IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 3, No. 1, March 2014, pp. 7~15
ISSN: 2252-8938

A Fast Genetic Algorithm for Solving University Scheduling Problem

Mortaza Abbaszadeh*, Saeed Saeedvand**
*Department of Computer Engineering, Ilkhchi Branch, Islamic Azad University, Ilkhchi, Iran
**Young Researchers Club, Ilkhchi Branch, Islamic Azad University, Ilkhchi, Iran

Article Info

Article history:

Received Oct 22, 2013
Revised Nov 01, 2013
Accepted Nov 03, 2013

Keyword:

Genetic Algorithm
Chromosome
Population
Fitness
Mutation

ABSTRACT

University course timetabling is a NP-hard problem which is very difficult to solve by conventional methods, we know scheduling problem is one of the Nondeterministic Polynomial (NP) problems. This means, solving NP problems through normal algorithm is a time-consuming process (it takes days or months with available equipment) which makes it impossible to be solved through a normal algorithm like this. In purposed algorithm the problem of university class scheduling is solved through a new chromosome structure and modifying the normal genetic methods which really improves the solution in this case. We include lecturer, class and course information in presented algorithm, with all their Constraints, and it creates optimized scheduling table for weekly program of university after creating primary population of chromosomes and running genetic operators. In the final part of this paper we conclude from the results of input data analysis that the results have high efficiency compared with other algorithms considering maximum Constraints.

Corresponding Author:

Mortaza Abbaszadeh,
Department of Computer Engineering, Ilkhchi Branch,
Islamic Azad University, Ilkhchi, Iran

1. INTRODUCTION

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

Also, we know that manual scheduling requires a lot of time and professionals who should spend several days to prepare an appropriate schedule whereas error probability is inevitable in manual operation, and this makes the task difficult. As we mentioned above, using Genetic Algorithm (GA) technique to solve scheduling problem is one of the under consideration ideas which have been used in numerous cases having ideas which have been used in numerous cases having notable results [5, 6, 7], moreover, there are other methods for solving TTP of which Graph Coloring Algorithms, Colony of ants and Memetic Algorithms are the most important ones [4, 9].

In the present paper we attempt to solve university courses scheduling problem through Genetic Algorithms with high efficiency and precision considering maximum constraint in scheduling, for which we determine time table of starting and finishing hours for any chromosome, then divide it into small time intervals, and schedule lecturer and instructor interval considering both form of soft and hard Constraint. It

should be mentioned that we can schedule examinations with acceptable efficiency and speed utilizing this presented method.

Numerous studies have been carried out in this field Sehraneh Ghaemi have proposed Genetic Algorithm that her main goal is to minimize the number of conflicts in the timetable and for it her purpose two approaches - modified GA and cooperative GA [2]. Edmund Burke, David Elliman, and Rupert Weare have proposed Genetic Algorithm at Department of computer Science, University of Nottingham, which only applies to acceptable time intervals and it is related to user interactively [4]. Khaled Mahar presented a genetic algorithm for solving courses scheduling problem in 2006, and realized it at a university and showed its results [5].

Wilhelm Erben & Jurgenkeppler also designed and realized optimized tables of weekly courses utilizing Genetic Algorithm [9]. Michael W. Carter divides problem into several sections in his proposed method and uses a greedy algorithm and an algorithm based on Lagrange function for allocating time intervals and allocating classrooms, respectively [3]. Considering computational complexity of Genetic Algorithms, actions have been taken to increase their speed, and in this direction, Abramson D. &Abela J have3 taken advantage of parallel Genetic Algorithms in their research to solve the problem of designing timetables for schools and they have reported relatively high time advantage compared with normal genetic algorithms [10], Varac J et al. have presented complete review on application of genetic algorithms in their research for designing time tables [11] and in our previous work in [15] we purpose a memetic algorithm to solving University Scheduling Problem that we improve final results here.

2. PROBLEM FORMULATION

Any response or timetable is considered as a chromosome in university course scheduling problem, which where, S, T, and R refer to course set, instructors set, and allowed time of classes, respectively; L refers to place or class, and F (T, S, R, L) means presenting the course S, by instructor T, on time R, and in place L.

