INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

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:
Prof. Artur Jorge Ferreira
Prof. 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 amount of time, with a better solution. Humans may take days to find a solution, or may not find a solution at all 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 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 parties (e.g. institution, students, nurses, train operators) in the produced 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 comprehending 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 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 [17]. This thesis ends by explaining the conclusions taken after implementing all the project's features and comparing the results with others approaches.

2

State of the Art

In this section, we review the state of the art of problem at hand. We start by describing why timetabling is a rather complex problem, some possible approaches on trying to solve the problem 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 named *feasible solutions*, *non feasible solutions*, *optimal solutions* or *sub-optimal solutions*. A feasible solution is a solution that 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. It's possible for a problem to 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 optimization problem [15]. As a search problem, the goal consist on finding a solution (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 [2]. Considering the graph coloring as an optimization problem, it is proven that the task of finding the optimal solution is a NP-Hard problem [2]. Graph Coloring problems are explained further in 2.2

2.2 Existing approaches

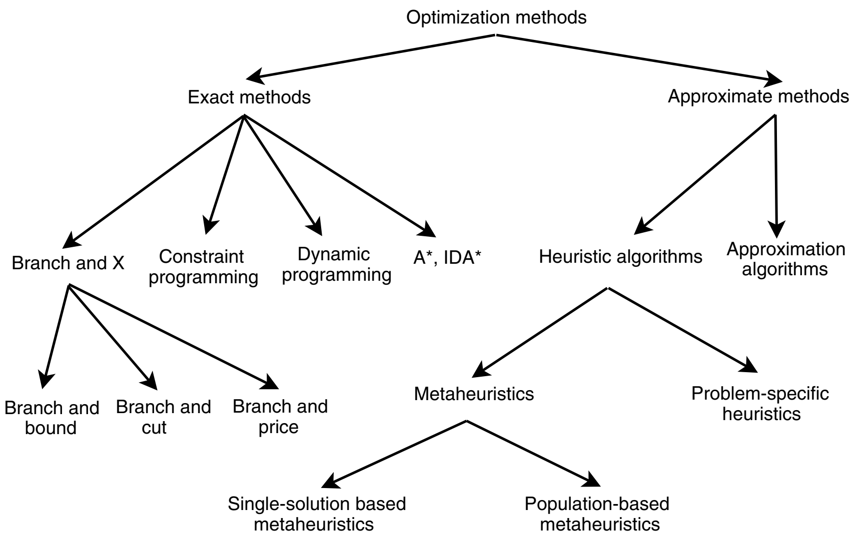

Figure 2.1: Types of algorithms adapted from [16].

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 [12]: *exact algorithms* (Branch-and-Bound, Dynamic Programming), *graph based sequential techniques*, *local search based techniques* (Tabu Search, Simulated Annealing), *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 Meta-heuristic algorithms used 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 satisfy both types of constraints at the same time;
- 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 focus only on solving hard constraints to obtain a feasible solution. The next phase is an attempt to find the best solution, trying to solve the highest number of soft constraints 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.

Constraint 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 [12].

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 [15] 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 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 [12].

Graph Coloring heuristics like *Saturation Degree Ordering* are very commonly used to get the initial solutions. Others like *First Fit*, *Degree Based Ordering*, *Largest Degree Ordering*, *Incident Degree Ordering* are also heuristic techniques for coloring graphs [6].

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* [16].

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 *SA*, *Variable-Neighborhood Search*, *Tabu Search*, *Guided Local Search* [16].

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* [16].

The common types of algorithms and how they are organized, are displayed in Figure 2.1

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 are evaluated in the following form. First, it is checked if the solution is feasible and a so-called distance to 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 the $Room_1$.

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]

The finalists are ranked based their rankings on the instances. For full details on the rankings system, please consult [8].

In this thesis, I'll be reviewing 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 [11] 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 [10] to obtain feasible solutions and Conflict-based Statistics [13] to prevent IFS from looping. The timetabling problems solved were specified as constraint satisfaction problems, where events (exams, courses) are represented by variables. The variables' values are the possible pairs (time slot, room) that don't cause violations in the hard constraints. For each iteration there's an attempt to sign a value to an unassigned variable. If it violates hard constraints, the conflicting variables are unassigned. The variable chosen in each iteration is randomized or parameterized to, for example, assign the most difficult assignable exam first. The Conflict-based Statistics was used to memorize some previously passed conflicts and avoid repeating those for each iteration.

The second phase consists in using multiple optimization algorithms. These algorithms are applied using this order: HC [14], Great Deluge (GD) [3] and optionally SA [5].

HC is used to optimize the first phase solution, resulting on a solution stuck at a local optimum. To leave local optimum area, GD is used in order to try and result in a better solution. For last, SA is used in a loop, keeping the temperature limit unchanged. After not getting a better solution for a limited time, the temperature is reheated (temperature limit gets higher) and HC phase is used again forming a loop in these 3 optimization algorithms.

