A Multi-Staged Algorithmic Process for the Solution of the Examination Timetabling Problem

Christos Gogos[1,2], Panayiotis Alefragis[1,3], Efthymios Housos[1].

[1]*University Of Patras-Greece. Dept. of Electrical and Computer Engineering.*

[2]*Technological Educational Institute of Epirus. Dept. of Finance and Auditing (Preveza. L. Ioanninon 210).*

[3]*Technological Educational Institute of Mesolonghi. Dept. of Telecommunication Systems and Networks (Varia, Nafpaktos).*

{cgogos|alefrag|housos@ece.upatras.gr}

Abstract. We present an approach for the examination timetabling problem as defined in the second International Timetabling Competition (http://www.cs.qub.ac.uk/itc2007). The solution approach can be considered as an implementation of the GRASP (Greedy Randomized Adaptive Search Procedure) method with the combination of several other metaheuristics. Three stages are employed. The first stage is responsible for the construction of a relatively high quality feasible solution while the second stage improves it using simulated annealing local search. The final stage uses mathematical programming and analyzes each examination period in isolation proposing movements of exams to other rooms resulting in further improvement of the solution quality. The procedure produces feasible solutions for each dataset provided under the runtime limit imposed by the competition's rules. Results are presented and analyzed.

Keywords: GRASP, simulated annealing, Kempe chains, integer programming.

1. Introduction

The initial goal of the research effort undertaken was to address the problem of examination timetabling for universities and attempt to bridge the gap between research and practice (McCollum, 2007). Examination timetabling has been an active area of research during the last decade and a number of solution approaches originating from various disciplines have been proposed. Analytical surveys for the examination timetabling problem can be found in (Schaerf, 1999) and in (Qu et al., 2006). A critical point that differentiates these approaches stems from individual problems definition, as each institutional examination timetabling problem is usually unique due to rules, regulations and quality factors incorporated. For example, the possibility to schedule more than one exam in the same room might be indifferent in one case while being of most importance in another. As a consequence, an effort started to formulate the examination timetabling problem in a general yet able to specialize way so as to be able to capture most of the aspects of the problem that most universities have to confront several times each year (Burke et al., 1996). It must also be noted that solution quality is measured in different ways depending on how

each institution views what a desired examination timetable is. Solution quality can usually be interpreted as timetables which have sufficiently long periods between exams for each student. It was fortunate that during our involvement with the problem, the second International Timetabling Competition (ITC07) was in progress and included a special track about examination timetabling (ETT) in universities. The objective of this track was to address university examination problems that included constraints which usually commence in real life. This fact provided valuable test data and insight in our approach. We believe that the problem model, as described in ITC07, is mature enough to capture most of the requirements typically encountered in reality. Our general objective was to produce an optimization platform that could be used for further research on timetabling problems. Due to the requirements of the competition, we also had to produce high quality feasible solutions in a strictly defined time frame.

2. Problem Description

As the description of the problem is published by the competition organizers, in this section a brief presentation is included for the sake of complicity. The interested reader should further consult the extensive description presented in (McCollum et al., 2007). The Examination Timetabling Problem (ETP) assigns exams to a specific set of time periods and examination rooms while satisfying a set of feasibility and quality constraints aiming to the maximization of the overall productivity and usability of the schedule. A key point in the examination timetabling problem formulation is whether a fixed or variable examination length can be produced as a solution. In our case, the examination period is scheduled for a fixed number of time periods that could span from several days to a few weeks. Each exam has a known enrolled set of students and has to be scheduled in one of the feasible periods while respecting constraints concerning student conflicts, capacity of individual rooms, period and exam length and sequencing and assignment constraints. In our problem, no penalized relaxation of the above constraints was allowed. Only solutions that satisfy hard constraints are compared according to soft constraints. The considered soft constraints could be divided according to the affected entity. The first soft constraint group considers the exam schedule of each individual student. These constraints take into consideration the existence of two immediately consecutive exams or two exams in the same day for the same student and the distance among every two examinations also for the same student. The second soft constraint group covers global constraints concerning the structure of the required solution and includes constraint about mixed duration of examinations within individual periods, constraints about large examinations appearing later in the timetable and constraints about the use of penalized periods and rooms. Users are able to assign penalty weights for each soft constraint and the set of all these weights is called the Institutional Model Index (IMI). The objective value used in the solution process is the weighted sum of the soft constraint violations. This objective function might produce low quality solutions for a number of students but it appears to be a reasonable method for the comparison of various algorithmic processes.

