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

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. 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 [2]. 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 *suboptimal 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 [3]. 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 [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 2.2

2.2 Existing approaches

Optimization methods

Exact methods

Approximate methods

Branch and X

Constraint programming

Dynamic programming

A*, IDA*

Heuristic algorithms

Approximation algorithms

Branch and bound

Branch and cut

Branch and price

Metaheuristics

Problem-specific heuristics

Single-solution based metaheuristics
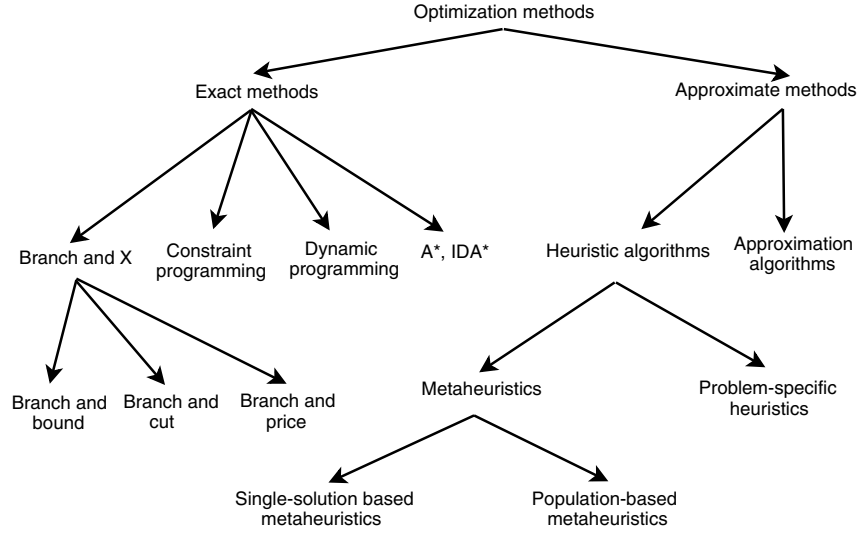
Population-based metaheuristics

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), *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 [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 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].

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 [10].

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 [12] to obtain feasible solutions and Conflict-based Statistics [13] to prevent IFS from looping. The second phase consists in using multiple optimization algorithms. These algorithms are applied using this order: Hill Climbing (HC) [14], Great Deluge (GD) [15] and optionally SA [16].

Christos Gogos' approach

Gogos was able to reach second place in Examination Timetabling track, right after Muller. Gogos' approach [17], like Müller's, was divided in two phases named *Construction Phase* and *Improvement Phase*, in which the first is used to construct a feasible solution and the second improves that solution, using local search algorithms. 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, resulting in the best feasible solution given a limited period of time. 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 [18] 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 [19] 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* [20]. 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* [21], 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.

2.2.5 Other approaches

(2009) Abdullah et al.'s approach

Salwani Abdullah, Hamza Turabieh, and Barry McCollum's approach [22] consists on using an hybridization of an electromagnetic-like mechanism and the GD algorithm. In this approach, the electromagnetism-like mechanism starts with a population of timetables randomly generated. Electromagnetic-like mechanism is a meta-heuristic algorithm using an *attraction-repulsion* mechanism [23] to move the solutions to the optimum.

(2009) Gogos et al.' approach

Christos Gogos, George Goulas, Panayiotis Alefragis and Efthymios Housos' approach [24] also used the two-phase algorithm. The first phase consists on creating the timetable using a greedy randomized positioning, with the use of a backtracking mechanism to help creating the solution. This construction phase is repeated and only the best solutions pass to the second phase. The second phase utilizes local search, SA, shaking and IP to improve the solutions. As mentioned by the author, this approach produced very good results and can be compared to Tomáš Müller's results in his approached used to win ITC 2007 competition.

(2009) McCollum et al.'s approach

B. McCollum, P.J. McMullan, A. J. Parkes, E.K. Burke and S. Abdullah's two phased approach [25] use an adaptive ordering heuristic from [26], proceeding with an *extended version* of GD. As the author stated, this approach is robust and general considering the results obtained on the benchmark datasets from ITC 2007 using this approach.