Formula for calculating chromosome size:

$$F_s = N_s * (\mathrm{S} * 2) \quad (1)$$

Where, F_s refers to chromosome size, S means course number which its doubling creates instructor's code according to each course, and N_s refers to divided time credit number which can be obtained from following formula.

Calculating time periods number in each chromosome:

$$N_s = ((End_T - Start_T) * 2) * X \quad (2)$$

Where, N_s refers to time period number which is assumed 30 minutes and it is changeable through changing required coefficient rate $(End_T - Start_T)$; $Start_T$ shows starting hour and End_T shows ending hour of scheduling which is receivable from user with 30 minutes precision. This means, the users can specify end hour and start hour of classes. X represents those days of week which will be scheduled.

2.1. Population Size Calculation

In the presented method to specify population size (parents and children population), if population size is very large, algorithm performing speed will decrease intensely. On the other hand, of the population size is small; the problem will converge on 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, increasing the number of events increases population size, and increasing the length of chromosomes decreases population number manually by users are possible.

$$P_{Size} = \frac{L*N_C}{E-1} * K \ , \ Ch_{Size} = P_{Size} * 1.5 \quad (3)$$

In the first relation, Ch_{Size} is children population size which is 1.5 times as much parent's population, and in the second relation, P_{Size} is the proposed parent's population. L is available class number, E refers to events number in chromosome, and as mentioned above, N_C is desired time periods (equation 2); and K is constant factor for regulating population size which has been attained through error and effort and it is approximately 80.

3. CONSTRAINTS

As mentioned above, each Constraint in scheduling courses can be divided into two basic parts, i.e. hard Constraint- that considered totally in scheduling- and soft Constraints- that applied to problem as far as possible. As we know, in maximum of presented papers, only some of basic Constraints have been applied and not all of Constraints have been used for scheduling [2] [7] [6] [5] [14].

However, we attempt to use maximum Constraints in scheduling for receive the best response in this paper that we will discuss them later. In this paper we assume all of considered Constraints in [15], (our previous work) and we work and improve implementing our purposed algorithm on that constraints in this paper.

4. CHROMOSOMES STRUCTURE

As we mentioned in section 2, in this research any time table has been assumed as a chromosome which every chromosome has been create as a two-dimensional array.

Actually, generating of the chromosome is transformation of phenotype space into chromosomes. As it is seen from the figure, columns of this chromosome specify desired class by class code which any 2 columns are considered as one class and creation of chromosome is done by permutation.

Is should be said that determining presented course group is take out considering predetermined course and it is extractable using the course feature which has not been shown directly in chromosome. Rows of this chromosome are created at two steps which dividing into the days of week is done at the first stage and dividing the days into time distance is done at the second stage. Formula (3) shows the method of achieve time intervals, i.e. from start of scheduling every day until end of scheduling on same day which are divided into changeable periods.

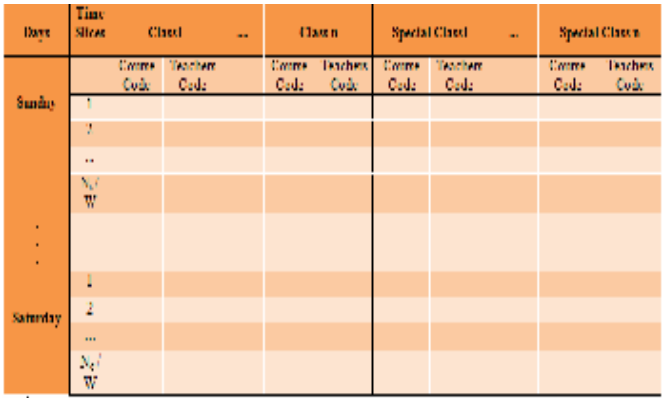


Figure 1. Displaying a chromosome

5. CROSSOVER OPERATOR

In the proposed algorithm, crossover is performed at two stages: at the first stage, we organize population considering fitness rate of any chromosome, formula (4), then by the rate of 15 percent of children population from chromosomes of parents population which have low fitness are transferred to the next generation without any change, i.e. better part, of parents population.

