Kütahya Dumlupınar University
Institute of Graduate Studies

Journal of Scientific
Reports-A
E-ISSN: 2687-6167
Number 49, June 2022

RESEARCH ARTICLE

A COMPLETE SOLUTION to EXAM SCHEDULING PROBLEM: A CASE STUDY

Abdullah ELEN*

*Bandırma Onyedi Eylül University, Faculty of Engineering and Nature Science, Devision of Software Engineering, aelen@bandirma.edu.tr, ORCID: 0000-0003-1644-0476

Receive Date:14.10.2021 *Accepted Date: 30.03.2022*

ABSTRACT

Exam scheduling is a complex task that higher education institutions (universities, colleges, etc.) must prepare each semester depending on their academic calendar. The preparation of exam schedules requires a multi-dimensional analysis and experience. It is also a quite time-consuming sequence of operations. Exam times should not overlap when preparing the schedules and needed constraints are expected to be complied with as much as possible. Therefore, it takes a long time to form a complete solution. In this study, a Genetic Algorithm based exam scheduling method was developed to create a complete solution for the Vocational School of T.O.B.B. Technical Sciences, Karabuk University. During the test phase, four different experiments were performed in different constraints and criteria. As a result of these experiments, all solutions gave appropriate results until 2000 iterations. There was no overlap in any of the exam schedules and significant success was achieved in the desired constraints.

Keywords: *Exam Scheduling, Genetic Algorithm, Metaheuristics, Karabuk University.*

1. INTRODUCTION

Universities are higher education institutions in a complex structure with many students, academic and administrative staffs. In these institutions, educational activities are carried out on course and exam schedules [1]. Therefore, it is necessary to prepare two or more (midterm, final exam etc.) exam schedules for all academic programs in each semester. When preparing exam schedules, distribution of exam halls (or classroom) according to departments, in which sessions will take part in lectures, the specific conditions of some courses, special requirements of examiners and course executives, capacity of exam halls, the number of students taking a course, possibility of some students being assigned to more than one exam session at the same time, exams can be held within a certain date range and limited time periods etc. many different situations have to be considered [2, 3]. Also, it may be necessary to revise the exam schedules depending on personal or institutional reasons. In summary, there are many parameters and constraints in exam schedules. Real-world optimization problems with many variables, such as exam/course/job scheduling, cannot be optimally solved at reasonable computational times [4]. Therefore, scheduling tasks are known as NP-hard [5] optimization problems. Also, it is difficult to talk about a general solution for this type of problem. The first studies on scheduling problems were done by Henry Laurence Gantt [6], an American mechanical engineer. These studies initially aimed at understanding the common problem structure by combining the

problems. These problems were tried to be produced with simple models. Towards the end of the 20th century, heuristic methods came to the fore according to the problem. Since 1990s, modern heuristic methods have been used for the solution of some problems [7, 8]. Heuristic algorithms do not guarantee that they will find the exact result, but they do guarantee that they will get a solution within a reasonable time [9]. In other words, the definitive solution is unknown, but the closest solution is a problem in which we want to determine what is good for determining quality of the solutions. For a problem to be solved by optimization algorithms, the constraints that the algorithm must comply with are defined. Then, an objective function is created by taking these constraints into consideration [10]. Two types of constraints are generally preferred for studies on exam scheduling problems in the literature. These are hard and soft constraints. Hard constraints are mandatory rules, while soft constraints are constraints are rules that are desired but not obligatory [11].

In recent years, meta-heuristic methods have been widely used in solving the exam scheduling problem. For instance, Santiago-Mozos et al. [12] proposed a two-step heuristic method for obtaining personalized schedules for some courses for a university in Spain. In this method, students can determine the groups they prefer on a given subject and their priority in the lessons. In the first stage of the method, the constraints of the problem are solved, and in the second stage, the solutions found are tried to be improved. Dammak et al. [13] developed a simple heuristic method to perform exam-classroom assignments to solve an exam scheduling problem. Their method tries to assign one exam in each classroom. If the method fails to provide this, it assigns a maximum of two exam halls. Pillay and Banzhaf [14] used the Informed Genetic Algorithm (IGA) as a two-step method for solving the exam scheduling problem. In the first stage, a GA is used to generate timetables that do not violate any hard constraints, and in the second stage, a GA is used to optimize the soft constraint costs of the schedules created in the first stage. Turabieh and Abdullah [15] proposed a hybrid approach that combines the principle of electromagnetic-like mechanism with the Great Deluge (GD) algorithm for the solution of the exam scheduling problem. The purpose of their proposed method is to move the sample points towards a high quality solution while avoiding local optimization using a calculated force value. This dynamically calculated value is evaluated as a distortion rate in determining the level within the GD algorithm. Shatnawi et al. [16] aimed to create the exam schedules of the Arab East College for High Education in Saudi Arabia with the least amount of overlap, considering some constraints. Their proposed method has two stages: the greedy algorithm and the genetic algorithm. They reported that by running these two algorithms in cooperation, they significantly reduced exam days and overlaps. Keskin et al. [17] proposed a two-step solution approach to Pamukkale University Faculty of Engineering's exam scheduling problem. They reported that the method they proposed produced faster results than commercial software IBM-CPLEX and Gurobi Optimization. Güler et al. [18] proposed mixed integer programming models for solving the exam scheduling and supervisor assignment problem. They stated that by integrating these models into a web-based decision support system, a complete timetable can be prepared in about two minutes. Aldeeb et al. [19] investigated the Intelligent Water Drops (IWD) algorithm for solving the university exam scheduling problem. They proposed a hybrid method based on local search algorithm to improve the disadvantages of the IWD. They reported that it achieved the best results on the three datasets compared to the best-known results in their experimental work. Hao et al. [20] A unified evolutionary multitasking graph-based hyper-heuristic (EMHH) framework is proposed in which the concept of evolutionary multitasking and graph heuristics are used as high-level search methodology and low-level heuristics, respectively. The EMHH was evaluated on the exam schedule and graphic coloring problems. Their experimental results indicate that the proposed unified framework improves efficiency, efficiency, and generality when compared to single-task hyper-heuristics.

In this study, the exam schedules were automatically generated by using the student course information of the 2018-2019 fall semester of Vocational School of T.O.B.B. Technical Sciences, Karabuk University. This paper is organized as follows; In the first section, creation of a relational database for solving the exam scheduling problem, determining the number of exam branches, creating the chromosome structure, and defining constraints and fitness function are mentioned. In the second section, four experimental studies were conducted according to different criteria and the results obtained were evaluated. In the last section, this paper is summarized.

2. MATERIALS AND METHODS

The categorical data in the database prepared in this study are as follows. There are 421 courses, 54 lecturers, 27 academic programs and 21 classrooms. In order to solve the problem, the model of the database was created with the E-R diagram shown in Fig. 1. This database was prepared based on the curricular relations of academic programs. Additional tables were also used to determine some specific limitations. For example, some lessons are taught in the laboratory only exams, common exams (for common courses taught in all departments) of the date/time intervals to be predetermined, according to the demands of lecturers are made according to the individual date/time criterion.

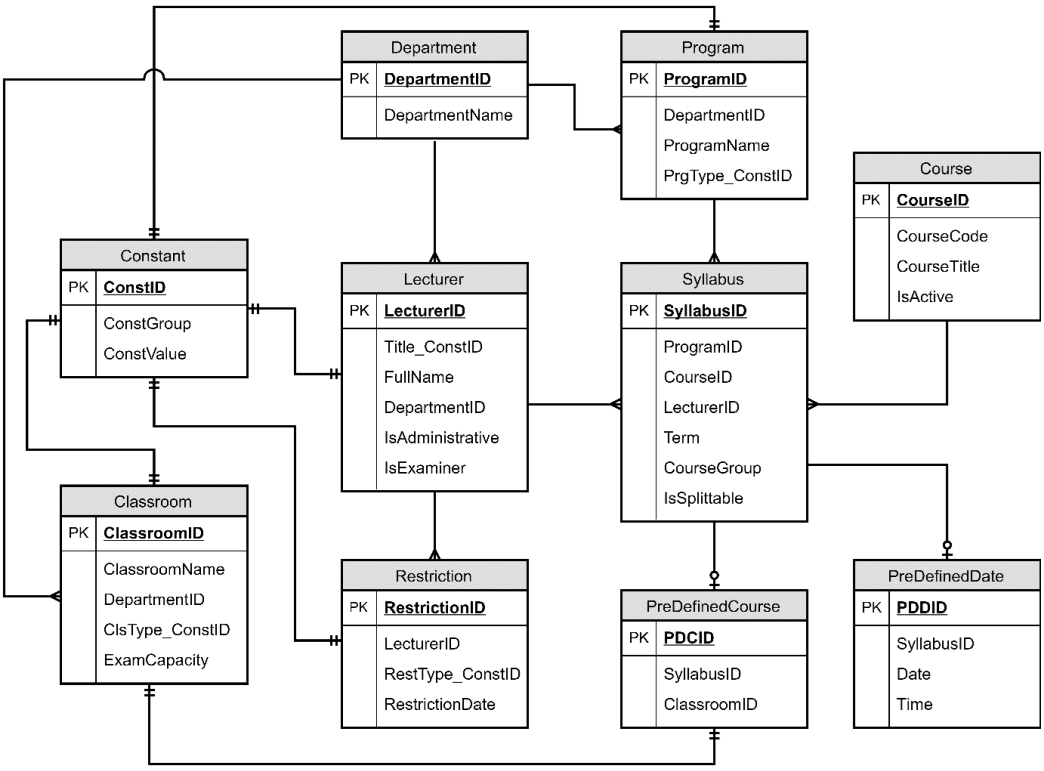

Fig. 1. E-R diagram of the database schema for exam scheduling problem.

