Final Exam Scheduler – FES

Ender Özcan
Dept. of Computer Engineering
Yeditepe University
Kayisdagi, Istanbul, Turkey
eozcan@cse.yeditepe.edu.tr

Ersan Ersoy
Dept. of Computer Engineering
Istanbul Technical University
Maslak, Istanbul, Turkey
eersoy@cse.yeditepe.edu.tr

Abstract- Timetabling problems are constraint optimization problems proven to be NP complete. Furthermore, evaluation of violations is costly, and there is no common data format for representing timetabling problem instances. In this paper, a framework for designing memetic algorithms (MAs) to solve timetabling problems is described and a tool, named Final Exam Scheduler (FES) is introduced. FES is the first tool that accepts Timetabling Markup Language (TTML) documents as input. It utilizes an MA with an adaptive violation directed hierarchical hill climbing method for solving examination timetabling problem instances. Experimental results on a set of benchmark data indicate the success of MA.

1 Introduction

Timetabling problems have been studied by numerous researchers due to its NP complete nature (Even et. al. 1976). There are varieties of timetabling problem classes on which variety of approaches are used. Most of the research in the area focuses on employee shift timetabling, especially; nurse rostering, course timetabling and examination timetabling (Werra 1985, Fang 1994, Schaerf 1996, Alkan et. al. 2003, Burke et. al. 2004, Ozcan 2005b). The aim in examination timetabling is to produce the most *appropriate* schedule for a set of examinations under a given set of constraints.

There is almost no common standard on how to specify a timetabling problem instance and its solution. Hence the results obtained by researchers cannot be compared and benchmarking becomes almost unattainable. Studies for a common data format for timetabling are initiated by Andrew Cumming at ICPTAT'95. Eventually, a language named SSTL (Kingston, Burke et. al. 1997) was emerged. For some reason, SSTL is not used by timetabling researchers. Ozcan (2003) proposed a promising XML based data format for timetabling problems; Timetabling Markup Language (TTML). TTML utilizes MathML for extendibility and generality. Some developers have already taken an interest in the XML standard; SchoolTool (http://www.schooltool.org), Tablix (http://www.tablix.org).

Tabu Search (Glover 1986), Simulated Annealing (Kirkpatrick et. al. 1983) and Genetic Algorithms (Holland 1975, Goldberg 1989) are the most common approaches used by researchers for solving different examination timetabling problem instances. Ross et al. (1994, 1996) introduced a set of violation directed mutation operators based on selecting a gene to mutate and an allele to mutate to, for evolutionary approaches. Their tests show that random selection of a gene, then selecting the allele by using a tournament selection performs the best. Usefulness of hill climbing and local search operators in population based algorithms is emphasized by many researchers (Moscato 1992, Raddcliffe et. al. 1994, Ozcan et. al. 1998, 2004). Burke et al. (1996) applied a light or a heavy mutation, randomly selecting one, followed by a hill climbing method. Investigation of various combinations of Constraint Satisfaction Strategies with GAs for solving examination timetabling problems can be found in Terashima-Marín (1998). Paquete et. al. (2001) applied a multiobjective evolutionary algorithm (MOEA) based on pareto ranking for solving examination timetabling problem in the Unit of Exact and Human Sciences at University of Algarve. Two objectives are determined as to minimize the number of conflicts within the same *group* and between *groups*. Wong et. al. (2002) used a GA utilizing a non-elitist replacement strategy to solve final examination timetabling problem at École de technologie supérieure. After genetic operators are applied, violations are repaired. This fixing process can be considered as a hill climbing procedure. In their experiments, they used a single problem instance.

Carter et. al. (1996) applied backtracking based on different heuristic orderings. Their experimental data is one of the benchmarks used in examination timetabling. Gaspero and Schaerf (2000) explored tabu search approach using graph coloring based heuristics. Merlot et al. (2003) proposed a hybrid approach for solving the final examination timetabling problem that generates an initial feasible timetable using constraint programming, and then applied simulated annealing with hill climbing to obtain a better solution. Burke et. al. (2004b) proposed a general and fast adaptive method that arranges the heuristic to be used for ordering examinations to be scheduled next. Their algorithm yields comparable results on a benchmark of problems with the current state of the art. Petrovic et. al. (2003) proposed a case based reasoning system to generate initial solutions to be used by great deluge algorithm.

In this study, we present a tool, named Final Exam Scheduler (FES) that accepts TTML input for solving final examination timetabling problems, often tackled by universities. FES is the first timetabling tool supporting TTML input. Furthermore, FES utilizes a memetic algorithm (MA), combining genetic algorithms and a violation directed hierarchical hill climbing method. MA, based on the very same framework that we have proposed in Alkan and Ozcan, is tested on a set of timetabling problem instances in TTML.