At the second stage, we use the next one-third of children population utilizing SB-PMX algorithm (Sector Base-Partially Mapped Crossover) which has been studied in detail by Enzehya, in [11], and run SB-PMX algorithm through selecting parents randomly from total parent's population, in which the permutation feature of genes is maintained in chromosomes. In SB-PMX, after crossover of two chromosomes with permutation, considering that time intervals of presented courses will be in contact, chromosome undergoes restoration, 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 mentioned that if the restoration of created chromosome is not possible, the created child will be deleted and crossover is performed again. At the same time, crossover of Constraint of normal and special classes is performed separately and in pairs, i.e. normal class part of first chromosome will be considered as a separate chromosome, and it will only crossover with special class part of second chromosome.

6. MUTATION

The function of Mutation operator is to prevent from homogeneity of population and entrapment of algorithms into local optimized position. In the proposed method, our idea about implementation of mutation is to do mutation at 3 different states: to determined mutation rate, so that in fact three mutation are considered as one mutation at 3 stages.

The first and second mutation is the one in which time of a introduced course and its instructor is replaced by the time of other course along with its instructor in acceptable limitation randomly (this action is execute only in special or normal class interval which leads to lack of invalid responses).

For this purpose, however, presented class time is checked with second course time so that 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. For interchanging of two genes, courses time should be equal or there should be sufficient free time before and after them, so there will be no interference and mutation on two genes becomes possible. It is noteworthy that if interchange was not possible, the other two genes would be selected randomly and operations would be repeated as it is observed in figure (2).

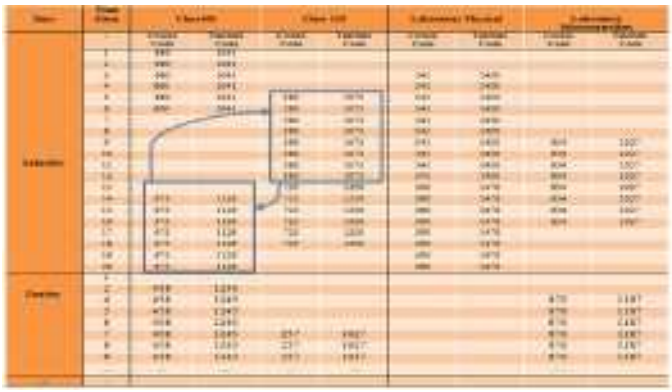


Figure 2. Display of type I mutation in a chromosome

In the second mutation, a number of instructors in any chromosome are interchanged by other instructors in the same chromosome, however, in this mutation, course group of instructor is considered with presented course and the skill of instructor for teaching new course, and instructor pair is selected from instructors list randomly, therefore course interchange of each instructor pair is possible. There are usually at least one instructor pair is possible. There is usually at least one instructor pair in each chromosome for interchanging their courses.

However, if this possibility does not exist, this stage of mutation will not occur. A sample of this mutation has been presented in figure (3).

Figure 3. Display of type II mutation in a chromosome

In the third mutation, a free space is selected randomly from successive time and then one of the presented courses along with instructor 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 4: Exhibits a sample of mutation.

Figure 4. Display of type III mutation 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 which will be discussed in Implementation part.

7. FITNESS FUNCTION

Fitness function indicates acceptability rate of chromosome. In this case, a fine has been considered for not following above mentioned Constraints which is defined as follows:

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

Where, i refers to proposed chromosome, Hi is total weight of violated hard Constraints related to desired chromosome, K_{Hard} is a constant for increasing the pressure on hard Constraints weight, Si refers to 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 $\boldsymbol{K_{Hard}} = 5 * \boldsymbol{K_{Soft}}$,$\boldsymbol{K_{Soft}}$ is value one for violation of any constraint.

8. SELECTION FUNCTION

In this section, we select once more the primary population from created children population chromosomes after running genetic operators on chromosomes with respect to fitness of each chromosome. Considering various tests which have been performed through different methods, the best result has been obtained from integrating two three-axis and tournament roulette wheel Method. So that selecting 60% population is done through creating three-axis Roulette wheel on total population. In figure (5), a sample for three chromosomes is presented which selection probability of each is specified in relation to fitness.

