Gazi University Journal of Science
GU J Sci
25(1):137-153 (2012)

ORIGINAL ARTICLE

A Genetic Algorithm Based Examination Timetabling Model Focusing on Student Success for the Case of the College of Engineering at Pamukkale University, Turkey

Can Berk KALAYCI[1], Aşkıner GÜNGÖR[1,♠]

[1]*Department of Industrial Engineering, College of Engineering, Pamukkale University, 20020 Denizli*

Received: 01.04.2011Revised: 27.06.2011 Accepted: 01.08.2011

Abstract

This study proposes a genetic algorithm (GA) based model to generate examination schedules such that they focus on students' success in addition to satisfying the hard constraints required for feasibility. The model is based on the idea that the student success is positively related to the adequate preparation and resting time among exams. Therefore, the main objective of this study is to maximize time length among exams (i.e., paper spread) considering the difficulties of exams. Two different genetic algorithm models were developed to optimize paper spread. In the first genetic algorithm model, a high penalty approach was used to eliminate infeasible solutions throughout generations. The second genetic algorithm model controls whether or not each chromosome joining the population satisfies the hard constraints. To evaluate the models, a set of experiments have been designed and studied using the data collected from the College of Engineering in Pamukkale University.

Keywords: *Genetic Algorithms, Examination Timetabling, Student Success*

1. INTRODUCTION

Timetabling has attracted the attention of the operational research and artificial intelligence research communities for more than 40 years. Perhaps the most widely studied class of timetabling problem is educational timetabling [1]. Timetabling problems are NP-complete in their general form, regarding their computational complexity, meaning that the difficulty to find a solution rises exponentially to their size and a deterministic algorithm, giving an optimal solution in polynomial time, cannot be found. University timetabling is a significant administrative issue that arises in academic institutions. The general term university timetabling typically refers to both university course and examination timetabling which are different in nature. In this study, we focus on examination timetabling problem which is a difficult and lengthy task for which universities devote a large amount of human and material resources. In some institutions, administrators approve the final timetable if exams are scheduled without any major conflicts in the available time period. It is widely accepted that exams are a part of learning process; therefore it is required to pay attention to other issues related to scheduling exams. In this study, we specifically focus on students' success when scheduling exams in addition to several hard constraints required for feasibility of the suggested schedule. Student success is positively related to the adequate preparation and resting time among exams. One of the desirable attributes of real-life examination timetabling solutions is the maximization of paper spread [2], which is a measure of the amount of study time that each student has among examinations. We extend this to include the idea that the amount of study time required by the students among examinations is positively related to the difficulty of exams. Therefore, the main objective of this study is to maximize paper spread (i.e., time length among exams) considering the difficulties of exams for better success rates of students. Two different genetic algorithm (GA) models were developed for this purpose. In the first GA model, a high penalty approach is used to get rid of infeasible

solutions throughout generations. The second GA model controls whether or not each chromosome joining the population satisfies the hard constraints. If the hard constraints are not satisfied, conflicts are removed from the chromosome with an embedded repair function. If repair fails, the chromosome is destroyed. The second method forces the genetic algorithm to search only in the feasible solution space. Comparative simulation results for both solution approaches are presented. The results show the effects of repair function and high penalty cost usage in solving NP-complete problems using GA based approaches and contribute to GA application literature.

The rest of this paper is organized as follows: The following section provides an overview of previous studies related to examination timetabling. Section 3 describes the problem under study and related constraints. Section 4 presents the proposed GA methods. Experimental design and computational results are given in section 5. Finally, in section 6 conclusions are presented.

2. LITERATURE REVIEW

In the literature, there has been a wide investigation of automated timetabling approaches in the domain of examination timetabling with various methodologies based on artificial intelligence and operational research techniques in order to produce better and feasible timetables. Qu et al. [3] provides an up to date overview of search methodologies and automated system development for examination timetabling. This section provides a survey of related work in the literature and clarifies the contribution of the proposed model.

A scheduling problem is simply to assign a set of tasks to a set of sources under a set of constraints. In an uncapacitated examination timetabling problem, exams represent tasks and time slots represent sources. Constraints are examined in two categories as hard and soft constraints. Hard constraints have to be assured in order to obtain feasible solutions. It is almost impossible to satisfy all of the soft constraints yet the quality of the solution is increased with the number of soft constraints satisfied. Therefore, choosing soft constraints wisely according to a certain goal becomes important. An example for a hard constraint in uncapacitated examination timetabling problem is considered to be "scheduling two exams with common students into the same time slot". Occurrence of this condition is "a conflict" and causes infeasibility in the solution obtained. On the other hand, one of the soft constraints, defined by Carter et al. [4], is concerned with spreading out the exams over a timetable so that students will not have to take exams that are too close to each other. The objective is to assign exams into timeslots while minimizing the cost of the violations of the hard and soft constraints. This is widely used in examination timetabling research to measure the solution quality. This measure is called "conflict density" and given in equation (1).

$$ConflictDensity = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left(\frac{\beta_{ij}}{N}\right)}{N} \quad (1)$$

where N is the number of exams and β_{ij} is a decision variable assuming value of 1 if exam i and exam j have at least one student in common while assuming value 0 otherwise.

There has been a wide investigation of different methods to minimize the cost on the violations of the constraint stated above using various techniques such as cased based heuristic selection [5], large neighborhood search approach [6], graph based heuristic [1], the great deluge metaheuristic [7] and fuzzy methodologies and heuristic orderings [8]. heuristic combinations [9], a random iterative graph based hyper-heuristic [10], variable neighborhood search and its hybridization with a genetic algorithm [11] and genetic algorithms by inducing solutions [12]. Gogos et al. [13] proposed a multi stage algorithmic process including hill climbing, simulated annealing, heuristics, metaheuristics and exact methods for the solution of examination timetabling problem considering the soft constraint provided by conflict density given in equation (1) to measure the final timetable quality in addition to hard constraints required for the feasibility provided by their proposed methods.

Exact methods and heuristic approaches have been widely applied to solve real-world timetabling problems as case studies in different universities considering specific constraints. Foulds and Johnson [14] designed a decision support system to construct university timetables considering specific circumstances to New Zealand University which detects if a room is double-booked, allocated to a course that exceeds the room capacity or a course is allocated to the same time slot. Dimopoulou and Miliotis [15] designed and implemented a computer network based timetabling system to aid the construction of a university course timetable at Athens University of Economics and Business. Dimopoulou and Miliotis [16] improved their previous system. Burke and Petrovic [17] described a method for decomposing large real-world timetabling problems and presented an approach that considers timetabling problems as multi-criteria decision problems. Room capacities, proximity of exams, time and order of exams have been considered as satisfaction criteria for the objective. Burke and Newall [18] adapted heuristic ordering based method to solve examination timetabling problem by minimizing the conflicts weighted descending by available periods among exams considering capacity constraints.

Integer programming formulation is very popular for timetabling problems since presenting candidate solutions fits easily and efficiently. Daskalaki et al. [19] presented a 0–1 integer programming formulation of the university timetabling problem with the objective

function that aims the satisfaction of expressed preferences regarding teaching periods, days of the week, and classrooms for specified courses. Daskalaki and Birbas [20] proposed a two-stage relaxation procedure that solves the integer programming formulation of a university timetabling problem with the objective function of minimizing penalty costs to drive the final solutions to better timetables. Avella and Vasil'Ev [21] applied branch and cut algorithm using integer programming formulation to maximize the satisfaction of teachers under a wide range of constraints for the solution of university course timetabling problem. MirHassani [22] presented a 0–1 integer programming formulation of the university timetabling problem and solved the real timetabling problem at Shahrood University of Technology considering the number of sessions necessary for a course as a hard constraint. Having one day off between two sessions of each course is considered as a soft constraint. The objective function of this model sets to minimize the infeasibility of the soft constraint and also penalize the redundant and non-preferred times. It is also stated that by changing the penalty functions it was possible to change the computation time, meaning that the optimization process may be guided faster to the optimal solution. MirHassani [2] pointed out the maximization of paper spread, which is a measure of the amount of study time that each student has between examinations as one of the most significant desirable attributes of real-life examination timetabling solutions. Al-Yakoob and Sherali [23] proposed a mixed-integer programming approach that aims to enhance existing manual approaches by minimizing class conflicts considering specific constraints of the university. Head and Shaban [24] proposed a heuristic approach to build the schedule and place the students into rooms simultaneously.

