INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Exam Timetabling

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. Doutor Artur Jorge Ferreira
    Prof. Nuno Miguel da Costa de Sousa Leite

Fevereiro de 2015

Contents

List of Figures

List of Tables

Listings

x

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. Search 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. These rules are called constraints in the timetabling subject, making harder to achieve a solution for the problem. 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

TODO

- Examination timetabling
- Course timetabling
- School timetabling

1.2 Objectives

1.3 Document Organization

Solutions can be divided in multiple types, like *feasible solutions*, *non feasible solutions*, *optimal solutions* or *sub-optimal solution*. 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.

The process of creating a timetable requires that the final solution follows a set of constraints. These can be 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 feasible 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.

The goal of the present project is to create an examination timetable generator using the ITC 2007 (International Timetabling Competition) specification. The solutions will be validated using a validator developed for this purpose in order to assess the quality of the solution. It is also required that the developed prototype may work with two exam epochs which can be considered an extension to the ITC-2007 formulation. In the end an (optional) Graphical User Interface will be created in order to allow the user to edit the current solution to fit the users needs and and to allow the optimization of the edited solution. The generator will be tested using data from ITC-2007 and some actual data from six different programs lectured at ISEL.

1.4 Examination timetabling: 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 ITC 2007 data.

1.4.1 Timetabling Problem

Timetabling is a subject that has been a target of research for about 50 years. Its problem may be formulated as a search or optimization problem [14]. 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 wants to find the best solution possible with time constraints. That is, 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 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 1.4.2.2

1.4.2 Existing approaches

Timetabling solution approaches are usually divided in the following categories [11] *exact algorithms* (Branch-and-Bound, Dynamic Programming), *graph based sequential techniques* (Saturation degree, Colour degree, Largest degree), *local search based techniques* (Tabu Search, Simulated Annealing (SA), *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 usually employed to solve this problem.

Real problem instances are usually solved by using both Heuristics and Meta-heuristics algorithms. 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. [6].

- 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 because is is divided in two phases (the reason for "Two-Stage" name). 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.

1.4.2.1 Exact methods

Some approaches may use *exact methods* which may be viewed as tree algorithms and can be proved that it can find the optimal solution (global optimal). Exact methods search for solutions in the whole search space, and they divide the global problem into simpler problems in order to

find the solution. This is not always used because these methods require a lot of computing time and they rarely produce results in a reasonable time in complex problems. Some approaches use some of these exact methods, namely: Constraint Programming Based Technique, Integer Linear Programming.

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. [11]

Integer Linear Programming

The Integer Linear Programming (ILP) 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 [14] surveys some approaches using the MILP technique to school, course, and examination timetabling.

1.4.2.2 Graph Coloring

The usual approaches start with using heuristics of Graph Coloring to get an initial solution that in most cases is a local optimum. Graph Coloring itself is not an heuristic or meta-heuristic but a method that designates a problem and its variants.

Graph Coloring Problems

The Graph Coloring (GC) algorithm is divided in two main sub-types, which are *vertex coloring* and *edge coloring*.

- The vertex coloring algorithm's main goal is to, given a number of vertices and edges, color the vertexes 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. [11]

- The edge coloring algorithm's main goal is equivalent to the vertex coloring algorithm, but this one is about coloring edges, so that no adjacent edges have the same color. As vertex's algorithm, the least colors possible, the better the solution is.

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.

Saturation Degree Ordering

The Saturation Degree Ordering heuristic colors the vertices with more constraints first. The coloring method is as follows: while choosing a vertice to color, the ones with higher saturation degree will be colored first. The saturation degree of one vertice is the number of differently colored vertices adjacent to this vertice or, in another words, the number of different colors of all adjacent vertices. In the case of a tie, the highest saturation vertice with higher number of adjacent vertices is chosen.

1.4.2.3 Meta-heuristics

Meta-heuristics, as mentioned above, usually provide good 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.* [15]

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* [15].

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*, *local search*, *neighborhood search*, *guided local search*[15].

