ARTIFICIAL INTELLIGENCE STUDIES
2019, 2 (2): 20-31

www.aistudies.org

Genetik Algoritma Temelli Çizelgeleme Programı

Muhammed Mutlu YAPICI[*,a], Omer Faruk BAY[b]

[a,*] *Ankara Üniversitesi, Bilgisayar Teknolojileri Bölümü, ANKARA, TÜRKİYE*

[b] *Gazi Üniversitesi, Teknoloji Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, ANKARA, TÜRKİYE*

MAKALE BİLGİSİ

Alınma: 19.07.2019
Kabul: 21.11.2019

***Anahtar Kelimeler**:*
Genetik algoritma,
ders çizelgeleme,
zaman çizelgeleme

[*]*Sorumlu Yazar:*
e-posta:
mutluyapici@ankara.edu.tr

ÖZET

Ders çizelgesi oluşturma problemi, bir çok farklı kısıtlamaları dikkate alarak dersliklere bir dizi ders ve eğitmen atamakla ilgilidir. Genellikle, bu problem el yordamıyla çözülmeye çalışılır ve içerdiği çok çeşitli kısıtlamalar ile karmaşık insan faktörleri nedeniyle, çok fazla zaman ve insan gücü gerektirir. Bu problem günümüzde üniversitelerin ve kolejlerin karşılaştığı en çok zaman alan problemlerden biri olarak kabul edilmektedir. Bu çalışmada gerçek bir dünya problemi olan ders çizelgeleme problemine çözüm getirecek genetik algoritma tabanlı bir ders çizelgeleme yazılımı geliştirme amaçlanmaktadır. Bu yazılım, kısıtlamaların kolayca girilmesine ve en uygun çözümlerin bulunmasına izin vermektedir. Optimizasyona en uygun çözümü bulmak için tam genetik algoritma ve kısmi genetik algoritma olmak üzere iki farklı çözüm yöntemiyle test edilmiştir. Test sonuçları, rastgele oluşturulmuş ilk popülasyondan tam genetik algoritmayla başlatıldığında uygun çözümü elde etmenin oldukça zaman aldığını göstermiştir. Kısmi genetik algoritma ile, tam genetik algoritmadan çok daha hızlı bir şekilde optimal çözüme ulaşılmıştır.

DOI: 10.30855/AIS.2019.02.02.01

Genetic Algorithm Based Timetabling Program

ARTICLE INFO

Received: 19.07.2019
Accepted: 21.11.2019

Keywords:
Genetic algorithm,
course scheduling,
time tabling

[*]***Corresponding Authors***
e-mail:
mutluyapici@ankara.edu.tr

ABSTRACT

Course Timetabling Problem is concerned with assigning a number of courses and instructors to classrooms by taking the constraints into consideration. Generally, this problem is typically resolved manually; and due to the large variety of constraints, resource limitations and complicated human factors involved, it takes a lot of time and manpower. It is considered as one of the most time-consuming problems faced by universities and colleges today. In this study, we aimed to develop a genetic algorithm-based timetabling software to bring a solution to course timetabling problem, which is a real world problem. This software allows constraints to be entered easily and allows that optimal solutions are found. To find the most suitable solution for optimization, two different solution methods, a full-genetic algorithm and a partial-genetic algorithm, were tested. Test results showed that when we start the full genetic algorithms from randomly generated initial population, it takes quite some time to obtain the appropriate solution. With the partial-genetic algorithm, an optimal solution was achieved much more quickly than the full genetic algorithm.

DOI: 10.30855/AIS.2019.02.02.01

1. INTRODUCTION *(GİRİŞ)*

Preparing a course schedule that students and teaching staff will both be pleased with is a problem at every university. This problem is typically resolved with more than one people working for many days at universities. In this study, we aimed to develop a genetic algorithm-based timetabling software to bring a solution to the timetabling problems primarily at universities and at other educational institutions like schools and private courses.

Course scheduling problem is a Problem of Resource Constrained Project Scheduling (PRCPS). When previous studies that aimed to solve the PRCPS problem are analyzed, it is observed that a lot of different artificial intelligence techniques were used. In recent years, it is observed that many researchers have been using methods of artificial intelligence such as Simulated Annealing solution algorithms (SA) [1, 2], Tabu Search (TS) [3], and Genetic Algorithms (GA) [4, 5] for the solution.

Genetic Algorithms (GA) are numerical optimization methods that are used to solve problems which are difficult or almost impossible to solve with the traditional methods [6]. Using GA for the solutions of problems that are difficult will eliminate the need for a comprehensive and complex calculation procedure [7].

In this study, a software based on genetic algorithm was developed for the solution of timetabling problem. Two different solution methods were analyzed and compared to improve the performance of the solution of the problem. The first method is Half (Partial) Genetic Algorithm, and the second one is Full (Complete) Genetic Algorithm method.