Christos Gogos' approach:

Gogos was able to reach second place in Examination Timetabling track, right after Muller. Gogos' approach is very different compared to Muller's. His approach, like Muller's, was divided in two phases named *Construction Phase* and *Improvement Phase*, in which the first is used do construct a feasible solution and the second improves the solution found in the first phase, using local search algorithms.

In the first phase, is starts using a Pre-processing stage. This stage deals with hidden dependencies, which means it adds dependencies to the problem that weren't there at the beginning but they exist and makes sense in the problem itself. For example, one hidden dependency may be one exam E_1 having an ordering constraint with exam E_2 stating that E_2 must be scheduled after exam E_1 and exam E_3 must be scheduled after exam E_2. So a dependency/constraint must be added so exam E_3 must be scheduled after exam E_1 as well. The key idea is that this pre-process method will be very helpful in the further stages when trying to find solutions.

After the pre-processing stage, a construction stage takes place. This method is used to create a complete timetable. In this stage, multiple solutions are attempted considering the available time in which only the best solution is passed to the next phase. The solution is constructed by constantly setting an exam and a room to a period until a feasible solution is found.

In the second phase, HC is used. This algorithm is used to generate neighbor solutions, accepting only better solutions compared to the current one, until reaching a local optimum. It is considered local optimum when HC cannot generate a better solution in a number of tries or time limit.

The next stage consists in utilizing SA to get out of local maximum. Simulated Annealing stops when it cannot generate better solutions after the specified period of time.

After Simulated Annealing, an IP formulation that uses Branch and Bound (BB) is used in a set of sub-problems. This stage is called "IP Sub-Problems Stage". This simply examines all periods trying to discover some possible improvements that can be done, for instance, swapping a room with high penalty with a room with no penalty or having lower penalty that is not currently being used.

The last stage is the Shaking Stage. This method "shakes" the current best solution in order to get equally good solutions and pass it to SA stage. This method is not used if the problem we're solving only has one room or the cost of each room is zero. Two heuristics are used in this stage. The first creates neighbor solutions but a solution is only accepted if the new solution is equally good and the room arrangement is better (total cost of room assignment is lower) as compared to the current best solution. The second heuristic reschedule a set of exams presented on the previous solution. These are first removed from the timetable and then scheduled using the same techniques used in the construction stage. This heuristic will probably generate a worse solution but it is accepted and used next on the SA stage.

Tabu Search (TS) [16] is also used in this approach, even though it's not used as a standalone method. This one is used along with HC and SA algorithms to prevent them from looping and creating the same results. TS avoids the creation of equal neighbor solutions by not letting the same period be selected for swapping after this being selected once. This period can only be selected again after a certain number of swaps on other periods.

3

Implementation

3.1 Loader Module

- Analysis of benchmark data, constraints, ...
  Class diagram

3.2 Solution Method

3.2.1 Simulated Annealing

SEE Lecture Notes

3.2.2 Neighbourhood Operators

SEE Muller

3.3 Planning

GHANTT diagram

4

Conclusions

References

1. Salwani Abdullah and Malek Alzaqebah. A hybrid self-adaptive bees algorithm for examination timetabling problems. *Appl. Soft Comput.*, 13(8):3608–3620, 2013.
2. Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
3. Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 1993.
4. Tommy R. Jensen. *Graph Coloring Problems*. John Wiley & Sons, Inc., 2001.
5. Scott Kirkpatrick, D. Gelatt Jr., and Mario P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.
6. Michael W. Carter Gilbert Laporte Sau Yan Lee. Examination timetabling: Algorithmic strategies and applications. *The Journal of the Operational Research Society*, 1996.
7. Rhydian Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum Volume 30, Issue 1 , pp 167-190*, 2007.
8. Barry McCollum, Andrea Schaerf, Ben Paechter, Paul McMullan, Rhyd Lewis, Andrew J. Parkes, Luca Di Gaspero, Rong Qu, and Edmund K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition, 2008.
9. Dr Barry McCollum, 2008.
10. Tomás Müller. *Constraint-based timetabling*. PhD thesis, Charles University in Prague Faculty of Mathematics and Physics, 2005.
11. Tomás Müller. ITC2007 solver description: a hybrid approach. *Annals OR*, 172(1):429–446, 2009.
12. Rong Qu, Edmund K. Burke, Barry McCollum, Liam T. G. Merlot, and S. Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *J. Scheduling*, 12(1):55–89, 2009.
13. Tomás Müller Roman Barták Hana Rudová. *Conflict-based Statistics*. PhD thesis, Faculty of Mathematics and Physics, Charles University Malostransk´e n´am. 2/25, Prague, Czech Republic, 2004.
14. Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
15. Andrea Schaerf. A survey of automated timetabling. *Artif. Intell. Rev.*, 13(2):87–127, 1999.
16. El-Ghazali Talbi. *Metaheuristics - From Design to Implementation*. Wiley, 2009.
17. James M. Wilson. Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2):430–437, 2003.