International Journal of Artificial Intelligence (IJ-AI)
Vol. 1, No. 2, June 2012, pp. 79~90
ISSN: 2252-8938

Solving University Scheduling Problem with a Memetic Algorithm

Mortaza Abbaszadeh*, Saeed Saeedvand*, Hamid Asbagi Mayani*
*Department of Computer Engineering, Sofian Islamic Azad University

Article Info

Article history:

Received May 31, 2012
Revised Jun 8, 2012
Accepted Jun 15, 2012

Keyword:

Memetic Algorithm
Genetic Algorithm
Chromosome
Population
Fitness

ABSTRACT

Scheduling problem is one of the Non-deterministic Polynomial (NP) problems. This means that using a normal algorithm to solve NP problems is so time-consuming a process (it may take months or even years with available equipment), and thus such an algorithm is regarded as an impracticable way of dealing with NP problems. The method of Memetic Algorithm presented in this paper is different from other available algorithms. In this algorithm the problem of a university class Scheduling is solved through applying a new chromosome structure, modifying the normal genetic methods and adding a local search, which is claimed to considerably improve the solution. We included the teacher, class and course information with their maximal constraints in the proposed algorithm, and it produced an optimized scheduling table for a weekly program of the university after creating the initial population of chromosomes and running genetic operators. The results of the study show a high efficiency for the proposed algorithm compared with other algorithms considering maximum Constraints.

Corresponding Author:

First Author,
Faculty Member of Computer Department,
Sofian Islamic Azad University,
5 University Road, Sofian Township, Tabriz County, Iran.
Email: M_A138@yahoo.com

1. INTRODUCTION

Scheduling algorithms or Timetabling Problem (TTP) are generally difficult problems with high computational complexity, which are known as NP-Complete problems. One of the most complex scheduling problems is the university scheduling and programming. It is very difficult and even impossible to solve such a scheduling problem through normal algorithms because of very large state space and numerous parameters with various constraints. Indeed, scheduling problem has been known as a NP-hard problem, which is a type of decision making problem that is solved through polynomial scheduling NP algorithms [1]. In university scheduling problem, scheduling mainly means assigning teachers to courses and placing them in classes at specified time intervals considering necessary constraints.

As stated above, using Genetic Algorithm (GA) technique to solve scheduling problem relying on Darwin evolutionary theory is one of the most popular and widely used algorithms which has been used in numerous cases and produced notable results [4, 5, 6, 7]; moreover, there are other methods for solving TTP, of which Memetic Algorithms, Graph Coloring Algorithms and Colony of ants are the most important [3, 8, 16].

Genetic algorithms similar other algorithms in first run steps identification an area of problem state space that local and global optimums are located very well but act some slow at continue way, which we offer a Memetic algorithm with new ideas in genetic operators that we show performance improvement algorithm in this research. Genetic algorithms, like other algorithms, appropriately identify in the initial

running steps an area of problem state space in which local and global optimums are located, but, proceeding towards global optimum, they act rather slowly. The present study aims to contribute to algorithm improvement by offering a Memetic algorithm with new ideas in genetic operators.

In the present paper, we attempt to efficiently solve the university courses scheduling problem with all the relevant constraints through a Memetic Algorithm, which is a combination of a new genetic algorithm and a local search. To this end, we determine the starting and finishing hours of the time table for each chromosome, and then divide it into small time intervals, in which teachers and courses are scheduled considering both soft and hard constraints.

Numerous studies have been carried out in this field. Ben Paechter & R.C. Rankin have presented a method based on Memetic Algorithms along with effective local search which has been applied to the first timetabling competition in 1994 [8]. Edmund Burke, David Elliman, and Rupert Weare have proposed Genetic Algorithm at the Department of computer Science, University of Nottingham, which only applies to acceptable time intervals and is related to user interactively [3]. Khaled Mahar introduced a genetic algorithm for solving courses scheduling problem in 2006. He applied it to a university and reported its results [4]. Wilhelm Erben & Jurgen keppler also designed and implemented optimized tables of weekly courses utilizing Genetic Algorithms in 1995 [9]. In his proposed method, Michael W.Carter (2001) divides a problem into several sections and uses a greedy algorithm and an algorithm based on Lagrange function for allocating time intervals and classrooms, respectively[2].

Considering computational complexity of Genetic Algorithms, measures have been taken to increase their speed. Abramson D. &Abela J (1992) have taken advantage of parallel Genetic Algorithms to solve the problem of designing timetables for schools, reporting a relatively high time advantage compared with normal genetic algorithms [10].

Varac J et al. (2002) have presented a complete review on the application of genetic algorithms for designing to designing time tables [11]. Also, Moschopoulosa et al. (2008), in Belgians, have proposed and used an adaptive algorithm on the base of evolutionary computations for designing schedules in Greek high schools [12].