3. Solution Process

The chosen approach for tackling the ETP problem can be regarded as a variation of the GRASP metaheuristic. GRASP (Feo and Resende, 1995) is an iterative process in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution whose neighborhood is investigated until a local minimum is found using a local search phase. A recent treatment of GRASP can be found in (Resende and Ribeiro, 2003). GRASP, with several improvements, has been successfully used in order to tackle the examination timetabling problem by (Casey and Thompson, 2003). In our approach, GRASP functions as the overall procedure that incorporates tabu search (Glover and Laguna, 1997), multiple ordering criteria (Burke and Newall, 2004) and simulated annealing (Dowsland, 1993). In addition, during the final optimization stage, an Integer Programming formulation that uses Branch and Bound as the solution mechanism is also used for a number of individual subproblems. In our approach the solution process is initiated using a pre-processing step followed by a sequence of optimization stages. The complete problem model is used at all stages due to the tight feasibility constraints of the problem. The construction phase is repeated a number of times and the best feasible solution achieved is recorded. Then, starting from this solution a local search phase tries to locate a better solution. By using simulated annealing moves to inferior solutions are admissible under certain conditions making possible the escape of local optima. After the local search phase, the algorithm orders the periods of the examination session by the amount of penalty regarding the use of certain rooms that should have been avoided plus the penalty that derives from the scheduling of exams with different lengths in the same room. Periods with higher penalties are handled with higher priority and an Integer Program is formulated and solved with the mathematical solver package GLPK (http://www.gnu.org/software/glpk/). Each one of the phases will be presented in detail in the following.

3.1 Pre-processing Step

The scheduling difficulty for an exam is a function of its enrollments, student conflicts, educational importance and other custom criteria. Exams with increased scheduling difficulty must be analyzed first because their late entry into the scheduling process would most likely increase the backtracking needs. The ordering of exams based on such difficulty criteria prior to their assignment to timeslots is a common technique and has been used by (Carter et al., 1996), (Burke and Petrovic, 2002), (Burke and Newall, 2004) and others. Our pre-processing procedure involves the creation of five ordered lists. The first list contains the exams sorted by their student size and the second list ranks the exams based on their conflict weight which is calculated as the sum of the nonempty intersections of all pairs of exams. The third list contains for each exam the number of all other exams that must take place in the same period. The list which follows contains for every exam a weight which is based on the sequence constraints that must be satisfied by the solution. Finally, the fifth list holds only the exams that have room constraints in order to be given certain

priority by the scheduling process. The lists are constructed at the beginning of the process and remain available thereafter. Whenever an exam is assigned to a timeslot, its presence in the lists is ignored.

3.2 First Stage – Achieving Feasibility

A feasible solution is constructed by adding to the schedule all the exams sequentially. The algorithm examines in a round robin manner each one of the five lists discussed in the previous section and at each step selects one of the exams from the currently active list. This is done by organizing a tournament among items of the active list which become what is known in the GRASP terminology as the Restricted Candidate List (RCL). More specifically RCL is constructed by selecting the N top items of the currently active list where N is a runtime parameter that in our experiments assumed the value of 10. The tournament approach is used in order to give advantage to exams that are highly ranked. More specifically, a linear bias is used that assigns a value of $1/r_i$ for each element of the list where r_i denotes the rank of element i. Consequently, the probability of each one of the first N elements of active list is given by

$$prob(i) = \frac{\frac{1}{r_i}}{\sum_{k \in RCL} \frac{1}{r_k}}$$

If one of the last three lists has no exams that are unassigned, then turn is passed to the next list. The above method of selecting exams occurs only when no high priority exams are present. High priority exams might be exams that have been removed from the program as a result of another exam assignment or exams that are of coincidence type with already assigned exams.