2.1. Pre-processing

Before start of the scheduling, several pre-processes are required. These processes are as follows; loading the data into the system completely, combining the grouped courses, defining the pre-defined exams which have a fixed time or place in the table. Some of the exams of the courses are grouped for a variety of reasons. For this, the process of joining the grouped courses is done as shown in Algorithm 1. Where C represents the entire course list. For each course is checked whether it belongs to a group. If a course has a group, the number of students taking the course is determined and added to the main course. The function of $getIndex(g)$ is used to determine the index of the main course. In the last step, the group courses that are completed in the merge process are removed from the list.

The courses with group joining are passed through a final phase and the exam branches are determined. The $CreateExamSections$ function in Algorithm 2 is used for this process. The input parameters of this function are as follows; C is course list, $splitVal$ is number of students to be placed in a classroom and $tolerance$ is placement tolerance for a classroom.

Algorithm 1. Merge all grouped courses.

```
1:   function MergeGroupedCourses(C: Course list)
2:       for i ← 1 to C.count() do
3:           set g ← getGroupCourse(C[i]);
4:           if g is not empty then
5:               set sci ← getStudentCount(C[i]);
6:               set scg ← getStudentCount(C[getIndex(g)]);
7:               scg ← scg + sci;
8:               setStudentCount(C[getIndex(g)], scg);
9:               C.remove(C[i]);
10:          end if
11:      end for
12:      return C;
13:  end function
```

Algorithm 2. Create exam sections using merged course list.

```
1:  function CreateExamSections(C, splitVal, tolerance)
2:      set S[], index = 1;
3:      for i ← 1 to C.count() do
4:          set section ← truncate(getStudentCount(C[i]) / splitVal);
5:          for j ← 1 to section do
6:              S[index] ← C[i];
7:              setStudentCount(S[index], splitVal);
8:              index ← index + 1;
9:          end for
10:         set remain ← getStudentCount(C[i]) mod splitVal;
11:         if (remain > tolerance) then
12:             S[index] ← C[i];
13:             setStudentCount(S[index], splitVal);
14:             index ← index + 1;
15:         else
16:             set part ← truncate(remain / section);
17:             set partRemain ← remain mod section;
18:             for j ← 1 to S.count() do
19:                 set sc ← getStudentCount(S[j]);
20:                 setStudentCount(S[j], (sc + part));
21                  if (partRemain > 0 and j = 1) then
22:                     setStudentCount(S[j], (sc + part + partRemain));
23:                 end if
24:             end for
25:         end if
26:     end for
27:     return S;
28: end function
```

The purpose of this function is to determine the number of students to be placed in each class according to the given input parameters and the examination sessions depending on these procedures. According to Algorithm 1, each course is examined separately in terms of the number of students taking the course, the number of students to be placed in a class and their tolerance. The parameters required for this algorithm are set as shown in Fig. 2 through the values specified in the *"Number of Students"* section.

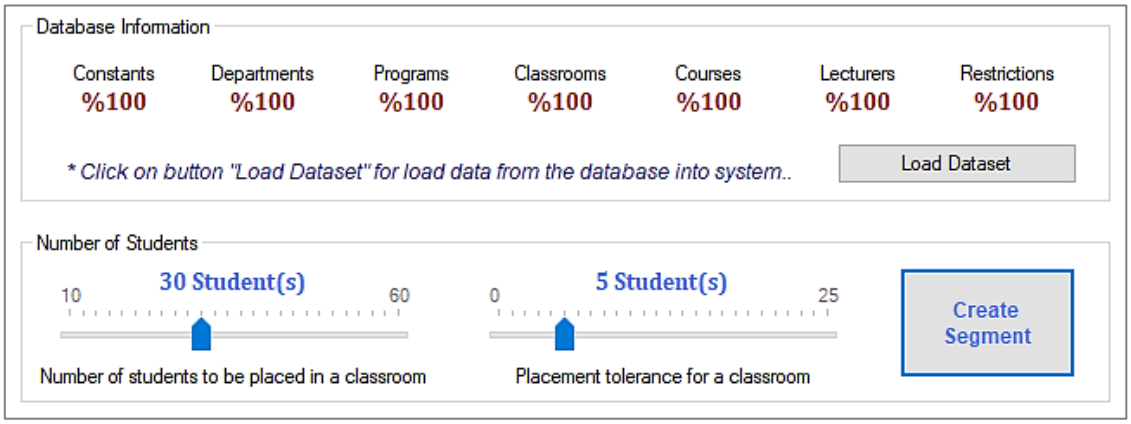

Fig. 2. Dataset loading and student segmentation process.

In Table 1, four different scenarios are given as an example for determining the number of exam branches *(EB)*. The calculations are made by using the information of 421 courses registered in the database as mentioned before. Where means S_P, number of students to be placed in a classroom and S_T, placement tolerance for a classroom. In summary, the EB values we obtain directly affect the structure (chromosome length) and performance of the algorithm to be used in the solution of the problem. The chromosome length of the algorithm varies dynamically according to EB.

Table 1. Determination of the number of exam sections according to S_P and S_T parameters.

Scenarios	S_P	S_T	ES
Sce-1	30	5	350
Sce-2	30	10	338
Sce-3	35	5	333
Sce-4	35	10	305

The final processing steps before running the scheduling algorithm are the determination of the exam schedule as in the Graphical User Interface (GUI) shown in Figure 3. From this interface, the active semester, the type of the exam, the start and end date interval of the exams (D_R), the time intervals to be used during the day (T_R) and the duration of the exams (T_D) are determined. Then holidays and/or school's closed times (D_C) are added to the list. Finally, it is determined whether the faculty member will be assigned for exams outside of their own courses (fmo) and all parameters are prepared. The exact dates for the preparation of the exam schedules (D_E) are determined as shown in Eq. (1).

$$D_E = \forall x \left(x \in D_R \wedge x \notin D_C \right) \tag{1}$$

Where x refers to any date in the start and end of the exams. According to the date range given in Fig. (3), the number of elements of the D_E (number of days) is equal to 10 when the days corresponding to the weekend are removed (From 5 to 16, except November 10 and 11).

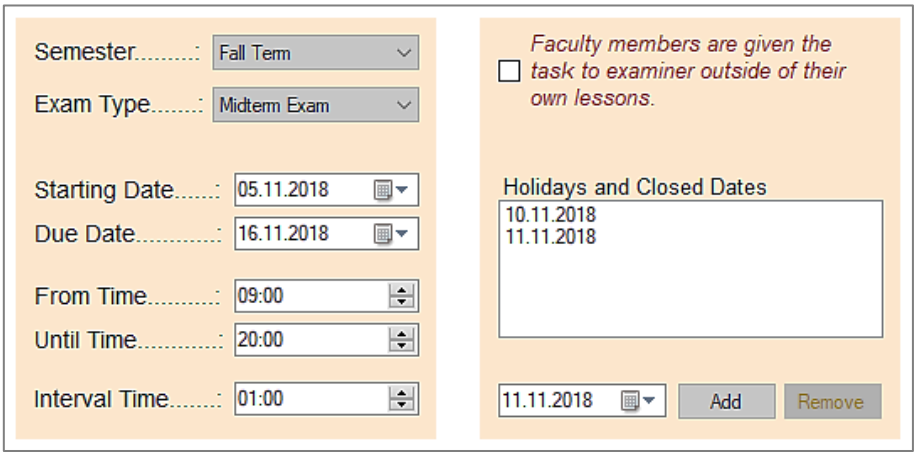

Fig. 3. Exam calendar and overall restrictions.

The flowchart of the proposed method and pre-process is as shown in Fig. (4). The first process collects the necessary information from the database. Then, group courses are combined into a single course. The user creates exam branches as shown previously in Fig. (2). The constraint rules are set in the last processing step before the algorithm is executed. Thus, all necessary parameters are provided for the genetic algorithm.