2. PROBLEM FORMULATION

In university course scheduling problem, any response or timetable is regarded as a chromosome and accordingly we can consider the scheduling problem functionally as follows:

$$F: T * C * P * L \rightarrow \{0, 1\} \quad (1)$$

Where C, T and P refer to course set, teachers set, and allowed time of classes, respectively; L refers to place or class, and F (T, C, P, L) means presenting the course C, by teacher T, on time P, and in place L.

We use the following formula to calculate chromosome size:

$$L_C = N_C * (\mathrm{C} * 2) \quad (2)$$

Where, L_C refers to chromosome size, C means course number, doubling of which creates teachers code according to each course, and N_C refers to the divided time credit number, which can be obtained from following formula.

Calculating time periods number in each chromosome:

$$N_C = ((E_{Time} - S_{Time}) * 4) * W \quad (3)$$

Where, N_C refers to time period number which is assumed 15 minutes intervals witch show each time period(4*15=60min=1hour) and is changeable through changing the required coefficient rate ($E_{Time} - S_{Time}$); S_{Time} shows starting hour and E_{Time} shows ending hour of scheduling, which is receivable from the user with 15 minutes precision. This means that the user can specify the opening and closing hours of classes. W represents those days of week which will be scheduled, and its rate is on the interval (0<W<8).

2.1. Population size calculation

When determining the population size (parents and children population) within the proposed method, if the population size is very large, the speed of algorithm run will greatly decrease. On the other hand, if the population size is small, the problem will converge into the answer which is not necessarily optimal.

The proposed solution is that population size is a function of the number of events and the length of chromosomes. This means that the more the number of events, the larger the population size is taken, and the longer the chromosomes, the smaller the population size is considered. The population size can also be adjusted manually.

$$P_{Size} = \frac{L*N_C}{E-1} * K \ , \ Ch_{Size} = P_{Size} * 1.3 \quad (4)$$

In the first equation, Ch_{Size} is the child population size, which is 1.3 times as large as parent population, which has been obtained through trial and error, and in the second equation, P_{Size} the proposed is parent population. L stands for the existing class number; E refers to number of events in a chromosome;N_C is the desired time periods (equation 3), as mentioned before; and K is constant factor for regulating the population size, which has been calculated through trial and error and is approximately 45.

3. CONSTRAINTS

As mentioned above, there are some constraints in scheduling courses, which can be divided into two basic parts: hard constraints, which are totally considered in scheduling, and soft constraints, which are applied to problems as far as possible. In most papers involving Memetic and Genetic algorithms, only some of basic constraints have been used for scheduling [1, 6, 5, 4, 15].However, we attempt to use maximum constraints in scheduling to obtain the best possible response.

Considering that the number of applied constraints could directly affect the algorithm speed and response finding, we attempt to minimize as far as possible the number of constraints through some methods. For this purpose, we deleted some of these constraints through establishing the initial population by using permutation method and applying genetic operators to chromosomes. Using permutation method implies that there should be no repeated gene in chromosome [14]. Therefore, deleted constraints due to the permutation maintenance in chromosomes during chromosome creation, genetic operations phases and isolation of specific[1] and normal classes are as follows:

- No course should be repeated twice a week.
- Offering all of the presented courses: all of the presented courses should be offered for scheduling in chromosome.
- Considering class type: each course which needs a specific class[1] should be held in this unique class.
- No class should be allocated for two courses at the same time.

3.1. Hard Constraints

- The coincidence of a teacher's different class times: One teacher should not be allocated for two different courses at the same time.
- Course interference of special entry students: The coincidence of the courses taken by the students of the same entrance
- Presentation of courses to special entry students should not coincide. A heavy fine will be imposed on for this constraint.
- Class capacity: class capacity allocated to a course should not be less than match the number of students enrolling on the course which is to be held in this place.

3.2. Soft Constraints

- Adapting class equipment to the course: when the course needs a special piece of equipment such as a computer or projector, it should be provided for the class.
- The number of intervals between the courses of a teacher: It would be better if there were no intervals between the courses.
- Determining maximum daytime teaching period for students in a term: As far as possible, teaching time should not exceed 6 hours a day during a term; however, this time limit could be changeable.

  -Determining maximum daytime teaching period for teachers in a term: More than 6 hours a day teaching period should not be allocated for teachers; however, this time limit could be changeable.

  -Maximum and minimum constraints of course unit in a term for each lecture: Any teacher at any university department have constraint about the number of course unit which should Not be more than maximum rate in scheduling or less than minimum rate of course unit (e.g.) it is considered not more than 23 units and not less than 16 units. For example, the number of course units taught by teachers ranges from 16 to 23.
- Presentation of non-laboratory three- unit courses for 3 successive hours: important courses should not be taught for three successive hours because it causes teaching quality to decrease.