Next a period and a room have to be assigned to the selected exam. This is done by examining for every valid period the room with the seat number that fits the exam best (leaves the smallest number of free seats in the same room). The penalty of the partial solution is computed and the first period with the minimum cost is preferred so as to schedule the exam in. If scheduling an exam in a period result in zero increment of penalty, then this period is selected without further investigation of the remaining periods. The construction process generates a feasible solution and its objective value is recorded. Then the construction phase is repeated considering another period as the start of the schedule. If during the new construction the objective value becomes worse than the recorded best solution, then the current construction process is dropped altogether and a new construction starts from scratch.

While assigning each exam in the schedule it is possible to that there will not be any more valid periods. If this is the case, then a backtrack method is executed and the selected exam is scheduled in a period and room while other exams that are now invalid are moved back to the list of unscheduled exams. An extra field (removals) is added in each exam recording the number of exams that the current exam forced to move in previous steps of the construction phase. This information is exploited in subsequent steps so that exams that have a history of major disruptions in the construction phase become relatively fixed in the schedule. So the backtracking procedure works by locating the period and room that will be used to schedule the new exam. For each

period a list is constructed using the scheduled exams that will have to be removed due to conflicts and room seats restrictions. Then a corresponding value is computed that is the summation of all its exams "removals" field value. The period that the selected exam will be put is selected stochastically among the list items with the smallest corresponding "removals" value. In order to avoid cycles of adding and then in subsequent steps removing the same exam a tabu list is implemented rendering each newly added exam unmovable for a number of successive iterations. On the other hand, removed exams are given high priority in order to return shortly to the schedule.

The backtracking mechanism proved to be successful and fast in finding feasible solutions for all eight of the provided datasets in ITC07. Information about the datasets examined and the solution achieved can be seen in Table 1. Details of all of these parameters can be found in (McCollum and McMullan, 2007). Conflict density is an important metric calculated from the input data in order to characterize the density level of the problem graph. The conflict density shows the percentage of conflicts among the exams. Two exams are considered to be in conflict if their enrollments share common students.

ITC07-DATASETS												
	Exams	Students	Periods	Rooms	Two In A Row Penalty	Two In A Day Penalty	Period Spread Penalty	No Mixed Durations Penalty	Number Of Largest Exams	Number Of Last Periods To Avoid	Frontload Penalty	Conflict Density
Dataset 1	607	7891	54	7	7	5	5	10	100	30	5	5,05%
Dataset 2	870	12743	40	49	15	5	1	25	250	30	5	1,17%
Dataset 3	934	16439	36	48	15	10	4	20	200	20	10	2,62%
Dataset 4	273	5045	21	1	9	5	2	10	50	10	5	15,00%
Dataset 5	1018	9253	42	3	40	15	5	0	250	30	10	0,87%
Dataset 6	242	7909	16	8	20	5	20	25	25	30	15	6,16%
Dataset 7	1096	14676	80	15	25	5	10	15	250	30	10	1,93%
Dataset 8	598	7718	80	8	150	0	15	25	250	30	5	4,55%

Table 1

3.3 Second stage – local search

Starting from a feasible solution provided by the previous stage, a local search phase follows. The neighborhood structure is defined by randomly selecting a pair of periods and moving certain exams from one period to the other. Kempe chains are used in order to select the appropriate exams (Thompson and Dowsland, 1996). Kempe chains are a by-product of the effort towards solving the infamous "four color problem". The vertices of a graph are grouped in sets according to the color that has been assigned to them. The goal is to produce new feasible solutions by moving vertices between sets. Such a movement might result in an infeasible solution because nodes of the same color must not be connected directly. By using Kempe chains we can move across feasible solutions only. This is done by selecting two sets and constructing a number of chains consisted of vertices belonging to either of the sets which are disconnected between each other. If a vertex of a chain is moved from one set to the other then all other vertices of the same chain must also move to the other set in order to maintain feasibility for the solution.