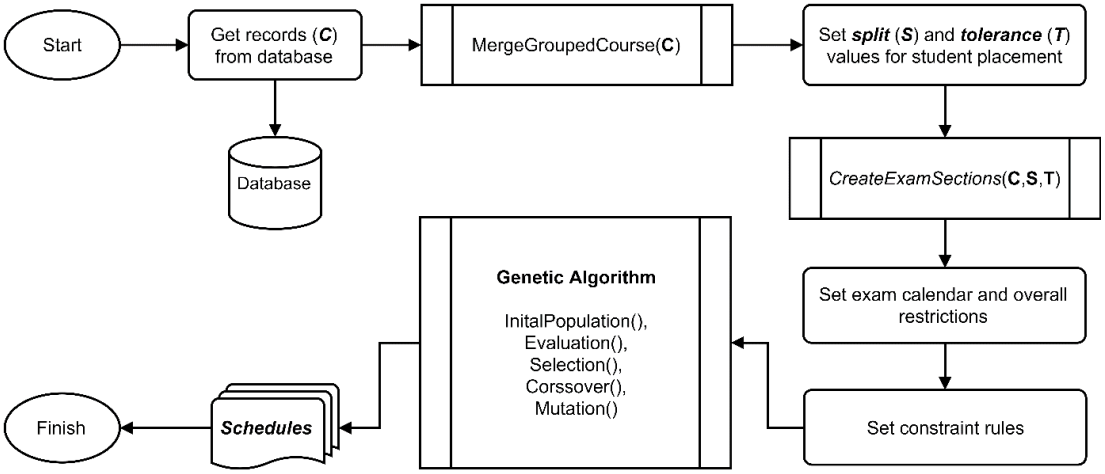

Fig. 4. Flowchart of the proposed method.

2.2. Structure of the Chromosome

The length of the chromosome in the GA to be used in solving the problem is determined by the interface as shown in Table 1 before. That is, the length of a chromosome is equal to EB. Fig. (5) shows a representative chromosome and the structure of a gene. Each gene has 14 different information. Some of these (from G_7 to G_{13}) are used for control purposes. G_0 and those for control purposes are not subject to any genetic processing.

JSR A
Journal of Scientific Reports

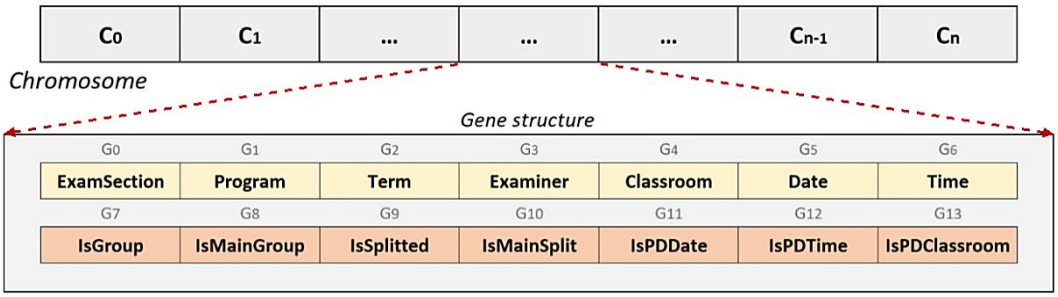

Fig. 5. Structure of the chromosome and gene.

While describing the location of the genes with the chromosome variable ($C_{i,j}$); where is $i \in \{0,1,\dots,n\}$ the gene index on the chromosome and $j \in \{0,1,\dots,13\}$ refers to the field index in the gene. C_i identifies a gene on the chromosome. The explanations of the features of the structure of the gene are given in Table 2. Where G_0 represents an exam branch. G_1 to G_6 are information that appears directly on the shedule. G_6 and G_7 determine whether a course has been tested. G_9 and G_{10} stores the information that the exam branch does not belong to a divided course.

Table 2. Descriptions of the gene parameters.

Index	$C_{i,j}$	Parameter	Description
G_0	$C_{i,0}$	ExamSection	*The exam that was created during pre-processing.*
G_1	$C_{i,1}$	Program	*Academic program.*
G_2	$C_{i,2}$	Term	*The academic term of the program.*
G_3	$C_{i,3}$	Examiner	*The lecturer who was tasked for the examination.*
G_4	$C_{i,4}$	Classroom	*The place where the examination will be held.*
G_5	$C_{i,5}$	Date	*The date when the examination will be held.*
G_6	$C_{i,6}$	Time	*The time when the examination will be held.*
G_7	$C_{i,7}$	IsGroup	*Is there a group of the course that is connected to this*
G_8	$C_{i,8}$	IsMainGroup	*Is main group that the course is connected to this exam?*
G_9	$C_{i,9}$	IsSplitted	*Is that the course was splitted connected to this exam?*
G_{10}	$C_{i,10}$	IsMainSplit	*Is main split that the course is connected to this exam?*
G_{11}	$C_{i,11}$	IsPDDate	*Is there a predefined date for this exam?*
G_{12}	$C_{i,12}$	IsPDTime	*Is there a predefined time for this exam?*
G_{13}	$C_{i,13}$	IsPDClassroom	*Are there any predefined classrooms for this exam?*

Where G_0 holds some information about the course to which an exam is attached. These statements keep the "*SyllabusID*" field, which is the primary key in the table (T_{SYL}) named "*Syllabus"*, in an array. It is an array that holds the "*CourseID*" field if it is a divided course and/or a group course. This information is used to establish the relationship between the curriculum and the exam shedules to be created. The parameters used in the genetic algorithm are as follows; The selection method is "*Elite*" the crossover rate is 90%, the mutation rate is 3%, the population is 100.

2.3. Constraints

Some rules must be fulfilled for the generation of genes in the chromosome. These rules and restrictions are presented separately under the four headings below. Also, it was divided into three main categories as "$soft$", "$middle$" and "$hard$" to define the constraints as more flexible. Objective functions calculate a fitness value based on constraint definitions. When calculating these fitness values, it can be maxima or minima depending on the type of problem.

2.3.1. Examiner and faculty member

The knowledge that every faculty member can be an examiner for the exam is stored in the table (T_{LEC}) named "*Lecturer*" in the database. Accordingly, the examiner list (L_E) to be used in the genes on the chromosome is as in Eq. (2).

$$L_E = \left\{ i \middle| i \in \{0,...,n\}, \left(T_{LEC}[i,5] \; is \; \text{True} \rightarrow T_{LEC}[i,0] \right) \right\} \tag{2}$$

Where n is the number of records in the table, $T_{LEC}[i,5]$ the table area (*IsExaminer*) that keeps track of whether the examiner is assigned for the faculty member in the i-th, $T_{LEC}[i,0]$ refers to the faculty member's ID (*LecturerID as primary key*). If faculty members only want to be examiners in their own courses, the examiner list is determined as in Eq. (3).

$$\begin{aligned} L_{FMT} &= \{'assist.prof','assoc.prof','prof'\}, \\ L_E &= \left\{ i \middle| i \in \{0,...,n\}, \left(\left(T_{LEC}[i,5] \; is \; \text{True} \wedge T_{LEC}[i,1] \notin L_{FMT} \right) \rightarrow T_{LEC}[i,0] \right) \right\} \end{aligned} \tag{3}$$

Where L_{FMT} represents the degree list of faculty members, $T_{LEC}[i,1]$ represents the degree of the faculty member in the *i*-th row/record line.

2.3.2. Pre-defined dates and classrooms

As mentioned earlier, some exams are stored in the database named $PreDefinedDate$ (T_{PDD}) table as a predetermined date. Accordingly, the date ($C_{i,5}$) and time ($C_{i,6}$) information of each gene that satisfies the $C_i \in T_{PDD}$ condition on the chromosome is updated according to the table T_{PDD}. Then the control fields of $IsPDDate$ ($C_{i,11}$) and $IsPDTime$ ($C_{i,12}$) are set to logic "1". If the place for the exam to be made, such as workshops, laboratories, etc. are made in the table (T_{PDC}) called $PreDefinedClassroom$; The Classroom ($C_{i,4}$) information of each gene that satisfies the $C_i \in T_{PDC}$ requirement on the chromosome is updated according to the table T_{PDC}. Then the control field of $IsPDClassroom$ ($C_{i,13}$) is set to logic "1". Pre-defined areas of these genes are not mutated during genetic procedures. Otherwise, this process has no meaning.

2.3.3. Examiner date and time ranges

The "$Restriction$" table (T_{RST}) in the database contains the date and time interval information that faculty members do not want to be assigned to the exam. As seen in Table 3, these constraints can be defined in three different ways. There are "$If\ possible$" and "$Absolutely$" options for each constraint. Accordingly, if "$If\ possible$" option is selected for any of these three constraints, P_M is used as penalty point and P_H is used if possible "$Absolutely$" option is selected.

Table 3. Date and time range constraints for examiners.

