

A Memetic Algorithm for Solving a Timetabling Problem: An Incremental Strategy

Ender Özcan, Alpay Alkan

Yeditepe University, Computer Engineering Department, İstanbul, Turkey,

{eozcan, aalkan }@cse.yeditepe.edu.tr

Timetabling problems are well known complicated constraint satisfaction problems. A new real-world course timetabling problem is described in this study. The difficulty in approximating to an optimal solution in a reasonable time increases for the large problem instances regardless of the algorithm used. An incremental strategy that aims to relieve this difficulty to some extent combined with a memetic algorithm (MA) is investigated for solving the new timetabling problem. The incremental MA is a multistage approach that enlarges the size of the candidate solutions by adding a selected subset of unscheduled course meetings at a stage. A solution is then sought for all the course meetings at hand. The initial results show that the incremental MA is promising.

Keywords: Memetic Algorithms, Timetabling, Multistage Approach, Hill Climbing.

1. Introduction

The high-school graduates in Turkey enter a country-wide *Student Selection Examination* (OSS), in order to get admitted to a higher education program at a university. The *Student Selection and Placement Center* (OSYM) has been providing a centralized system for fair access and placement of students to higher education programs since 1974. Currently, OSYM administers OSS. The students are placed into the higher education programs according to their OSS score and set of choices. OSS is arranged every year. Less than 40% of the applicants can continue their education in an undergraduate degree program. This situation causes a high-level competition among students. For this reason, there are many private OSS preparation schools in Turkey established as a support mechanism for the candidates. Such schools opening more branches started to deal with the difficult problem of scheduling the course meetings properly subject to a set of constraints.

The timetabling problems are NP complete [11] real-world problems. A comprehensive survey on timetabling can be found in [7] particularly discussing the MAs. Course timetabling problems (CTPs) are a subset of timetabling problems that require an optimal arrangement of resources for the employees (e.g., teachers), employers (e.g., school administration) and students at an educational institution subject to a set of constraints. There are two types of constraints in timetabling: *hard* and *soft*. Hard constraint violations represent highly undesirable situations, while soft constraints are less essential preferences. The constraints can be categorized further for the practical timetabling. In this paper, a new course timetabling problem is introduced, University Exam (OSS) Preparation School Timetabling Problem (PSTP). PSTP is analyzed initially by Alpay Alkan during his studies on timetabling.

Alkan and Ozcan introduced a violation directed hierarchical hill climbing method (VDHC) in [3] for solving the university course timetabling problem. VDHC performs a local search over the *constraint oriented neighborhoods* as described in [29] based on the constraint types, their violations and structure of the problem. Ozcan extended the study and suggested a heuristic template for designing a set of operators in [22]. A variety of genetic operators and self-adjusting hill-climbers based on this template have already been experimented within the Memetic Algorithm (MAs) for solving different timetabling problems in [21] and [23]. The VDHC methods within the MAs that managed multiple constraint based hill climbers turned out to be successful in timetabling. In this study, an incremental MA is proposed for solving PSTPs. The performance of the incremental MA is compared to the conventional one both using the VDHC on real PSTP instances, each requiring a large set of course meetings to be scheduled.

2. Preliminaries

2.1. High-school Course Timetabling Problem

The university and high-school course timetabling problems are the subsets of CTPs that are commonly dealt with by numerous researchers. There is a variety of approaches used for solving such problems that have different types of constraints. In general, the high-school course timetabling differs from the university course timetabling. Besides the set of constraints, the timetable for a student at a high-school has less empty time-slots as compared to a student at a university. PSTPs form another subset of CTPs that are more similar to the high-school timetabling problems.

There are exact and inexact approaches for solving the high-school timetabling problems. The researchers focus on the inexact methods more, including heuristics, meta-heuristics and *hyper-heuristics* [5], [19]. Hyper-heuristics perform a search over a set of heuristics. Most of the existing hyper-heuristic combine two decision making strategies. A heuristic selection mechanism is used for choosing a heuristic from a lower set of heuristics, while an acceptance mechanism is used for deciding whether to accept or reject a candidate solution [4]. Different meta-heuristics for high-school course timetabling have been studied by many researchers, such as simulated annealing [1], [2], evolutionary algorithms [10], [13], [25], tabu search [16], [28]. Colomi et. al. [9] compares these approaches over an Italian high-school data.