This study consists of five sections. In the second section, we focused on the working procedure of the genetic algorithm and discussed the course scheduling problem. In the third chapter, a simulation study was carried out. Half Genetic Algorithm performance and Full (Complete) Genetic Algorithm performances were examined separately. The performances of the two genetic algorithm methods were compared in accordance with the data obtained. In the fourth section, the program interfaces were introduced, and the program output tables of the results were given. In the fifth section, which is the last section, the analyzed the results of the program.

2. GENETIC ALGORITHM AND COURSE SCHEDULING PROBLEM*(GENETİK ALGORİTMA VE DERS ÇİZELGELEME PROBLEMİ)*

Genetic algorithm is a numerical optimization method based on the principle of natural selection. The algorithm is characterized by a number of parameters such as chromosome, gene, population, fitness rate, selection, crossover and mutation. Genetic algorithm is carried out by repeating operations until we get the optimal solution. Flow chart of the genetic algorithm is shown in Figure 1.

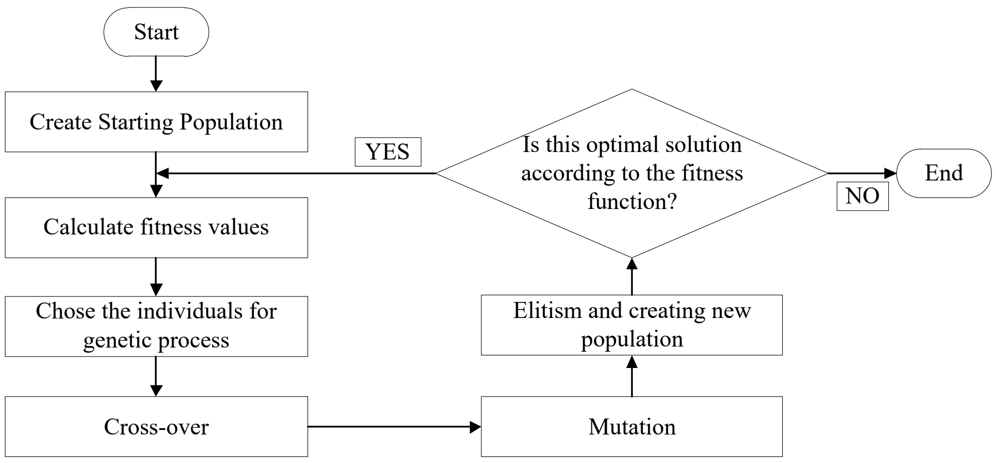
Start
Create Starting Population
Calculate fitness values
Chose the individuals for genetic process
Cross-over
Mutation
Elitism and creating new population
Is this optimal solution according to the fitness function?
YES
NO
End

Figure 1. Flow chart of the genetic algorithm *(Genetik algoritma akış şeması)*

2.1. Chromosome *(Kromozom)*

Genetic Algorithm is started by defining the parameter array and the chromosomes (individuals) which will be optimized.The parameters that will take place on a chromosome should be designed in such a way that will include all of the information that is required for the best solution [8]. Thus, it will be possible to produce effective results with genetic operations [7]. To encode the chromosomes, the binary and decimal number systems are used [9]. Especially the decimal coding is preferred for systems that have so many number of decision variables that are too long for the binary system [7]. Because of this, in this study, the decimal coding system was preferred.

In this study, two different chromosome structures have been tested. The first chromosome model that was used was designed according to the number of the rooms and the number of the days and the number of the hours. According to this model, each chromosome is composed of a gene which was composed of a multiplied number of classrooms with the number of the days and with the number of the hours in a day. Thus, each time of each classroom is represented by a gene. Operating mode of this chromosome model is based on the logic of assigning activity ID info into each gene (array element). Then, the mapping of the classrooms, days, hours, branches, courses and lecturers takes place. The basic model of chromosome structure is shown in Figure 2. The following chromosome structure is designed for N classrooms, 5 days from Monday to Friday, and 11 hours in a day.

Chromosome Length= Number of classrooms x 5 x 11

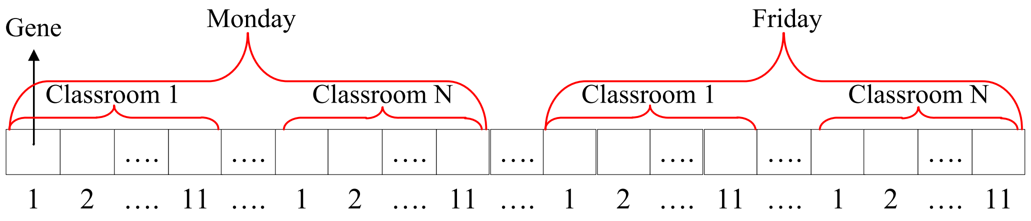

Figure 2. First chromosome structure *(İlk kromozom yapısı)*

In the second chromosome model, every gene in the chromosome represents an activity. The value of these genes represents the info as day, hour and classroom ID. The model of this chromosome structure is shown in Figure 3.