ID	Examiner constraints	If possible	Absolutely
K_{BN}	*I do not want to be tasked before-noon*		
K_{AN}	*I do not want to be tasked afternoon*	P_M	P_H
K_{SD}	*I do not want to be tasked on the date specified*		

Accordingly, the penalties for each constraint are calculated as shown Eq. (4), (5) and (6).

$$K_{BN} = p\sum_{i=0}^{n}\sum_{j=0}^{m}\begin{cases}1, & \left(T_{RST}\left[j,1\right]=C_{i,3}\right)\wedge\left(C_{i,6}<time\left(12:00\right)\right), \quad p\in\left\{P_M,P_H\right\}\\ 0, & otherwise\end{cases} \tag{4}$$

$$K_{AN} = p\sum_{i=0}^{n}\sum_{j=0}^{m}\begin{cases}1, & \left(T_{RST}\left[j,1\right]=C_{i,3}\right)\wedge\left(C_{i,6}>time\left(12:00\right)\right), \quad p\in\left\{P_M,P_H\right\}\\ 0, & otherwise\end{cases} \tag{5}$$

$$K_{SD} = p\sum_{i=0}^{n}\sum_{j=0}^{m}\begin{cases}1, & \left(T_{RST}\left[j,1\right]=C_{i,3}\right)\wedge\left(C_{i,5}\neq T_{RST}\left[j,3\right]\right), \quad p\in\left\{P_M,P_H\right\}\\ 0, & otherwise\end{cases} \tag{6}$$

Where, P_M is the middle level penalty point, n the number of genes in the chromosome, m the number of records in the table, $T_{RST}[j,1]$ the examiner ID found in the j -th record line in the table, $T_{RST}[j,3]$ the date information in the same row of the table, $C_{i,3}$ examiner information in the i-th gene of the chromosome, $C_{i,5}$ and $C_{i,6}$ refer to the date and time information in the gene, respectively.

2.3.4. Other constraints

Under this heading, other constraints are mentioned. The GUI shown in Fig. (6) has three different constraints. "Low, Middle and High" options are given for each constraint feature to be used. The penalties of these options are equal to P_S, P_M, P_H respectively.

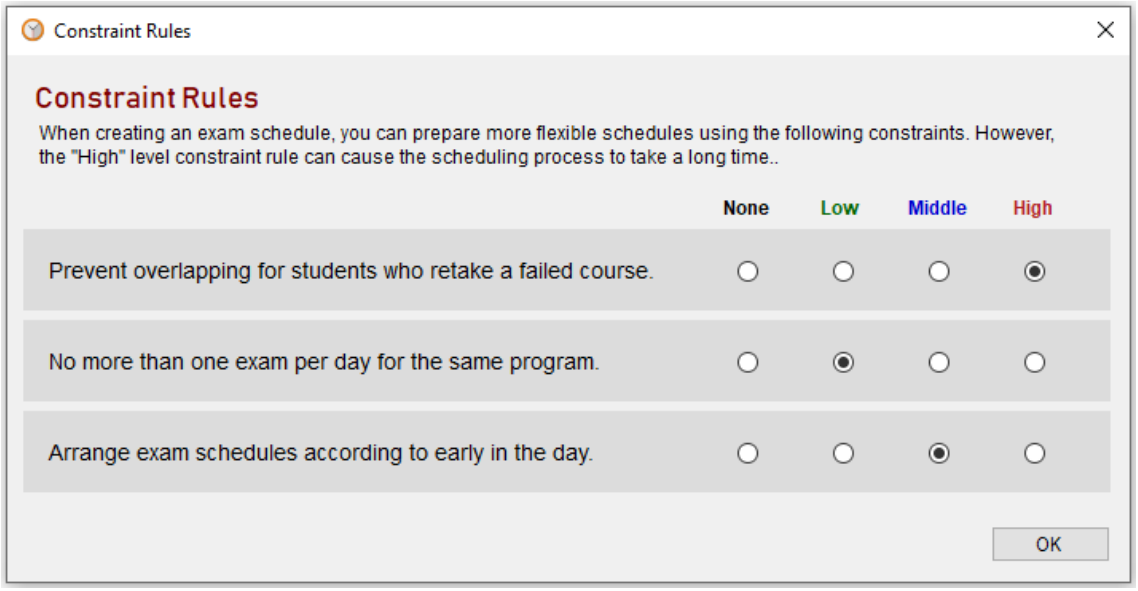

Fig. 6. Constraint rules for the exam scheduling.

The functions that calculate the constraint penalties are shown in Eq. (7) "Prevent overlapping for students who retake a failed course" (K_{SE}), shown in Eq. (8) "No more than one exam per day for the same program" (K_{PE}) and shown in Eq. (9) "Arrange exam schedules according to early in the day" (K_{DE}), respectively. The operating principle of these functions is like sorting algorithms. Two consecutive gene ($eg: C_{i,5} - C_{i+1,5}$) information on a chromosome are passed through comparative logical tests. If the necessary conditions are met, the penalty value is increased, otherwise it is ineffective.

$$K_{SE} = p\sum_{i=0}^{n-1}\sum_{j=i+1}^{n}\begin{cases}1, & C_{i,1}=C_{j,1}\wedge C_{i,5}=C_{j,5}\wedge C_{i,6}=C_{j,6},\ p\in\{P_S,P_M,P_H\}\\ 0, & otherwise\end{cases} \tag{7}$$

$$K_{PE} = p\sum_{i=0}^{n-1}\sum_{j=i+1}^{n}\begin{cases}1, & C_{i,1}=C_{j,1}\wedge C_{i,2}=C_{j,2}\wedge C_{i,5}=C_{j,5},\ p\in\{P_S,P_M,P_H\}\\ 0, & otherwise\end{cases} \tag{8}$$

$$K_{DE} = \frac{p}{\mathrm{count}(T_R)}\sum_{i=0}^{n}\begin{cases}\dfrac{C_{i,6}-\mathrm{median}(T_R)}{\max(T_R)-\mathrm{median}(T_R)} & C_{i,6}\geq\mathrm{median}(T_R),\ p\in\{P_S,P_M,P_H\}\\ 0 & otherwise\end{cases} \tag{9}$$

Where p is the optional penalty value, n is the number of genes in the chromosome, $C_{i,1}$, $C_{i,2}$, $C_{i,5}$ and $C_{i,6}$ the academic program in the i-th gene, academic period, the date, time and T_R express the time interval of the exam.

2.4. Fitness Function

For scheduling problems, the most important parameters of the fitness function are undoubtedly overlapping. Three different methods are used for examiners, students and classrooms to calculate penalties in these overlaps. In these calculations, each is considered as a separate layer, as shown in Fig. (7).

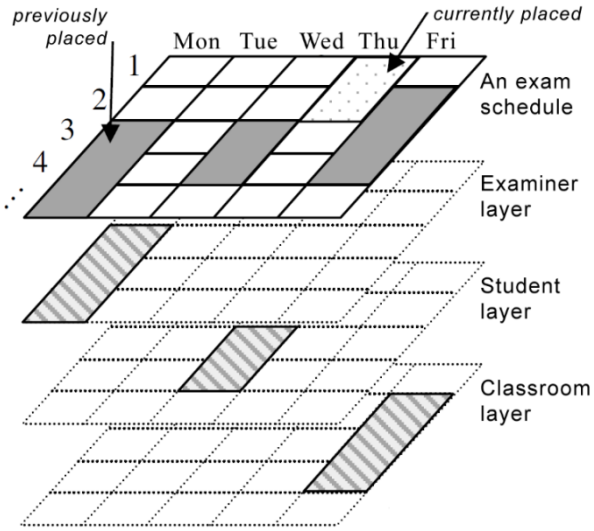

Fig. 7. Presentation of an exam schedule as a layer.

Accordingly, to avoid overlaps, the rules that must be followed in each layer are as follows.

- **Examiner layer:** A examiner cannot be found in different places (*classrooms*) in the same time frame.
- **Student layer:** A student cannot take more than one exam at the same time-period and only one exam can be done in the same period for the relevant semester of the academic program to which he/she is affiliated.
- **Classroom layer:** In a classroom, more than one exam cannot be done at the same time-period.