In our algorithm a graph is constructed with vertices and edges corresponding to exams and conflicts between exams respectively. Each period can be regarded as a color and the purpose is to colorize the graph so as not to have directly connected vertices with the same color. Since we already have a feasible solution we also have a proper coloring of the underlying graph. So Kempe chains can be used in order to produce new feasible solutions according to conflicts between exams. Each solution produced in this way is further checked regarding other constraints like room capacities and in case it is valid its cost is computed. If the cost is better than the best cost found so far then it is accepted as the new best solution. Even if the cost is not better, there is a possibility of accepting this solution as the next solution of the local search phase. This happens according to the Simulated Annealing local search scheme that accepts inferior solutions that are close to the best solution with a probability that degrades while the process continues. Finally the best solution encountered is the result of the local search stage.

It has to be mentioned that a number of Kempe chains are produced for every two selected periods. Our algorithm starts by processing the longer chains first in an effort to disrupt more the current solution. In case a better solution is found remaining Kempe chains of the periods under consideration are not examined. A critical point for the success of the search method has to do with the mechanism of selecting periods that will be used in order to search for Kempe chains. Poor selection might result in cycles of non improving steps and in ignoring parts of the search space that could have been useful otherwise. Our design of this part of the application proposes in each step period pairs with relatively minor involvement in previous steps and also periods that in the past resulted in successful moves. A memory structure is used in order to achieve this. The first period of a pair is selected according to the relative success of previous moves examining all the periods. After selecting the first part of a pair then all the other periods are examined according to past successful moves for both the periods.

Another point of interest is that our use of the simulated annealing metaheuristic involves a reheating scheme additional to the typically used cooling scheme. When a number of local search steps is examined without producing a solution to replace the currently best one, parameter temperature rises resulting in greater probability of acceptance of inferior solutions. Our experiments showed that a starting temperature of 10 degrees Celsius, cooling and heating schemes of geometric nature and a typical exponential function of accepting solutions with less good objective function values produce relatively good results.

3.4 Third stage – improvements per period

The timetable produced when the local search phase has finished is further examined in a period by period basis so as to discover and remedy situations that might result in further cost improvements. Cost can be reduced by not using rooms with high penalty and avoiding putting in the same room exams with mixed durations. The following Integer Programming model was formulated. Let E be the set of exams, R the set of rooms and K the set of discrete exams durations. Then s_i is the number of students enrolled in exam i, c_j is the capacity of room j, p_j is the penalty of using room j, Pnmd is the penalty of having mixed exams durations in a room and d_{ik} is a parameter having value 1 when exam i has duration equal to the k^{th} discrete duration from set K

and 0 otherwise. Let x_{ij} be a binary variable which is supposed to take value 1 when exam i is scheduled in room j and 0 otherwise. Let y_j be an integer variable that counts the number of exams scheduled in room j and finally let z_{kj} be a binary variable with value 1 when distinct duration k is among the durations of the exams scheduled in room j.