Multi-objective optimization approaches mostly focus on minimizing the timetable length while simultaneously optimizing the spread of examinations. van den Broek et al. [25] solved a real-world timetabling problem at the Department of Industrial Design of the TU Eindhoven with the objective to assign courses as high as possible on preference lists and to have students spread as equally as possible over the sections. Birbas et al. [26] aimed to satisfy the teachers' preferences, assign core courses towards the beginning of each day, balance the sum of teaching and idle periods in addition to hard constraints required for feasibility. Kahar and Kendall [27] developed a heuristic solution approach for capacitated examination timetabling problem given the constraints at University Malaysia Pahang in addition to generally used constraints and compared the results of their university's current software solutions. Sarin et al. [28] applied Benders' partitioning approach using integer programming formulation to minimize the total distance that faculty members have to travel from their offices to the classrooms where the courses are scheduled to sole university timetabling problem at College of Engineering of Virginia Tech University. Rudová et al. [29] solved complex university timetabling at Purdue University using generic iterative forward search, branch and bound algorithm considering rooms, room equipments, instructors availability and time precedence between classes. Wang et al. [30] designed and implemented a decision support system includes a greedy heuristic to create initial schedules and a variable bilinearization and decomposition technique that allows the improver module to improve the initial timetabling solutions 0-1 linear programming formulation at United States Military Academy/West Point with the objective function that attempts to minimize the total number of makeup exams due to short time period availability.

Metaheuristics and hybrid combinations have been applied to solve examination timetabling problems. Smith et al. [31] modified neural networks approach to solve school timetabling problems and compared with results obtained using the greedy search, simulated annealing and tabu search. Genetic algorithms have been the most studied evolutionary algorithms in terms of exam timetabling research. Dave Corne [32] investigated direct and indirect approaches of genetic algorithms in terms of search space, speed of the algorithm and quality of the solution. Naji Azimi [33] used a direct approach by integrating a high penalty model to satisfy hard constraints and applied genetic algorithms, simulated annealing, tabu search and ant colony system techniques and hybrid combinations of these approaches to examination timetabling problem considering a similar objective to the general density function given in equation (1). Ross et al. [34] stated the weakness of the use of direct coding in genetic algorithms. Erben [35] proposed a grouping genetic algorithm using a swap mutation operator to exchange the positions of two randomly chosen groups in the chromosome and a greedy algorithm to satisfy capacity constraints to solve graph coloring problems that correspond to modeling the exam timetabling problem with only hard constraints. Wong et al. [36] applied fitness based evaluation counting consecutive night and next day morning exams provided by binary tournament selection operator, uniform crossover operator, random mutation operator with heuristic repair and a reinsertion operator. Sheibani [37] tried to maximize the interval between exam subjects using genetic algorithms with the minimum number of clashes. Wong et al. [38] proposed a hybrid multi-objective evolutionary algorithm in which crossover is replaced by two local search operators. Côté et al. [39] applied a multi objective evolutionary algorithm to simultaneously minimize timetable length and proximity cost. Santiago-Mozos et al. [40] presented the application of a two-phase heuristic evolutionary algorithm to obtain personalized timetables in a Spanish university as a case study using the objective function to minimize the total number of non-assigned subjects and make obtained timetables as compact as possible satisfying the maximum number of student preferences. Chiarandini et al. [41] proposed a hybrid metaheuristics algorithm including heuristics, tabu search, variable neighborhood descent and simulated annealing approaches to solve university course timetabling problem. Ülker et al. [42] applied genetic algorithms using linear linkage encoding representation with greedy partition crossover, lowest index first crossover and lowest index max crossover operators. Beligiannis et al. [43] developed an adaptive algorithm based on evolutionary computation techniques in order to solve high school timetabling

problem in Greece using the objective function to minimize the total cost of idle hours for all teachers and maximize the satisfaction of teachers. Pongcharoen et al. [44] developed a stochastic optimization timetabling tool for university course timetabling using genetic algorithms and simulated annealing, included a repair process, which ensures that all infeasible timetables are rectified in order to prevent clashes and determine rooms with the sufficient seating capacity. Mumford [45] presented candidate solutions to a multi-objective memetic algorithm as orderings of examinations and a greedy algorithm to construct violation free timetables from permutation sequences of exams for solving examination timetabling problem. Cheong et al. [46] proposed a multi objective evolutionary algorithm including genetic algorithms and a hill climber local search operator to minimize the number of clashes and timetable length considering capacity constraints. De Causmaecker et al. [47] suggested a decomposed metaheuristics approach.

As presented above, the timetabling literature is quite rich and each new insight into the problem is considered to be a contribution to the related reseach field. We contribute to the literature by extending the MirHassani's [2] maximization of paper spread idea by considering the difficulty of each exam which is positively linked to the required amount of study time according to our main focus of students' success. Taking difficulties of exams into account requires more complex GA structures. We used two different GA modeling approaches and compared them. A conflict formulization similar to equation (1) has been used including three types of exam clashing (i.e., two exams in the same time slot, two exams on the same day and two exams on two consecutive days).

3. PROBLEM DEFINITION and FORMULATION

Variables used in the definition and formulation of the examination timetabling problem are given as follows: N is the number of exams, E_i is an exam where $i \in \{1,\ldots,N\}$, D is the number of days, T is the given number of available timeslots, t_i specifies the assigned time slot for E_i where $1 \le t_i \le T$. dc_i specifies the difficulty coefficient for E_i where $dc_i = [1,..,10]$ and integer. The value of 10 represents the most difficult exam, the value of 1 represents the easiest one, $C = (c_{ij})_{N\times N}$ is a conflict matrix where each element denoted by c_{ij} is the number of students taking E_i and E_j, a_{it} is 1 if E_i is allocated to t_i, 0 otherwise, $S = (s_{ij})_{N\times N}$ is the shared exam matrix where each element is denoted by 1 if E_i and E_j are shared exams, 0 otherwise, and finally L is the seating capacity for each period.

Hard and soft constraints considered in this problem can be written as follows:

i. No student can sit in more than one exam at the same time.

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \times b(t_i,t_j) = 0 \quad \text{where} \quad b(t_i,t_j) = \begin{cases} 1 & if\,(t_i = t_j) \\ 0 & otherwise \end{cases} \tag{2}$$

ii. All of the planned exams must be scheduled among the available time slots.

$$1 \le t_i \le T \text{ for every } E_i \text{ where } i \in \{1,\ldots,N\} \tag{3}$$

iii. Every exam can only be scheduled once in any timetable.

$$\sum_{t=1}^{T} a_{it} = 1 \quad \text{where } i \in \{1,\ldots,N\} \tag{4}$$

iv. All of the shared exams of different departments have to be scheduled at the same time.

$$\sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij} \cdot (t_i - t_j) = 0 \tag{5}$$

v. The maximum amount of time between exams as much as possible among available time slots is necessary. This soft constraint is formulated into equation (8).

vi. Difficult exams should be assigned to considerably far time slots than easier exams according to each other's position. This soft constraint is formulated into equation (8).

vii. There must be sufficient seats for every exam scheduled in the time period.

$$\sum_{i=1}^{N} a_{it} c_i \le L \quad \text{where } t = \{1,...,T\} \tag{6}$$

Constraints i, ii, and iii are generally accepted hard constraints for uncapacitated examination timetabling problems. By applying these constraints, all exams of each student are assigned to different time slots in the available time period. Constraint iv is a hard constraint suitable for the case of College of Engineering at Pamukkale University (PAUCOE). In this case, a lecturer teaching the same course in different departments requires the exam to be scheduled to a time

slot so that all students can take the exam at the same time. This means that exams for more than one course need to be scheduled in the same time slot. We name this type of exams as "shared exams". In order to increase the quality of the feasible solution, MirHassani [2] presented the soft constraint v to maximize the *paper spread* which is a measure of the amount of study time that each student has among examinations. By the help of *paper spread*, students will have more time to study or to relax among exams. In this study, we focus on the thought that difficult exams require more study and resting time than easier exams. Therefore it is also required to consider the difficulties of exams during the construction of paper spread which brings out the modified soft constraint given in constraint vi. When the final schedule is obtained by satisfying this soft constraint as much as possible, students will be able to get enough study and resting time among difficult exams. The capacity constraint is also a hard constraint to be satisfied for each period (i.e., constraint vii).

