *shakes* the current solution creating an equally good solution that is given to the HC, creating a loop on the used optimization methods.

Mitsunori Atsuta, Koji Nonobe and Toshihide Ibaraki's approach

Mitsunori Atsuta, Koji Nonobe and Toshihide Ibaraki ended up in third place on the Examination Timetabling track and won third and second place on the other tracks as well, utilizing the same approach for all of them. The approach [17] consists on applying constraint satisfaction problem solver adopting an hybridization of TS and Iterated Local Search (modification of local search method), handling weight constraints ?.

Geoffrey De Smet's approach

Geoffrey De Smet's approach [18] differs from all others because he decided not to use a known problem-specific heuristic to obtain a feasible solution, but instead used what is called the *Drool's rule engine*, named *drools-solver* [19]. By definition, the drools-solver is a combination of optimization heuristics and meta-heuristics with very efficient score calculation. After obtaining a feasible solution, Geoffrey opted to use a local search algorithm to improve the solutions obtained using the drools-solver.

Nelishia Pillay's approach

Nelishia Pillay opted to use a two-phase algorithm variation, using *Developmental Approach based on Cell Biology* [20], which the goal is to form a well-developed organism by the process of creating a cell, proceeding with cell division, cell interaction and cell migration. In this approach, each cell represents a timeslot. The first phase represents the creating of the first cell, cell division and cell interaction, and the second phase represents the cell migration.

References

1. E. Talbi, *Metaheuristics - From Design to Implementation*. Wiley, 2009.
2. J. M. Wilson, "Gantt charts: A centenary appreciation," *European Journal of Operational Research*, vol. 149, no. 2, pp. 430–437, 2003.
3. A. Schaerf, "A survey of automated timetabling," *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 87–127, 1999.
4. T. R. Jensen, *Graph Coloring Problems*. John Wiley & Sons, Inc., 2001.
5. S. Arora and B. Barak, *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
6. R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *J. Scheduling*, vol. 12, no. 1, pp. 55–89, 2009.
7. R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems," *OR Spectrum Volume 30, Issue 1 , pp 167-190*, 2007.
8. M. W. Carter, G. Laporte, and S. Y. Lee, "Examination timetabling: Algorithmic strategies and applications," *The Journal of the Operational Research Society*, 1996.
9. D. B. McCollum, "Evaluation function," 2008.
10. T. Müller, "ITC2007 solver description: A hybrid approach," *Annals OR*, vol. 172, no. 1, pp. 429–446, 2009.
11. T. Müller, *Constraint-Based Timetabling*. PhD thesis, Charles University in Prague Faculty of Mathematics and Physics, 2005.
12. T. Müller, R. Barták, and H. Rudová, *Conflict-Based Statistics*. PhD thesis, Faculty of Mathematics and Physics, Charles University Malostransk´e n´am. 2/25, Prague, Czech Republic, 2004.
13. S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
14. G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *Journal of Computational Physics*, 1993.
15. S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simmulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
16. C. Gogos, P. Alefragis, and E. Housos, "An improved multi-staged algorithmic process for the solution of the examination timetabling problem," *Annals OR*, vol. 194, no. 1, pp. 203–221, 2012.
17. M. Atsuta, K. Nonobe, and T. Ibaraki, *ITC-2007 Track2: An Approach using General CSP Solver*. PhD thesis, Kwansei-Gakuin University, School of Science and Technology, 2-1 Gakuen, Sanda, Japan 669-1337 ady67525, ibaraki@kwansei.ac.jp Hosei University, Faculty of Engineering & Design 2-17-1 Fujimi, Chiyoda-ku, Tokyo, Japan 102-8160 nonobe@hosei.ac.jp, 2007.
18. G. D. Smet, "Drools-solver," 2007.
19. T. J. Drools, "Drools planner user guide."
20. N. Pillay, "A developmental approach to the examination timetabling problem," 2007.