(2011) Malek Alzaqebah and Salwani Abdullah's approach

Malek Alzaqebah and Salwani Abdullah's two phased approach [27] starts by using a graph coloring heuristic (largest degree ordering) to generate the initial solution and ends by utilizing the *Artificial Bee Colony* search algorithm to optimize the solution.

(2011) Hamza Turabieh and Salwani Abdullah's approach

Hamza Turabieh and SalwaniAbdullah's approach [28] utilize two phase algorithm. The first phase consists on constructing initial solutions by using an hybridization of graph coloring heuristics (least saturation degree, largest degree first and largest enrollment ? first). The second phase utilizes an hybridization of electromagnetic-like mechanism and GD algorithm, just like in (2009) Abdullah et al.'s approach.

(2012) Hamza Turabieh and Salwani Abdullah's approach

Hamza Turabieh and Salwani Abdullah created another approach in 2012 [29]. It utilizes a Tabu-based memetic algorithm which consists on an hybridization of a genetic algorithm with a Tabu Search algorithm. The author states that this approach produces some of the best results when tested on ITC 2007's datasets.

(2012) Demeester et al.'s approach

P. Demeester, B. Bilgin, P. De Causmaecker and G. V. Berghe used an hyper-heuristic approach [30]. The heuristics utilized are 'improved or equal' (hill climbing?), SA, GD and an adapted version of the *late acceptance* strategy [31]. These heuristics are used on already-created initial solutions. Initial solutions are constructed using an algorithm which does not guarantee the feasibility of the solution.

(2012) McCollum et al.'s approach

B. McCollum, P.J. McMullan, A. J. Parkes, E.K. Burke and S. Abdullah's 2012 approach [32] to ITC 2007 problem was based on IP formulation. This approach though, was not capable of solving the competition instances.

(2012) Sabar et al.'s hyper-heuristic of hybridizations approach

N. R. Sabar, M. Ayob, R. Qu and G. Kendall utilized a graphical coloring hyper-heuristic on its approach [33]. This hyper-heuristic is composed of four hybridizations of these four methods: last degree, saturation degree, largest colored degree and largest enrollment. This approach seems to compete with the winners' approaches from ITC 2007, considering the benchmark results.

(2012) Sabar et al.'s honey-bee approach

N. R. Sabar, M. Ayob, G. Kendall, R. Qub's approach [34] utilizes a two phase algorithm. Starts by using an hybridization of graph coloring heuristics to obtain feasible solutions and a variant of honey-bee algorithm for optimization. The hybridization is composed of least saturation degree, largest degree first, largest enrollment first applied in this order.

(2013) Salwani Abdullah and Malek Alzaqebah's approach

Salwani Abdullah and Malek Alzaqebah's opted to create an hybridization approach [35], mixing the utilization of a modified bees algorithm with local search algorithms (i.e. SA and late acceptance HC)

(2014) Salwani Abdullah and Malek Alzaqebah's approach

Salwani Abdullah and Malek Alzaqebah in 2014 constructed an approach [36] that utilizes an hybridization of a modified artificial bee colony with local search algorithm (i.e. late acceptance HC).

(2014) Burke et al.'s approach

Edmund K. Burke, Rong Qu and Amr Soghier's approach [37] uses an hyper-heuristic with hybridization of low level heuristics (neighbor operations) to improve the solutions. The low level heuristics are *move exam*, *swap exam*, *kempe chain move* and *swap timeslot*. After applying this hyper-heuristic with the hybridizations, the hybridization with the best results will be tested with multiple exam ordering methods, applying another hyper-heuristic with hybridizations. The heuristics applied are *largest degree*, *largest weighted degree*, *saturation degree*, *largest penalty* and *random ordering*.

(2014) Ali Hmer and Malek Mouhoub's approach

Ali Hmer and Malek Mouhoub's approach [38] uses a multi-phase hybridization of meta-heuristics. Works like the two phase algorithm but includes a pre-processing phase before the construction phase. This pre-processing phase is divided in two phases: the propagation of ordering constraints and implicit constraints discovery. The construction phase utilizes a variant of TS. The optimization phase uses hybridization of HC, SA and an extended version of GD algorithm.

