UNIVERSITY EXAM TIMETABLING USING GENETIC ALGORITHMS

Ozan Akı[1]
[1] *Trakya University*

Abstract

Exam timetabling problems are one of the optimization problems and these are also known as NP-Hard problems. An exam timetabling consist of different constraints and satisfactions from course timetabling. In university exams, one of the significant difference from courses are exams can be done used multiple classrooms while courses use only one. In the exam timetabling, satisfying classes are more important than course timetabling. The number of exams per day, distributing hard exams, optimizing classroom usages are soft constraints but these are important as hard constraints. In this paper, solving an exam timetabling problem for universities has been discussed and examined. Genetic algorithms have been used for the generation candidate solution. Results have been discussed related to defined constraints.

Keywords: exam, timetable, genetic algorithm

INTRODUCTION

Exam timetables are just another optimization problem. These types of optimization problems are classified as NP-hard [1]. Solving by linear programming by testing all possibilities takes excessive time [2]. Exam time tables are similar to the course time tables but some requirements differ. In universities, course time tables must be done before each season. In Turkey, there are usually two semesters, fall and spring. Some universities have an extra summer semester. Course timetables are prepared two or three times per each year. On the other hand, depending on the number of exams in one semester, a typical university in Turkey must have to prepare a minimum of three exam timetables per semester and a minimum of 6 exam timetables per year. Furthermore, exam timetables have some additional complexities from course timetables [1].

Most of the exam timetables in universities are generating manually because each university or department has its unique physical resources and requirements. Generating exam timetables automatically can help the staff that dedicated to making timetables. Also, the resources of universities can be used efficiently.

There are many studies about generating course timetables with different approaches. But there are few papers specifically about exam timetabling.

Al-Hawari and et al. have been used a three-phase integer linear programming (ILP) approach for solving timetable problems [1]
Edis and et al. have been used two-phase integer linear programming to solve exam timetabling problems [3].
Huang and et al. have been proposed a new Memetic algorithm for generating exam timetables [4]. Burke and et al., and Soghier and et al. have been evaluated an adaptive selection of heuristics for improving exam timetables [5, 6]. Mansour and et al. have been used the scatter search technique for exam timetabling [7]. Komar and et al., have been used distributed evolutionary computation for solving exam timetabling [8].

This paper aims to generate real-world exam timetables automatically using genetic algorithms. Experiments have been performed with real-world data and constraints for testing algorithms' performances. For evaluating the generated exam timetable performance, Trakya University Ipsala vocational school exam tables that have been made by manually are used as reference and comparison.

PROBLEM DESCRIPTION

Exam timetabling differs from course timetabling in many respects. An exam

timetable consists of exams in timeslots such as courses. Dependent on the number of students of the exam, one or more classrooms can be used for examination. Also, examinators must be assigned to each dedicated classroom of the exam. In vocational schools, examinators are mostly lecturers. So, lecturers can have two different roles in the examination week. Assigning classrooms and balancing workloads of observers are as important as arranging exams in the timetable.

An exam timetable has two types of constraints. Hard constraints must have fulfilled. A class's exams cannot be in the same timeslot. There must be empty classrooms in the timeslot that total capacities meet the number of students. These are some examples of hard constraints.

Soft constraints improve the satisfaction of the exam timetable. A balanced distribution of class's exams over days, preventing hard exams arranged successively, spare timeslots for lecturers and students, lecturer preferences are some examples of soft constraints.

When hard constraints are once met, soft constraints will increase the satisfaction of the timetable in respect of defined conditions [2].

In this paper, hard constraints are defined as follows:

- Exams of the same class cannot be in the same timeslot
- A lecturer cannot have more than one exam in the same timeslot unless there is a combined exam.
- A lecturer cannot assign as an observer if there is own exam in the same timeslot.
- An exam cannot be assigned to the timeslot when there is no available classroom to meet the number of students in this exam.

Soft constraints are defined as follows:

- A balanced distribution of exams of classes along exam days.
- Avoiding assigning hard exams on the same day for any class.
- Selecting the fittest capacity of the combination of classrooms (chose classrooms that total capacity should closer to the number of students)
- Distribute workloads of the observers evenly.
- Try to maximize lecturers total coefficiency for each timeslot
- Try to maximize classes total coefficiency for each timeslot

The algorithm aims to fulfill the hard constraints first and then maximize the overall soft constraints' score.

METHODOLOGY

Genetic Algorithms have been used to solve optimization problems such as timetables. Core elements of genetic algorithms are genomes. Chromosomes are consist of genes and have all information about any candidate solution. A population consists of a certain number of chromosomes. Genetic operators, crossover, and mutation apply over to some selected chromosomes from the population. The new generation obtaining by adding new modified chromosomes to the current population. These steps recurring over and over until reached the threshold level.