The common types of algorithms and how they are organized, are displayed in Figure 1.1
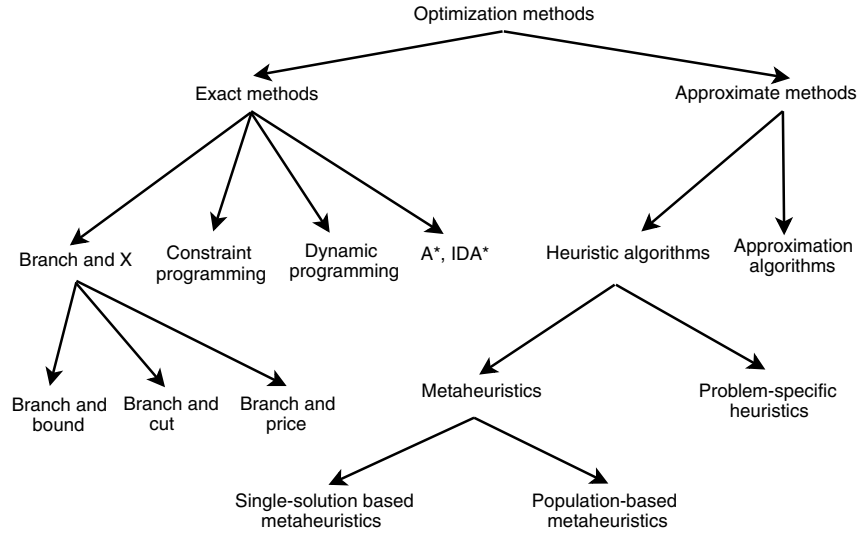
Figure 1.1: Types of algorithms adapted from [15].

The Figure 1.1 represents the organization of Optimization methods. These methods are divided into *Exact methods* and *Approximate methods*. Exact methods and Heuristic algorithms are explained above in this topic.

1.4.2.4 ITC 2007 Examination timetabling problem: some approaches

I will briefly describe some techniques used in ITC 2007 - Examination timetabling track, as detailed explanations about these techniques are properly presented in the articles which are referenced below.

The ITC 2007 Examination timetabling problem is divided in 3 tracks, which are *Examination timetabling*, *Post Enrolment based Course Timetabling* and *Curriculum based Course Timetabling*. The main focus will be the first track - Examination Timetabling.

This 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 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 lowest solution fitness value wins. A solution is considered deemed acceptable if the value of distance to feasibility is zero, and so followed all the hard constraints. In order to have an acceptable solution, this must be feasible and so it is required to follow the following set of hard constraints:

- 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 directly adjacent exams (straight timeslots) in the same day;
- Two exams in a day: A student should not be assigned to be in two non-directly 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 spread 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 [8] and Abdullah's survey [1]

The designation of the winners is divided in two parts. The first part consists on choosing 5 finalists considering the top 5 results provided by all the competitors, which ran the tests on the first 10 instances (not including the hidden ones) on their machines following a certain set of rules [7]. The second part involves the algorithms of these 5 finalists being compared using the competition organizer's machines and running in all 12 instances, including the hidden ones.

The first part consists on comparing the results provided by the competitors (distance to feasibility and values of fitness) for each of the ten instances, ranking the competitors. While in the second part results were generated by running, for each instance, 10 times each the finalist's algorithm. In total were generated 50 solutions for each of the 12 instances. Each of the 50 solutions were ranked based on the results generated (50 to 1). Considering each algorithm has 10 ranks, the rank of the finalist, for that instance, is given by comparing the average of those ranks generated for their algorithm. The lower the average, the better the rank is.

The finalists are ranked based their rankings on the instances. For full details on the rankings system, please consult [7].

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 [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 [9] to obtain feasible solutions and Conflict-based Statistics [12] 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 [13], 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 SA, an Integer Programming (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) [15] 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.

2

Recommender Systems

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. Rhydian Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum Volume 30, Issue 1 , pp 167-190*, 2007.
7. 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.
8. Dr Barry McCollum, 2008.
9. Tomás Müller. *Constraint-based timetabling.* PhD thesis, Charles University in Prague Faculty of Mathematics and Physics, 2005.
10. Tomás Müller. ITC2007 solver description: a hybrid approach. *Annals OR*, 172(1):429–446, 2009.
11. 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.
12. 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.
13. Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.).* Pearson Education, 2010.
14. Andrea Schaerf. A survey of automated timetabling. *Artif. Intell. Rev.*, 13(2):87–127, 1999.
15. El-Ghazali Talbi. *Metaheuristics - From Design to Implementation.* Wiley, 2009.