2 Examination Timetabling Problem

Examination timetabling problems (ETPs) can be represented as a constraint optimization problem by a 3-tuple (V, D, C), where V is a finite set of examinations, possibly examination of courses in a department, faculty or university, D is a finite set of domains of variables and C is a set of constraints to be satisfied:

$V=\{v_1, v_2, \dots, v_M\}$

$D=\{d_1, \dots, d_i, \dots, d_M\}$

$C=\{c_1, c_2, \dots, c_K\}$

Let $G=\{t_1, t_2, \dots, t_N\}$ represent a set of examination start times, then a possible domain of each variable can be $d_i \subseteq G$. Domain of a variable can be a product of sets, each representing a different resource. For example, $d_i \subseteq G x S$ can be a domain of a variable, where S represents the set of classrooms. In this paper, resources other than time will be ignored. ETP can be described as a search for finding the best *assignment* (v_i, t_j) for each variable $v_i \in V$, such that, all the constraints are satisfied. The assignment implies that the examination represented by v_i starts at t_j. The search space size is enormous; M^N, causing the failure of traditional methods.

2.1 Constraints

Constraints can be categorized into two groups; *hard* and *soft*. Hard constraints are the ones that are required to be satisfied, whereas, soft ones represent preferences. Similar to the rest of the timetabling problem instances six different constraint types can be identified for ETPs: *exclusions*, *presets*, *edge constraints*, *ordering constraints*, *event-spread constraints* and *attribute constraints*.

Exclusions represent the excluded domain members of the variables. For example, "CSE 211 exam should not be scheduled on Tuesday(s)", or "ES 112 exam should not be scheduled in the afternoons". Presets represent the predetermined assignments for some variables. For example, "CSE 311 exam must on Friday (in the first week) at 14:00-17:00". Edge constraints are the most common constraint types, representing a pair of course exams that should be scheduled without a clash. Assuming a single timeslot assignment for each course meeting, an edge constraint might require a pair of course meetings v_i and v_k to be assigned to (v_i, t_j) and (v_k, t_l), respectively, such that, $t_j \neq t_l$. Ordering constraints represent an ordering between examinations. For example, "Exams CSE 252 and CSE 416 should start at the same time", or "CSE 311 exam must be scheduled after CSE 103 exam". Event-spread constraints deal with the way how the exams are spread out in time. For example, "Exams of the courses in the last term of a departmental curriculum must be arranged earlier than rest of the courses". Attribute constraints represent restrictions that apply between the attributes of a course and/or the attributes of its assignment. For example, assuming an attribute for a course is the total number of students entering the exam for that course, and an attribute for a classroom is its capacity, a possible constraint would be "Total number of students entering the exam should not exceed the capacity of the classroom". Attribute constraints are ignored, since only G is assumed as a domain of a variable.

2.2 Final Exam Timetabling Problem of Faculty of Engineering and Architecture at Yeditepe University

Students in the Faculty of Engineering and Architecture (FEA) at Yeditepe University choose their courses based on a curriculum. A curriculum consists of 8 terms and 7 courses per term on average. According to the regulations, irregular students with low cumulative grade point averages are enforced to register less than the number of term courses. In the first year (2 terms), almost all the courses are common. In the following terms, there are approximately 20 more courses that are taken in common. Although a course might be offered in more than one section, generally, final examinations are held in common. Sections are merged. All the final examinations must be completed at most in two weeks during fall (1) and spring semesters (2), in one week during summer semesters (3). Each day, 3 hour time slots (9:00-12:00, 13:00-16:00, 16:00-19:00) are allocated for the examinations, hence at most 3 consecutive final examinations can take place in a day. FEA is a growing faculty. Recently, 2 more departments are joined. Arranging final examinations based on the constraints started to become a burden, taking a week or so. Constraints required by FEA is similar to the ones described in Carter et. al. (1996) and Burke et. al. (1996). Ignoring presets and exclusions, we consider the following constraints:

1) No student should be scheduled to two final examinations at the same time.
2) No student should be scheduled to two final examinations in adjacent periods in a day. There must be a free slot between them.
3) During a period, a maximum seat capacity must not be exceeded.

Due to the second constraint, students can attend at most two final examinations per day. FEA requires all the constraints to be satisfied.

3 A Framework for Designing Operators for Solving Timetabling Problems

In most of the timetabling problems, it is possible to identify a hierarchical arrangement of events and resources to be scheduled. Generally, such arrangements are *static*. Dynamic arrangements can be also produced by taking the timetable structure and the assignments into account while searching for an optimal solution. For example, all the events scheduled at each day in a timetable constitute a dynamic arrangement of events. During the search, each new candidate or neighboring solution might generate different groupings in the arrangement. Obviously, more than one such arrangements can be identified for a given timetabling problem instance. These are useful especially, while defining the constraints. For example, Figure 1 illustrates static arrangements arising due to the curriculum based university (faculty) course timetabling problem instance as described in Alkan and Ozcan 2003 at FEA. An example constraint is that no clashes should occur among the courses in a term. We should emphasize that at a level in the hierarchy, although it is possible, a subset does not have to be a partition. For example, the real data contains common technical elective courses in terms 6, 7 and 8.