2.2. University Exam Preparation School Timetabling Problem (PSTP)

OSS is a single stage exam having two parts, in which two aptitudes of the entering candidates are assessed: verbal and quantitative. Computing the score of a candidate's exam requires some transformations of the raw result by taking into account the grade-point average of the student at school, the difficulty level of each question that is determined statistically, etc. After the transformations are applied, three different composite score types can be computed: *verbal*, *quantitative* and *equally weighted* OSS scores. One of these scores is used in the selection of those candidates who will be considered for the placement to the undergraduate programs. Each department at a Turkish university admits students according to a specific OSS score type. For example, a computer engineering department accepts students based on their quantitative OSS scores. Hence, the students aim to maximize one of those OSS score types to get admitted to a department according to their wish at a university by correctly answering the relevant questions. The private preparation schools (PPSs) act as a support mechanism for the students that will enter the OSS. A student can be admitted to a PPS at any time during his/her high-school education. The high-school education has been extended from 3 to 4 years in Turkey.

In a PPS that has several branches at different locations, there are teachers that circulate around these branches, where each registered student attends a set of courses at a specific branch. A student gets prepared to maximize one of the scores, verbal, quantitative or equally-weighted. A *division* indicates the score type that a student aims to collect during OSS. The curriculum of a verbal second-grade (*year*) high-school student in a PPS differs from that of a verbal or quantitative third-grade high-school student. Hence, the set of courses offered for each student differs according to his/her division and *grade*. At a branch, depending on the number of registered students, several sections might be arranged to cover them all. Depending on the high-school, the classes might be held before noon, in the afternoon or during the whole day. Moreover, some OSS applicants are high-school graduates. Consequently, PPSs have to arrange course meetings accordingly. For example, eight different sections might be required for all quantitative second-grade high-school students. Some of these sections might require the courses to be scheduled in the morning and some of them in the afternoon during the weekdays. So, all grades are further divided into grade sections. The third-year high-school students that are in the quantitative division must take mathematics, natural sciences and Turkish language courses. All the students that are in the verbal or equally weighted divisions must take some courses in social sciences such as geography and history as well as mathematics and Turkish language courses. But, the number and length of the meetings that must be assigned to the grade sections (of different divisions) can differ. For example, a student in a grade section of an equally weighted division must attend four meetings of the

geography course, whereas a student in a grade section of a verbal division must attend six meetings of the geography course. Each grade section groups a set of course sections that a student must attend. A course section denotes a set of meetings for a course.

In a University Exam Preparation School Timetabling Problem (PSTP), an optimal assignment of all events that places course-section meetings into p periods in a timetable of d days by t time-slots per day is searched. This assignment is subject to a set of constraints. If the total number of course meetings is M , then the search space size becomes M^p . The traditional approaches might fail to obtain an optimal solution. An assignment in a PSTP is an ordered pair (x,y) , where $x \in E$ (set of course-section meetings) and $y \in T$ (domain, set of periods). The interpretation of this assignment in terms of PSTP is: "Course section meeting x starts at time y ". Most of the teachers in a PPS are part time teachers having preferences that are treated as hard constraints. There might be inexperienced teachers who are organized to enter some courses together with the experienced teachers. Each teacher is assigned to a course section beforehand and can take responsibility for more than one course-section. Some of these course sections might be held in different branches. Hard constraints are as follows:

- C1. Each meeting of a course section should be assigned to a period in a different day
- C2. Meetings in a grade section cannot overlap
- C3. Meetings of a teacher cannot overlap
- C4. A teacher can prefer certain days and/or periods
- C5. The administration can prefer certain days and/or periods for some course meetings
- C6. Each grade section excludes some periods from before or after noon
- C7. Some course meetings can be scheduled at the same time for some grade sections
- C8. More than one teacher can be assigned to a section of a course

Soft constraints are as follows:

- C9. There shouldn't be an interleave of more than some given number periods between the meetings of a teacher in each day so that a teacher would not wait too long for its next meeting
- C10. There is a minimum and a maximum load per day for teachers

Solving the PSTP requires the generation of a *feasible* timetable containing a collection of assignments one per course section meeting with the minimum number of constraint violations. A few number of C5, C7 and C8 constraints appear among all constraints in a problem.