Min $$\sum_{j=1}^{R}\left(p_j y_j + \sum_{k=1}^{K} z_{kj} Pnmd \right)$$

Subject to $$\sum_{i=1}^{E} s_i x_{ij} \le c_j \qquad \text{j in 1..R}$$

$$\sum_{j=1}^{R} x_{ij} = 1 \qquad \text{i in 1..E}$$

$$\sum_{i=1}^{E} x_{ij} = y_j \qquad \text{j in 1..R}$$

$$\sum_{k=1}^{K} d_{ik} z_{kj} \ge x_{ij} \qquad \text{i in 1..E, j in 1..R}$$

$x_{ij} \in \{0,1\}$, $z_{kj} \in \{0,1\}$, $y_j \in Z^+$

We solve the above model for each one of the periods using the open source mathematical solver GLPK. GLPK employs a branch and bound algorithm with LP relaxations for solving IP problems. A few parameters had to be tuned in order to get good results in a timely manner. So in our problem branching occurs on the most fractional integer variables while the backtrack scheme used is depth first search. Experimenting with the model we also find out that the solution process is greatly accelerated by using what GLPK calls advanced MIP solver which is also a branch and bound with the enhancements of presolving the MIP problem and generating cutting planes to improve its LP relaxation. We have observed that due to small objective value variations in the feasible solution space, GLPK tends to spend a significant amount of time without proving optimality and so we decided to enforce a time limit criterion for each subproblem.

Our incentive for programming the third stage was that the solutions produced by the previous stages were likely to be consisted of suboptimal assignments of exams to rooms revealed by examining each period in isolation. Those suboptimal fragments of the timetable are easily located by a human operator and their presence in a final solution might give the impression of a low quality outcome for the whole process.
Unfortunately the time available for this stage is limited so periods are ordered according to the value of their penalties resulting from rooms and mixed durations.

4. Experiments

Our experiments used the eight provided datasets from ITC07 which according to the organizers of the competition represent a relatively real model of the examination timetable problem in terms of data, constraints and evaluation. Four more datasets which are called 'hidden' will be released at a time later to the writing of this paper. The variability of the provided datasets is significant not only according to the size of the problems but also with respect to the IMI index.
Among the datasets, dataset 4 is rather special because it has only one room rendering stage 3 of our procedure useless and a much higher value of conflict density compared with the remaining datasets. High conflict density value makes successful backtracking harder. In our initial experiments the backtracking mechanism was implemented by examining all periods and choosing the one that resulted in the fewer number of dislocating already scheduled exams. Unfortunately when the above mechanism operated in dataset 4, it led to cycles of removing big exams of the timetable and after a number of steps inserting the same exam again resulted in mass removals of smaller exams. The above problem was not alleviated even with the presence of taboo lists. Finally, the adoption of the backtracking mechanism described in 3.2 eliminated this problem.
The program ran on a simple workstation equipped with java runtime environment. In this machine the benchmarking routine provided by the organizers of the competition showed that the acceptable run time for our application had to be no more than 422 seconds. Our application adhered to this limit and distributed time into the three stages giving 30% of the time to the first stage, 50% to the second and the remaining 20% to the third. In special cases like dataset 4 where stage 3 is obsolete its time was appended to stage 2.

4.1 Programming Issues

In order to achieve the limited available execution time a number of enhancements was considered. Redundant data structures were designed and optimized versions of the functions that compute the penalty incrementally were coded. A preprocess step creates for each exam extra data that makes positioning exams and backtracking easier. For example if an after constraint exists for exam E1 that links it with exam E2 and E1 has a coincidence constraint with E3 then the same after exam constraint for E3 to E2 is added. Another example also about coincidence exams is the following. If two or more exams with coincidence constraints between them have different duration then the longest duration must be assumed for all the exams.

Our program is implemented in Java 5 despite the slowest runtime performance of Java compared to compiled programming languages. We believe that this disadvantage is balanced by the ease of programming complex algorithms and Java's rich API especially regarding data structures.