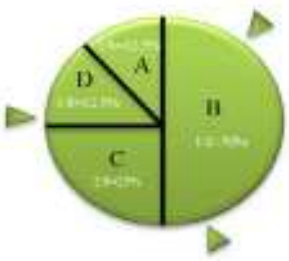


Figure 5. Display of three-axis Roulette wheel

In this method, fitness rate of each chromosome determines selection chance of each chromosome and remained 40 percent are selected through selecting tournament where Chromosomes compete with each other in pairs. It should be mentioned that selection is done from 1.5- folds of primary population, and children number is more than parent's number according to formula (3).

9. INITALIZATION

We use possible method randomly and independent from fitness but considering performance of some of hard Constraint having permutation quality, for initialize primary population chromosomes, and we take into account some of soft Constraints during creation of primary population for obtaining good results. Chromosomes generally suffer from low fitness in 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, creation of primary population will be very difficult and inapplicable, and if we consider none of them in creating population, speed and chance of finding optimized response will decrease, therefore, we consider none of them in creating population, speed and chance of finding optimized response will decrease, therefore, we consider initialization semi random rather completely random.

Considering what was mentioned, initially some conditions are checked during creation of primary population which there is no need to solve them as Constraints with respect to permutation in genetic operators. They are as follows:

- Each special or normal class is placed in its own part.
- No course has time interference with other course in a class.
- Those courses are considered that should be scheduled totally or separately.
- 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.

Items that influence finding of response in creating primary population none randomly are as follows.

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

10. IMPLEMENTATION

In this section, we deal with method, order and stages of realizing presented algorithm. As we consider in the initialization section, after determining number of parents and children population we initialize parents as primary population semi randomly, then we compute fitness rate of each chromosome (according to formula (4)), and finally we run goal test function on population. It is considerable that goal test function is a function that checks fitness rate of chromosomes individually to determine whether the related chromosome response is optimal or not. Here, criterions of goal test function are to test existence zero of hard Constraint fitness rate, existence zero of fitness rate of some soft Constraints and existence minimized of the other Constraints so that if goal chromosome was found in this function, at first sample of that chromosome is stored as a answer, and production components of some other Constraints so that if goal chromosome is found as an answer, and generation components of some other generation are given to algorithm before stopping the run in order that there will be more probability to create optimal answer by running genetic operators on optimal chromosome.

At the next step, we run crossover operator on primary population, and as we discussed in the step of crossover operation, children population is created, and then we run fitness run calculation and target test functions on population. Mutation operator is run on children population. We should know if mutation rate is constant, there will be the risk of being trapped in local maximum; therefore, mutation rate increases or decrease and it is run on population with high rate in order to avoid this problem.

At the next step, after running fitness computational function on chromosomes, selection function is run on children chromosomes, and it replaces with primary population through above-mentioned method. As it was represented in formulation section, children population size was obtained using the formula $Ch_{Size} = P_{Size} * 1.5$, therefore creating more children and increasing the possibility of creating good response and selecting new primary population from children become possible. Here, in this purposed algorithm we will observe results of Implementation section that using this method will lead to satisfactory results. Above-mentioned steps execute until the specified rate for generation number ends or optimal response would be found.

11. RESULT OF IMPLEMENTATION

As we mentioned above, in proposed genetic algorithm, initialization is semi randomly in special chromosome structure, and through various tests, different mutation and integration compared with normal genetic algorithms have been considered. 15 percent of population is replicated in the next generation directly in order to cause good chromosomes to be not lost.

For the purpose of comparing a normal genetic algorithm with advanced genetic algorithm which is more complete than normal genetic algorithm and proposed genetic algorithm, tests have been performed the results of which are explained and studied afterwards.

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, Black lines show minimum fitness resulted from evaluation function, and red lines specifies mean fitness of population. In all of tests, facilities are equal; faculty has 10 classes, 3 laboratories and 2 computer site. Also, all specifications and information about courses, classes and instructors are similar and they have been extracted from courses set of Engineering faculty of Tabriz University. Moreover, integration and mutation with independent probabilities are performed on all chromosomes. It is noteworthy that the proposed algorithm has been testes on courses of Islamic University, Ilkhchi Branch, which has represented acceptable scheduling proposed algorithm.