3. An Incremental Strategy and Memetic Algorithms for PSTPs

Memetic Algorithms (MAs) represent a set of hybrid algorithms that combine Genetic Algorithms (GAs), advocated by Holland [17] and local search. An *individual* (chromosome) in a GA represents a candidate solution for a given problem. Each *gene* that composes an individual receives an *allele* value from a set of values. For example, in a binary representation, a gene gets an allele value from the set $\{0,1\}$. In an iterative cycle, a randomly generated *population* of individuals evolves towards an optimal solution by undergoing a set of genetic operators, namely *crossover*, *mutation* and *selection*. In a conventional MA, a hill climbing method is applied after the mutation occurs. The GAs and MAs are successfully used in solving a set of difficult search and optimization problems, including the real-world timetabling problems ([3], [14], [18]–[25]). In most of the meta-heuristic approaches, as the problem size increases, the speed might become an issue. A *reasonable* amount of time should be spent on obtaining a *reasonable* schedule for the timetabling of large problem instances. That's why Carter uses a divide-and-conquer approach in solving such problems [8]. The traditional approaches such as integer programming are suggested for solving sufficiently small problem instances during the conquering step. Weare [30] and Burke and Newall [6] showed the potential in applying a multistage approach for solving timetabling problems.

Most of the approaches for timetabling can be converted into a multistage approach based on a strategy that somehow selects and handles a subset of events during each stage. Three different types of such strategies can be identified. In the first type, the stages can be arranged such that the approach is applied only to a selected subset of events. Once a satisfactory solution is obtained, based on some criteria only for these events subject to the constraints, the assignment for each

event is fixed. Then the next subset is processed as shown in Figure 1(a). As a second type of strategy, the approach can be applied to the union of an unprocessed subset indicating some unscheduled events and a subset of some previously processed events as illustrated in Figure 1(b). Still some scheduled events are fixed in this strategy and they are not processed further by the approach. Both of these strategies are evaluated in [30] and [6]. They do not seem to be that promising for most of the common real-world timetabling problems such as nurse rostering, or course timetabling. Whenever a solution to a subset is fixed, the approach is disallowed for exploring some part of the search landscape, which might contain a promising area. A third extreme strategy is utilized in this study. No assignment of events is fixed. At each stage, a subset of unscheduled events is incrementally added to be processed by the approach along with the previously processed subset of events as in Figure 1(c). The aim of this study is to inquire whether such an incremental multistage approach can provide a better performance as compared to its single stage version.

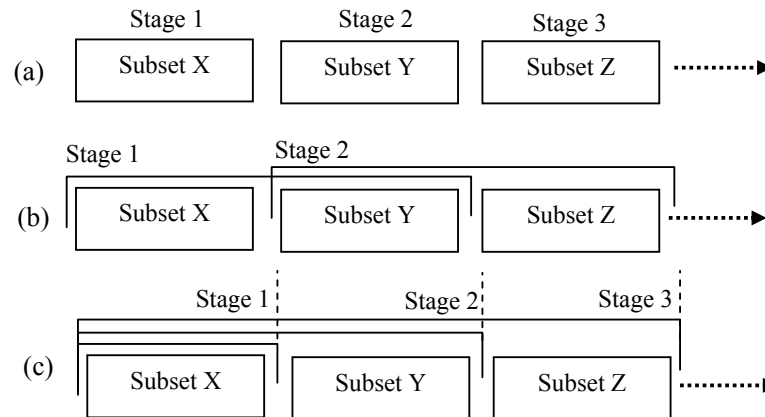


Figure 1. Possible ways to arrange stages in a multistage approach for timetabling

As an incremental multistage approach, an MA is used for solving PSTPs as illustrated in Figure 2. At each stage, a subset of new (unscheduled) events is selected using certain criteria. Un-scheduled events for the selected subset of events are randomly generated within a population of candidate solutions. This population is exposed to the traditional MA operators. Whenever some stage termination criteria is satisfied, then another subset of new events is chosen. This process is repeated until all events are scheduled and some additional termination criteria is satisfied. After the first pass, each individual in the initial population is a partial solution and has a small size. At each stage, the size of individuals incrementally grows as the new subset of events is added for optimization. In this way, no portion of the search landscape is ignored.