- Presentation of non-laboratory three- unit courses for one day: Besides the previous condition, specialized and important courses should not be taught in one day; rather, the time of teaching should be divided into two days.
- Presentation of non-laboratory three- unit courses for two successive days: time of three-unit courses should be distributed during a week, including a one- hour and a two- hour time.
- excluding intervals between courses: There should be no intervals between class times. There should not be vacancy among class times.
- The days of a term during which the students and teachers are present: Considering the university schedule and other constraints, the students' as well as teachers' attendance days during a term should reach a minimum.
- Constraint of the teacher's attendance days: each teacher announces the days of his attendance in the faculty.
- Constraint of the lecture's teacher's attendance time on attendance days: teachers can inform attendance time constraint on attendance days.
- Class availability: some classes may belong to special group or faculty at certain times.
- Time intervals between taught courses over a term: There should be a time interval between the courses being taught during a certain term. For instance, for 6 successive courses, there should be a 15- minute interval.
- Holding a normal class in a special class: If the normal class is held in the special class, soft constraints will be added to the fitting by low coefficient. The reason behind this is to prevent its occurrence as far as possible.
- Teaching priority: paying attention to teachers' preferences for the courses they are to teaches an important issue, which is done separately in each group, and is applied to the fitting with lower coefficient.
- Meals time: The time which should be given to students and teachers to have a meal.

In the case of the meal time, it should be mentioned that a half-hour interval is set to solve the meal time problem using the course scheduling chart so that no class should be met during this half-hour time in a given term, i.e. at first, courses of several terms from the chart at one half-hour, and students of the next several terms from the same chart at other half-hour, etc., are given meal time. The start and the end hours of meal serving are time intervals equal to the specified period in formula (3), which is determined by the user from beginning of desired meal time.

In this method, not only is the students' lunch time of is taken into account but also a considerable amount of time is saved for teaching the courses taken by the students of other terms. In the case of the meal time for teachers, it should be mentioned that at a certain meal interval, the taught courses of each teacher are checked so that a free time is made for having meal at this interval; otherwise, it will be added to the soft fitting in a lower coefficient.

it is noteworthy that non-connectivity of course hours in some cases such as 3-hour laboratory courses and 3-unit courses 2 hours of which should be connected is considered the very first time should be considered from the very beginning and we regard those course hours as successive courses. Moreover, there are some courses which are run two hours a week and these two hours should also be successive. Finally, total fitness includes the weight sum of hard and soft fines. Hard fines have more weight than the soft, and their computation method is presented in formula (5).

4. SIDPLAYING OF CHROMOSOMES

As mentioned in the formulation section, in this research each timetable has been assumed as a chromosome, and every chromosome is defined as a two-dimensional array. Indeed, producing a chromosome is defined as the transformation of phenotype space into chromosomes. As it is seen in figure, columns of this array specify the intended class by class code, in which two columns are considered one class (the first column of each class specifies the course code, and the second column specifies the teacher code of that course).The development of a chromosome is done through permutation.

Is should be mentioned that specifying which group the presented is carried out by considering a predetermined course, which is extractable by using the course feature which has not been shown directly in chromosome. Lines of this chromosome are developed at two stages: dividing into the days of week is done at the first stage, and dividing the days into time intervals is done at the second stage. Formula (3) shows the method of setting time intervals, i.e. from the start of each day's scheduling to its end, which is divided into changeable periods.

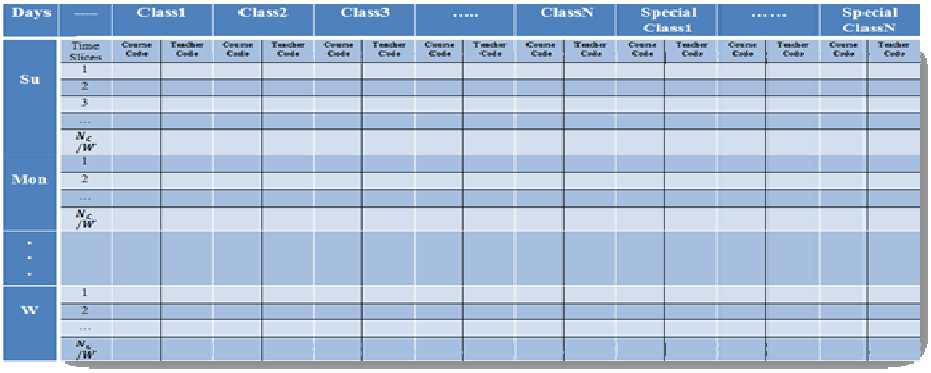

Figure 1.Displaying a chromosome

5. CROSSOVER OPEATOR

In the proposed algorithm, crossover is performed at three stages: at the first stage, we organize the population considering the fitness rate of any chromosome (formula 5), Then, as much as 10% of the child population, the chromosomes of parent population, which are of low fitness, are transferred without making any changes to the next generation, i.e. the better part of the parent population (Elitism).

At the second stage, we use the next 40% of children population utilizing SB-PMX algorithm (Sector Base-Partially Mapped Crossover), which has been studied in detail by Enzehya (2002), in [13], and run SB-PMX algorithm through selecting parents randomly from total parent population, in which the permutation feature of genes is maintained in chromosomes. In SB-PMX, following the crossover of two chromosomes with permutation, chromosomes undergoes restoration, given that time intervals of presented courses will be in contact, and the presented courses which are in contact with other course are transferred to other free time in the same chromosome in order to solve the problem.