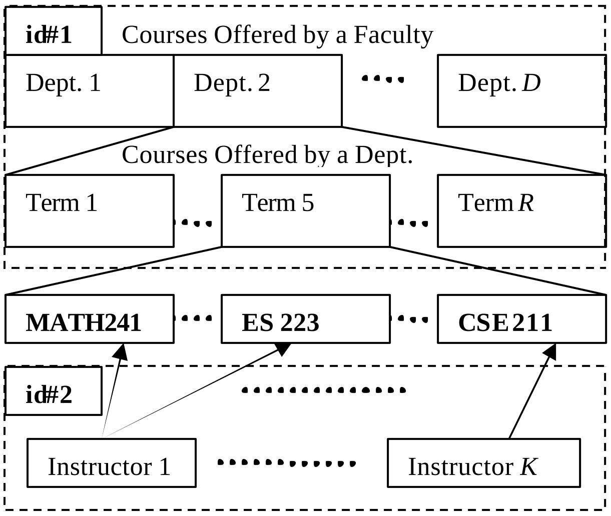


Figure 1. An example of two different arrangements of events (courses) to be scheduled

Structured problem instances as explained above allow the design of a rich set of genetic and hill climbing operators:

- A *classifier* (subset of variables) at any level of an arrangement can be considered as a single unit. Operators can be applied on or within all classifiers at a *selected* hierarchy level in a *selected* arrangement.
- Operators can be applied only within the *selected* regions in a *selected* classifier.

There are many selection mechanisms to be employed. A fixed arrangement and/or a hierarchy level can be utilized (predetermined selection) during the design of an operator. An arrangement and/or a hierarchy level and/or a classifier and/or a region can be selected randomly. Using a biased selection mechanism based on overall violations (cost/fitness), or violations due to a specific constraint type seems to be promising. Hence, ranking, tournament and rhoulette wheel selection methods become useful. As another alternative, a different method can be utilized at any point of selection. Some of the suggested genetic operators, exploiting the structure of data, are already experimented by Alkan et. al. (2003) and Ozcan (2005b). In Alkan et. al. (2003), operators are designed based on a fixed arrangement with id#1 as shown in Figure 1. Results indicate that traditional crossover and mutation operators perform the best. Yet, the rest of the operators are still promising. Choosing the appropriate operator dynamically or adaptively and discovering arrangements of events automatically is left as a further study. Ignoring the rest of the constraints, assuming there are only edge constraints, then a timetabling problem reduces to a graph coloring problem (Leighton 1979). A clique is defined as a strongly connected subgraph. Terashima-Marín et. al. (1999) investigated some promising clique-based crossover operators for solving timetabling problems. Similarity between these operators and the proposed operators are that classifiers are cliques in a structured data. Clique-based crossover awaits for further attention. Burke et. al. (2003b) applied variable neighborhood search using a set of different perturbation methods and local search algorithms on randomly generated schedules. Lewis et. al. (2004) tested several crossover operators for solving university course timetabling problems. According to the results conflicts-based crossover (uses a dynamic arrangement) performs better than sector-based (Enzhe et. al. 2002), day-based and student-based crossover. Most of these operators use the framework described above. A *hyper-heuristic* is a method for selecting a heuristic among a set of heuristics to solve an optimization problem (Burke et. al. 2003b). Recently, timetabling researchers started to investigate different types of hyper-heuristics. Our framework might be useful in designing more heuristics and even hyper-heuristics.

In some timetabling problems, it is not possible to build static arrangements of events. For example, there is a single arrangement of events in most of the examination timetabling problem instances due to the students as shown in Figure 2. We propose a modified version of violation directed hierarchical hill climbing method (Alkan et. al. 2003) to be applied to timetabling problems discarding any arrangements of events.

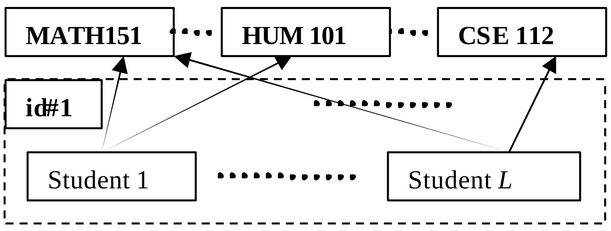


Figure 2. Arrangement of events (exams) for an examination timetabling problem instance

4 Memetic Algorithms for Solving Examination Timetabling Problems

Memetic Algorithms (MAs) combine Genetic Algorithms and hill climbing. A memetic algorithm with a violation directed hierarchical hill climbing (VDHC) is designed to solve examination timetabling problems.