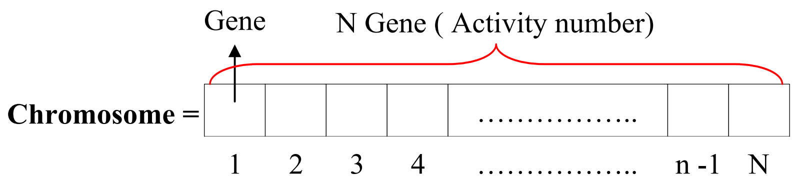

Figure 3. Second chromosome structure *(İkinci kromozom yapısı)*

The comparison of the two chromosome structures is shown in Table 1. When Table 1 is analyzed, it is seen that the first chromosome gene structure is longer than the second chromosome gene structure, and it takes more processing time for each iteration. The second chromosome model, which works more efficiently, was preferred for genetic operations.

Table 1. Usability Comparison of the two chromosome structure *(Kromozomların kullanılabilirlik karşılaştırması)*

	First Chromosome Model	Second Chromosome Model
Chromosome Length	36 x 5 x11 = 1980	257
Estimated number of Operation for an Iteration	8 x 1980 = 15 840	8 x 257 = 2056
Advantages of the model	There is no any possibility of two different activity representations in a classroom in the same time. Thus, there is no any conflict of classroom.	There is no any possibility of activity representation in two different classroom in the same time. Thus, there is no any conflict of activity.

2.2. Gene *(Gen)*

Gene is the smallest meaningful information in chromosome. Each parameter constitutes a gene in chromosome. The gene string stores specific information of the problem. Each element is called gene, and is usually named as a string of variables [10]. In this study, two different gene structures are modelled on two different chromosomes models. In the first chromosome model, every gene holds an activity ID. The sampling of the situation is shown in Figure 4.

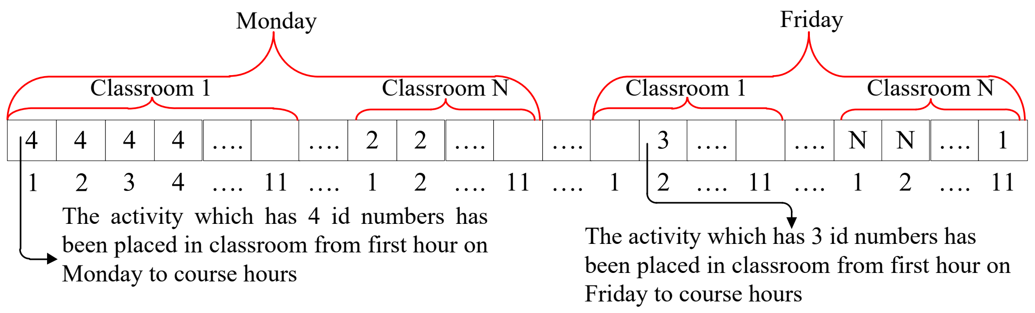
Monday
Friday
Classroom 1
Classroom N
Classroom 1
Classroom N
4 4 4 4 …. …. 2 2 …. …. 3 …. …. N N …. 1
1 2 3 4 …. 11 …. 1 2 …. 11 …. 1 2 …. 11 …. 1 2 …. 11
The activity which has 4 id numbers has been placed in classroom from first hour on Monday to course hours
The activity which has 3 id numbers has been placed in classroom from first hour on Friday to course hours

Figure 4. Gene structure that belongs the first chromosome model *(İlk kromozom modeline ait gen yapısı)*

In the second model, each gene represents an activity and holds a number in which classroom ID and the day and time information is encrypted. The sampling of the situation is shown in Figure 5.

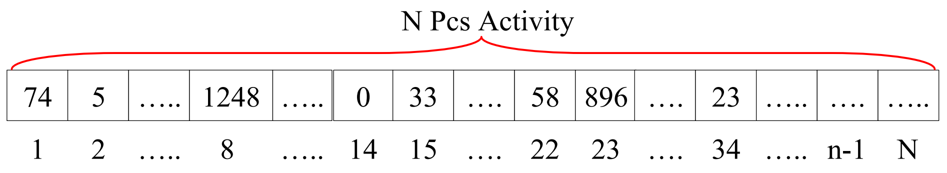

Figure 5. Gene structure that belongs the second chromosome model *(İkinci kromozom modeline ait gen yapısı)*

1248 represents classrooms, the day and the time to which 8th Activity has been assigned. Information number of 1248 was arranged with a certain formula. The formula that determines the value of the gene is shown in Eq 1.

$$\text{Position} = D \times Nc \times TH + Cid \times TH + H \qquad (1)$$

Position is value in the gene, D represents number of the day that is defined for activity. Nc is total classroom number. TH represents total hour number in a day. Cid is ID number of the classroom that is defined for activity. H represents lesson beginning hour that is defined for activity.

Since the second chromosome model was preferred for the solution of the problem, the genetic algorithm operations were conducted in accordance with the second gene model.