It should be noted that if the restoration of the created chromosome is not possible, the created child will be deleted and crossover is performed again. At the same time, the crossover of the constraint of normal and special classes is performed separately and in pairs, i.e. the normal class part of first chromosome will be considered a separate chromosome, and it will only crossover with special class part of a second chromosome.

At the third stage, we use cycle crossover algorithm (figure 2) and complete the last part of the population assuming that every gene together with its place from one parent gets to the chromosome. The procedure for each of the constraints of normal and special classes in chromosomes is independent.

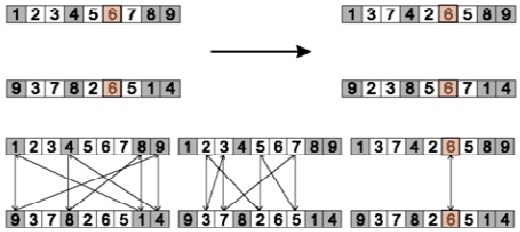

Figure 2. Displaying crossover cycles

Since the time of some courses may overlap in the method, the same restoration function as SB-PMX is performed on children created in cycle crossover.

6. MUTATION

The task of Mutation operator is to avert the homogeneity of population and entrapment of algorithms into local optimized places. In the proposed method, our idea about Implementation of mutation operator is to do mutation at three different states: In the suggested method, to implement the mutation operator, the mutation is performed in three different stages: when we say a mutation occurs in a chromosome, three different mutations occurring in one chromosome are applied to the children population

according to determined mutation rate, so that in fact three mutation are considered one mutation at three stages.

Mutation 1: The primary mutation is the one in which time of a presented course and its teacher is randomly replaced by the time of other course along with the teacher within acceptable limits this action is performed only in a special or normal class interval, which leads to the elimination of irrelevant responses for this purpose, however, the presented class time is checked with second course time so that the interchange of two genes would be possible. If interchange is not possible, the free time after and before each course will be checked, respectively, in order that there will be sufficient free time for interchange. The interchange of two genes requires that times of the courses be equal, or there should be sufficient free time before and after them, so that there will be no interference and mutation in the two genes. It is noteworthy that if the interchange was not possible, the other two genes would be randomly selected and operations would be repeated, as it is seen in figure (3).

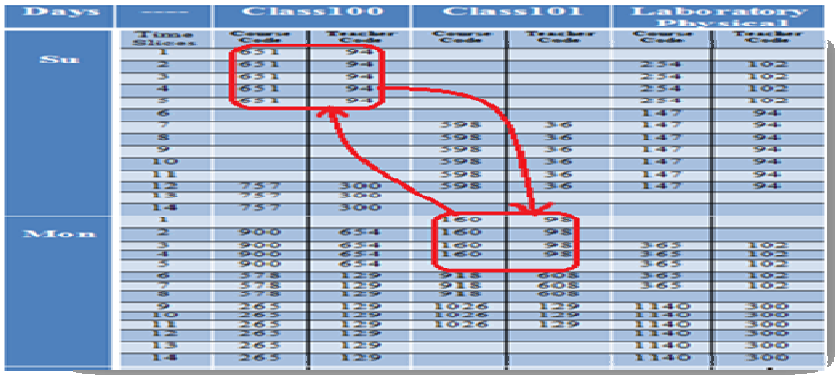

Figure 3. Display of mutation 1 in a chromosome

Mutation 2: In the second mutation, a number of teachers in each chromosome are interchanged by other teachers in the same chromosome; however, in this mutation, the course group of teacher is considered with the presented course and the skill of teacher for teaching a new course. A teacher pair is randomly selected from teachers list; therefore, the course interchange of each teacher pair is possible. At least one teacher pair is often possible. There is usually at least one teacher pair in each chromosome to interchange their courses.

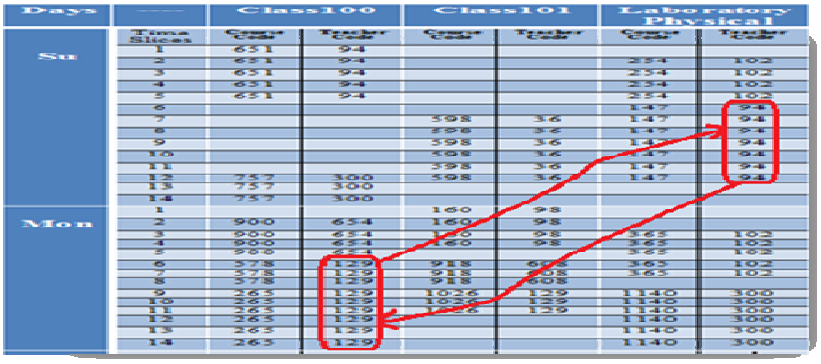

Figure 4. Display of mutation 2 in a chromosome

However, if there is no such a possibility, this stage of mutation will not occur. A sample of this mutation demonstrated in figure (4).