4.1 Representation and Initialization

Direct representation is used to encode a candidate solution. Each gene receives an allele, indicating when the examination for a specific course will start. For example, assume that the first gene of a chromosome represents the variable for MATH151 as shown in Figure 3. Then (3, 2) can be a possible allele assignment for that specific locus, indicating MATH151 exam starts in the third day, second timetable slot. The chromosome length corresponds to the number of exams.

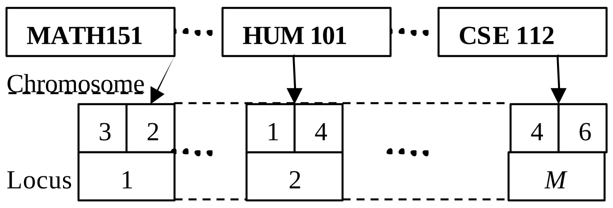


Figure 3. A sample individual

Initial population is generated randomly. A random allele value is chosen within the discrete domain of a corresponding variable for the gene in question. Population size is a factor of the chromosome length.

4.2 Genetic Operators

As a crossover operator traditional one point crossover (1PTX) is used. Two different mutation operators are implemented. As a traditional mutation operator, RAND randomly assigns a time slot for an exam. A modified version of RAND is also utilized. SWAP operator swaps two alleles if the values are within the domains of related variables, otherwise RAND is invoked.

4.3 Fitness Function

An optimum timetable is the one satisfying all the constraints. Fitness function is defined as follows:

$$f(T) = \frac{1}{1 + \sum_{\forall j \forall i\{\in j\}} w_j \, p_j \, g(T)} \qquad \text{(Eq. 1),}$$

where p_j is the penalty associated with the constraint $c_i \in C$ belonging to the constraint type j, $g_i(T)$ is the number violations in the timetable T due to the constraint c_i, and w_j is the weight applied to the constraint type j.

MA attempts to satisfy constraints with higher penalties with respect to lower ones and/or constraints that cause more violations. Unless it is mentioned p_j is set to 1 in all the experiments.

4.4 Selection

As a mate selection method, ranking (RANK) and tournament (TOUR) strategies are implemented. Steady state and transgenerational strategies based on weak elitism are utilized as a replacement strategy. Steady state (SSMA) strategy requires two offspring to be produced. Then the parents are replaced by the best individuals among parents and offpsring. Trans-generational (TGMA) approach requires creation of an offspring pool of size (*population_size*-2). During the replacement, 2 best individuals from the old generation are kept; the rest is replaced by offspring.

4.5 Violation Directed Hierarchical Hill Climbing

The idea behind our hill climbing approach is to create a hill climbing method for each type of constraint and combine them under a single hill climbing method, denoted as VDHC. Starting from a high resolution, as long as the fitness improves, VDHC stays at that level, otherwise, VDHC lowers the resolution, restricting the aim. In Alkan et. al. (2003) and Ozcan (2005b), the number of hill climbing steps depends on the hierarchy levels in an arrangement. Since there is almost no arrangement for the examination timetabling problem instances, 3 improvement steps are identified, representing 3 hierarchical levels of resolution as presented in Figure 4. In the second step, part of the chromosome is selected randomly. In the third step, an event pair is marked for hill climbing, since we do not have any unary constraints.

VDHC_HILL_CLIMB

*Step*1. Select a constraint type. Apply the related hill climbing function to the whole chromosome (all events). If individual is improved go to *Step*1.

*Step*2. Select a constraint type. Apply the related hill climbing function to a part of a chromosome (some of the events). If individual is improved go to *Step*2.

*Step*3. Select a constraint type. Apply the related hill climbing function to a single gene of a chromosome (single event or a pair of events). If the individual is improved go to *Step*3.

Figure 4. Violation directed hierarchical hill climbing method used in the MA

As a constraint type selection method tournament with tour size 2 is used. Three types of constraints should be satisfied; hence, three hill climbing methods are implemented corresponding to each constraint as described in Section 2.2: HC1, HC2 and HC3. HC1 captures overlapping pair of examinations and reschedules one of them using its domain information. If scheduling fails for the selected examination, HC1 reschedules the second one in the same way. HC2 attempts to schedule adjacent examinations using one of the patterns based on predetermined probabilities. Either examinations are assigned to the first and the third slot in the same day, or they are randomly rescheduled in a similar fashion to HC1.

Given a period, if the maximum seat capacity is exceeded, HC3 reschedules an examination to a random period. In all hill climbing methods, maximum number of unsuccessful attempts is bounded. Constraint type selection provides adaptivity. Hill climbing method corresponding to a constraint type producing higher number of violations within the range is more likely to be selected.