We now formulate the objective function for the proposed GA models as follows:

$$Min \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij}[(dc_i + dc_j) \cdot \lambda(t_i, t_j)] \qquad (7)$$

Where;

$$\lambda(t_i, t_j) = \begin{cases} 1000 & , if\ t_i = t_j \\ 2^{4-\left|(t_i - t_j)/D\right|} & , if\ Mod(t_i, D) = Mod(t_j, D) \\ 1 & , if\ \left|Mod(t_i, D) - Mod(t_j, D)\right| = 1 \\ 0 & , otherwise \end{cases} \qquad (8)$$

Equation (8) presents a proximity value assigned to two corresponding exams inspired by Carter et al. [4]. For example, if it is accepted that there are four possible time slots for each exam in a day, then the proximity values associated with two corresponding exams may take the values of 1000, 8, 4, 2, 1 or 0 depending on the proximity to each other. In this equation, the cost of violating this hard constraint is fixed to 1000 as Naji Azimi [33] did to avoid infeasible solutions through generations so that candidate solutions satisfy the constraint i.

4. THE PROPOSED GENETIC ALGORITHMS

Genetic Algorithms (GA) are based on an analogy of biological evolution, in which the fitness of an individual determines its ability to survive and reproduce [48]. Each individual which represents a candidate solution in the population is repeatedly evaluated by genetic operators such as crossover and mutation. In this evaluation process, survival of each individual is determined according to a fitness function. Bad individuals, i.e., low qualified solutions, are destroyed through generations. This "survival of the fittest principle" is the idea behind GA [49]. When the ending criteria are satisfied, the chromosome that has the best gene combination at the last generation represents the best solution for the problem.

In order to solve the exam timetabling problem defined in Section 3, a GA based model has been designed and implemented and detailed in the following subsections.

4.1. Gene encoding

It is well known that timetabling problems falls into the NP-Complete class of combinatorial optimization problems. When the problem size increases, the solution space is exponentially increased and an optimal solution in polynomial time cannot be found. Therefore, it is necessary to use alternative methods in order to reach (near) optimal solutions faster. Metaheuristics such as GAs seem to be particularly suited for this task because they process a set of solutions in parallel, possibly exploiting similarities of solutions by recombination that provides an alternative to traditional optimization techniques to locate optimum solutions in complex landscapes [12, 35, 38, 39, 46].

In order to apply a GA to a particular problem, an internal representation for the solution space is required. The choice of this structure is one of the most critical aspects for the success of the GA operators for the problem. As explained in sections 2 and 3, in our study, different from the related literature, the case of "shared exams" need to be taken into account when determining the GA structure. This is a common requirement in government universities in Turkey. For example, Exams of Calculus 101 offered in all departments should be scheduled at the same time slot. As it is demonstrated in Figure 1, unlike Cheong et al. [46] each gene of a chromosome represents the assigned time slot for the related exam. The selection of the chromosome encoding focused on the following concern. A chromosome shall occupy as small space as possible in relevance with the information it encodes and to be easy to preserve as many characteristics of the timetable concerning hard constraints as possible during the reproduction processes. Shared exam slots are placed as the first chain in the chromosome structure and then other non-shared exams of each department are added to the structure respectively. Shared and other exams' gene combinations form an individual in the GA.

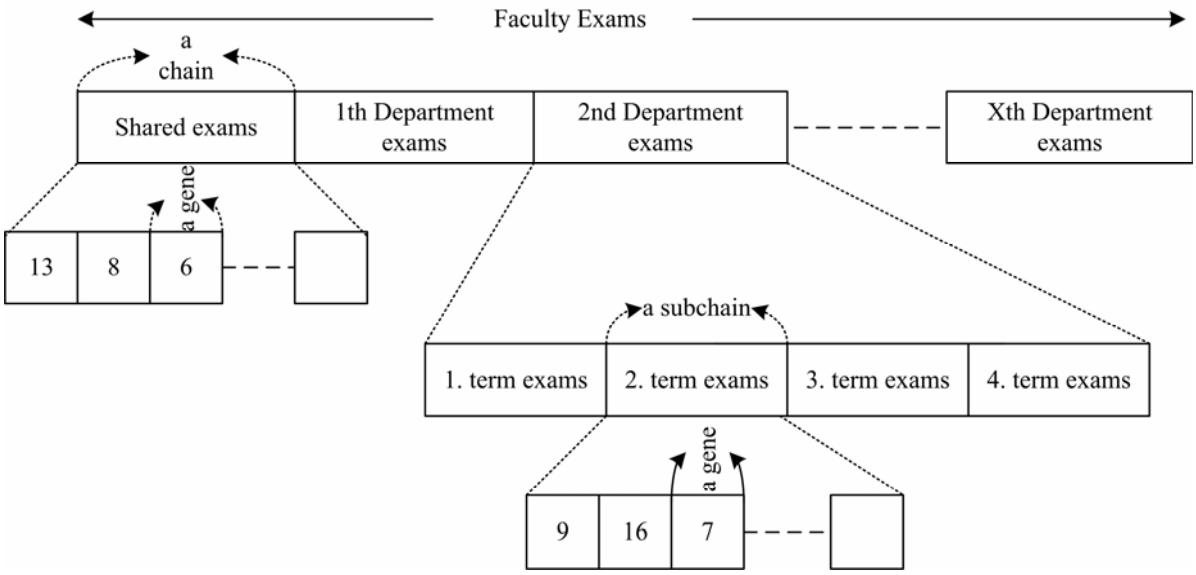

Figure 1. The chromosome representation of a candidate solution to PAUCOE exam timetabling problem

The integer interval of gene values is determined by available time slots in a day and available days in the schedule. If we assume that there are 4 different time slots available each day at a 9 days exam period (as in the case of Pamukkale University), 36 possible time slot assignments exist. Therefore, gene values may take integer values between [1, 36].

In Figure 2, time slot assignments are demonstrated for t time slots available each day at a D days' period.

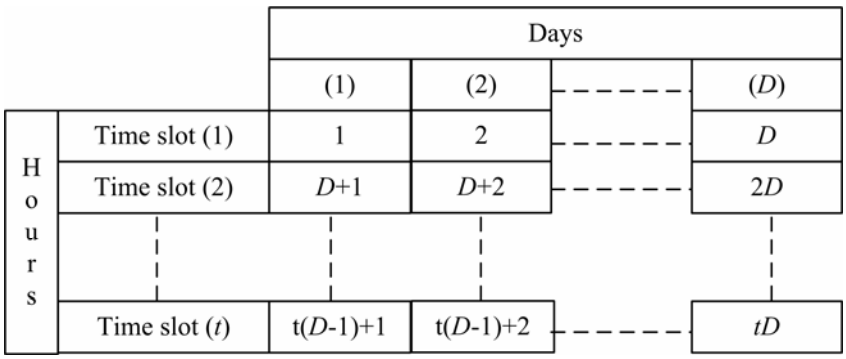

Figure 2. Symbolic representation of t time slots available each day of a D days period

4.2 Chromosome initialization

The creation of the initial population has an importance regarding diversity. In this problem formulation, all exams which belong to a term have to be scheduled at different time slots, so chromosomes that form the initial population have to be adjusted to take different gene values in chains that represent terms. If two or more genes have same values in a chain, direct violation of hard constraints occur. For this reason, during the creation of the initial population, randomization process is modified to block conflicts by assuring different gene values. By this modification, there will not be any conflicts on chain basis. However, there may still be other conflicts because of the interactions among other chains. In order to remove these conflicts, fitness function selection structure or a special repair function is used. Detailed information can be found in the following subsections.

4.3 Fitness function

The fitness function is one of the most important parts of the GA, because it decodes the chromosome into timetable and calculates a fitness value of each chromosome that points out how well it solves the problem under study.

In our problem, the fitness is formed in order to maximize students' success. There are many students who share the same exam schedule at each term. Instead of checking each student's schedule separately in the database, we preferred to group students based on chains and shared exams. Thus, subchains and their connected shared exams represent a group of students who have the same exam schedule.

As it is demonstrated in Figure 3, the fourth subchain includes exams that belong to a subgroup of students. With the appropriate query on shared chain that is automatically executed by the system, it is found out that *S2* and *S4* are shared exams which also belong to this group of students. Thus, *S2* and *S4* exams are added to the fourth subchain as *E5* and *E6*. In some cases, students are obligated to retake exams from previous terms. All exams that student subgroups are responsible for are added to the assignment array and fitness value is calculated. Details of these different cases are given in section 4.5. Each exam has a time slot and a difficulty coefficient. Difficulty coefficients of each exam are obtained on a scale between 1 and 10, integer values, by conducting surveys with students and also considering European Credit Transfer System (ECTS) credits of each course. Basically, the greater coefficient represents the more difficult exam on the schedule.