Mutation 3: In the third mutation, a free space is randomly selected from successive times and then one of the presented courses along with teacher is transferred to it. It should be mentioned that if there is no appropriate free time for mutation in chromosome, this stage of mutation will not occur. Figure 5 shows a sample of mutation.

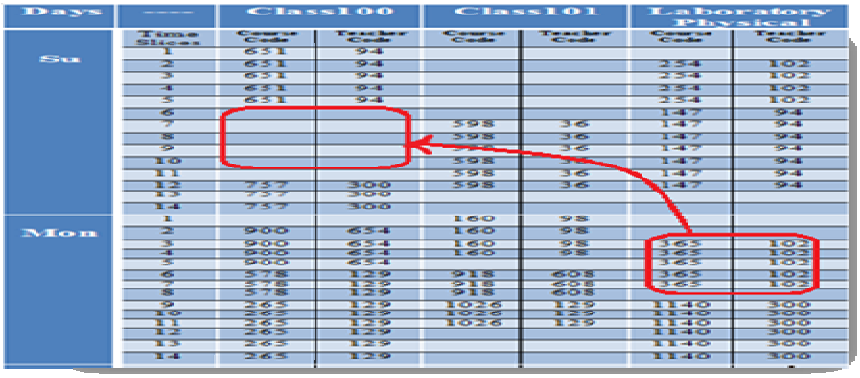

Figure 5. Display of mutation 3 in a chromosome

It is noteworthy that if the population moves toward homogeneity through repeating generations, mutation rate gradually increases in order to prevent homogeneity (This will be discussed later in the Implementation part).

7. FITNESS FUNCTION

Fitness function indicates the acceptability rate of a chromosome. In this case, a fine is imposed on not following the above mentioned constraints. Fitness is defined as follows:

$$Fitness(n) = \sum \boldsymbol{H_i} * \boldsymbol{K_{Hard}} + \sum \boldsymbol{S_j} * \boldsymbol{K_{Soft}} \quad (5)$$

Where, n refers to the proposed chromosome; H_i is the total weight of violated hard constraints related to the desired chromosome;K_{Hard} is a constant for increasing the pressure on hard constraints weight; S_j stands for the total weight of violated soft constraints related to desired chromosome, and K_{Soft} is a constant for determining the pressure on soft constraints weight. We always have($K_{Hard} > K_{Soft}$).

8. SELECTION FUNCTION

In this section, we select once more the primary population from the created child population chromosomes after running genetic operators on chromosomes with respect to the fitness of each chromosome. Considering various tests which performed through different methods, the best result has been obtained by integrating two single-axis and tournament roulette wheel Method so that 72% of the population is selected through creating a single-axis Roulette wheel on total population. In this method, the fitness rate of each chromosome determines the selection chance of each chromosome and the remaining 28% are selected through choosing the tournament where Chromosomes compete with each other in pairs. It should be mentioned that selection is done from 1.3- folds of the primary population, and children outnumber parents according to formula (4).

9. LOCAL SERACH

A local search algorithm (like figure 6) searches from an initial zero state to a neighbor state P_i, which is a chromosome. At every loop, the search starts with an initial P state and extends to the search space located in the vicinity ofP_i+1 area. In every stage, the search, relying on the fitness function, examines the quality of every state and continues to find the best chromosome nearby. There are three local searches called Hill climbing, simulated flux and uniform motion. These three types of searches, or a combination of them, are employed in different presented Memetic algorithms used to deal with different optimization problems.

Considering that the local searches generally have a high time complexity, we use an innovative local search in the proposed algorithm. In the implemented method, considering the three hard constraints addressed in the 'Constraints' section, by local search, the limitation of the adjacent chromosomes for finding the chromosomes that have not violated these three constraints takes place, and regarding the maintenance of the previous conditions of the existing chromosome and other constraints, the knowledge obtained from the detected chromosome will be used in making the necessary changes intelligently in the chromosome.

Implementing this method for generations improves the chromosomes, and in general, gives rise to the improvement of algorithm performance.

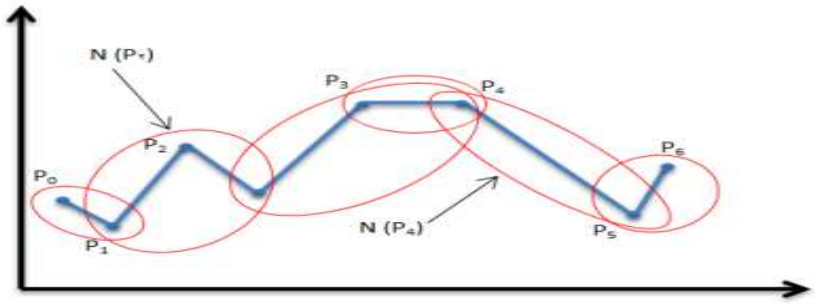

Figure 6.Display of Local Search

10. INITIALIZATION

To initialize the primary population chromosomes, we use possible method randomly and independently from fitness, but considering the effect of some of the hard constraints having permutation quality, we are to take into account some of the soft constraints during the creation of the primary population to obtain good results. Chromosomes generally suffer from low fitness in the primary population because soft and hard constraints are not applied completely to initialization. This is because if we want to apply all of the soft and hard constraints, it will be impossible to create the primary population, and if we consider none of them in creating the population, the speed and chance of finding optimized response will decrease; therefore, we do not consider any them in creating population; thus, we regard initialization as semi random rather completely random.