Timetabling problems are considered as multi-criteria optimization problems. While reducing the violations due to a constraint, overall fitness of an individual can be worsen. Hill climbing attempts to move an individual to a local optimum. Applying a hill climbing seems to be computationally expensive in timetabling problems due to fitness calculations. Yet, if the total computation time reduces and/or the quality of solution increases, a hill climbing approach becomes preferable. Furthermore, depending on the constraints, it might be possible to calculate the fitness of an individual faster by considering the changes within itself, after an operator is applied (Ross et. al. 1994a).

5 Final Exam Scheduler (FES)

The details of Timetabling Markup Language (TTML) can be found in Ozcan (2005). Final exam scheduler (FES) is a TTML processor, embodying a *parser*, *problem solver* and a *solution interpreter* as shown in Figure 5. As a programming language Java is used to implement FES. FES accepts an examination timetabling problem instance in TTML format and attempts to solve it. As a problem solver, an MA as described in Section 4 is embedded. FES allows the same input document to be modified to include a proposed solution by the solver. Various parameters of the MA can be also set via the graphical user interface of FES.

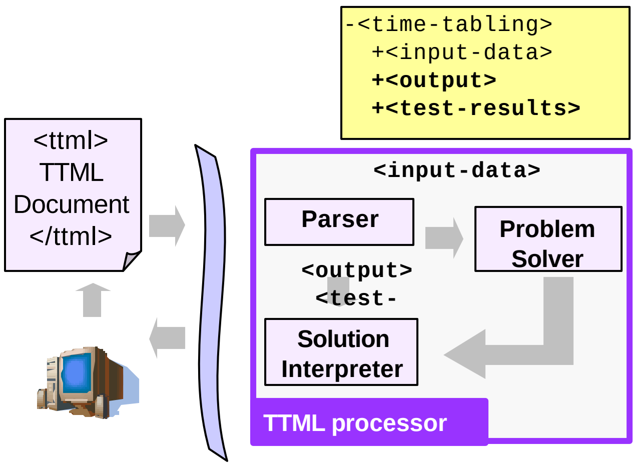
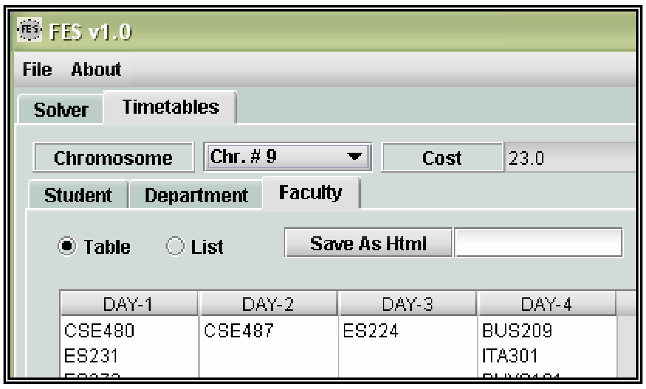


Figure 5. A TTML processor for solving timetabling problem instances

Solution interpreter component represents the utility for viewing the results. During the evolution, pausing is supported by FES at any time. After the execution is paused, any candidate solution (chromosome) within the population can be selected to be viewed. Total number of violations (cost) and violations due to each constraint type for a candidate solution are also provided. Depending on the classifiers defined in the TTML document used as input, FES supports generation of different timetables for viewing (Figure 6). For example, defining student classifiers in a TTML document, where each subset represents courses registered by a student; final examination timetable of each student can be displayed. Additionally, a user might prefer to prepare a TTML document with curriculum term classifiers, grouping courses for a department for each curriculum term and lecturer classifiers, grouping courses offered by each lecturer. By that way, a user is enabled to define constraints using these classifiers, and display different final examination timetables. Then, timetables containing the assignments of courses for a selected curriculum term or a selected lecturer can be displayed. Output can be viewed as either a list or a table. Users are allowed to save a selected solution embedding it into the input document in TTML format. Furthermore, generated timetables can be saved in HTML format as well.



DAY-1	DAY-2	DAY-3	DAY-4
CSE480	CSE487	ES224	BUS209
ES231			ITA301

Figure 6. A snapshot of the FES graphical user interface during execution

We are hoping for a single problem solver for solving different types of timetabling problems and we have taken some serious steps towards achieving this. Ultimately, our aim is to design such a system that would accept TTML input and solve it, without any data and problem type dependency. Developing an XML based standard, namely TTML and FES was the first step. Currently, we are working on the framework of the problem solver. It is important to emphasize that feeding TTML input into the TTML processor does not have to be a local process. TTML processor can be designed as a web service.

6 Experiments

Experiments are performed in two stages on Pentium 4, 2.6GHz Win2000 machines with 256 Mb memories. In the first stage, best configuration is obtained, while in the second stage the MA using the best configuration VDHC is tested on more data.