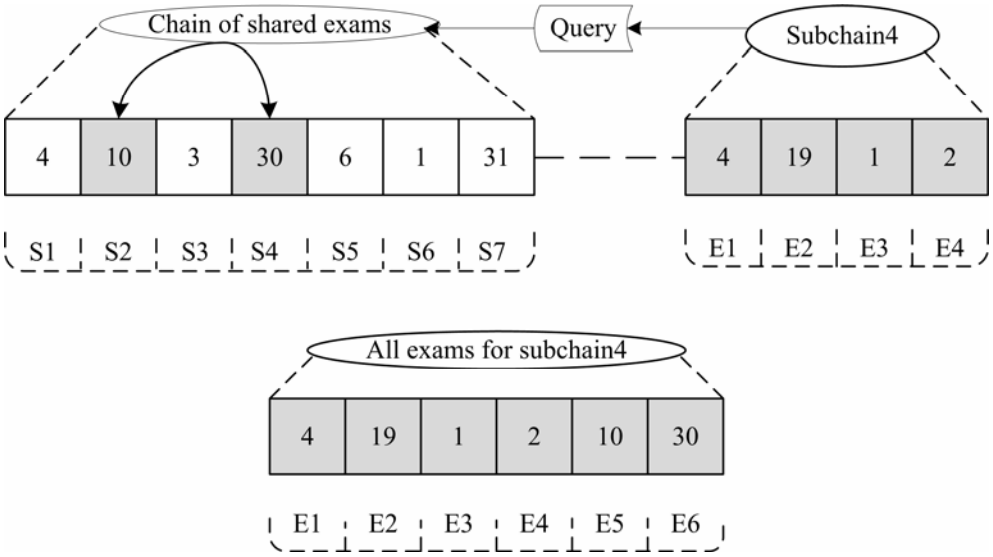

Figure 3. Assignment array of all exams for a subchain

4.4. Genetic Operators

Beligiannis et al.[43] used linear ranking selection and Erben [35], Wong et al. [36], Côté et al. [39], Cheong et al. [46], Pillay and Banzhaf [12] used tournament selection operator in order to create the intermediate population (mating pool) for reproduction. Among the different types of selection procedures existing, we have chosen the one known as roulette wheel [11, 33, 40, 44, 48, 50], in which the probability of an individual to be selected for the next generation depends on its current fitness value. Before applying genetic operators, each individual in the population is sorted according to their fitness values and the individuals that have values greater than a specified average fitness value are put into a mating pool. New fitness values of individuals are calculated according to equation (9). Then, the individuals with negative fitness values are removed from mating pool for the current iteration of GA and the problem is converted to a maximization problem from a minimization problem.

The variables used in equation 9 are described as follows: F_n is the new fitness value of the current chromosome; F_a is the average fitness value of the current population; co : 0,1,2 or 3, sd is the standard deviation of the current population; and F_o is the old fitness value of the current individual.

$$F_n = F_a + co \times sd - F_o \tag{9}$$

In equation 9, when *co* = 0, the individuals with fitness values below the average are taking out of the mating pool. The individual with the least fitness value in general population has the greatest fitness value in the mating pool by this transformation. Individuals with their new fitness values in the mating pool become ready to enter roulette wheel process. The greater fitness value means the bigger part of the roulette wheel is occupied by that individual. Thus, better chromosomes have a greater chance to be selected. Roulette wheel selects the first individual as a mate, and the next one afterwards. No individual is allowed to mate itself, because variation to better or worse is expected during crossover. This selection procedure is continued until the required number of mates for crossover is reached.

4.4.1 Crossover operator

Naji Azimi [33] applied one and two point crossover operator, Terashima-Marin et al. [51] applied clique-based crossover operator, Santiago-Mozos et al. [40] applied partially matched crossover operator, Ülker et al. [42] applied greedy partition crossover and lower index first-max crossover operators, Pongcharoen et al. [44] applied one, two point and position based crossover operators, Puente et al. [50] introduced a specific multi point crossover operator that can only work exchanging information between entire work weeks to solve timetabling problems by GA. Beligiannis et al. [43] and Pillay and Banzhaf [12] did not use any crossover operator at all because of their experimental results have shown that crossover in this specific problem and chromosome encoding does not contribute satisfactorily, while it adds too much complexity and time delay. In this algorithm, we have used a modified uniform crossover operator which has been experimented to give satisfying results in Erben [35] and Wong et al. [36] for the solution of timetabling problems in GA. New individuals created by crossover operation, called children, should satisfy the same constraints as previous individuals, called parents. Genes that are associated with zero value of randomly created binary chain are copied from the first parent. The remaining genes of the first child are copied from the second parent from the second parent starting from the first gene. If there are similar genes in second parent's chromosome, these genes are jumped over from copying to the child in order to prevent conflicts in the chain of the new chromosome. Crossover operation is demonstrated as an example in Figure 4.

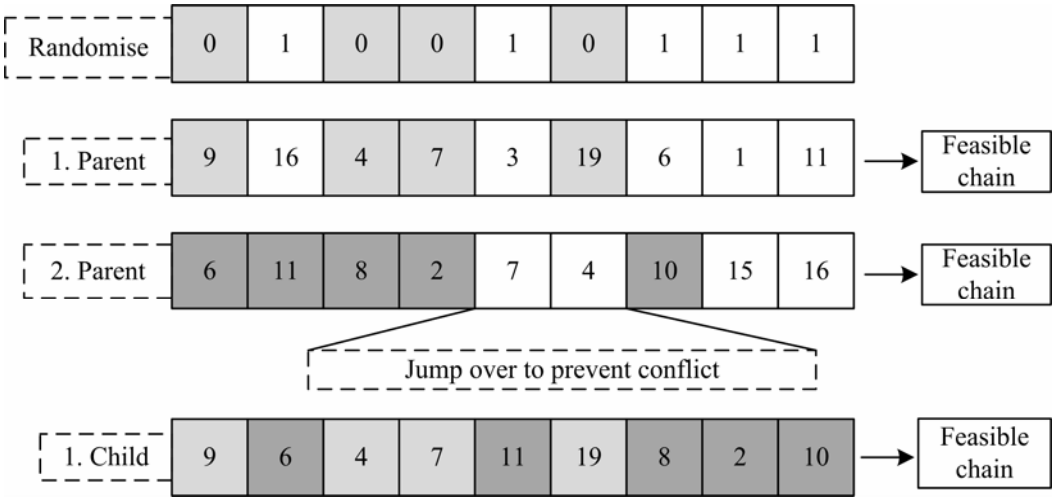

Figure 4. A creation of a child with uniform crossover

4.4.2. Mutation operator

Côté et al. [39] used uniform mutation operator, Erben[35], Santiago-Mozos et al. [40], Cheong et al. [46], Puente et al. [50], Pillay and Banzhaf [9] used swap mutation operator, Beligiannis et al. [43] created a mutation operator that swap and randomize, Pongcharoen et al. [44] used a day shift change mutation operator. Since diversity cannot be obtained by crossover operation, a mutation operator is necessary for the variation of chromosomes in the population. For the chromosome structure under implementation, swap mutation operator is not able to provide full diversity as it only interchanges the values inside the chromosome. We selected and modified the random replacement mutation operator [33, 36] to use in GA. It is demonstrated in Figure 5 as an example that randomly selected gene is replaced with a new value in the selected chromosome. This new value cannot be same as one of the values in the belonging chain, so these gene values are removed from the pool of gene values.