According to these general rules, functions that calculate overlap penalties are as follows; The examiner overlaps (O_E) shown in Eq. (10), the student overlaps (O_S) shown in Eq. (11), and classroom overlaps (O_C) shown in Eq. (12).

$$O_E = P_H \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \begin{cases} 1, & C_{i,3} = C_{j,3} \wedge C_{i,5} = C_{j,5} \wedge C_{i,6} = C_{j,6} \\ 0, & otherwise \end{cases} \tag{10}$$

$$O_S = P_H \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \begin{cases} 1, & C_{i,1} = C_{j,1} \wedge C_{i,2} = C_{j,2} \wedge C_{i,5} = C_{j,5} \wedge C_{i,6} = C_{j,6} \\ 0, & otherwise \end{cases} \tag{11}$$

$$O_C = P_H \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \begin{cases} 1, & C_{i,4} = C_{j,4} \wedge C_{i,5} = C_{j,5} \wedge C_{i,6} = C_{j,6} \\ 0, & otherwise \end{cases} \tag{12}$$

Where the values mean; n is the number of genes in the chromosome, P_H is the penalty value for hard constraints, $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, $C_{i,4}$, $C_{i,5}$ and $C_{i,6}$ the academic programs academic period, examiner, classrooms, date and time expresses the information in the i-th gene respectively. A logical "AND" operator was used for each gene in the chromosome. Another issue that needs to be considered when preparing the exam schedules is that the number of exam tasks per examiner should be as balanced as possible. As shown in Eq. (13) as part of the fitness function, average distribution of tasks (μ_E), standard deviation (σ_E) and balance coefficient (B_E) for the examiners were calculated. Accordingly, if all examiners take equal number of exams, the standard deviation value will be equal to zero. As a result, the balance coefficient will be zero.

$$\begin{aligned} e_k &= \sum_{i=0}^{n} \begin{cases} 1, & C_{i,3} = T_{LEC}[k,0] \\ 0, & otherwise \end{cases}, \quad \mu_E = \frac{1}{m} \sum_{k=0}^{m} e_k, \quad \sigma_E = \sqrt{\frac{1}{m} \sum_{k=0}^{m} (e_k - \mu_E)^2}, \\ B_E &= \left(\sum_{k=0}^{m} \begin{cases} e_k, & e_k > \lceil \mu_E \rceil \\ 0, & otherwise \end{cases} \right) \Bigg/ \left(\sum_{k=0}^{m} \begin{cases} 1, & e_k > \lceil \mu_E \rceil \\ 0, & otherwise \end{cases} \right) \sigma_E P_S \end{aligned} \tag{13}$$

The parameters of Eq. (13) have the following meanings; e_k is the number of examinery duties of k-th faculty members, n is the number of genes in the chromosome, T_{LEC} is the "$Lecturer$" table in the database, $T_{LEC}[k,0]$ is the faculty members ID found in the k-th line in the table, m is the number of examiners, P_S is soft restricted penalty score. Another balance element in the shedule is that the exams of academic programs are distributed as balanced as possible within the calendar. When this criterion is not taken into consideration, a program may have multiple exams on the same day. This negatively affects the performance of students. The function that calculates the program/exam balance according to the calendar interval is shown in Eq. (14).

$$\begin{aligned} t_k &= \sum_{i=0}^{n} \begin{cases} 1, & C_{i,1} = T_{PRG}[k,0] \\ 0, & otherwise \end{cases}, \\ d_k &= \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \begin{cases} 1, & C_{i,1} = C_{j,1} \wedge C_{i,1} = T_{PRG}[k,0] \wedge C_{i,5} = C_{j,5} \\ 0, & otherwise \end{cases}, \\ B_P &= P_S \sum_{k=0}^{m} \frac{d_k}{(t_k - 1)}, \; t_k > 1 \end{aligned} \tag{14}$$

The parameters of Eq. (14) have the following meanings; t_k is the total number of exams for the k-th program, d_k is the number of exams performed more than once in the same day for the k-th program, n *is* the number of genes in the chromosome, $C_{i,1}$ and $C_{i,5}$ is the academic program and date information in the i-th gene, respectively. T_{PRG} is the program table in the database, $T_{PRG}[k,0]$ is the

program ID on k-th row in the table. According to Eq. (14), the ratio of d_k and t_k for each program is summed by multiplying the P_S soft constraint as penalty. Thus, we calculate the program balance coefficient (B_P). The total penalties calculated for a chromosome are shown in Eq. (15).

$$\begin{aligned} f_{Overlap} &= O_E + O_C + O_P, \\ f_{Constraint} &= K_{BN} + K_{AN} + K_{SD} + K_{SE} + K_{PE} + K_{DE}, \\ f_{Balance} &= B_E + B_P, \\ F &= f_{Overlap} + f_{Constraint} + f_{Balance} \end{aligned} \tag{15}$$

Where $f_{Overlap}$ is the total overlap in the chromosome, $f_{Constraint}$ is the sum of the constraints, the $f_{Balance}$ is total balance coefficient, and F is the sum of all the calculated all penalties points. The fitness function of Eq. (16), which is used in the solution of the problem, aims to find the minimum penalty points.

$$fitness = \min\left\{ i \middle| i \in \{0,1,...,n\}, F_i \right\} \tag{16}$$

Where n is the number of the population. Accordingly, the fitness function finds the chromosome with the lowest penalties of the population in each iteration. The smallest fitness score of all time is found in Eq. (17).

$$fitness_{BEST} = \begin{cases} fitness_i & fitness_i < fitness_{BEST} \\ fitness_{BEST} & otherwise \end{cases}, \; i \middle| i \in \{0,1,...,n\} \tag{17}$$

Where n is the number of iterations. Accordingly, if the fitness value in i-th iteration is smaller than the best, the current fitness is determined to be the best. This process is repeated continuously in each iteration until the algorithm terminated.

3. EXPERIMENTAL RESULTS

Tests of the proposed method were carried out in consideration of the following items. Accordingly, four different experimental sets were prepared, and evaluations were made for each. The prepared test parameters are shown in Table 4. The summary information for T.O.B.B. Vocational School of Technical Sciences, Karabuk University is as follows:

- The total number of courses is 421. Since 32 of these courses are related to profession practice, their exams are held in special places such as workshops and laboratories.
- There are 21 educational places: 10 classrooms, 1 amphitheater, 3 laboratories, 6 workshops and 1 seminar room. The usage rate of 10 exam halls, which are classrooms, in exam schedules is approximately 95%. Other places are used for practice exams or for private purposes only. In addition, capacities of these classrooms vary between 35-40 students.

- The number of academic programs is 27. However, as some programs are no longer available for students, the number of students can vary greatly depending on academic programs. For this reason, some courses are grouped together, and the exams are asked to be made jointly.
- The number of registered faculty member for the exam system is 54. However, only 37 of them are examiners. In addition, the 12 of them have the degree of faculty member *(Assist.prof., Assoc.prof., Prof.).*
- The mid-term exams for 2018-2019 fall semester were held between 05/11/2018 and 16/11/2018 (for 2 weeks). However, it was decided not to hold an exam on 11/11/2018 (Sunday). It was also reported that exam hours would be between 09:00 and 18:00.
- It was stated that six of the faculty members will not take part in exams. In addition, three people declared that they did not want to task in the afternoon if possible.

According to the exam distribution, the number of examiners and classrooms for the number of examiners was 37 and the number of classrooms was 10. The calculated numerical distributions according to the *EB* value for each experiment are as in Table 4.

Table 4. Tested parameters of the four different experiments.

Experiments	S_P	S_T	ES	D_E	$p(K_{SE})$	$p(K_{PE})$	$p(K_{DE})$	fmo
Exp-1	30	5	350	11	none	none	low	True
Exp-2	30	10	338	11	middle	low	none	False
Exp-3	35	5	333	11	middle	low	low	True
Exp-4	35	10	305	11	high	middle	low	False

For each experiment, graphs of fitness function, determined constraints and balance conditions were drawn. In balance status graphs, it is required that the exam task per examiner is as equal as possible and for the academic programs, the exams should approach to zero in order to make the optimal distribution according to the determined calendar days. It is expected that each criterion will be reduced to a minimum level in the graphs where the constraint rules are shown.

3.1. Experiment I

In this experiment, a total of 350 exam sessions with 30 students in each class were created for the exam branches (tolerance is 5). The placement of the exams based on the early hours was determined as a constraint. For Exp-1, the fitness, constraint and balance values in each iteration are shown in Fig. 8. According to obtained results; the number of exam tasks per examiner is between 9-11; 21 faculty members with 9 exams, 15 faculty members with 10 exams, one faculty members with 11 exams. The exams evenly were equally distributed according to dates. However, as there were common exams (pre-determined exam dates) on November 5-6, 2018, there was an accumulation in these two days. The exams were placed into the early hours by 80% according to K_{DE} criteria. The number of overlaps in this test table is zero. As a further constraint, all of the K_{SD} constraints were successfully applied for six faculty members. Only one of the K_{AN} constraints could not be fulfilled for three faculty members.