Considering what was stated, some conditions are initially checked during the creation of the primary population. There is no need to solve them as constraints with respect to permutation in genetic operators. The constraints are as follows:

- Each special or normal class is placed in its own part.
- No course has time overlap with other course in a class.
- Those courses are considered that should be scheduled totally or separately. Only the courses that should be scheduled totally or separately are counted.
- All the presented courses should be included in chromosome, and if this is not possible, the user will be informed after searching and computation that this rate of courses cannot be scheduled in these numbers of classes.

The Items that influence the finding of a response in non-random creating of the primary population are as follows:

- Class capacity is taken into account during course allocation.
- Teachers priorities for course presentation are taken into account as far as possible.
- Class facilities and equipment are taken into account as far as possible.

11. IMPLEMENTATION

In this section, we deal with manner, order and stages of realizing the presented introduced algorithm. As we discussed in the initialization section, we initialize parents as the primary population semi-randomly after determining the number of parent and child populations, then we compute the fitness rate of each chromosome according to formula (5), afterwards we run the target test function on the primary population. The target test function is a function that checks the fitness rate of chromosomes individually to specify whether the related chromosome response is optimal or not. Here, the criteria of the target test function are test whether or not the hard constraint fitness rate is zero, whether or not the fitness rate of some soft constraints is zero and whether or not the other constraints are minimized so that if the target chromosome is found in this function. At first, a sample of that chromosome is stored as a response, and before stopping the run, the algorithm is allowed to produce further generations so that the probability of creating optimal responses through running genetic operators on optimal chromosome could increase.

In the next stage, we run the crossover operator on the primary population, and as we discussed in the section of crossover operation, after creating the child population, we run fitness run computation and target test functions on the population. Mutation operator is run only on the child population. Concerning the

mutation operator, it seems necessary to note that if mutation rate is constant, there will be the risk of being trapped in local maximum; therefore, the mutation rate is raised and is run on population with high rate in order to avoid this problem.

In the next stage, we run the local search on the child population so that the knowledge of chromosomes could be shared with one another, and finally, after running the fitness rate computational function on chromosomes, the selection function is run, replacing the initial population with new chromosomes by using the method discussed in the section 'selection function'. As stated in the formulation section, the child population size was obtained using the formula $Ch_{Size} = P_{Size} * 1.3$. This it means that the child population has become 1.3-fold; therefore, creating more children and increasing the possibility of creating a good response and selecting a new primary population from children become possible. Here, it seems that algorithm speed will generally decrease because of an increase in the number of children; however, as we will observe in results of Implementation section, using this method in the introduced algorithm will lead to satisfactory results.

The above-mentioned stages are performed until the specified rate for replicating generation number ends, or the optimal response could be found and stops after running several generations.

12. RESULTS OF IMPLEMENTATION

In proposed Memetic algorithm, initialization is done semi randomly in the special chromosome structure, and through various tests, mutation and integration are treated differently from normal genetic algorithms. The rate of mutation varies depending on the conditions. it takes place in three forms so that different mutation states are satisfied. One third of the population is directly replicated in the next generation in order to cause good chromosomes to be not lost. For the purpose of comparing a normal Memetic algorithm with genetic algorithm and proposed Memetic algorithm, a number of tests have been performed.

In the following diagrams, horizontal axis indicates generation iteration and vertical axis indicates fitness function so that when fitness rate approaches zero, it shows good results. Also, blue lines show minimum fitness resulted from evaluation function, and red lines indicate mean fitness of population. In all of the tests, the facilities are equal; (i.e. the faculty has 10 classes, 3 laboratories and 2 computer sites). Also all the specifications and information about courses, classes and teachers, extracted from courses set of Engineering faculty of Tabriz University, are similar. Moreover, integration and mutation with independent probabilities are performed on all chromosomes. It is noteworthy that the proposed algorithm has been tested on the courses offered in the Islamic Azad University, Sofian Branch and has produced acceptable scheduling results.

12.1. Running a Genetic Algorithm

We have opted for a completely random initialization in running the genetic algorithm. We assume the crossover rate 65% and mutation rate 5%, and the number of primary population is equal to that of children. In this stage, genetic operators are applied as follows:

- Mutation is done as interchanging two genes irrespective of the interference probability.
- Crossover as a 3-point crossover has been considered on the total population in which permutation is not taken into account.
- New generation selection is realized through generational method in which total primary population is replaced by children.