Repeat

Select a group (subset) of new events based on some criteria
Generate random assignments for the new events in all individuals
Repeat

Apply Crossover, Mutation and then Hill Climbing

Until stage termination criteria₁ are satisfied

Until all events are scheduled and termination criteria₂ are satisfied

Figure 2. Pseudo-code for an incremental Memetic Algorithm for timetabling

An individual contains all course (section) meetings to be scheduled and its physical implementation reflects the same logical arrangement in Figure 3. A gene corresponds to a course section in the representation used. All course section meetings are generated randomly without any clash for all individuals within the initial population. Similarly, the traditional mutation randomly reschedules the meetings in a course section with no clash using a probability of $1/\text{no_of_course_sections}$. By this way, C1 is satisfied at all times. Two crossover operators are implemented forming two new individuals. Modified one-point crossover (m1PTX) swaps parts of

selected individuals at a selected point. As a crossover point, one of the start points of the grade sections is randomly chosen. A modified uniform crossover (mUX) exchanges course meetings in each grade section as a whole with a probability of 1/2.

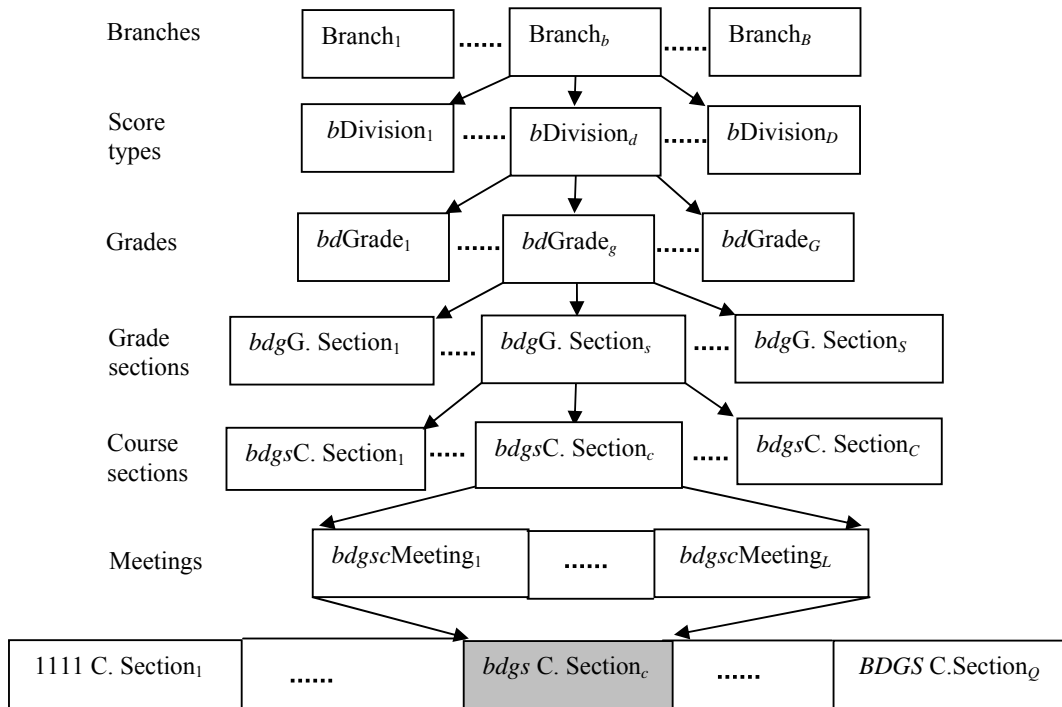


Figure 3. The logical organization of the course (section) meetings in a PSTP and the representation used in the MAs, where “*bdsg C. Section_c*” denotes all course meetings of the c^{th} course section of the s^{th} grade section from the g^{th} grade of the d^{th} division in the b^{th} branch of a school

For C7, the course sections that should be scheduled to the same periods have just a single gene in the representation pointed by them. Similarly, each course section that will be taught by more than one teacher has a single gene pointed by the related teachers for C8. The relevant violations are detected by using the list of each teacher and course section assignments. No assignment, out of the restricted domain of each course meeting due to C4-C6 is allowed during the evolution. The MAs utilize a set of constraint based hill climbers for C2, C3 and C9, C10. Each hill climber attempts to remove the violations due to a constraint by random rescheduling. A limited number of possibilities are tested and the best one is accepted. The hill climber for C2 is applied after all genetic operators.