6.1 Parameter Settings

Each experiment is repeated 50 times. A run is terminated if all the constraints are satisfied or maximum number of generations is exceeded that is a factor of the chromosome length. Crossover and mutation rates are chosen as 1.0 and 1/*chromosome_length*, respectively. Population size is selected as half of the chromosome length. Maximum number of unsuccessful hill climbing (HC1, HC2 and HC3) steps is fixed as 5. Maximum number of successful VDHC attempts is also fixed as the chromosome length. In all the experiments, the virility for RANK and the tour size for TOUR are set to 4.

6.2 Experimental Data

Two problem instance sets are used during the experiments. The first set is a reduced form of student course registrations of in FEA at YU during 2001/2002 and 2002/2003 educational years. YU data is labeled as yue#educational year#semesterid. Data reduction is performed by eliminating sections. Most of the existing final examination timetabling data consists of two files. One file includes a list of students and the courses they are assigned and the other includes a list of classrooms and their capacity. Constraints are generally provided in the context of related publications. Using TTML, we have combined the input data and the constraints in a single file for each problem instance. The second problem instance set is retrieved from Carter's benchmark (Carter et. al. 1996). *Similar* problem instances staf83 and utes92 are used during the experiments. It is known that there is at least one solution for all the examination problem instances used during the experiments.

Characteristics of the experimental data are summarized in Table 1, including constraint related parameters. Number of days in the timetables used for YU data is based on the FEA requirements.

Data Label	**No. of exams**	**No. of st.**	**Avr. *neps***	***dcm***	***msc*, days**
yue20011	140	559	6,24	0,14	500,10
yue20012	158	591	6,27	0,14	500,10
yue20013	30	234	1,91	0,19	500,4
yue20021	168	826	6,97	0,16	800,12
yue20022	187	896	6,54	0,16	800,12
yue20023	40	420	1,88	0,19	800,4
yue20031	177	1125	5,97	0,15	800,12
yue20032	210	1185	5,77	0,14	800,12
staf83	139	611	9,41	0,14	500,10
utes92	184	2749	4,29	0,08	500,10

Table 1. Characteristics of experimental data, where *neps* is the number of examinations per student, *dcm* is the density of the conflict matrix and *msc* is the maximum seat capacity

Both problem instance sets are converted into TTML format, using a java parser, named as CONFETI (a converter for examination timetabling textual data). CONFETI applet and the timetabling data repository can be freely reached at:

http://cse.yeditepe.edu.tr/~eozcan/research/TTML

6.3 Experimental Results

In the first stage, eight experiments are conducted using each data; yue20011, yue20012, yue20013, yue20023, staf83 and utes92. MAs are tested utilizing RANK and TOUR, SSMA and TGMA, RAND and SWAP operators as mate selection, replacement and mutation operators, respectively. Define *success rate* (*s.r.*) as percentage of the successful runs. Considering the success rate, in all cases tournament mate selection performed better than ranking strategy as presented in Table 2.

In order to compare the replacement strategies utilized, yue20022 is added to the test cases. According to the results of the four experiments for each data in which TOUR is fixed as a mate selection method, TGMA performs better than SSMA as a replacement strategy in all cases. Outcomes are presented in Table 3. TGMA visits less number of states as compared to SSMA on average.

	TOUR		**RANK**	
Data Label	**Avr. No. of States Vis.**	**Avr. *s.r.***	**Avr. No. of States Vis.**	**Avr. *s.r.***
yue20011	967.909	0,99	829.507	0,99
yue20012	4.358.394	0,94	4.952.054	0,57
yue20013	1.851	1,00	734	1,00
yue20023	3.621	1,00	2.697	1,00
staf83	1.255.874	0,99	1.459.840	0,91
utes92	89.537	1,00	144.780	1,00
Avr.	1.112.864	**0,99**	1.231.602	0,91

Table 2. Comparison of mate selection methods; TOUR and RANK, based on average number of states visited per run and average success rate for each problem instance, considering all runs

	SSMA		**TGMA**	
Data Label	**Avr. No. of States Vis.**	**Avr. *s.r.***	**Avr. No. of States Vis.**	**Avr. *s.r.***
yue20011	1.904.463	0,98	31.354	1,00
yue20012	8.516.845	0,89	199.942	0,99
yue20013	3.422	1,00	279	1,00
yue20022	18.364.709	0,84	521.309	0,88
yue20023	6.442	1,00	801	1,00
staf83	2.472.994	0,98	38.755	1,00
utes92	170.020	1,00	9.053	1,00
Avr.	4.491.271	0,96	114.499	**0,98**

Table 3. Comparison of replacement strategies; SSMA and TGMA, based on average number of states visited per run and average success rate for each problem instance, considering all runs where TOUR is used as a mate selection method

MA experimental results, in which TOUR and TGMA are utilized, indicate that traditional mutation operator RAND performs almost the same as SWAP with respect to the average success rate. In five out of seven test cases MA runs using RAND perform slightly better than SWAP considering average number of generations. Hence, RAND is favored during further experiments.