2.3. Population and Crossover *(Popülasyon ve Çaprazlama)*

All chromosomes (individuals) create a population by coming together. So the sum of the chromosomes (individuals) is called as a population. The population size, in the other words, the number of the chromosomes (individuals) is one of the important factors affecting GA's success or access to the duration of optimum solution [10]. Selection of a small population can increase the operation speed. However, this will reduce the probability of finding better solutions. So, better solutions tend to be overlooked. Although big populations are better to illustrate the entire solution space, it can cause that the program slows down and the solution may be found late. Therefore, the selection of a suitable population size is important for the program.

To investigate the potential of the existing gene pool, crossover operator is used to create new chromosomes that have better features than the chromosomes of the previous generation. Crossover is usually applied to chromosomes which are chosen with a probability equal to crossover rate [11, 12]. Depending on the type of the problem, there are different crossover models as Single-Point Crossover and Multi-Point Crossover.

In the developed program, a method that can be changed interactively has been preferred. The crossover rate and the number of crossover points can be changed on the program interface. Two-point crossover model which is used in the program is shown in Figure 6. According to the model, randomly selected genes are being changed mutually between two chromosomes. As shown in Figure 6, after the crossover process, two new chromosomes are obtained.

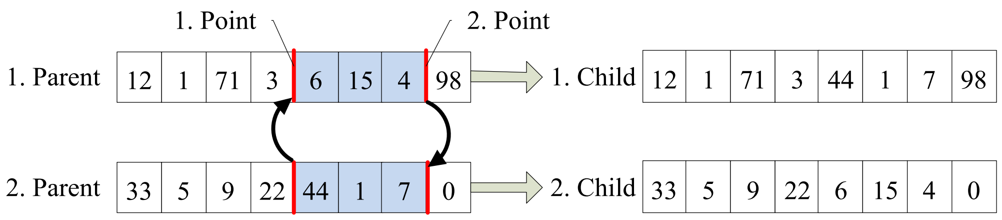

Figure 6. 2 point crossover model *(İki noktalı çaprazlama modeli)*

2.4. Mutation *(Mutasyon)*

The overall aim of the mutation is providing or maintaining genetic diversity. Mutation operator's duty is providing a protection against the loss of a good solution [13]. In the developed program, mutation rate and the mutation point number can be changed on the program interface. In the design phase of the mutation, new gene values are formed by using the formula given in Eq. 1.

The four-point mutation model which is used in the program is shown in Figure 7. According to the mutation function, the new position of the randomly selected genes is recalculated and changed. In this study, we changed the mutation function process at half GA. We changed the gene value with the new value which is recalculated according to the conditions of the fitness function. Thus, we aimed to eliminate the useless values. In this way, we also aimed to achieve faster genetic solution.

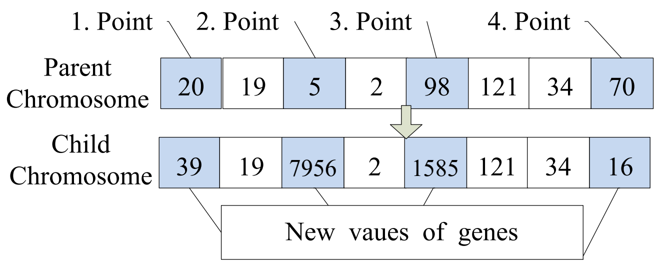

Figure 7. 4 point mutation model *(Dört noktalı mutasyon modeli)*

2.5. Fitness Function *(Uygunluk Fonksiyornu)*

Fitness function covers all the conditions determined for the solution of the problem. In this study, we conducted an assessment based on punishment by calculating the error value for each chromosome. The error value of each chromosome is calculated according to Eq. 2.

$$f(k) = \sum_{i=0}^{g} \sum_{j=0}^{t} P_{(ij)} \qquad (2)$$

$f(k)$ is the fitness value for kth chromosome, g is the total number of the genes in the chromosome, t is the number of the total constraints, and $P_{(ij)}$ is the error value which belongs to jth constraints of ith gene.

There are two types of conditions as the main conditions and the special conditions for course scheduling problem. The main conditions refer to the criteria that must be definitely met to solve the problem. The special conditions increase the quality of the problem solution. Fitness function contains all of the main and special conditions according to the precedence.

2.6. Half - Genetic Solution Function *(Kısmi Genetik Çözüm Fonksiyornu)*

In this study, two different solution methods as Half-Genetic and Full (Complete) Genetic Solutions have been developed to solve the course scheduling problem. The stage of the genetic algorithm is applied as described in the previous section. In Half-Genetic solution, after creating the initial population, Half-Genetic solution function is implemented to population for a while. And then genetic operations (crossover and mutation) are activated. In the Half-Genetic solution, fitness values of population individuals are improved up to a certain value. Thus, it is aimed to accelerate the genetic algorithm to reach a solution.