The idea of using a violation directed mutation operator in a genetic algorithm is tested over a set of real and syntactic data for lecture and examination timetabling by Ross et. al. ([26], [27]). Their results show that the random selection of a gene and then the selection of an allele to be assigned by using tournament perform the best. The tournament strategy favors the assignment that produces less number of violations. Extending this idea further, a mechanism called violation directed hierarchical hill climbing (VDHC) as described in [3] and [21] is used to manage the hill climbers for C3, C9 and C10. More details on the VDHC can be found in [22]. At each step, the number of violations due to a constraint type is computed and a hill climber is randomly selected using a tournament strategy (with a tour size of 2) based on this information. Each hill climber aims to correct the violations of the related constraint type. Then the selected hill climber is applied to a group of course meetings. The VDHC makes a hierarchical traversal over the groups of course meetings based on the static organization of events (Figure 3) until there is no improvement or a maximum number of steps is exceeded. The fitness function is a weighted sum of the number

of all constraint violations. Each conflict counts as one violation in C2 and C3, while the number of violations is the total deviation from the limits for C9 and C10. For example, if a teacher has a load of 1 in a day and the minimum load is per day is 3; this generates a violation of 2. In the incremental MA, all grade sections receive an additional selected course section simultaneously at a stage. The course section with the largest number of meetings in each grade is added. In case of equality, a random choice is made. The MA at a stage terminates whenever all hard constraints (C1-C8) are satisfied by an individual or a maximum number of steps is exceeded.

4. Experiments

Pentium IV 3 GHz. windows machines having 2 Gb of memory are used during the experiments. All runs are repeated fifty times. *Success rate* (s.r.) indicates the ratio of successful runs, achieving the expected fitness to the total number of runs. A real data obtained from *Final Dershanesi*, a private PPS is used during the experiments. This data is divided into smaller pieces and mixed to perform the initial experiments as summarized in Table 1. Even these small problem instances are difficult enough, necessitating the application of a nontraditional approach. Each problem instance requires an optimal schedule to be generated for more than 1100 course meetings. There are 8 days and 12 hours per day in the timetable. For most of the courses, one hour is required for each meeting. The students in a grade section attend 45 to 48 course meetings. Each course section requires 2 to 8 course meetings. The interleave in C9 is fixed as one. The minimum and maximum load of a teacher imposed by C10 is set to 2 and 6 hours per day, respectively. The benchmark data is available at <http://cse.yeditepe.edu.tr/~eozcan/research/TTML/>.

Table 1. The characteristics of the experimental data set, where *minl* and *maxl* denote the minimum and maximum total load of the teachers for a given problem, respectively

<i>label</i>	<i>No. of Meetings</i>	<i>No. of Branches</i>	<i>No. of Divisions</i>	<i>No. of Grades</i>	<i>Grade Sections</i>	<i>Course Sections</i>	<i>No. of Teachers</i>	<i>minl</i>	<i>maxl</i>
fd1	1107	1	3	1	30	374	41	5	46
fd2	1128	2	4	3	30	322	49	6	40
fd3	1131	2	4	3	30	322	50	8	42
fd4	1163	2	3	2	30	342	42	6	46
fd5	1166	2	3	2	30	342	42	8	41
fd6	1187	1	4	3	30	290	48	3	46

The incremental MA is compared to the conventional MA, which attempts to schedule all course meetings simultaneously. For a fair comparison between the approaches, the experiments are terminated if the execution time exceeds 600 CPU seconds or the expected global optimum is achieved. If there are no constraint violations, 0 fitness value is expected. Equal weights are used within the fitness function. Tournament strategy is used to choose individuals for crossover with a tour size of four. Crossover is applied to the individuals with a probability of 0.5. As a replacement strategy, the best two individuals in a generation are passed to the next one. The rest of the population is generated using the genetic operators. Some preliminary experiments are performed using the multistage MA. Population size of 16 and 32 are compared. The results show that a small population is a slightly better choice. It is also observed that the crossover mUX is slightly better than m1PTX on average. For this reason, mUX is chosen as the crossover and a population size of 16 is used during the further experiments. The rest of the settings are kept the same. The comparison of the MAs is presented in Table 2. The incremental approach performs better than the conventional MA in almost all cases. For all of the problem instances all constraints are left unresolved, except fd5. The success rate for the incremental MA on fd5 is 0.62, while it is 0.48 for the conventional MA. For fd2, the incremental MA achieves the same quality solution as the conventional one by visiting less number of states on average.