- **Running Normal Genetic Algorithm**

We have considered initialization completely random in running normal genetic algorithm. We assume integration rate as 65% and mutation rate as 6%, and number of primary population and number of children are equal. At this stage, genetic operators are applied as follows:

- Mutation is done as interchanging of tow genes irrespective of interference probability.
- Crossover as 3-point crossover has been considered on 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.

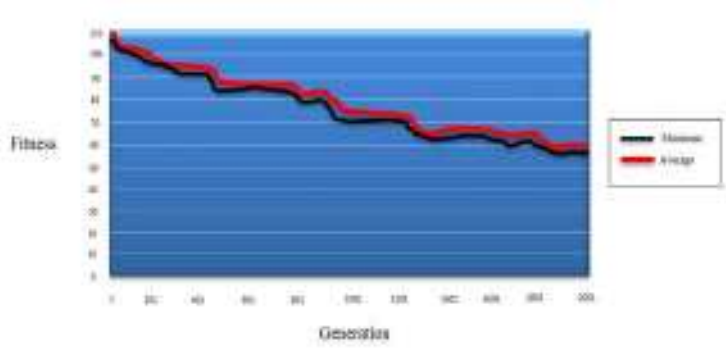


Figure 6. Fitness curve of running normal genetic algorithm

As it is shown is figure (6), primary tables have very low fitness and even after 2000 generation, there are 55 tables of this type which is very low compared with fitness rates obtained from the proposed algorithm; in contrast to proposed method, 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 change in normal genetic algorithm increases intensively the quality of produced timetable.

- **Running Advanced Genetic Algorithm Respecting Some Constraints**

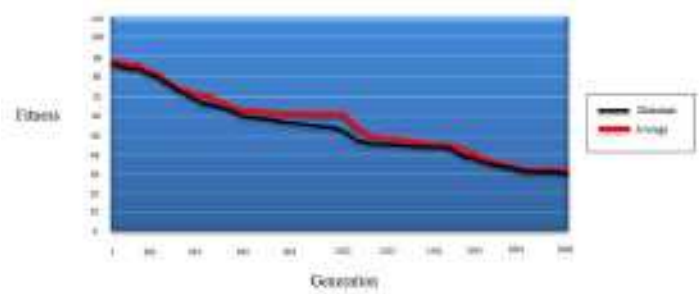
Although rates and information on problem and previous runs are the same, classes are scheduled in two separate parts, i.e. normal class and special class. As it is seen in figure 8, the initial rate of chromosomes fitness is different in comparison with normal genetic algorithms.

- **Running conditions are as follows:**

We consider initialization as semi random in running advanced genetic algorithm. We assume integration, mutation rate for initialization, and variable in relation to homogeneity of fitness rates as 66%, 4% and 8%, respectively. Also, primary population and children population are equal.

- **At this stage, genetic operators are run as follows:**
  - Mutation is done as displacement of two genes considering interference possibility.
  - Crossover as PMX is considered on total population in which permutation is taken into account.

- Selection of new generation is realized through 45% direct transfer method from better part of children population to primary population and selection as tournament on remained 65% of population.



Figuer 7. Fitness curve of running advanced genetic algorithm

After running this algorithm as figure (7), results of running seem better than normal genetic algorithm. Fitness rate approached 29 after running 2000 generations and as it is observed the curve is descending, and due to direct duplication of half of better population to parent population, the probability of optimal chromosomes loss is eliminated.

12. THE RESULTS OF RUNNING PROPOSED NEW ALGORITHM CURVE OF RESULTS OBTAINED FROM RUNNING PROPOSED

Algorithm for presented courses has been shown in figure (8). The results of this test is better that using standard genetic 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 rat of chromosomes to decrease. As you see, in testing with more advanced genetic algorithms, fitness is limited to above 29 after about 2000 generation iterations, however, in running proposed algorithm, fitness reaches below 10 after about 1401 generation iteration; and finally it reaches 8. Since on third of population is replication directly on the next generation through proposed method, fitness rate of new population will not be less than fitness rate of population in previous generations. Rather, there is fluctuation in mean fitness of population.