Way of functioning of Half-Genetic solution is similar to the mutation function. Actually, in the half GA, we used the improved mutation function. This is the fundamental difference between the half GA and Full GA. Normally, in the mutation functions, we choose a random gene points on the selected chromosome, and then change the gene value with randomly re-generated value. But in this study, we changed the gene value with a new value which was produced according to the conditions of the fitness function. Thus, we aimed to eliminate the useless values. In this way, we aimed to achieve faster genetic solution. Chromosome with the best fitness value in the population is selected and sent to the half-genetic solution function. The main purpose of this function is to reach better values of the individual which has the best fitness value in the population. In the function, firstly the defective genes (the ones with bad values) of the chromosome are selected and then selected genes' IDs are held in an array. Subsequently, a gene selected from the defective genes and a new position value is produced for this gene. While the new position value is produced, new values for Classroom, Day and Time information are produced according to the conditions instead of being produced randomly. It is aimed to perform a controlled mutation operation. In this way, chromosomes with the best fitness value in the population are recuperated for better values. As mentioned above, after the fitness value reaches 15000 or an even better fitness value, genetic algorithm is applied for the whole population. After the controlled mutation process has been applied to chromosome which was sent to the half-genetic function, fitness value of the chromosome is calculated and added to the population.

The Half-Genetic solution function is applied only to chromosomes with the best fitness value instead of the whole population. Thus, the aim is to protect the sampling feature on different points in the solution space.

3. EXPERIMENTS and RESULTS

In this part, the effects of different population sizes, the effects of different mutation probabilities and the effects of the numbers of the mutation points are investigated. In addition, performance comparisons were made by comparing the half GA results and full GA results.

The simulation results were obtained according to the data from the Computer Engineering Department of a Virtual University in spring semester in 2019-2020 academic year. In total, a course scheduling for 257 courses with 36 classrooms (22 classrooms + 14 labs), and for 27 academic staff was made. In addition, these results were obtained for 36 different groups in Computer Engineering Department.

3.1. Population Size *(Popülasyoon Sayısı)*

The results of the program were obtained for 10, 25, 50, 75, and 100 populations, respectively at 10,000 generations. These results were obtained under the same conditions for half-genetic algorithm and full genetic algorithms, respectively. The effects of the number of the population are shown in Table 2 for each of the two genetic solutions. The comparison of the performances for each genetic solutions is shown in Figure 8.

Table 2. Comparison of fitness values for half and full genetic algorithm *(Kısmi ve tam genetik algoritma uygunluk değerlerinin karşılaştırması)*

Population Size	**Error Values for Half Genetic Algorithm**	**Error Values for Full Genetic Algorithm**
10	6457	31762
25	5512	40768
50	5615	58529
75	5614	67804
100	5606	77851

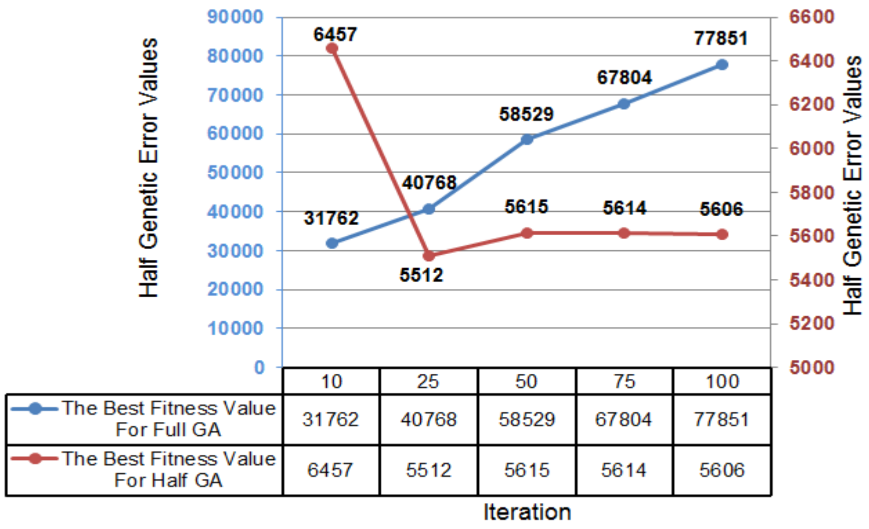

Figure 8. Performance comparison for half and full genetic algorithm *(Kısmi ve tam genetik algoritma performans karşılaştırması)*

The program was run several times for each stage until the optimum results were obtained. As a result, when the data and graph were analyzed, it was seen that the best population size was 25 for both two solution methods. The result of the experiment showed that incrementing the size of the population affects the solution negatively; and generally decreases the performance for both two solution methods. The negative effect is much higher for the Full Genetic Solution. At Half-Genetic solution, when the population ratio is under the certain value, the error rate gets high. When the full and half genetic solution results were analyzed under equal conditions, Half-Genetic solution appears to be much more successful in solving the problem.