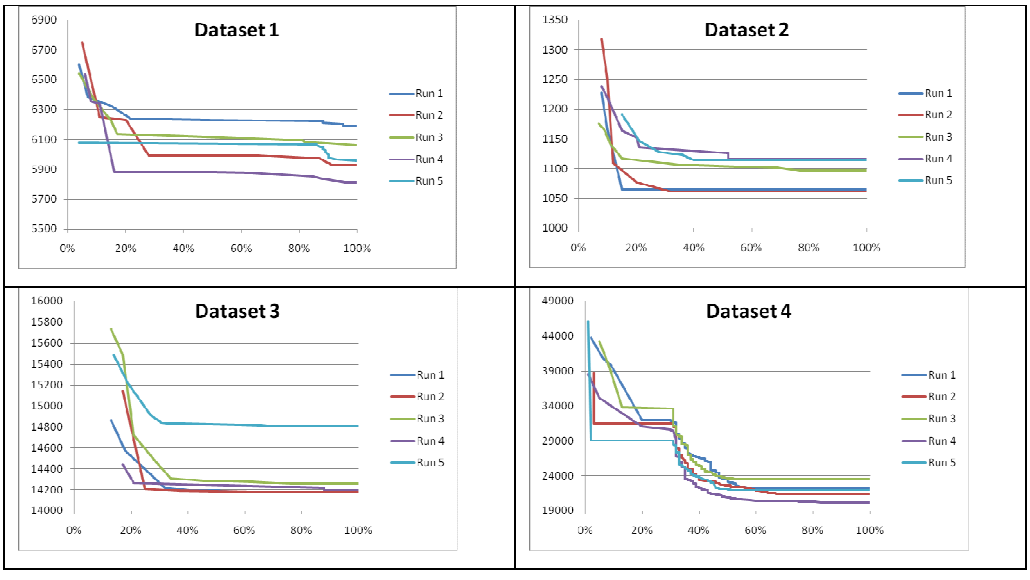
4.2 Results

The results obtained using the approach of this paper can be seen in Table 2. The second and third columns show the best and the average penalty that our implementation produced after a number of runs respecting the same limit in execution time for each individual run. We can observer that the best and average values are relatively close between them indicating robustness for the approach.

Five runs for each one of the eight datasets are presented in Table 3. For some of the datasets the second and third stage of the algorithm fail to reduce the cost significantly. We believe that this happens because the proposed search on a neighborhood structure which is based on Kempe chains does not visualize all the problem constraints. The problem is that in some cases local search moves are able to locate improved solutions without exam conflicts but other constraints like room constraints, period constraints, etc failed to be satisfied because a more global view of the problem is needed. As a result a significant percentage of intermediate solutions during the local search process are invalid.

	Best Penalty	Average Penalty
Dataset 1	5.814	5.914
Dataset 2	1.062	1.091
Dataset 3	14.179	14.336
Dataset 4	20.207	21.846
Dataset 5	3.986	4.167
Dataset 6	27.755	28.361
Dataset 7	6.885	7.010
Dataset 8	10.449	10.796