Table 2. Comparison of the MAs, where *viol.*, *gen.* denote violations and generations, respectively

label	Incremental MA					Conventional MA				
	best	avr. viol.	std.	avr. gen.	std.	best	avr. viol.	std.	avr. gen.	std.
fd1	19	28.7	5.4	1323.3	5.4	39	52.1	8.4	1264.0	34.1
fd2	2	4.3	1.9	1152.0	20.7	2	5.4	3.1	1301.4	36.7
fd3	7	12.8	3.0	1165.1	28.6	12	19.1	4.2	1292.9	17.8
fd4	27	48.4	8.0	1204.0	25.6	49	69.2	9.3	1213.3	34.9
fd5	0	0.6	1.0	888.4	339.9	0	0.9	1.1	1030.8	472.8
fd6	2	9.7	3.9	1128.0	34.1	11	19.8	5.0	1285.9	21.5

5. Conclusions

A new course timetabling problem is presented in this paper: University Exam Preparation School Timetabling Problem (PSTP). The search space is extremely immense even for a *small* PSTP instance. A set of such problem instances is provided, each requiring an optimal schedule for more than 1100 course meetings subject to a set of hard and soft constraints. An incremental strategy is presented to convert a single stage approach into a multistage approach for timetabling. In order to obtain a *good* solution in a *reasonable* amount of time for a PSTP instance, a memetic algorithm based on such a strategy is proposed. For this purpose, in addition to the incremental approach, the population size is kept small within the MA. The comparison between the incremental MA and its conventional version shows that the incremental one achieves better results. Both MAs use the same operators, including the VDHC; a heuristic that decides the most appropriate hill climber to apply whenever necessary from a set of constraint based hill climbers.

In a PSTP, the search landscape is highly multimodal and immense due to the number of course meetings to be scheduled and the constraints to be satisfied. It is likely that an approach for solving this problem might get stuck at a local optimum. It seems this is what happens in case the conventional MA is utilized with a small population size. The proposed incremental strategy acts as a diversification mechanism within the MA due to the newly introduced events and their random scheduling at each stage. The search space size also grows gradually along with the problem size. This process seems to alleviate the burden of solving a large problem. At each stage, a new search starts over an enlarged space using the partial solutions from the previous stage. The proposed incremental strategy is a universal strategy that can be adapted easily by the existing approaches for timetabling and scheduling. Different mechanisms for the subset selection and the termination criteria at each stage can be investigated further.

Acknowledgement

This research is funded by TÜBİTAK (The Scientific and Technological Research Council of Turkey) under the grant number 107E027.

References

- [1] D. Abramson (1991), Constructing School Timetables Using Simulated Annealing, Sequential and Parallel Algorithms. *Management Science* **37**(1), 98 – 113.
- [2] D. Abramson, H. Dang, and M. Krisnamoorthy (1999), Simulated Annealing Cooling Schedules for the School Timetabling Problem, *Asia–Pacific J. of Op. Res.* **16**, 1 – 22.
- [3] A. Alkan and E. Ozcan (2003), Memetic Algorithms for Timetabling, *Proc. of IEEE Congress on Evolutionary Computation*, 1796 – 1802.
- [4] B. Bilgin, E. Ozcan, E.E. Korkmaz (2006), An Experimental Study on Hyper-Heuristics and Exam Scheduling, *Proc. of the 6th Int. Conf. on PATAT*, 123 – 140.
- [5] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu (2007), A Graph-Based Hyper Heuristic for Educational Timetabling Problems, *EJOR* **176**(1), 177 – 192.