3.2. Mutation Rate *(Mutasyon Oranıı)*

The program performance was measured by changing the mutation rates. Thus, the performance data were obtained for different mutation rates. Two different mutation rate data were obtained from Full Genetic Solution and Half Genetic Solution for comparison.

3.2.1. Mutation results for full genetic solution *(Tam genetik çözüm için mutasyon sonuçlarıi)*

When the obtained results are analyzed, it is seen that if mutation probability rate is lower than a certain rate, the optimal solution of the problem gets hard and the error rate gets too high for 10000 generations. In the same way, while the program is running on the full genetic solution mode, it is seen that when mutation probability rate is higher than a certain rate, the optimal solution of the problem gets hard and the error rate gets too high for 10000 generations, too. The optimum value was obtained at the range of 30% to 40% mutation rate for the solution. The obtained results are shown in Table 3. The chart of the results is shown in Figure 9.

Table 3. Mutation rates and change of error values for full genetic solution (Generation Number: 10 000) *(Mutasyon oranlarına göre tam genetik algoritma için hata değerlerinin değişimi)*

Mutation Rates (%)	Error Values
1	0
2	58217
5	52376
10	46744
15	42130
20	43110
25	42884
30	41115
40	40660
50	41639
65	42071
80	44228
99	49297

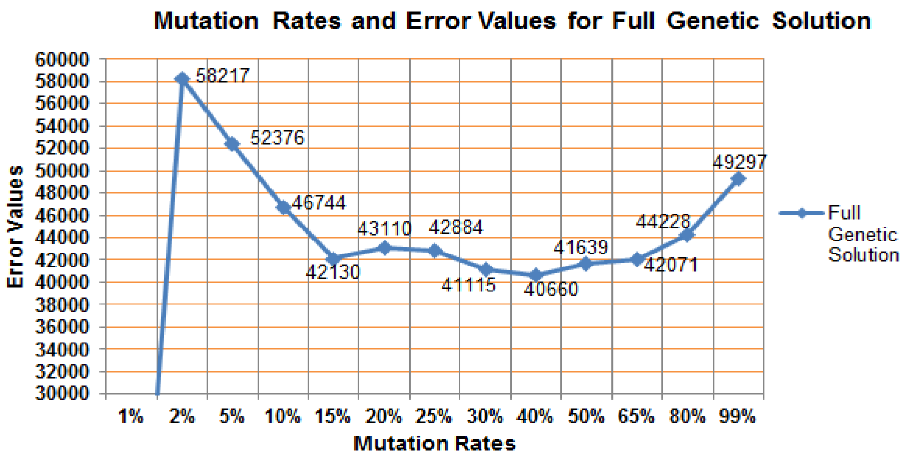

Figure 9. The chart of mutation rates and error values *(Mutasyon oranlarına göre hata değerlerinin değişim grafiği)*

3.2.2. Mutation results for half genetic solution *(Kısmi genetik çözüm için mutasyon sonuçları)*

For Half Genetic Solutions, it is seen that, when the mutation rate is higher or lower than a certain rate, it affects the solution negatively, as it is the case in the Full Genetic Solution. While the mutation rate is higher or lower than a certain value, it was seen that reaching an optimum solution gets harder and the error rate gets higher for 10000 generations. In the experiments, the mutation rate was observed to be around 50% of the value of the best for Half Genetic Solution. The effect of the mutation rates on the error values is shown in Table 4.The chart of the results for these values is shown in Figure 10.

Table 4. Mutation rates and change of error values for half genetic solution (Generation Number: 10 000) *(Mutasyon oranlarına göre kısmi genetik algoritma için hata değerlerinin değişimi)*

Mutation Rates (%)	Error Values
1	0
2	6387
5	5880
10	6380
15	5431
20	5594
25	6024
30	5400
40	5735
50	4320
65	6355

80	6983
99	6435

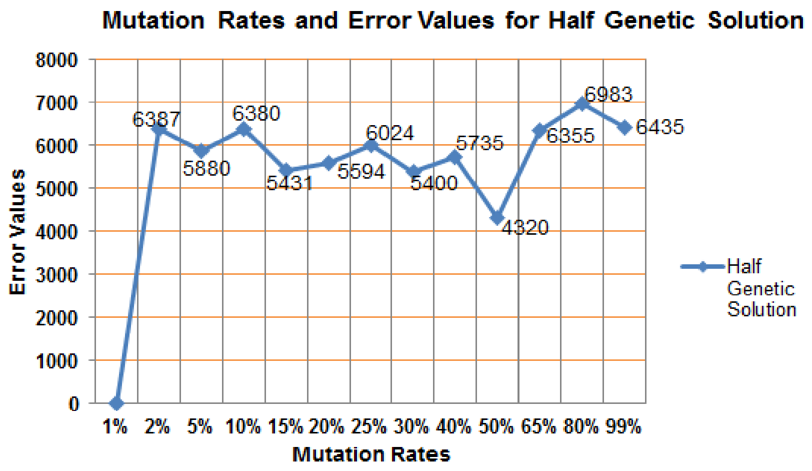

Figure 10. The chart of mutation rates and error values for half genetic solution *(Kısmi genetik çözüm için mutasyon oranlarına göre hata değerlerinin değişim grafiği)*

3.3. Number of Mutation Point *(Mutasyon Nokta Sayısı)*

The effect of the number of the mutation point was measured by changing the numbers of the mutation points. The results have been obtained by testing more than once for different mutation point numbers. Two different data were obtained for Full Genetic Solution and Half Genetic Solution. The effect of the number of the mutation point for full genetic solution is shown in Table 5. These data are shown as a chart in Figure 11.