In the second stage experiments, yue20031, yue20032 and yue20033 are added to the test cases. The best MA configuration is experimented on the additional problem instances. Experimental results with the best MA (vMA) utilizing TOUR, RAND and TGMA operators are summarized in Table 4. The vMA utilizing VDHC discovers optimal solutions in all cases with high success rates.

Furthermore, experiments are performed for observing the influences of hill climbing and crossover. Maximum number of generations is trippled during Genetic Algorithm runs without hill climbing (sGA) for a fair comparison. The same MA settings are used for the rest of the parameters. When the hill climbing operator (VDHC) is disabled, the performance of sGA deteriorates significantly in almost all cases as shown in Table 4. Moreover, the average number of states visited per run increases for the problem instances yue20013 and utes92, unless VDHC is used. When the crossover operator is disabled (xMA), performance of the MA worsens for the problem instances yue20021 and yue20022 (Table 4). As a result, we can safely conclude that hill climbing and crossover operators together are very useful during the search for an optimal solution.

Data Label	**vMA**	**xMA**	**sGA**
yue20011	1,00	1,00	0,86
yue20012	0,96	0,96	0,36
yue20013	1,00	1,00	1,00
yue20021	1,00	0,00	0,00
yue20022	0,90	0,88	0,32
yue20023	1,00	1,00	0,98
yue20031	0,98	1,00	0,70
yue20032	1,00	1,00	0,74
staf83	1,00	1,00	0,96
utes92	1,00	1,00	1,00

Table 4. Success rates for the MA with TOUR, RAND, TGMA and VDHC (vMA), the MA without crossover (xMA) and without VDHC (sGA)

7 Conclusions and Future Work

Final examination timetabling problem at the Faculty of Engineering and Architecture is described. The only difference between the problem and other available problem instances is that all the constraints are hard constraints. A tool is introduced for solving examination timetabling problems, named Final Examination Scheduler (FES). The major components of the tool are its document parser, timetable viewer and an MA problem solver. Timetabling Markup Language documents are accepted as input by FES. Output viewer generates and displays a timetable for a subset of variables as defined in the TTML document. Any view can be saved as an HTML file. The MA solver with VDHC is tested using different operators on a set of benchmark data. The best MA configuration is achieved by combining one point crossover, tournament mate selection, elitist transgenerational replacement strategy and traditional mutation. In almost all cases, the MA successfully produces optimal timetables, satisfying all the imposed constraints. Modified violation directed hierarchical hill climbing operator utilizes a framework proposed in a previous study (Alkan et. al. 2003). VDHC turns out to be a promising hill climbing method, even if no arrangements or logical groupings of data are available before the runs.

More MA operators will be considered within the framework described to improve the GA solver. First operator to be tested next is the uniform crossover. A robust solver with a high success rate and low average number of generations (evaluations) for solving a problem instance is the goal. In most of the recent publications, importance of initialization is emphasized for solving timetabling problem instances. Hence, heuristics for initializing a population will be considered. VDHC is an adaptive operator based on the cooperation of different hill climbing methods. As a next step, its performance will be compared to a *multimeme strategy* (Krasnogor 2002). More benchmark problems and larger instances will be attacked with the methods devised.

Bibliography

Alkan, A. and Ozcan, E. (2003) “Memetic Algorithms for Timetabling”, Proc. of 2003 IEEE Congress on Evolutionary Computation, pp. 1796-1802.

Burke, E., Elliman, D., Ford, P., Weare, B. (1996a) “Examination Timetabling in British Universities- A Survey”, The Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1153, pages 76–90. Springer-Verlag.

Burke, E., Newall, J.P., and Weare, R.F. (1996b) “A Memetic Algorithm for University Exam Timetabling”, Lecture Notes in Computer Science, 1153:241-250, Springer.

Burke, E. K., Pepper, P. A. and Kingston, J. H. (1997) “A Standard Data Format for Timetabling Instances”, Springer Lecture Notes in Computer Science, 1408:213-222.

Burke, E.K., De Causmaecker, P., Petrovic, S., Vanden Berghe G. (2003a) “Variable Neighbourhood Search for Nurse Rostering Problems”, in Metaheuristics: Computer Decision-Making (edited by Mauricio G.C. Resende and Jorge Pinho de Sousa), chapter 7, Kluwer, pp. 153-172.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. and Schulenburg, S. (2003b) “Hyper-heuristics: an emerging direction in modern search technology", Handbook of metaheuristics, chapter 16, pp. 457-474. Kluwer Academic Publishers.

Burke, E.K., De Causmaecker, P. and Vanden Berghe, G., Van Landeghem, H. (2004a) "The State of the Art of Nurse Rostering", Journal of Scheduling, 7:441-499.

Burke, E. and Newall, J. P. (2004b) "Solving Examination Timetabling Problems through Adaption of Heuristic Orderings: Models and Algorithms for Planning and Scheduling Problems", (Editors: Philippe Baptiste, Jacques Carlier, Alix Munier, Andreas S. Schulz) Annals of Operations Research, vol. 129, no. 1-4, pp. 107-134(28).