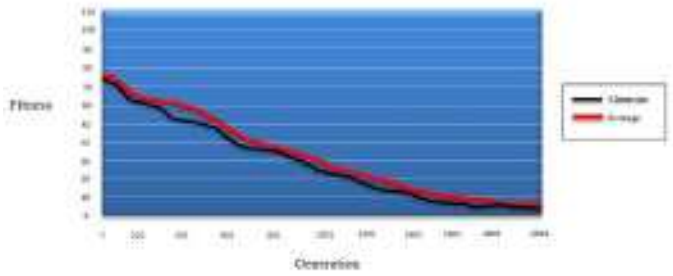


Figure 8. Fitness curve of running proposed genetic algorithm

13. CONCLUSION

The present paper examined genetic algorithm properties as an appropriate tool for optimizing TTP responses that even show better result than memetic algorithms. Then the problem of designing TTP for university courses was introduced as an applied problem, and a standard genetic 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 better one 15 percent population to the next generation, semi random initialization and other changes applied to normal genetic algorithm and some changes from last purposed memetic algorithm. Also, it

become clear that avoiding completely random initialization and attempting to avoided local minimums through manipulating mutation parameter and deleting repeated chromosomes and displacing them by remaining better chromosomes in children population increased 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 result shows that proposed genetic algorithm is effective in designing timetables for a college, and produced tables can reach better fitness than responses created through normal genetic algorithms. Research results concentrate on efficiency of localized methods based on normal and standard evolutionary algorithm. It should out on timetables of college examinations the favorable results of which verify the results and reasoning's presented in the present paper.

REFERENCES

[1] Cooper T, Kingston J. "The Complexity of Timetable Construction Problems". Lecture Notes in Computer Science. 1996; 1153: 281-295.
[2] Sehraneh Ghaemi. Using a genetic algorithm optimizer tool to solve University timetable scheduling problem. *International Conference on IEEE Xplore*. 2007.
[3] Carter M. "A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo". *Lecture Notes in Computer Science*. 2001; 2079: 64-82.
[4] 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. 1994: 35-40.
[5] Khaled Mahar. "AUTOMATIC GENERATION OF UNIVERSITY TIMETABLES: AN EVOLUTIONARY APPROACH". 2006: 2-5.
[6] Aldy Gunawan, KM Ng, HL Ong. "*A Genetic Algorithm for the Teacher Assignment Problem for a University in Indonesia*". 2008; 19(1): 1-16.
[7] Sadaf Naseem Jat, Shengxiang Yang. "A Guided Search Genetic Algorithm for the university Course Timetabling Problem". *MISTA*. 2009.
[8] Z Bratkovic, T Herman, V Omrcen, M Cupic, D Jakobovic. "*University Course timetabling with Genetic Algorithm*". a Laboratory Exercises Case Study. 2008.
[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: 198-211.
[10] Abramson D, Abela J. *A Parallel Genetic Algorithm for Solving the School Timetabling Problem*. Proceedings of the 15th AustralianComputer Science Conference, Hobart, Australia. 1992: 101.
[11] Vorac J, Vondrak I, Vlcek K. *School Timetabling Using Genetic Algorithm*. Technical Report, VSB-Technical university of Ostrava, Czech Republic. 2002.
[12] EnzheYa And Ki SeokSung. A Genetic Algorithm For a University Weekly courses Time Tabling Problem. *Intl. Trans in Op .Res. 9*. 2002.
[13] Whitely D. A Genetic Algorithm Tutorial. *Journal of Statistics and Computing*. 1994; 4: 65-85.
[14] Spyros Kazarlis, Vassilios Petridis and Pavlina Fragkou. *Solving University Timetabling Problems Using Advanced Genetic Algorithms*. 2-3
[15] Mortaza Abbaszadeh, Saeed Saeedvand, Hamid Asbagi Mayani. Solving University Scheduling Problem With a Memetic Algorithm. *IAES International Journal of Artificial Intelligence*. 2012; 1(2).