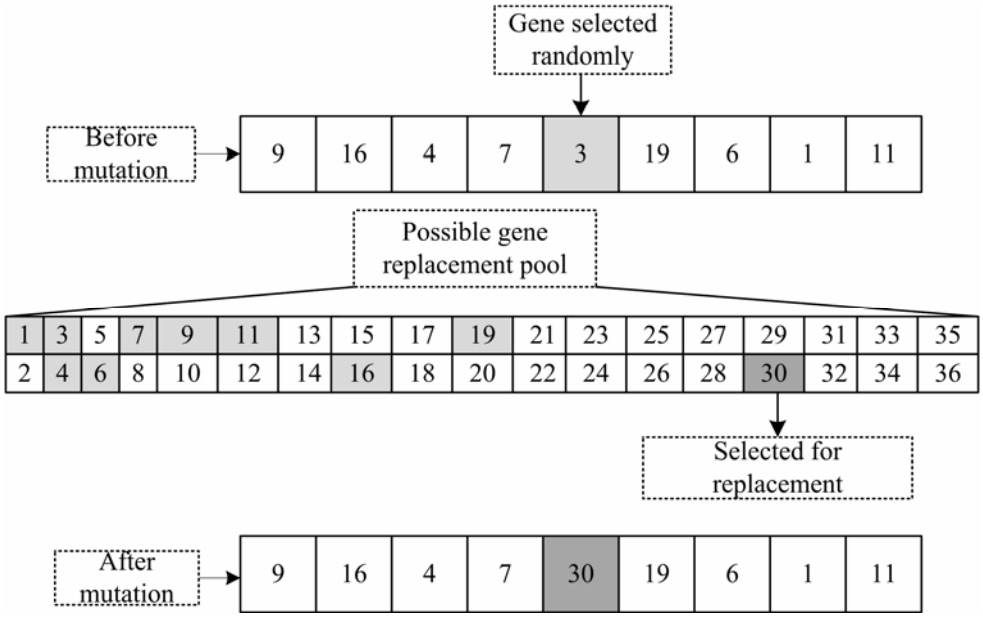

Figure 5. Mutation in a chain of selected chromosome

Mutation is a very sensitive operator that may quickly create better candidates and slow down the whole operation. That's why it is required to pay much more attention to this component. The question here is how much diversity is more appropriate for the problem under study. In order to answer this question, a new parameter is added to the set of experiments called mutation tactic. This parameter decides to replace 1, 2 or 4 genes at a time.

4.5 Embedded repair function to remove infeasibilities

High penalty cost [33] which is added to fitness value for hard constraints, described in section 3, aims to prevent infeasible chromosomes to survive at next generations. During experiments with high penalty cost model, it has been seen that this idea works well to get rid of infeasible solutions. However, we got the sense of GA performance may have been affected worse due to letting infeasible chromosomes to be occupied in general population even though they are not going to be eventually selected [32]. Thus, we came up with an alternative model that assures feasible chromosomes in general population called repair function model. This repair technique is commonly used to block hard constraints for GA [12, 36, 39, 42, 44, 46, 50]; so we designed a special repair algorithm for the problem under study and embedded it to the GA.

Chromosome repair operation is executed regarding the information of related shared exams and also whether or not each exam is taken from upper and lower terms. In the example given in Figure 6, it is assumed that $O1$ exam is a shared exam belonging to xth chain. Before repair operation, xth chain has the genes with values 4, 19, 1 and 2, respectively. $S1$ exam is accepted as a defect and the time slot of 4 is going to be unloaded because $O1$ exam already requires that time slot. Thus, "shared exam and chain" conflict is prevented. One of the possible time slots is selected randomly and assigned to the xth chain's empty gene. This process is very similar to mutation operation described in the previous subsection. Other related chain value adjustments are done with the same logic by checking the data matrix of shared exams.

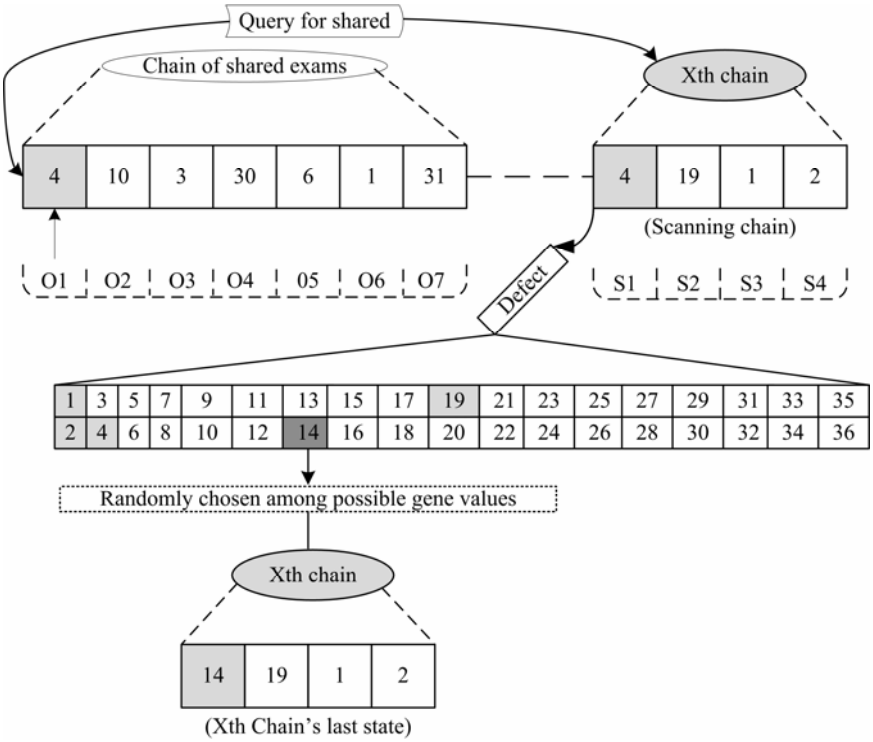

Figure 6. Query for shared exams and repair process

In the example given in Figure 7, it is assumed that *A*1 exam is both taken from one and two upper terms. Since *A*1 exam belongs to the *x*th chain, chains (*x*+1) and (*x*+2) should be checked. In (*x*+1)th chain, there is no conflict; but exam *C*1 in chain (*x*+2) is assigned to the same time slot of *A*1. Therefore, *A*1 exam is labeled as a defect and the time slot for *A*1 has to be replaced. The new time slot should be determined so that the time slot will be different that the values in chains *x*, (*x*+1) and (*x*+2).

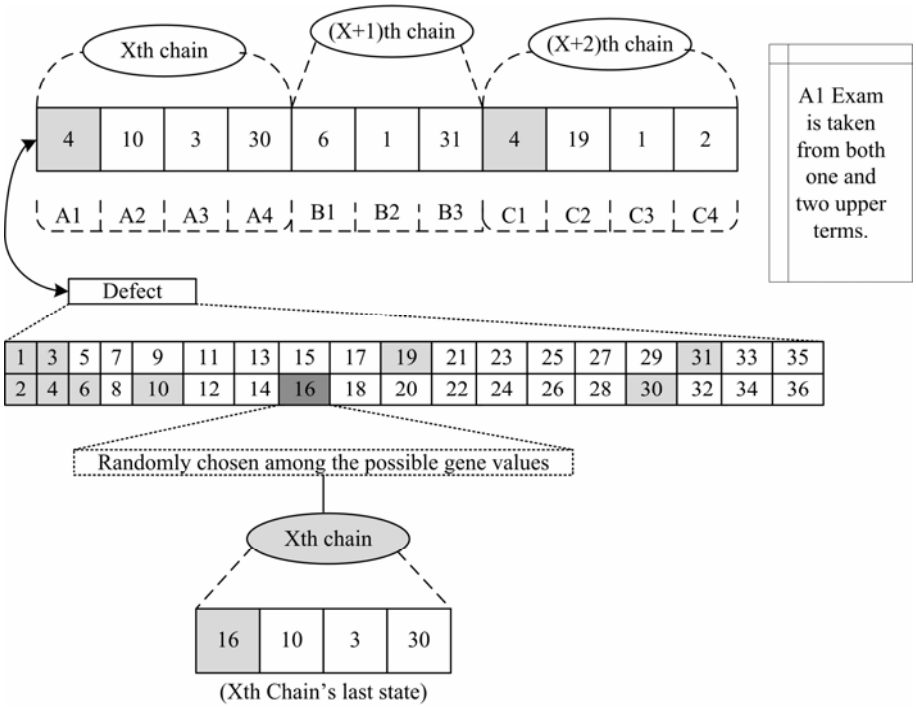

Figure 7. Scan upper chains and repair

First, each gene in the chromosome is checked for shared exams, lower and upper terms conflicts. If any

conflicts are met, repair process is applied for that gene. When the repair process is applied from first gene to the last one, any change in the chromosome that has been done is recorded because the following changes may interact with previous changes. Therefore, a replacement is done considering any possible conflict to ensure the solution is feasible.

4.6 Chromosome selection and flow of the proposed GA

The pseudo code for general flow of the proposed GA is given in Table 1.