Table 2



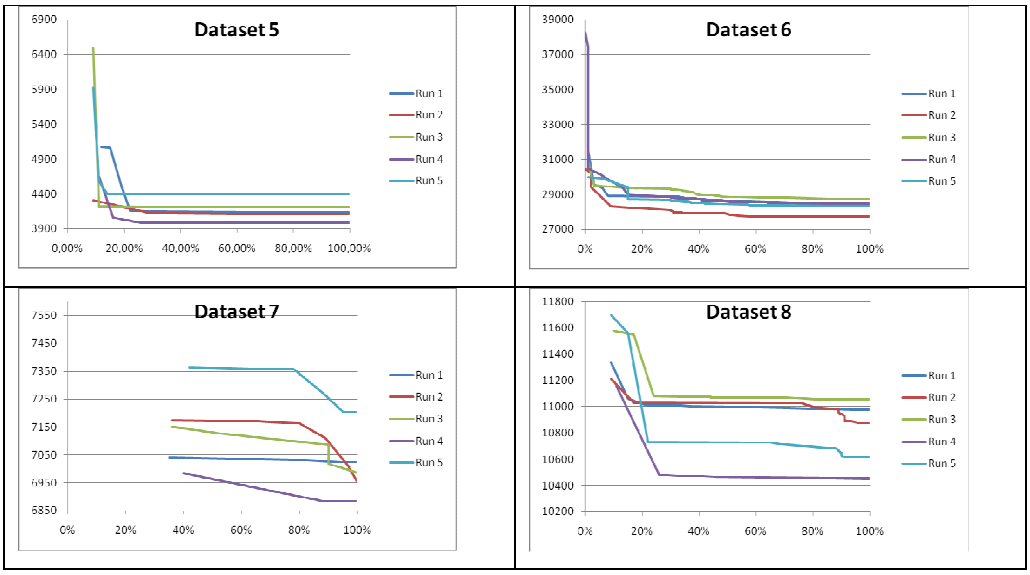


Table 3

5. Conclusions - Future Work

In this paper we presented an algorithmic approach for the solution of the examination timetabling problem as it was defined in the first track of ITC07. The solution process has been partitioned into stages and a number of heuristics, metaheuristics and exact methods have been used. The limited execution time was a critical factor that heavily affected our approach. The solutions appear satisfactory but it is also clear that further improvements are possible. This is especially true if the execution time limit is relaxed. We plan to experiment with fuzzy multiple ordering (Asmuni, Burke and Garibaldi, 2005) during the construction phase because two or more ordering criteria can be used simultaneously using a fuzzy expert system. The local search phase can also be improved by using a variable neighborhood search schema (Hansen and Mladenovic, 2003) that will enable the systematic change of the neighborhood using as an alternative neighborhood structure the one resulting from single exam moves. We plan to capitalize on the experience gained from our involvement with ITC07 by modifying our algorithm in order to solve the examination problem for the colleges and universities in Greece. The main modeling extensions to the ITC07 competition involve the existence of an availability schedule for the lecturers and the various rooms and the uniformity of the duration of the exams.

References

Asmuni H., Burke E., Garibaldi J. and McCollum B. (2005). Fuzzy multiple heuristic orderings for examination timetabling. PATAT 2005, LNCS 3616, pp 334-353. Springer-Verlag Berlin Heidelberg.

Burke E.K. and Newall J.P. (2004). Solving examination timetabling problems through adaptation of heuristic orderings. Annals of operations research 129, 107-134. Kluwer Academic Publishers, Netherlands.

Burke E.K. and Petrovic S. (2002). Recent research directions in automated timetabling. European Journal of Operational Research, 140, 266-280.

Carter M.W., Laporte G., Lee S.Y. (1996). Examination timetabling: Algorithmic strategies and applications. Journal of Operational Research Society, 47, 373-383.

Casey S. and Thompson J. (2003). GRASPing the Examination Scheduling Problem. PATAT 2002, LNCS 2740, pp. 232-244. Springer-Verlag Berlin Heidelberg.

Dowsland K.A. (1993). Simulated Annealing. Modern heuristic techniques for combinatorial problems by C. Reeves. John Wiley & Sons Inc. New York, USA. ISBN 0-470-22079-1.

Feo T.A. and Resende M.G.C. (1995) Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, 6, 109-133.

Glover F., Laguna M. (1997). Tabu Search. Kluwer Academic Publishers, Boston, MA.

Hansen P. and Mladenovic N. (2003). Chapter 6. Variable Neighborhood Search. Handbook of Metaheuristics by F. Glover and G. Kochenberger. Kluwer. ISBN: 978-1-4020-7263-5

McCollum B. (2007). A Perspective on Bridging the Gap Between Theory and Practice in University Timetabling. PATAT 2006, LNCS 3867, pp 3-23, ISBN 978-3-540-77344-3.

McCollum B. and McMullan M. (2007). The Second International Timetabling Competition: Examination Timetabling Track. Technical Report: QUB/IEEE/Tech/ITC2007/Exam/v4.0/17. September 20, 2007.

Qu R., Burke E., McCollum B. Merlot L. and Lee S. (2006). A Survey of Search Methodologies and Automated System Development for Examination Timetabling. Computer Science Technical Report No. NOTTCS-TR-2006-4.

Resende M.G.C. and Ribeiro C.C. (2003). Chapter 8. Greedy Randomized Adaptive Search Procedures. Handbook of Metaheuristics by F. Glover and G. Kochenberger. Kluwer. ISBN: 978-1-4020-7263-5.

Thompson J.M. and Dowsland K.A. (1996). Variants of Simulated Annealing for the Examination Timetabling Problem. Annals of Operations Research Volume 63, Number 1. Springer.