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

Enzhe, Y. and Sung, K. (2002) "A Genetic Algorithm for a University Weekly Courses Timetabling Problem", International Transactions in Operational Research, 9:703-717.

Even, S., Itai, A. and Shamir, A. (1976) "On the Complexity of Timetable and Multicommodity Flow Problems", SIAM J. Comput., 5(4):691-703.

Fang, H. L. (1994) "Genetic Algorithms in Timetabling and Scheduling", PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Scotland.

Glover, F. "Tabu search - Part I". ORSA Journal on Computing, 1(3):190-206, 1989.

Gaspero, L. Di and Schaerf, A. (2000) "Tabu Search Techniques for Examination Timetabling", LCNS archive Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling, pp. 104 - 117.

Goldberg, D. E. (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading (MA).

Holland, J. H. (1975) Adaptation in Natural and Artificial Systems, Univ. Mich. Press.

Kingston, J.H. (2001) "Modeling timetabling problems with STTL", Springer LCNS, 2079:309.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). "Optimization by simulated annealing", Science, vol. 220, no. 4598, pp. 671-680.

Krasnogor, N. (2002) "Studies on the Theory and Design Space of Memetic Algorithms", PhD Thesis, University of the West of England, Bristol, United Kingdom.

Leighton, F. T. (1979) "A graph coloring algorithm for large scheduling problems", Journal of Reasearch of the National Bureau of Standards, 84:489.

Lewis, R. and Paechter, B. (2004) "New Crossover Operators for Timetabling with Evolutionary Algorithms", 5th International Conference on Recent Advances in Soft Computing (RASC 2004), vol. 5, pp 189-195.

Merlot, L., Boland, N., Hughes, B. and Stuckey P. (2003) "A hybrid algorithm for the examination timetabling problem", Lecture Notes in Computer Science, vol. 2740, Gent, Belgium, Springer-Verlag, pp. 207-231.

Moscato, P. and Norman, M. G. (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, pp. 177-186.

Ozcan, E. and Mohan, C. K. (1998) "Steady State Memetic Algorithm for Partial Shape Matching", LNCS, Evolutionary Programming VII, 7th International Conference, EP98, San Diego, CA, USA, Springer, Berlin, vol. 1447, pp. 527-536.

Ozcan, E. and Alkan, A., (2002) "Solving Time Tabling Problem using Genetic Algorithms", Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling, pp.104-107.

Ozcan, E. and Onbasioglu, E. (2004) "Genetic Algorithms for Parallel Code Optimization", Proc. of 2004 IEEE Congress on Evolutionary Comp., vol. 2, pp. 1775-1781.

Ozcan, E. (2005a) "Towards an XML based standard for Timetabling Problems: TTML", Multidisciplinary Scheduling: Theory and Applications, 2005, Selected Volume of the 1st MISTA (2003), to appear.

Ozcan, E. (2005b) "Memetic Algorithms for Nurse Rostering", The 20th International Symposium on Computer and Information Sciences, in review.

Petrovic, S., Yang, Y., Dror, M. (2003) "Case-Based Initialisation of Metaheuristics for Examination Timetabling", pp. 137-154., Proc. of 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2003), pp. 137-154.

Paquete, L. F. and Fonseca C. M. (2001) "A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms", Proc. of the 4th Metaheuristics International Conference (MIC 2001), pp. 149-154, Porto.

Radcliffe, N. J. and Surry, P.D. (1994) "Formal memetic algorithms", Evolutionary Computing: AISB Workshop, Springer Verlag, LNCS 865, pp. 1-16.

Ross, P., Corne, D. and Fang, H-L. (1994a) "Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation", Proc. of PPSN III, pp. 556-565.

Ross, P., Corne, D. and Fang, H-L. (1994b) "Fast Practical Evolutionary Timetabling", Proc. of AISBWorkshop on Evolutionary Computation.

Schaerf, A. (1999) "A survey of automated timetabling", Artificial Intelligence Review, 13(2):87-127.

Terashima-Marín, H. (1998) "Combinations of GAs and CSP Strategies for Solving the Examination Timetabling Problem", PhD thesis, Computer Systems Engineering, Tecnológico Monterrey, Mexico.

Terashima-Marín, H., Ross, P. and Valenzuela-Rendón, M. (1999) "Clique-Based Crossover for Solving the Timetabling Problem with Gas", Proc. of the Congress on Evolutionary Computation, pp. 1200-1206.

Werra, D. De, (1985) "An introduction to timetabling", European Journal of Operations Research, 19:151-162.

Wong, T., Cote, P. and Gely, P. (2002) "Final exam timetabling: a practical approach", Canadian Conference on Electrical and Computer Engineering, Winnipeg, CA, vol.2, pp. 726- 731.