Table 1. Pseudo code for general flow of the proposed algorithm

```
SET initial population size to double of population size
SET number of offspring to crossover rate times population size
SET elite size to elitism rate times population size
CALL generate initial population with initial population size RETURNING initial population
SET number of mutants to chromosome length times mutation rate times population size
FOR each chromosome in the initial population
        CALL determine fitness value with input variables RETURNING fitness value and conflict matrix
END FOR
SORT fitness values of each chromosome in initial population in descending order.
DETERMINE general population from initial population
SET trial to zero.
REPEAT
        CALL create mating pool with standard division coefficient RETURNING Pool
        IF Pool has not enough chromosomes THEN
            RETURN
        ELSE
            CONTINUE
        END IF
        CALL crosswhom with Pool and number of offspring RETURNING Mating Set
        CALL crossover with Population, Mating Set RETURNING Offspring
        CASE mutation tactic OF
            1: CALL mutate 1 gene with Population, number of mutants, maximum time slot RETURNING Mutants
            2: CALL mutate 2 genes with Population, number of mutants, maximum time slot RETURNING Mutants
            4: CALL mutate 4 genes with Population, number of mutants, maximum time slot RETURNING Mutants
        END CASE
        CALL select best of chromosomes with Population, Offspring, Mutants, Elites RETURNING Population
UNTIL trial < number of generation
```

Children created by crossover operator, mutants created by mutation operator and the general population are inserted into the selection process. Best of these chromosomes are passed onto the next generation, the rest is destroyed. In order to save some of better individuals in the general population, elites are directly transferred to the next generation without any modification. The only difference between high penalty cost model and the repair model is the integration of the repair function in the creation of initial population and after the creation of new children and mutants. Since high penalty cost is redundant when repair process is applied, it was not necessary to modify fitness formulation. Selection of new generation and the general flow of repair model are demonstrated in Figure 8. This loop is continued until two ending criteria: (1) Upper limit of number of generations is reached; (2) Improvement rate of the fitness value through ten generations is satisfied. The experimental results and analysis for both models are given in section 5.

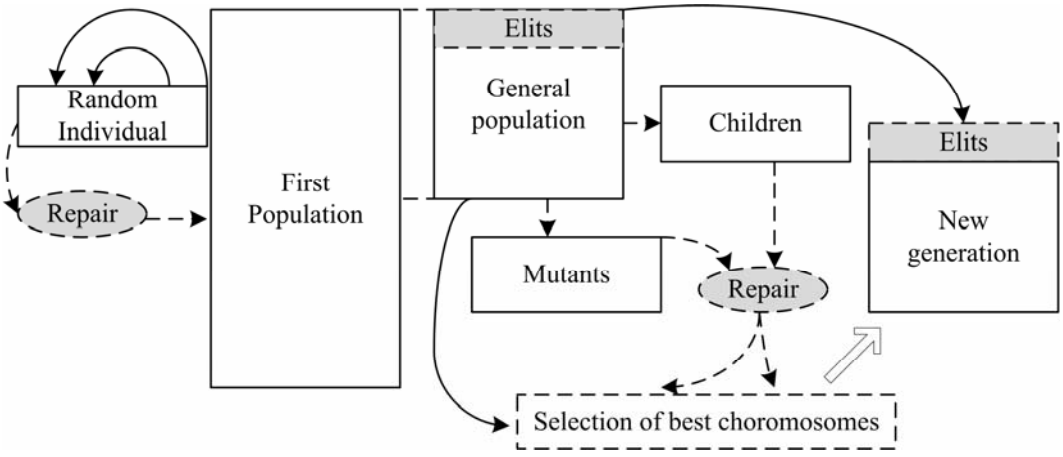

Figure 8. Selection of new generation with repair function model

4.7 Allocating exams to rooms

Although the room allocation is not in the focus of our study, for the sake of completeness we developed a simple heuristic to assign exams into rooms. Every room is considered to be identical except their seating capacities. After the final timetable is constructed by GA, a simple heuristic is applied to assign exams to rooms for each period without allowing double-booking. When the seating capacity of one room is not enough, two or more rooms are combined as one. The heuristic creates every possible combination of rooms, called virtual rooms, and allocate where necessary. If the room assigning heuristic fails because of the capacity limit given in constraint vii, it checks for other solutions in the database with same or approximate fitness values. If it no allocation can be made, the heuristic automatically calls the GA again with the best known parameters by taking some of the feasible good solutions as initial population in order to come up with a solution for which the room allocation can be made.

5. EXPERIMENTAL RESULTS and ANALYSIS

We have tested our proposed algorithms on data sets collected from the College of Engineering of Pamukkale University. We have investigated the effects of GA parameters and operators, compared the algorithms' performance and time complexity. The design of each experiment, the results and analyses are provided in the following subsections.

5.1 Investigation on the GA parameters and operators

The experimental factors and levels considered in experiments are shown in Table 2. Population size (*ps*), elitism rate (*er*), crossover rate (*cr*), mutation rate (*mr*), mutation tactic (*mt*) and standard deviation coefficient (*co*) are the factors that are used to investigate GA performance.

Table 2. GA experimental factors and its levels

			Levels		
	#	**Name**	**Low**	**Medium**	**High**
Factors	1	*ps*	20	40	80
	2	*er*	0.1	0.2	-
	3	*cr*	0.3	0.6	0.9
	4	*mr*	0.01	0.05	0.1
	5	*mt*	1	2	4
	6	*co*	0	3	-

In order to carry out a full factorial experimental analysis with the factors and levels provided in Table 2, 1620 experiments are totally required with 5 replications for each experiment set. Replication is necessary, because randomness may eventually cause different performance. For one experiment with our data set, high penalty cost model required approximately 10 minutes and repair function model required average of 25 minutes to complete on a Pentium IV 1.80 GHz Intel processor. A full factorial analysis would take approximately 40 days of computing time. This is just too long. So we preferred to modify Taguchi's L_{27} experimental design table which is well suited to experimental factors and levels for selected parameters. Modification of Taguchi's L_{27} table is necessary because in the original table there are 13 factors yet in our case there are 6 factors. Final experiment sets are shown in Table 3 after removing unnecessary columns and applying appropriate dummy treatment where necessary in order to get the general idea of factor affecting the GA model. Therefore 135 experiments are executed for each proposed GA models.

Table 3. Experiment Sets

Experiment	*ps*	*er*	*cr*	*mr*	*mt*	*co*
1	20	0.1	0.3	0.01	1	0
2	20	0.1	0.3	0.01	2	3
3	20	0.1	0.3	0.01	4	3
4	20	0.2	0.6	0.05	1	0
5	20	0.2	0.6	0.05	2	3
6	20	0.2	0.6	0.05	4	3
7	20	0.2	0.9	0.1	1	0
8	20	0.2	0.9	0.1	2	3
9	20	0.2	0.9	0.1	4	3
10	40	0.1	0.6	0.1	1	3
11	40	0.1	0.6	0.1	2	3
12	40	0.1	0.6	0.1	4	0
13	40	0.2	0.9	0.01	1	3
14	40	0.2	0.9	0.01	2	3
15	40	0.2	0.9	0.01	4	0
16	40	0.2	0.3	0.05	1	3
17	40	0.2	0.3	0.05	2	3
18	40	0.2	0.3	0.05	4	0
19	80	0.1	0.9	0.05	1	3
20	80	0.1	0.9	0.05	2	0
21	80	0.1	0.9	0.05	4	3
22	80	0.2	0.3	0.1	1	3
23	80	0.2	0.3	0.1	2	0
24	80	0.2	0.3	0.1	4	3
25	80	0.2	0.6	0.01	1	3
26	80	0.2	0.6	0.01	2	0
27	80	0.2	0.6	0.01	4	3

Experimental results for high penalty cost model are shown in Table 4 and in Table 5 for the repair function model in ascending order according to average fitness values and standard deviations of last generation respectively. LF_a is the average of fitness values in the last generation, LF_{sd} is the standard deviation of fitness values in the last generation, t_{cpu} is CPU time in seconds during the total operation of each experiment and G_a represents average generation number executed during the operation of designed models. 10th and 22nd experiment sets appear to provide best results regarding average fitness values, standard deviation, CPU time and number of generations for both models designed.