- [6] E.K. Burke, J.P. Newall (1999), A Multistage Evolutionary Algorithm for the Timetable Problem, *IEEE Trans. on Evolutionary Computation* **3**(1), 63 – 74.
- [7] E.K. Burke, J.D. Landa Silva (2004), The Design of Memetic Algorithms for Scheduling and Timetabling Problems. Krasnogor N., Hart W., Smith J. (eds.), *Recent Advances in Memetic Algorithms*, Studies in Fuzziness and Soft Computing, vol. 166, Springer, 289 – 312.
- [8] M.W. Carter (1983), A Decomposition Algorithm for Practical Timetabling Problems, Dept. of Industrial Engineering, University of Toronto, Working Paper 83-06, April.
- [9] A. Colomi, M. Dorigo and V. Maniezzo (1992), A genetic algorithm to solve the timetable problem, Politecnico di Milano, Italy, Tech. rep. 90-060 revised.
- [10] W. Erben, J. Keppler (1995), A Genetic Algorithm Solving a Weekly Course– Timetabling Problem, *Proc. of the First Int. Conf. on the Practice and Theory of Automated Timetabling (ICPTAT)*, Napier University, Edinburgh, 21 – 32.
- [11] S. Even, A. Itai, and A. Shamir (1976), On the Complexity of Timetable and Multicommodity Flow Problems, *SIAM J. Computing* **5**(4), 691 – 703.
- [12] H.L. Fang (1994), *Genetic Algorithms in Timetabling and Scheduling*, PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Scotland.
- [13] G.R. Filho, L.A.N. Lorena (2001), Constructive Evolutionary Approach to School Timetabling. *Lecture Notes in Computer Science* **2037**, Springer, 130 – 139.
- [14] D.E. Goldberg (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, Reading (MA).
- [15] C. Head and S. Shaban (2007), A heuristic approach to simultaneous course/student timetabling *Computers & Operations Research* **34**(4), 919 – 933.
- [16] A. Hertz (1992), Finding a feasible course schedule using a tabu search. *Discrete Applied Mathematics* **35**, 255 – 270.
- [17] J.H. Holland (1975), *Adaptation in Natural and Artificial Systems*, Univ. Mich. Press
- [18] N. Krasnogor (2002), *Studies on the Theory and Design Space of Memetic Algorithms*, Ph.D. Thesis, University of the West of England, Bristol, United Kingdom.
- [19] S.A. MirHassani (2006) A computational approach to enhancing course timetabling with integer programming, *Applied Mathematics and Computation* **175**(1), 814 – 822.
- [20] P. Moscato and M.G. Norman (1992), A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message–Passing Systems, *Parallel Computing and Transputer Applications*, 177 – 186.
- [21] E. Ozcan (2005) Memetic Algorithms for Nurse Rostering, P. Yolum (Eds.): *Lecture Notes in Computer Science 3733*, Springer-Verlag, The 20th ISCRIS, 482 – 492.
- [22] E. Ozcan (2006), An Empirical Investigation on Memes, Self-generation and Nurse Rostering, *Proc. of the 6th Int. Conf. on the PATAT*, 246 – 263.
- [23] E. Ozcan, E. Ersoy (2005), Final Exam Scheduler-FES, *Proc. of 2005 IEEE Congress on Evolutionary Computation* **2**, 1356 – 1363.
- [24] E. Ozcan and E. Onbasioglu (2007), Memetic Algorithms for Parallel Code Optimization, *Int. J. on Parallel Processing* **35**(1), 33 – 61.
- [25] B. Paechter, R.C. Rankin, A. Cumming and T.C. Fogarty (1998), Timetabling the Classes of an Entire University with an Evolutionary Algorithm, *Proc. of Parallel Problem Solving from Nature (PPSN V)*, 865 – 874.
- [26] P. Ross, D. Corne and H.-L. Fang (1994), Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation, *Proc. of PPSN III*, 556 – 565.
- [27] P. Ross, D. Corne and H.-L. Fang (1994), Fast Practical Evolutionary Timetabling, *Proc. of AISB Workshop on Evolutionary Computation*, 250 – 263.
- [28] A. Schaerf (1996), Tabu Search Techniques for Large High– School Timetabling Problems, *Proc. of the Fourteenth National Conference on AI*, 363 – 368.
- [29] A. Viana, Pinho de Sousa J., Matos M.A. (2003). GRASP with constraint neighbourhoods - an application to the unit commitment problem. *Proceedings of the 5th MIC*.
- [30] R.F. Weare (1995), *Automated Examination Timetabling*, Ph.D. dissertation, University of Nottingham, Department of Computer Science.