(2014) Ryan Hamilton-Bryce et al.'s approach

Ryan Hamilton-Bryce, Paul McMullan and Barry McCollum opted to use a non-stochastic method on their approach [39] when choosing examinations in the neighborhood searching process on the optimization phase. Instead, it uses a technique to make an intelligent selection of examinations using information gathered in the construction phase. This approach is divided in 3 phases. The first phase uses a *Squeaky Wheel* constructor which generates multiple initial timetables and a weighted list for each timetable generated. Only the best timetable and its weighted list is passed to the second phase. The second phase is the *Directed Selection Optimization* phase which uses the weighted list created in the construction phase to influence the selection of examinations for the neighbor search process. ("DSO utilizes the weighted list to influence the selection of the examination for optimization".. ??? Não deveria ser seleção de exames para criar soluções vizinhas ao qual estas vão para optimização? Ou a melhor vai para optimização..). Only the best timetable is passed onto the next phase. The third phase is the *Highest Soft Constraint Optimization* phase which is similar to the previous phase but weighted list values are calculated based on the solution's individual soft constraints penalty.

(2014) Syariza Rahman et al.'s constructive approach

S. A. Rahman, E. K. Burke, A. Bargiela, B. McCollum and E. Özcan's approach [40] is a constructive approach. This divides examinations in sets called *easy sets* and *hard sets*.

Easy sets contain the examinations that are easy to schedule on a timetable and on the contrary, hard sets contain the ones that are hard to schedule and so are identified as the ones creating infeasibility. This allows to use the examinations present on the hard sets first on future construction attempts. There's also a sub-set within the easy set, called *Boundary Set* which helps on the examinations' ordering and shuffling. Initial examinations' ordering are accomplished by using graph coloring heuristics like largest degree and saturation degree heuristics.

(2014) Syariza Rahman et al.'s adaptive linear combination approach

Syariza A. Rahman, Andrzej Bargiela, Edmund K. Burke, Ender Özcan, Barry McCollum and Paul McMullan's approach [41] utilizes adaptive linear combinations of graph coloring heuristics (largest degree and saturation degree) with heuristic modifier. These adaptive linear combinations allows the attribution of difficulty scores to examinations depending on how hard their scheduling is. The ones with higher score, and so harder to schedule, are scheduled using two strategies: using single or multiple heuristics and with or without heuristic modifier. The authors conclude that multiple heuristics with heuristic modifier offers good quality solutions, and the presented adaptive linear combination is a highly effective method.

-

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. 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. C. G. L. 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. B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke, "Setting the research agenda in automated timetabling: The second international timetabling competition," 2008.
11. T. Müller, "ITC2007 solver description: A hybrid approach," *Annals OR*, vol. 172, no. 1, pp. 429–446, 2009.
12. T. Müller, *Constraint-Based Timetabling*. PhD thesis, Charles University in Prague Faculty of Mathematics and Physics, 2005.
13. T. M. R. B. H. Rudová, *Conflict-Based Statistics*. PhD thesis, Faculty of Mathematics and Physics, Charles University Malostransk´e n´am. 2/25, Prague, Czech Republic, 2004.
14. S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
15. G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *Journal of Computational Physics*, 1993.
16. S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simmulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
17. 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.
18. M. A. K. N. 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.
19. G. D. Smet, "Drools-solver," 2007.