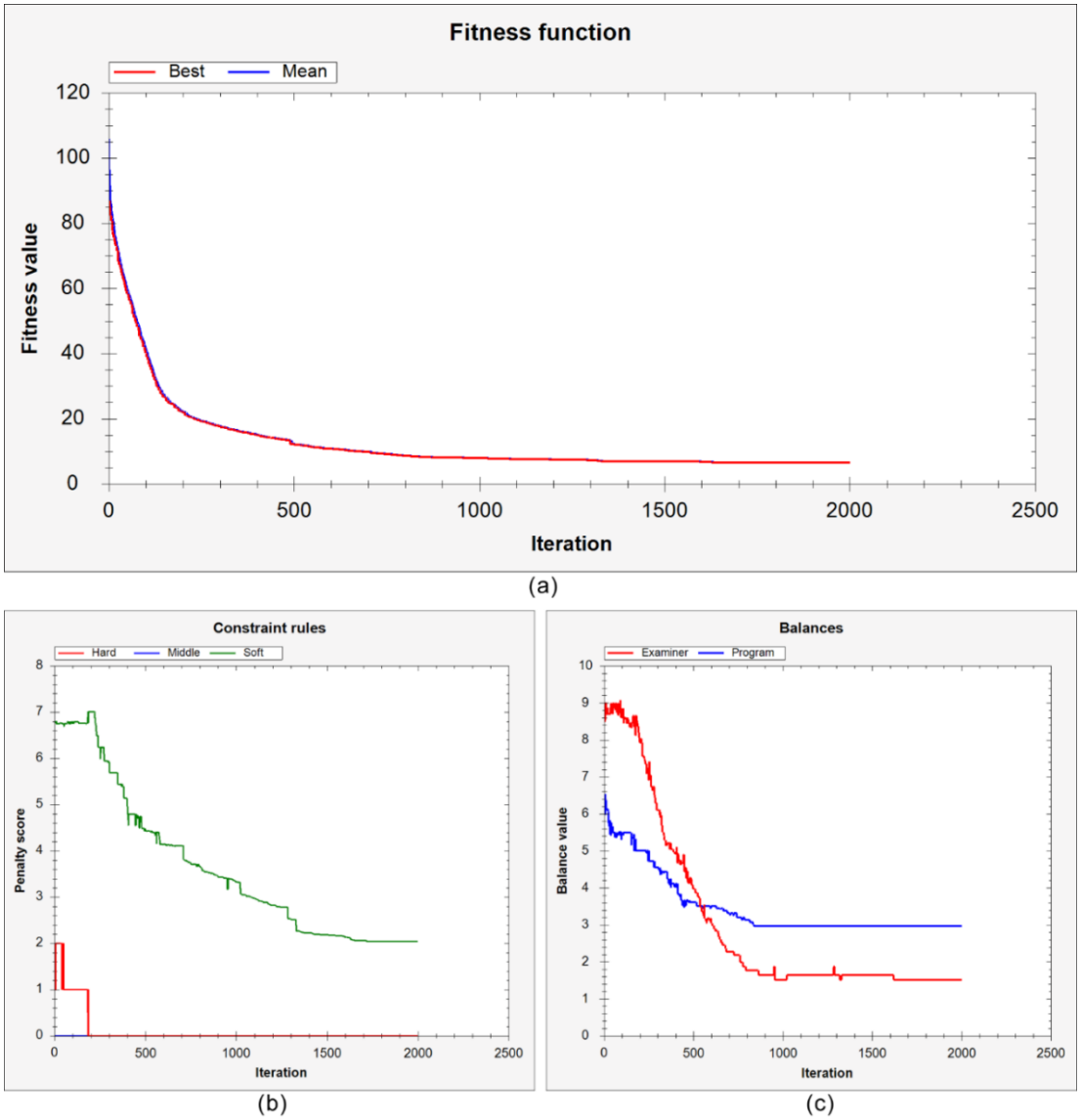

Fig. 8. Exp-1; (a) fitness function, (b) constraint rules, (c) balances.

3.2. Experiment II

In this experiment, a total of 338 exam sessions with 30 students in each class were created for the exam branches (tolerance is 10). In the exams, students who took courses from previous periods were asked not to overlap the exam times in different periods of the same program. In addition, an academic program was asked not to do more than one exam per day. Finally, faculty members were asked not to take exams task outside their own courses. For Exp-2, fitness, constraint, and balance values in each iteration are shown in Fig. (9).

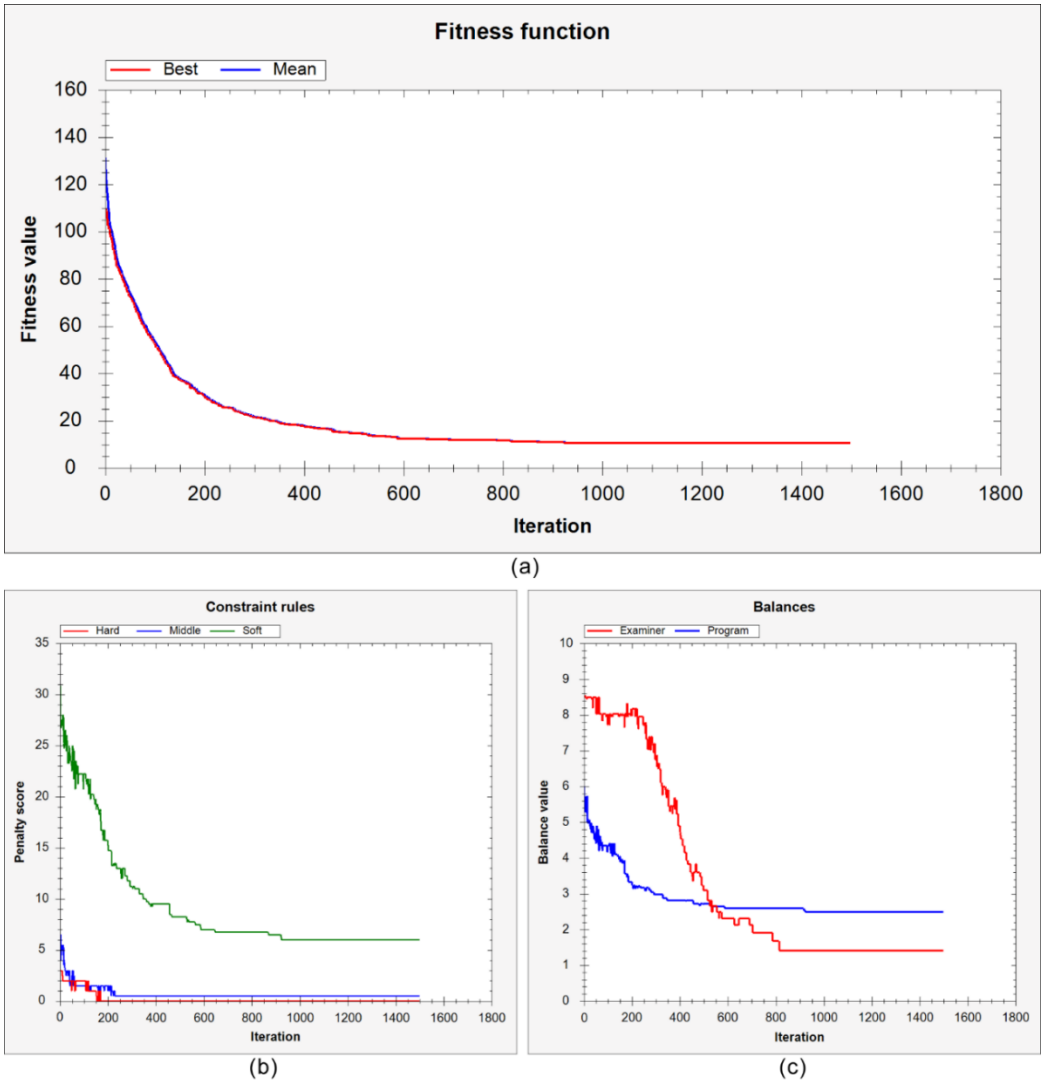

Fig. 9. Exp-2; (a) fitness function, (b) constraint rules, (c) balances.

According to obtained results; the number of exam tasks per examiner (except for 12 faculty members) is between 11-13; 23 faculty members with 11 exams, one faculty members with 12 and 13 exams. According to the K_{SE} criterion, exams are conducted at the same time-period for only 1 of the 338 exams for the students taking the courses from previous periods. According to the K_{PE} criterion, there are 24 exams in all academic programs on the same days. Finally, all of the K_{SD} (for six faculty members) and K_{AN} (for three faculty members) constraints were successfully applied.

3.3. Experiment III

In this experiment, a total of 333 exam sessions with 35 students in each class were created for the exam branches (tolerance is 5). The desired process in the experiment is that; K_{SE}, K_{PE} and K_{DE}

criteria are fulfilled at the specified levels. For Exp-3, fitness, constraint, and balance values in each iteration are shown in Fig. (10).