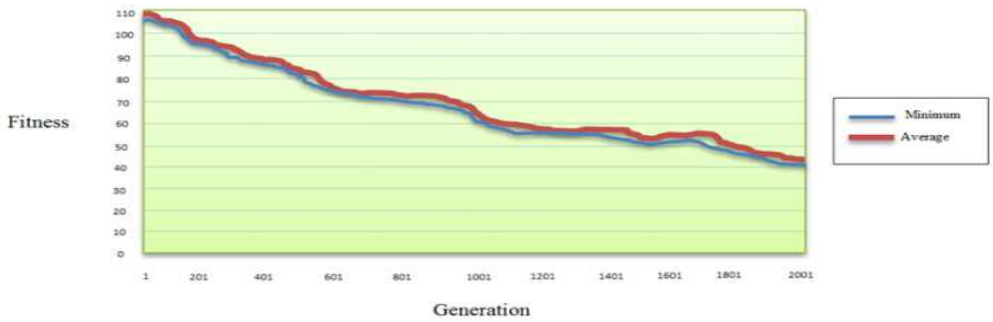

Figure 7. Fitness curve of running genetic algorithm

As shown is figure (7), initial tables have a very low fitness and even after 2000 generations, there are 40 tables of this type, which is very low compared with fitness rates obtained from the proposed

algorithm. In contrast to the proposed method, the fitness curve is not completely descending because there is no guarantee for losing optimal chromosome in each generation due to lack of direct duplication of some better chromosomes. Later, we will see that in a Memetic algorithm, by making some changes in the genetic parts and adding a local search, the quality of the developed timetables increases considerably.

12.2. Running Memetic Algorithm Considering Some Constraints

Although rates and information on the problems and previous runs are the same because we only considered hard constrains in the initialization section, classes are scheduled into two separate parts, i.e. normal class and special class, and a local search is run on population As seen in figure 8, the initial rate of chromosomes fitness is different in comparison with genetic algorithms.

12.2.1. Running conditions

We consider initialization as semi-random in running the advanced genetic algorithm. The integration and mutation rates for initialization are assumed 65%, 3% respectively; the variable in relation to the homogeneity of fitness rates is assumed to be a maximum of 8%, respectively. Also, primary population and children population are equal.

12.2.2. Running genetic operators

- Mutation is done as displacement of two genes considering interference possibility.
- Crossover as PMX is considered in total population, in which permutation is taken into account.
- Selection of a new generation is implemented through 50% direct transfer method from the better part of the child population to the primary initial population selection in the form tournament on the remaining 50% of the population.
- The Considered local search is hill climbing.

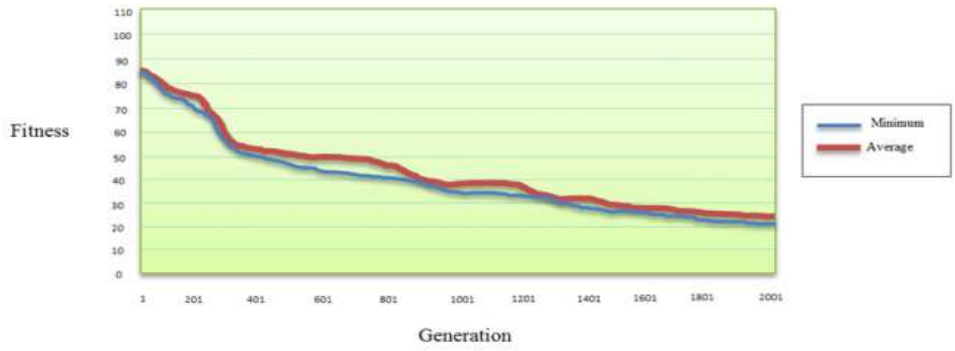

Figure 8. Fitness curve of running advanced genetic algorithm

After running this algorithm as shown in figure (8), the results seem better than those of the genetic algorithm. Fitness rate approached 21 after running 2000 generations and, as it is observed, the curve is descending. Furthermore, due to direct duplication of half of better population to parent population, the probability of optimal chromosomes loss is eliminated.

12.3. The Results of Running the Proposed New Algorithm

Figure 9 shows the curve of results obtained from running the proposed Algorithm for the presented courses. The results of this test is better than using genetic and previous Memetic algorithm, and as we mentioned above, in addition to hard constraints, some soft constraints have also been considered in creating primary population, which has caused the initial fitness rate of chromosomes to decrease. As you see, in testing with Memetic algorithms, fitness is limited to above 20 after about 2000 generation iterations; however, in running the proposed algorithm, fitness reaches below 20 after about 1010 generation iteration, and finally it reaches 7. Since, in the proposed method, one third of the population is replicated directly in the next generation, through the proposed method, the fitness rate of the new population will not be less than fitness rate of the population in previous generations. Rather, there is a fluctuation in the mean fitness of the population.