20. T. J. Drools, "Drools planner user guide."
21. N. Pillay, "A developmental approach to the examination timetabling problem," 2007.
22. S. Abdullah, H. Turabieh, and B. McCollum, "A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems," in *Hybrid Metaheuristics, 6th International Workshop, HM 2009, Udine, Italy, October 16-17, 2009. Proceedings* (M. J. Blesa, C. Blum, L. D. Gaspero, A. Roli, M. Sampels, and A. Schaerf, eds.), vol. 5818 of *Lecture Notes in Computer Science*, pp. 60–72, Springer, 2009.
23. N. Javadian, M. G. Alikhani, and R. Tavakkoli-Moghaddam, "A discrete binary version of the electromagnetism-like heuristic for solving traveling salesman problem," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, 4th International Conference on Intelligent Computing, ICIC 2008, Shanghai, China, September 15-18, 2008, Proceedings* (D. Huang, D. C. W. II, D. S. Levine, and K. Jo, eds.), vol. 5227 of *Lecture Notes in Computer Science*, pp. 123–130, Springer, 2008.
24. C. Gogos, G. Goulas, P. Alefragis, and E. Housos, "Pursuit of better results for the examination timetabling problem using grid resources," in *2009 IEEE Symposium on Computational Intelligence in Scheduling, CISched 2009, Nashville, TN, USA, April 1-2, 2009*, pp. 48–53, IEEE, 2009.
25. B. McCollum, P. J. McMullan, A. J. Parkes, E. K. Burke, and S. Abdullah, "An extended great deluge approach to the examination timetabling problem," in *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2009), 10-12 Aug 2009, Dublin, Ireland* (J. Blazewicz, M. Drozdowski, G. Kendall, and B. McCollum, eds.), pp. 424–434, 2009. Paper.
26. E. K. Burke and J. P. Newall, "Solving examination timetabling problems through adaption of heuristic orderings," *Annals OR*, vol. 129, no. 1-4, pp. 107–134, 2004.
27. M. Alzaqebah and S. Abdullah, "Hybrid artificial bee colony search algorithm based on disruptive selection for examination timetabling problems," in *Combinatorial Optimization and Applications - 5th International Conference, COCOA 2011, Zhangjiajie, China, August 4-6, 2011. Proceedings* (W. Wang, X. Zhu, and D. Du, eds.), vol. 6831 of *Lecture Notes in Computer Science*, pp. 31–45, Springer, 2011.
28. H. Turabieh and S. Abdullah, "An integrated hybrid approach to the examination timetabling problem," *Omega*, vol. 39, no. 6, pp. 598 – 607, 2011.
29. S. Abdullah and H. Turabieh, "On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems," *Information Sciences*, vol. 191, no. 0, pp. 146 – 168, 2012. Data Mining for Software Trustworthiness.
30. P. Demeester, B. Bilgin, P. D. Causmaecker, and G. V. Berghe, "A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice," *J. Scheduling*, vol. 15, no. 1, pp. 83–103, 2012.
31. E. K. Burke and Y. Bykov, "A late acceptance strategy in hill-climbing for examination timetabling problems," in *In Proceedings of the conference on the Practice and Theory of Automated Timetabling(PATAT*, 2008.
32. B. McCollum, P. McMullan, A. J. Parkes, E. K. Burke, and R. Qu, "A new model for automated examination timetabling," *Annals OR*, vol. 194, no. 1, pp. 291–315, 2012.
33. N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph coloring constructive hyper-heuristic for examination timetabling problems," *Appl. Intell.*, vol. 37, no. 1, pp. 1–11, 2012.
34. N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A honey-bee mating optimization algorithm for educational timetabling problems," *European Journal of Operational Research*, vol. 216, no. 3, pp. 533–543, 2012.
35. S. Abdullah and M. Alzaqebah, "A hybrid self-ddaptive bees algorithm for examination timetabling problems," *Appl. Soft Comput.*, vol. 13, no. 8, pp. 3608–3620, 2013.
36. M. Alzaqebah and S. Abdullah, "An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling," *J. Scheduling*, vol. 17, no. 3, pp. 249–262, 2014.
37. E. K. Burke, R. Qu, and A. Soghier, "Adaptive selection of heuristics for improving exam timetables," *Annals OR*, vol. 218, no. 1, pp. 129–145, 2014.

38. A. H. M. Mouhoub, "A multi-phase hybrid metaheuristics approach for the exam timetabling," *(?)Department of Computer Science University of Regina Regina, Canada fh-mer200a,mouhoubmg@cs.uregina.ca*, 2014.

39. R. H.-B. P. M. B. McCollum, "Directed selection using reinforcement learning for the examination timetabling problem," *(?)*, 2014.

40. S. A. Rahman, E. K. Burke, A. Bargiela, B. McCollum, and E. Özcan, "A constructive approach to examination timetabling based on adaptive decomposition and ordering," *Annals OR*, vol. 218, no. 1, pp. 3–21, 2014.

41. S. A. Rahman, A. Bargiela, E. K. Burke, E. Özcan, B. McCollum, and P. McMullan, "Adaptive linear combination of heuristic orderings in constructing examination timetables," *European Journal of Operational Research*, vol. 232, no. 2, pp. 287–297, 2014.