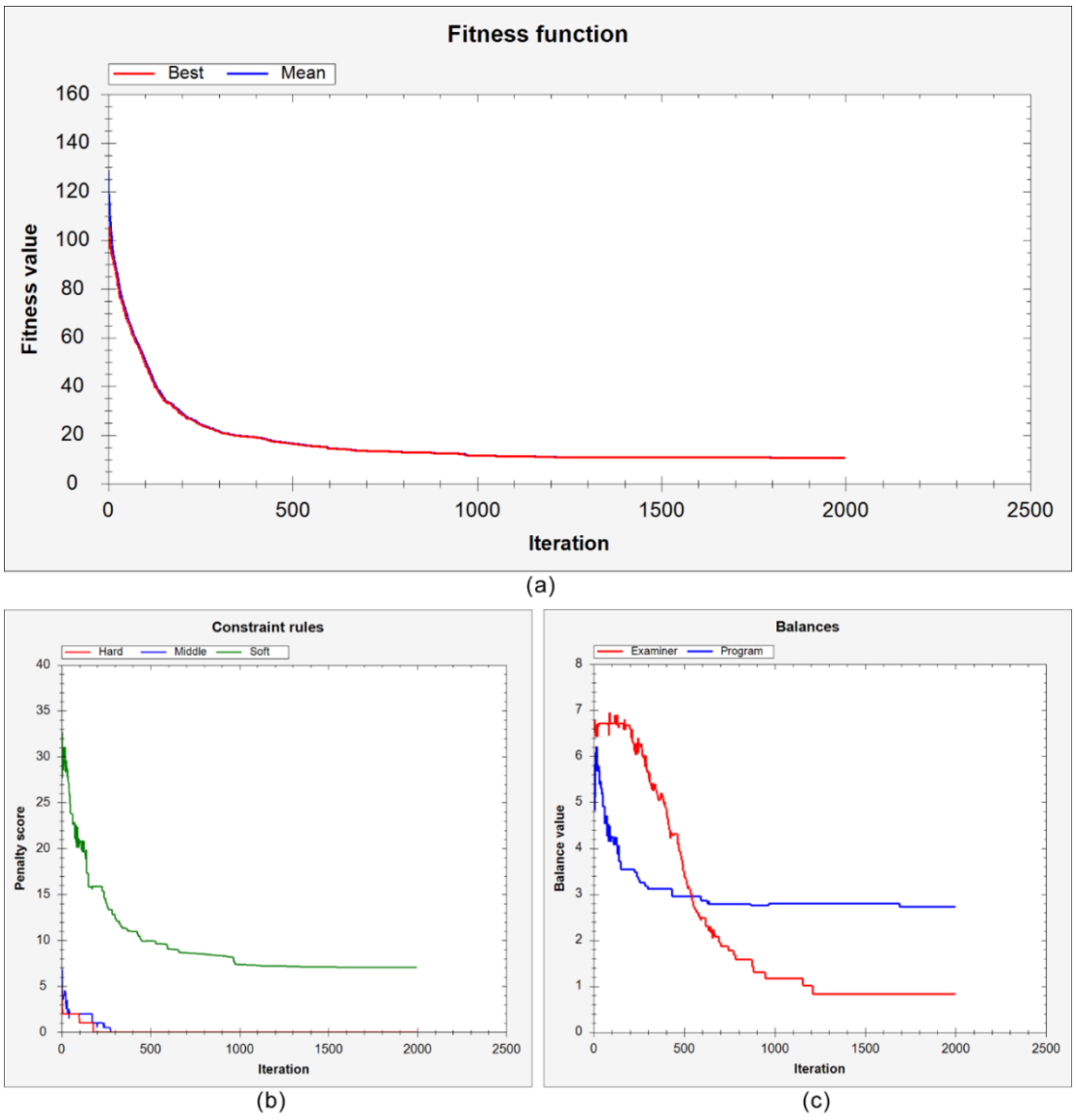

Fig. 10. Exp-3; (a) fitness function, (b) constraint rules, (c) balances.

According to obtained results; the number of exam tasks per examiner is between 8-10; 2 faculty members with 8-10 exams, 33 faculty members with 9 exams. Exams According to the K_{SE} criterion, all exams were placed appropriately for students who took courses from previous periods. According to the K_{PE} criterion, there are 22 exams conducted in the same days in all academic programs. In addition, all of the K_{SD} (for 6 faculty members) and K_{AN} (for 3 faculty members) constraints were

successfully applied. Finally, the exams were placed into early hours with 85% success according to K_{DE} criteria.

3.4. Experiment IV

In this experiment, a total of 306 exam sessions with 35 students in each class were created for the exam branches (tolerance is 10). The desired process in the experiment is that; K_{SE}, K_{PE} and K_{DE} criteria are fulfilled at the specified levels. In addition, faculty members were asked not to take exams task outside their own courses. For Exp-4, fitness, constraint, and balance values in each iteration are shown in Fig. (11).

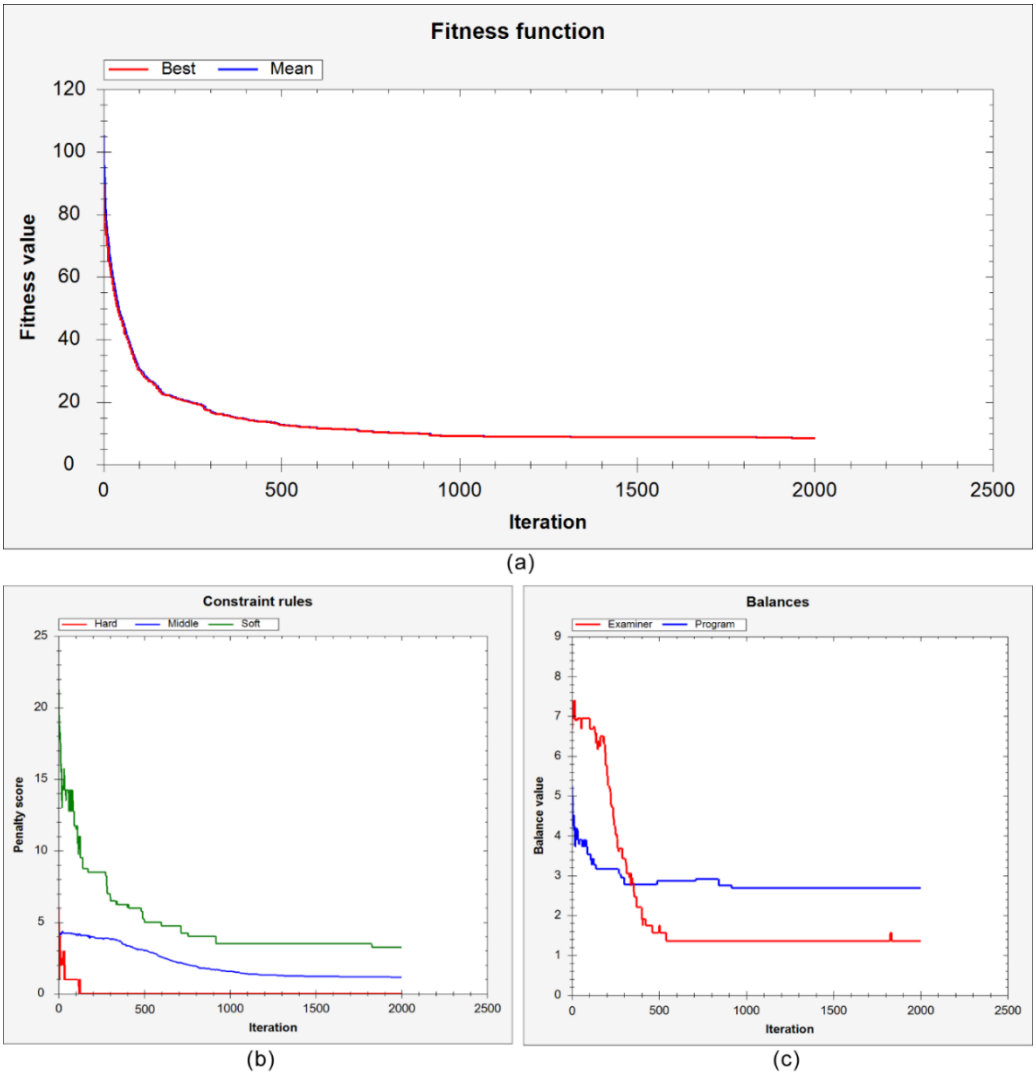

Fig. 11. Exp-4; (a) fitness function, (b) constraint rules, (c) balances.

According to obtained results; the number of exam tasks per examiner (except for 12 faculty members) is between 9-11; six faculty members with 9 exams, 18 faculty members with 10 exams, 1 faculty members with 11 exams. Exams According to the K_{SE} criterion, all exams were placed appropriately for students who took courses from previous periods. According to the K_{PE} criterion, there are 13 exams conducted in the same days in all academic programs. Finally, the exams were placed into early hours with 85% success according to K_{DE} criteria.

3.5. Numerical Evaluation of Experiments

The numerical results obtained from the four experimental studies done above were kept with tables. In the Experimental Results section (Experiments 1 to 4), some values from Prevent overlapping for students who retake a failed course (K_{SE}), Arrange exam schedules according to early in the day" (K_{DE}), No more than one exam per day for the same program (K_{PE}) and I do not want to be tasked before-noon (K_{BN}) were not determined as optional constraints, so no values were specified in the table below. The fitness values obtained from the experimental studies are shown in Table 5.

Table 5. Fitness values of experiments.