Table 4. Experiment Sets Results with high penalty cost model

High penalty cost model				
Experiment #	LF_a	LF_{sd}	t_{cpu}	G_a
10	4344	58	1084	200
22	4345	30	2749	205
19	4354	35	1635	260
25	4387	65	858	426
16	4399	69	680	270
13	4412	59	359	324
17	4415	70	963	639
11	4419	36	1065	387
23	4435	79	1718	284
7	4455	68	304	129
4	4471	65	196	159
8	4526	54	333	248
20	4527	126	537	148
5	4563	68	271	339
14	4774	132	273	289
2	4859	134	171	454
26	4907	178	271	163
1	4974	158	103	224
24	5131	307	1606	473
21	5260	270	535	205
9	5633	199	157	173
27	5711	406	449	298
6	5964	376	117	200
18	6134	359	152	142
3	6257	605	117	349
12	6432	478	165	96
15	7520	884	71	83

Table 5. Experiment Sets Results with repair function model

Repair Function Model				
Experiment #	LF_a	LF_{sd}	t_{cpu}	G_a
10	2217	16	2529	140
22	2218	20	6564	160
17	2231	27	2687	596
19	2236	15	3683	188
23	2236	30	4264	223
25	2238	16	1625	343
13	2246	35	728	278
11	2246	42	2937	326
16	2268	32	1503	178
8	2276	59	941	210
7	2291	56	849	101
5	2292	35	555	234
20	2310	58	1261	120
4	2322	51	449	107
2	2403	116	289	413
14	2429	43	403	209
1	2491	85	153	144
26	2493	53	349	104
21	2570	81	1262	192
24	2578	140	4633	465
27	2582	106	616	233
9	2714	242	340	128
12	2752	163	374	73
6	2792	135	304	206
18	2957	214	324	120
3	2984	148	128	243
15	3191	188	105	64

5.2. Complexity and performance of the algorithms

Detailed experiment results for five different trials of 10th experiment set are shown in Table 6 and Table 7 for high penalty cost and repair function model, respectively. FF_b is the best fitness value in the first generation; LF_b is the last fitness value in the last generation; t_{cpu} is the CPU time during the algorithm's operation and G is the number of generations. A conflict matrix is used to calculate and explain the meaning of each fitness value. We considered five different types of exam clashing for students: CT_1 : two exams are offered in the same time slot; CT_2 : two exams are offered on two consecutive time slots on the same day; CT_3 : two exams are offered with one time slot break on the same day; CT_4 : Two exams are offered with two time slots break on the same day; and CT_5 : two exams are offered on two consecutive days. Only CT_1 is related to a hard constraint, other conflict types are linked to soft constraints.

Table 6. Detailed experiment results for high penalty cost model

FF_b	LF_b	t_{cpu}	G	CT_1	CT_2	CT_3	CT_4	CT_5
19878	4388	824	152	0	0	5	50	265
19488	4414	1057	195	0	0	0	57	267
21636	4324	1226	227	0	0	0	49	290
20358	4327	1059	196	0	0	1	49	278
19891	4267	1256	232	0	0	5	49	276

Table 7. Detailed experiment results for repair function model

FF_b	LF_b	t_{cpu}	G	CT_1	CT_2	CT_3	CT_4	CT_5
6136	2202	2127	118	0	0	0	17	171
6160	2230	2218	123	0	0	1	17	163
6023	2215	2758	153	0	0	2	20	155
5910	2236	2744	152	0	0	0	17	169
6053	2201	2797	155	0	0	1	19	161

As the main focus of this study is to maximize students' success, conflict types CT_5, CT_4, CT_3, CT_2, CT_1 should be preferred, respectively. In order to understand the value of these results, fitness value and conflict matrix of the timetable which is manually constructed is given in Table 8. It can be easily stated that both of the models provide better timetables than manual construction. However, repair function model provided far better results of all. Repair function model, even with the worst parameter set, provided better timetables than high penalty cost model with best parameter set.

Table 8. Fitness value and conflict matrix of manually constructed timetable

LF_b	CT_1	CT_2	CT_3	CT_4	CT_5
9100	0	35	39	7	345

At worst case, it has been calculated that high penalty cost model requires 20,294,251,207 processes and repair function model requires 94,381,970,167 processes on the most complex case for 10th experiment set values of *ps*=40, *er*=.1 , *cr*=.6, *mr*=.1, *mt*=1, *co*=3. According to this complexity result, high penalty cost model works approximately 4.65 times faster than the repair function model. The worst time complexity was considered during the calculation of complexity. This value changes according to the difficulty of timetable construction. For instance, for the

10th parameter set repair function model was approximately 3.3 times slower than the high penalty cost model. As a result, in our case, high penalty cost model constructed faster timetables, but showed low performance. In Figure 9, the change in the fitness values for best parameter results through generations is shown for both algorithms to demonstrate performance differences. While both algorithms start with the initial value of 23482, penalty function model is stuck to 4789 at the 390th iteration and repair function model is stuck 2331 at the 354th iteration under 500 iterations at most.

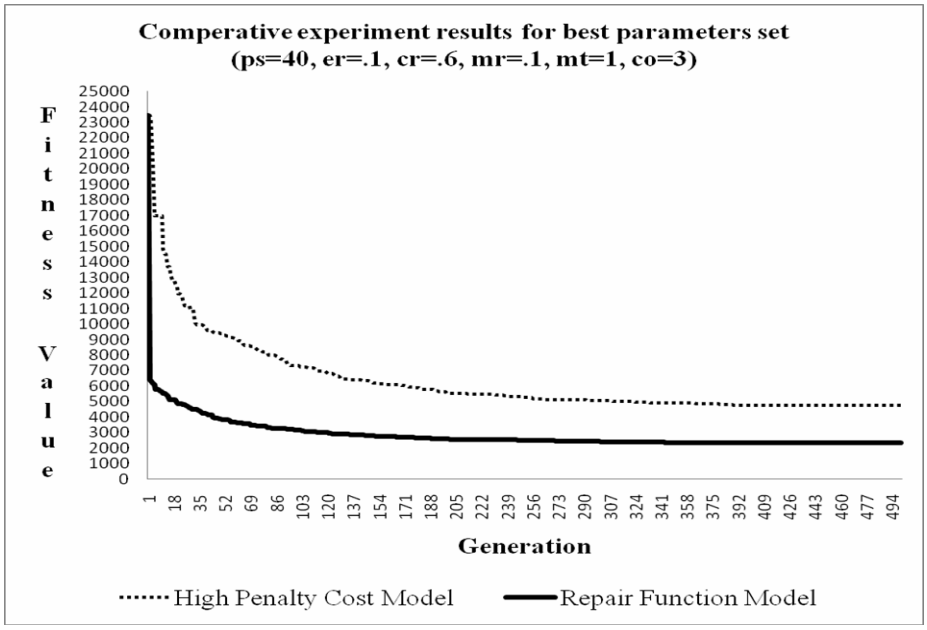

Figure 9 Comperative experiment results for best parameters set

-6. Conclusions and Future Work

In this paper, a GA based examination timetabling model focusing on students' success has been designed and examined using the data collected from the College of Engineering at Pamukkale University. This paper extended the MirHassani's [2] maximization of *paper spread* idea by considering the difficulty of each exam which is positively linked to the students' success. Considering difficulties of exams requires more complex GA structures. We used two different GA modeling approaches, clearly explained them in the paper and compared them with each other and the exam schedule created manually. First GA model works connected to the high penalty value minimization and the second model includes a repair function that forces the algorithm to work on only feasible solutions by assuring all of the hard constraints to be satisfied in each stage of the algorithm.

The timetabling literature is extensive. Even small contributions are valuable including applications of the models to real life problems. This paper includes the effects of difficulties of exams into the objective function. The model also deals with shared exams and offers a real life application case in a university in Turkey. Almost in all universities in Turkey, exam timetabling is a very difficult task mostly done by "research assistants" causing waste of valuable research time. This is a valuable contribution of proposed study on the application side. The model presented can be implemented in software to make it use practical. On the theory side, the models presented can be extended to include the fuzziness that may arise when understanding the difficulties of exams. In addition, new soft constraints may be added to problem such as student preferences. So far in the existing literature, only constraints set by the institutions and lecturers have been taken into account.

Acknowledgments : This research was partially funded by Scientific Research Projects Division of Pamukkale University under grant number 2007FBE0014.