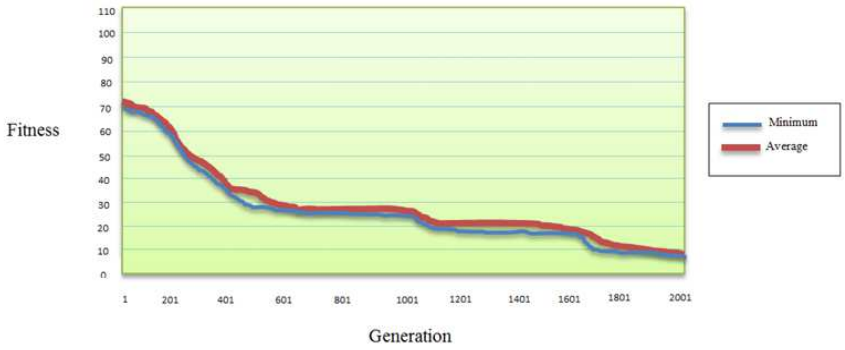

Figure 9. Fitness curve of running proposed Memetic algorithm

13. CONCLUTION

The present paper examined Memetic algorithm properties as an appropriate tool to optimize TTP responses. Then the problem of designing TTP for university courses was put forth as an applied problem, and a Memetic algorithm solution was represented to it. We prevented the algorithm from converging through creating changes in mutation, and desired results were obtained through integration; chromosome structure, obtained through integration; chromosome structure; transferring a better one- third of population to the next generation; semi-random initialization and other changes applied to the presented solution in the Local search and genetic algorithm. It was also shown that avoiding complete random initialization as well as local minimums through manipulating mutation parameter increased the genetic algorithm efficiency.

In particular, non-global searches such as genetic algorithm, precocious convergence leads to population uniformity and being trapped in a local optimum. The results show that the proposed Memetic algorithm is effective in designing timetables for a college, and developed tables can reach better fitness than responses created through genetic algorithms and Memetic algorithms. The result of this study focuses on the efficiency of localized methods based on normal and standard evolutionary algorithm.

References

[1] Cooper T., Kingston J., "The Complexity of Timetable Construction Problems, Lecture Notes in Computer Science", Vol. 1153, pp. 281-295, 1996.
[2] Carter M., "A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo", Lecture Notes in Computer Science, Vol. 2079, pp. 64-82, 2001.
[3] Burke E., Elliman D., Wearer R.," A Genetic Algorithm based University Timetabling System, Proceedings of the 2nd East-West International Conference on Computer Technologies in Education, pp. 35-40", 1994.
[4] KhaledMahar, "AUTOMATIC GENERATION OF UNIVERSITY TIMETABLES: AN EVOLUTIONARY APPROACH", pp. 2-5, 2006.
[5] AldyGunawan, K. M. Ng, H. L. Ong, "A Genetic Algorithm for the Teacher Assignment Problem for a University in Indonesia",Volume 19, Number 1, pp. 1-16, 2008.
[6] SadafNaseemJat, Shengxiang Yang, "A Guided Search Genetic Algorithm for the university Course Timetabling Problem", MISTA 2009.
[7] Z. Bratkovic, T. Herman, V. Omrcen, M. Cupic, D. Jakobovic, "University Course timetabling with Genetic Algorithm", a Laboratory Exercises Case Study, 2008.
[8] Rossi-Doria O., Paechter B," A Memetic Algorithm for University Course Timetabling", Proceedings of the CO2004 Conference, Lancaster, UK, 2004,p. 65.
[9] Erben W., Keppler J., "A Genetic Algorithm Solving a Weekly Course-timetabling Problem", Proceedings of The First International Conference on The Practice and Theory of Automated Timetabling, Edinburgh, UK, 1995, pp. 198-211.
[10] Abramson D., Abela J., "A Parallel Genetic Algorithm for Solving the School Timetabling Problem", Proceedings of the 15th Australian Computer Science Conference, Hobart, Australia, 1992, pp.101.
[11] Vorac J., Vondrak I., Vlcek K., "School Timetabling Using Genetic Algorithm", Technical Report, VSB-Technical university of Ostrava, Czech Republic, 2002.
[12] Beligiannisa G., Moschopoulosa C., Kaperonisa G., Likothanassisa D., "Applying Evolutionary Computation to the School Timetabling Problem: The Greek Case", Journal of Computers & Operations Research, Vol. 35, 2008, pp. 1265–1280.
[13] EnzheYa And Ki SeokSung ,"A Genetic Algorithm For a University Weekly courses Time Tabling Problem" , Intl. Trans in Op .Res. 9, 2002.
[14] Whitely D., "A Genetic Algorithm Tutorial", Journal of Statistics and Computing Vol. 4, 1994, pp. 65-85.
[15] Spyros Kazarlis, Vassilios Petridis and PavlinaFragkou, "Solving University Timetabling Problems Using Advanced Genetic Algorithms" pp.2-3.

[16] NysretMusliu, Werner Schafhauser, Magdalena Widl, "A Memetic Algorithm for a Break Scheduling Problem", MIC The VIII Metaheuristics International Conference, 2009.
[17] Olivia Rossi, Doria and Ben Paechter, "A Memetic algorithm for University Course Timetabling",Evolutionary Computation (GSICE),2002.