Experiments	O_E	O_S	O_C	K_{AN}	K_{BN}	K_{SD}	K_{SE}	K_{PE}	K_{DE}	B_E	B_P	F
Exp-1	0	0	0	0,25	—	0,00	—	—	1,79	1,51	2,96	6,51
Exp-2	0	0	0	0,00	—	0,00	1,00	6,75	—	1,14	2,89	11,78
Exp-3	0	0	0	0,00	—	0,00	0,00	5,5	1,53	0,82	2,72	10,57
Exp-4	0	0	0	0,00	—	0,00	0,00	3,25	1,17	1,35	2,68	8,45

Table 5 shows that there is no overlap even in different rules and situations. That is, the scheduling process was performed 100% accurately without any overlap. Table 6 shows the number of examiners and the number of exams per classroom. Table 7 shows distribution of the number of exams according to calendar dates. Table 8 shows distribution of the number of exams according to the hours intervals.

Table 6. The number of examiners and exams per classroom (exam hall).

	Exp-1		**Exp-2**		**Exp-3**		**Exp-4**	
	Examiner	**Exam**	**Examiner**	**Exam**	**Examiner**	**Exam**	**Examiner**	**Exam**
Min	9	22	11	26	8	24	9	20
Max	11	44	13	42	10	40	11	36
μ	9,46	33,6	11,12	32,4	9	31,8	9,8	29,2
σ	0,56	6,39	0,44	5,46	0,33	4,82	0,5	4,66

Table 7. Distribution of exam numbers according to calendar dates.

Experiments	05	06	07	08	09	10	11	12	13	14	15	16
Exp-1 (350)	42	37	29	29	30	34	—	33	29	29	29	29
Exp-2 (338)	28	31	29	28	29	27	—	29	32	35	32	38
Exp-3 (333)	26	42	28	31	30	29	—	30	28	30	28	31
Exp-4 (305)	32	28	26	28	27	26	—	29	28	27	27	27

Table 8. Distribution of the numbers of exams according to hours intervals.

Experiments	09:00-10:59	11:00-12:59	13:00-14:59	15:00-16:59	17:00-18:59	K_{DE}
Exp-1 (549)	117	109	78	25	21	80%
Exp-2 (528)	68	82	67	64	57	—
Exp-3 (490)	122	104	75	21	11	85%
Exp-4 (486)	101	105	66	14	19	85%

4. CONCLUSIONS

Today, while planning in any branch of business (education, production, transportation, service, finance, etc.), time scheduling problem is encountered. The developed application can be used not only in universities but also in any scheduling distribution. In this respect, the developed application has a dynamic structure, and it can easily be scheduled under different conditions. Academic units at universities prepare exam schedules to make exams at a specific time in the academic education period. The preparation process of the exam schedules is still carried out in many faculties, colleges, and vocational schools with the help of the spreadsheet program such as Microsoft Excel. This process is both difficult and time consuming. According to the findings obtained from the experimental studies conducted in this study, the preparation time of the exam schedules for Karabük University (T.O.B.B. Technical Sciences Vocational School) without any overlap is average 4 minutes and below 2000 iterations. The performance of the proposed method was tested with 421 courses, 54 lecturers, 27 academic programs and 21 classrooms, and the exam schedules were automatically prepared correctly. Also, the proposed method was tested in different situations. Other studies on this issue in the literature are generally based on departments. The proposed method can produce solutions to all kinds of rules and constraints at a faculty and vocational school level. The proposed method accurately creates exam schedules without overlapping, depending on the constraints determined. Thanks to flexible constraint and rule options, any university can perform exam planning. In addition, it is extremely important in terms of time cost compared to traditional methods.

ACKNOWLEDGEMENTS

The dataset in this study has been prepared on basis of the 2018-2019 fall term student information system of the Vocational School of T.O.B.B. Technical Sciences, Karabuk University.

REFERENCES

[1] Elen, A., & Çayıroğlu, İ., (2010), Solving of Scheduling Problem with Heuristic Optimization Approach. Teknoloji (Engineering Science and Technology, an International Journal), 13(3), 159-172.

[2] Soghier, A., & Qu, R., (2013), Adaptive selection of heuristics for assigning time slots and rooms in exam timetables. Applied Intelligence, 39(2), 438–450. https://doi.org/10.1007/s10489-013-0422-z

[3] Çayıroğlu, İ., & Elen, A., (2012), A Heuristic Optimization Approach for A Real-World University Timetabling Problem. Advances in Computer Science and Engineering, 9(2), 103-131.

[4] Brunato, M., & Battiti, R., (2019), Combining intelligent heuristics with simulators in hotel revenue management. Annals of Mathematics and Artificial Intelligence. https://doi.org/10.1007/s10472-019-09651-9

[5] Taheri, G., Khonsari, A., Entezari-Maleki, R., & Sousa, L., (2020), A hybrid algorithm for task scheduling on heterogeneous multiprocessor embedded systems. Applied Soft Computing, 106202. https://doi.org/10.1016/j.asoc.2020.106202

[6] Gantt, H.L., (1910), "Work, Wages and Profit". Engineering Magazine. New York.; republished as Work, Wages and Profits. Easton, Pennsylvania: Hive Publishing Company. 1974. ISBN 0-87960-048-9.

[7] Coit, D. W., & Zio, E., (2018), The Evolution of System Reliability Optimization. Reliability Engineering & System Safety. https://doi.org/10.1016/j.ress.2018.09.008

[8] Saldarriaga, J., Páez, D., Salcedo, C., Cuero, P., López, L. L., León, N., & Celeita, D., (2020), A Direct Approach for the Near-Optimal Design of Water Distribution Networks Based on Power Use. Water, 12(4), 1037. https://doi.org/10.3390/w12041037

[9] Khosravanian, R., Mansouri, V., Wood, D. A., & Alipour, M. R., (2018), A comparative study of several metaheuristic algorithms for optimizing complex 3-D well-path designs. Journal of Petroleum Exploration and Production Technology. https://doi.org/10.1007/s13202-018-0447-2

[10] Crown, W., Buyukkaramikli, N., Sir, M. Y., Thokala, P., Morton, A., Marshall, D. A., Tosh, J. C., Ijzerman, M. J., Padula, W. V., & Pasupathy, K. S., (2018), Application of Constrained Optimization Methods in Health Services Research: Report 2 of the ISPOR Optimization Methods Emerging Good Practices Task Force. Value in Health, 21(9), 1019–1028. https://doi.org/10.1016/j.jval.2018.05.003

[11] Chávez-Bosquez O, Hernández-Torruco J, Hernández-Ocaña B, Canul-Reich J., (2020), Modeling and Solving a Latin American University Course Timetabling Problem Instance. Mathematics 8(10), 1833. https://doi.org/10.3390/math8101833

[12] Santiago-Mozos, R., Salcedo-Sanz, S., DePrado-Cumplido, M., & Bousoño-Calzón, C., (2005), A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a Spanish university. Computers & Operations Research, 32(7), 1761–1776. https://doi.org/10.1016/j.cor.2003.11.030

[13] Dammak, A., Elloumi, A., & Kamoun, H., (2006), Classroom assignment for exam timetabling. Advances in Engineering Software, 37(10), 659–666. https://doi.org/10.1016/j.advengsoft.2006.02.001

[14] Pillay, N., & Banzhaf, W., (2010), An informed genetic algorithm for the examination timetabling problem. Applied Soft Computing, 10(2), 457–467. https://doi.org/10.1016/j.asoc.2009.08.011

[15] Turabieh, H., & Abdullah, S., (2011), An integrated hybrid approach to the examination timetabling problem. Omega, 39(6), 598–607. https://doi.org/10.1016/j.omega.2010.12.005

[16] Shatnawi, A., Fraiwan, M., & Al-Qahtani, H. S., (2017), Exam scheduling: A case study. 2017 Ninth International Conference on Advanced Computational Intelligence (ICACI). https://doi.org/10.1109/icaci.2017.7974498

[17] Keskin, M. E., Döyen, A., Akyer, H., & Güler, M. G., (2018), Examination timetabling problem with scarce resources: a case study. European J. of Industrial Engineering, 12(6), 855. https://doi.org/10.1504/ejie.2018.096394

[18] Güler, M. G., Geçici, E., Köroğlu, T., & Becit, E., (2021), A web-based decision support system for examination timetabling. Expert Systems with Applications, 183, 115363. https://doi.org/10.1016/j.eswa.2021.115363

[19] Aldeeb, B. A., Azmi Al-Betar, M., Md Norwawi, N., Alissa, K. A., Alsmadi, M. K., Hazaymeh, A. A., & Alzaqebah, M., (2021), Hybrid Intelligent Water Drops Algorithm for Examination Timetabling Problem. Journal of King Saud University - Computer and Information Sciences. https://doi.org/10.1016/j.jksuci.2021.06.016

[20] Hao, X., Qu, R., & Liu, J., (2020), A Unified Framework of Graph-based Evolutionary Multitasking Hyper-heuristic. IEEE Transactions on Evolutionary Computation, 25, 1. https://doi.org/10.1109/TEVC.2020.2991717