REFERENCES

[1] Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. A graph-based hyper-heuristic for educational timetabling problems. ***European Journal of Operational Research****, 176*, 177-192 (2007)

[2] MirHassani, S. A. Improving paper spread in examination timetables using integer programming. ***Applied Mathematics and Computation****, 179*, 702-706 (2006)

[3] Qu, R., Burke, E., McCollum, B., Merlot, L., & Lee, S. A survey of search methodologies and automated system development for examination timetabling. ***Journal of Scheduling****, 12*, 55-89 (2009)

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

[5] Burke, E., Petrovic, S., & Qu, R. Case-based heuristic selection for timetabling problems. ***Journal of Scheduling,*** *9*, 115-132.(2006)

[6] Abdullah, S., Ahmadi, S., Burke, E., & Dror, M. Investigating Ahuja–Orlin's large neighbourhood search approach for examination timetabling. ***OR Spectrum****, 29*, 351-372 (2007)

[7] Petrovic, S., Yang, Y., & Dror, M. Case-based selection of initialisation heuristics for metaheuristic examination timetabling. ***Expert Systems with Applications****, 33*, 772-785 (2007)

[8] Asmuni, H., Burke, E. K., Garibaldi, J. M., McCollum, B., & Parkes, A. J. An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables. ***Computers & Operations Research****, 36*, 981-1001 (2009)

[9] Pillay, N., & Banzhaf, W. A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. ***European Journal of Operational Research****, 197*, 482-491 (2009)

[10] Qu, R., Burke, E. K., & McCollum, B. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. ***European Journal of Operational Research****, 198*, 392-404 (2009)

[11] Burke, E. K., Eckersley, A. J., McCollum, B., Petrovic, S., & Qu, R. Hybrid variable neighbourhood approaches to university exam timetabling. ***European Journal of Operational Research****, 206*, 46-53 (2010)

[12] Pillay, N., & Banzhaf, W. An informed genetic algorithm for the examination timetabling problem. ***Applied Soft Computing****, 10*, 457-467 (2010)

[13] Gogos, C., Alefragis, P., & Housos, E. An improved multi-staged algorithmic process for the solution of the examination timetabling problem. ***Annals of Operations Research*** (2010)

[14] Foulds, L. R., & Johnson, D. G. SlotManager: a microcomputer-based decision support system for university timetabling. ***Decision Support Systems****, 27*, 367-381 (2000)

[15] Dimopoulou, M., & Miliotis, P. Implementation of a university course and examination timetabling system. ***European Journal of Operational Research****, 130*, 202-213 (2001)

[16] Dimopoulou, M., & Miliotis, P. An automated university course timetabling system developed in a distributed environment: A case study. ***European Journal of Operational Research****, 153*, 136-147 (2004)

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

[18] Burke, E. K., & Newall, J. P. Solving Examination Timetabling Problems through Adaption of Heuristic Orderings. ***Annals of Operations Research****, 129*, 107-134 (2004)

[19] Daskalaki, S., Birbas, T., & Housos, E. An integer programming formulation for a case study in university timetabling. ***European Journal of Operational Research****, 153*, 117-135 (2004)

[20] Daskalaki, S., & Birbas, T. Efficient solutions for a university timetabling problem through integer programming. ***European Journal of Operational Research****, 160*, 106-120 (2005)

[21] Avella, P., & Vasil'Ev, I. A Computational Study of a Cutting Plane Algorithm for University Course Timetabling. ***Journal of Scheduling****, 8*, 497-514 (2005)

[22] MirHassani, S. A. A computational approach to enhancing course timetabling with integer programming. ***Applied Mathematics and Computation****, 175*, 814-822 (2006)

[23] Al-Yakoob, S. M., & Sherali, H. D. A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations. ***European Journal of Operational Research****, 180*, 1028-1044 (2007)

[24] Head, C., & Shaban, S. A heuristic approach to simultaneous course/student timetabling. ***Computers & Operations Research****, 34*, 919-933 (2007)

[25] Van den Broek, J., Hurkens, C., & Woeginger, G. Timetabling problems at the TU Eindhoven. ***European Journal of Operational Research****, 196*, 877-885 (2009)

[26] Birbas, T., Daskalaki, S., & Housos, E. School timetabling for quality student and teacher schedules. ***Journal of Scheduling****, 12*, 177-197 (2009)

[27] Kahar, M. N. M., & Kendall, G. The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. ***European Journal of Operational Research, In Press, Accepted Manuscript*** (2010)

[28] Sarin, S., Wang, Y., & Varadarajan, A. A university-timetabling problem and its solution

using Benders' partitioning—a case study. ***Journal of Scheduling**, 13*, 131-141 (2010)

[29] Rudová, H., Müller, T., & Murray, K. Complex university course timetabling. ***Journal of Scheduling*** (2010)

[30] Wang, S., Bussieck, M., Guignard, M., Meeraus, A., & O'Brien, F. Term-end exam scheduling at United States Military Academy/West Point. ***Journal of Scheduling**, 13*, 375-391 (2010)

[31] Smith, K. A., Abramson, D., & Duke, D. Hopfield neural networks for timetabling: formulations, methods, and comparative results. ***Computers & Industrial Engineering**, 44*, 283-305 (2003)

[32] Dave Corne, P. R., Hsiao-lan Fang. Evolutionary timetabling: Practice, prospects and work in progress. In P. Prosser (Ed.), ***Proceedings of the UK Planning and Scheduling SIG Workshop*** (1994)

[33] Naji Azimi, Z. Hybrid heuristics for Examination Timetabling problem. ***Applied Mathematics and Computation**, 163*, 705-733 (2005)

[34] Ross, P., Hart, E., & Corne, D. Some Observations about GA-Based Exam Timetabling. ***Lecture Notes in Computer Science***, Vol. 1408 (pp. 115) (1998)

[35] Erben, W. A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling. ***Lecture Notes in Computer Science***, Vol. 2079 (pp. 132-156) (2001)

[36] Wong, T., Cote, P., & Gely, P. Final exam timetabling: a practical approach. In ***Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on*** (Vol. 2, pp. 726-731 vol.722) (2002)

[37] Sheibani, K. An Evolutionary Approach For The Examination Timetabling Problems. D. C. E. K. Burke (Ed.), ***Proceedings of the 4th international conference on practice and theory of automated timetabling*** (Vol. 2740/2003, pp. 387–396). Gent, Belgium: Springer Berlin / Heidelberg (2002)

[38] Wong, T., Cote, P., & Sabourin, R. A hybrid MOEA for the capacitated exam proximity problem. In ***Evolutionary Computation, 2004. CEC2004. Congress on*** (Vol. 2, pp. 1495-1501 Vol.1492) (2004)

[39] Côté, P., Wong, T., & Sabourin, R. A Hybrid Multi-objective Evolutionary Algorithm for the Uncapacitated Exam Proximity Problem. ***Lecture Notes in Computer Science***, Volume 3616 (pp. 294-312) (2005)

[40] Santiago-Mozos, R., Salcedo-Sanz, S., DePrado-Cumplido, M., & Bousoño-Calzón, C. A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a Spanish university. ***Computers & Operations Research**, 32*, 1761-1776 (2005)

[41] Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. An effective hybrid algorithm for university course timetabling. ***Journal of Scheduling**, 9*, 403-432 (2006)

[42] Ülker, Ö., Özcan, E., & Korkmaz, E. Linear Linkage Encoding in Grouping Problems: Applications on Graph Coloring and Timetabling. PATAT'06 Proceedings of the 6th international conference on Practice and theory of automated timetabling VI Springer-Verlag Berlin, Heidelberg, (2007)

[43] Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., & Likothanassis, S. D. Applying evolutionary computation to the school timetabling problem: The Greek case. ***Computers & Operations Research**, 35*, 1265-1280 (2008)

[44] Pongcharoen, P., Promtet, W., Yenradee, P., & Hicks, C. Stochastic Optimisation Timetabling Tool for university course scheduling. ***International Journal of Production Economics**, 112*, 903-918 (2008)

[45] Mumford, C. A multiobjective framework for heavily constrained examination timetabling problems. ***Annals of Operations Research*** (2008)

[46] Cheong, C., Tan, K., & Veeravalli, B. A multi-objective evolutionary algorithm for examination timetabling. ***Journal of Scheduling**, 12*, 121-146 (2009)

[47] De Causmaecker, P., Demeester, P., & Vanden Berghe, G. A decomposed metaheuristic approach for a real-world university timetabling problem. ***European Journal of Operational Research**, 195*, 307-318 (2009)

[48] Goldberg, D. E. ***Genetic Algorithms in Search, Optimisation and Machine Learning.*** Boston: Addison-Wesley Longman Publishing Co., Inc.(1989)

[49] Michalewicz, Z. Genetic algorithms + data structures = evolution programs (3rd ed.): ***Springer-Verlag*** (1996)

[50] Puente, J., Gómez, A., Fernández, I., & Priore, P. Medical doctor rostering problem in a hospital emergency department by means of genetic algorithms. ***Computers & Industrial Engineering,*** *56*, 1232-1242 (2009)

[51] Terashima-Marin, H., Ross, P., & Valenzuela-Rendon, M. Clique-based crossover for solving the timetabling problem with GAs. In ***Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*** (Vol. 2, pp. 1206 Vol. 1202) (1999)