Table 5. Number of mutation point – error values for full genetic solution (Generation Number: 10 000) *(Tam genetik çözüm için mutasyon nokta sayısı ve hata değerleri)*

Number of Mutation Point	Error Values
1	0
2	41115
3	46439
4	49405
5	51225
7	51218
9	53583
10	55827

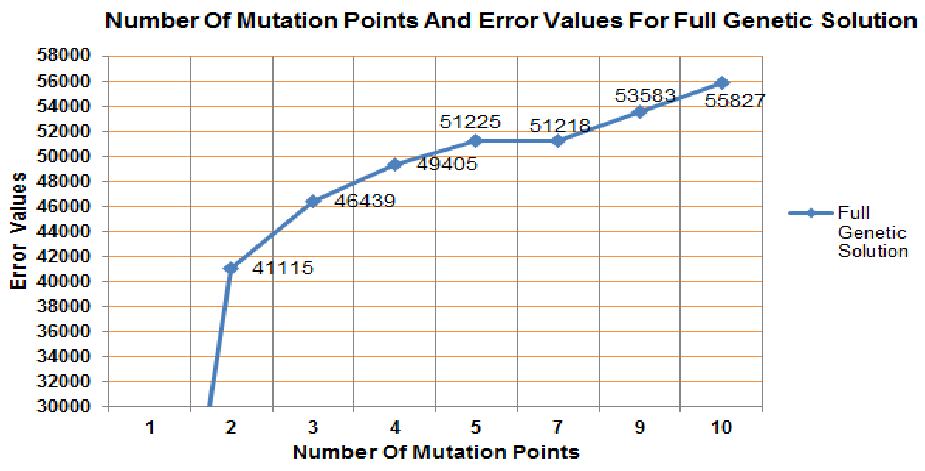

Figure 11. Numbers of mutation points and change of error values for full genetic solution *(Tam genetik çözüm için mutasyon nokta sayısına göre hata değerlerinin değişim grafiği)*

When the results were analyzed, the effect of the increase in the number of the point mutations in the solution of the course scheduling problems was found to be negative. When the mutation point number is higher than a certain number, it gets harder to find a solution, and the error values get higher for 10000 generations. The best error value was obtained at the two mutation point numbers on the Full Genetic Solution mode.

For the Half Genetic Solution, the change of the error values was observed by running the program more than once on the different numbers of the mutation points. The obtained data is shown in Table 6. The graphical view of the data is shown in Figure 12. When the table and the chart are analysed, it is seen that, while the mutation point number increases, it gets difficult to reach a solution. On the Half Genetic Solution mode, while the mutation point number gets higher, the error values increases. The best solution was obtained at the two numbers of the mutation points for Half Genetic Solution.

Table 6 Numbers of mutation points - error values for half genetic solution (Generation Number: 10 000)) *(Kısmi genetik çözüm için mutasyon nokta sayısı ve hata değerleri)*

Mutation Point Number	Error Values
1	0
2	5400
3	5920
4	6862
5	6297
7	6250
9	6460
10	6270

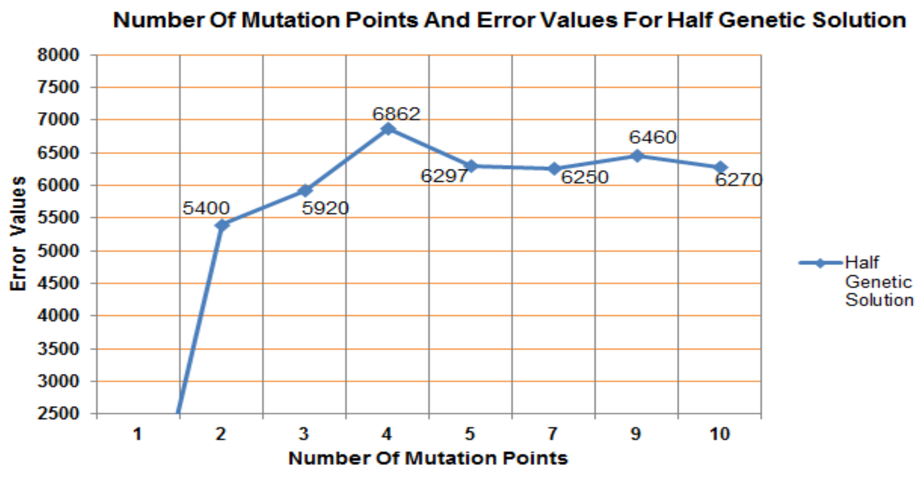

Figure 12. Chart of numbers *of* mutation points and change of error values for half genetic solution *(Kısmi genetik çözüm için mutasyon nokta sayısına göre hata değerlerinin değişim grafiği)*

4. CONCLUSION

In this study, a scheduling program has been developed to solve the course scheduling problem at schools and at other educational institutions. The program has been developed with Genetic Algorithm Technique. Two different genetic solution methods as Full and Half Genetic Solutions were compared. The performance of the solution methods under equal conditions was compared for different population numbers, different mutation rates, and different mutation point numbers. According to the obtained data, the best genetic solution method, the best population number, the best mutation rate, and the best mutation point number have been determined for the best performance of the developed software.

In conclusion, it was seen that the Half Genetic Solution Method is faster and more successful than Full Genetic Solution Method. The best population number was determined as 25, and the best mutation point number was determined as 2 for both solution methods. The best solution has been obtained at 40 percent mutation rate for Full Genetic Solution Method and at 50 percent mutation rate for Half Genetic Solution Method.

REFERENCES *(KAYNAKLAR)*

[1] R. W. Eglese and G. K. Rand, "Conference seminar timetabling," *J. Oper. Res. Soc.*, vol. 38, no. 7, pp. 591–598, 1987.

[2] D. Abramson, "Constructing school timetables using simulated annealing: sequential and parallel algorithms," *Manage. Sci.*, vol. 37, no. 1, pp. 98–113, 1991.

[3] D. Costa, "A tabu search algorithm for computing an operational timetable," *Eur. J. Oper. Res.*, vol. 76, no. 1, pp. 98–110, 1994.

[4] E. K. Burke, D. Elliman, and R. Weare, "A genetic algorithm based university timetabling system," in *Proceedings of the 2nd east-west international conference on computer technologies in education*, 1994, vol. 1, pp. 35–40.

[5] W. Erben and J. Keppler, "A genetic algorithm solving a weekly course-timetabling problem," in *International Conference on the Practice and Theory of Automated Timetabling*, 1995, pp. 198–211.

[6] G. Schmidt and T. Ströhlein, "Timetable construction--an annotated bibliography," *Comput. J.*, vol. 23, no. 4, pp. 307–316, 1980.

[7] S. Paksoy, "Genetik algoritma ile proje çizelgeleme," *Yayınlanmamış Doktora Tezi, Çukurova Üniversitesi Sos. Bilim. Enstitüsü*, *Adana,* 2007.

[8] M. Mori and C. C. Tseng, "A genetic algorithm for multi-mode resource constrained project scheduling problem", *European Journal of Operational Research,* 100(1), 134-141, 1997.

[9] M. Mori and C. C. Tseng, "A genetic algorithm for multi-mode resource constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 100, no. 1, pp. 134–141, 1997.

[10] M. Gen and L. Lin, "Genetic algorithms," *Wiley Encycl. Comput. Sci. Eng.*, pp. 1–15, 2007.

[11] G. G. Emel and Ç. Taşkin, "Genetik Algoritmalar ve Uygulama Alanlari," *Uludağ Üniversitesi İktisadi ve İdari Bilim. Fakültesi Derg.*, vol. 21, no. 1, pp. 129–152, 2002.

[12] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, "Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [Book Review]," *IEEE Trans. Automat. Contr.*, vol. 42, no. 10, pp. 1482–1484, 1997.

[13] D. E. Goldberg, "Genetic algorithms and Walsh functions: Part I, a gentle introduction," *Complex Syst.*, vol. 3, no. 2, pp. 129–152, 1989.

Muhammed Mutlu YAPICI[*]
Muhammed Mutlu YAPICI received the Master's degree in Computer and Electronic Education from Informatics Institute, Gazi University, Turkey in 2012. Currently, he is pursuing his Ph.D. Degree in Computer Engineering Department, Gazi University. He is working as a lecturer at the Computer Technologies Department, Ankara University, Turkey. His main research area includes Artificial Intelligence, Machine Learning, Deep Learning, Image Processing and their applications on signature verification and object recognition.
ORCID: 0000-0001-6171-1226

Prof. Dr. Ömer Faruk BAY
Dr. Bay received the B.Sc. degree in Electrical and Electronics Education from Gazi University, Turkey in 1985, M.Sc. and Ph.D. degrees in Electronics Engineering from Erciyes University, Turkey in 1992 and 1996 respectively. He is a full professor at the Department of Electronics and Computer Education in Gazi University. His research interests include artificial intelligence and their applications; control and instrumentation; intelligent systems and BCI for home automation. He has been worked at some national and international Projects as a manager and researcher. He has published more than 85 articles and published one book.
ORCID: 0000 0002 6823 145X