In this study, chromosome length is determined by the number of exams. Each genome is consists of a timeslot, a list of classroom indices, and an observers indices list for each classroom index. A section of the exam chromosome is shown in figure 1.

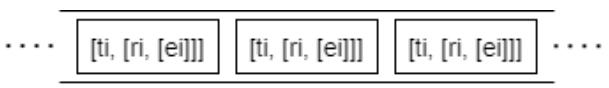

Fig. 1. *A section of an exam chromosome*

In figure 1, *ti* is the timeslot index that indicates the day and period in the timetable. *ri* is classroom index in timeslot *ti*. timeslot index *ti* is calculated from day *d* and period *p* as shown by formula 1.

$$ti = d \cdot N_p + p \qquad (1)$$

In formula 1, N_p is the number of periods per day. By using this form of representation, a 1-dimensional chromosome has been obtained. Thus, genetic operations and fitness calculations on a chromosome have been simplified. Recurred lists in each genome seem to increase calculation iteration but inner lists mostly will not have items more than a few.

The overall fitness value has been calculated as weighted sums of fitnesses of lecturers L_f, classes C_f, classrooms R_f, *and* observers O_f. Chromosome fitness value K_f is calculating as formula 2.

$$K_f = \omega_l L_f + \omega_c C_f + \omega_r R_f + \omega_e O_f \qquad (2)$$

Lecturers, classes, classrooms, and observers can have different weights in the sum of fitness calculation dependent on the importance of objectives.

Lecturers' fitness values are calculating from coefficient values for each timeslot entered the database along with lecturers' data. These coefficients are can vary between -1.0 to 1.0. These coefficients are indicated the lecturers' happiness for each timeslot. Negative values represent the ratio of unhappiness while positives ones are happiness. Zero value indicates the neutral or doesn't care state. L_h lecturer fitness value is calculating as formula 3.

$$L_f = 1 - \frac{\sum_{i=0}^{Nts} L_{tci}}{\sum_{i=0}^{Nts} X_{tpi}} \qquad (4)$$

In this formula, L_f is the lecturer fitness value, N_{ts} is the number of timeslots in the exam table, L_{tci} is the lecturer's happiness coefficient value for the assigned exam in timeslot i and has been shown in formula 5. X_{tpi} is the coefficient values assigned exams for timeslot I and has been shown in formula 6.

$$L_{tci} = \begin{cases} L_{tci}, & L_{tci} \geq 0 \\ 0, & L_{tci} < 0 \end{cases} \qquad (5)$$

$$X_{tpi} = \begin{cases} X_{tpi}, & X_{tpi} \geq 0 \\ 0, & X_{tpi} < 0 \end{cases} \qquad (6)$$

The fitness value of a class is dependent on a balanced distribution of exams along days in the exam timetable. The value of C_h is calculating as in formula 7.

$$C_f = 1 - \frac{H_{cb}}{\sum_{i=0}^{Nd} \left| X_{ni} - \frac{N_x}{N_d} \right|} \qquad (7)$$

In formula 7, N_d is the number of days in the exam timetable, N_x is the total number of the exam of class, X_{ni} is the number of exams in the day i for class. H_{cb} is the best exam arrangement value.

Classrooms' fitness values are related to its fullness rate and calculated as formula 8.

$$R_f = \frac{R_{tu}}{N_r} \cdot 1 - \frac{N_s}{\sum_{r=0}^{Nr} R_{cr}} \qquad (8)$$

In this formula, R_f is a classroom fitness value. R_{tu} is the number of used classrooms for the exam. N_r is the total number of classrooms. N_s is the number of students for the exam. R_{cr} is the total capacity of assigned classrooms for the exam.

Observers' fitness values depend on the equality of workloads and are calculated by formula 9.

$$O_f = \frac{\sum_{e=0}^{Ne} \frac{O_{tp}}{N_e} - O_{tpi}}{O_{tp}} \qquad (9)$$

In formula 9, O_f is the fitness value of observers. N_e is the number of the observer (also it is the number of lecturers if all lecturers are observers). O_{tp} is the total number of tasks. O_{tpi} is the observer's number of tasks.

The genetic algorithm flowchart is shown in figure 2. By this flowchart, the initial population is creating first. Created new chromosomes always fulfill the hard constraints thanks to the chromosome creation algorithm prevents the violating of hard constraints. Thus, the calculation of fitness value represents only soft constraints' fulfillment. The genetic algorithm tries to minimize total fitness value K_h. The selection of chromosomes from the population is based on the elitism approach. Only chromosomes that have the best fitness values are selecting for the mating pool. Then doing crossover and mutation genetic operations over mating pool chromosomes. Then modified mating pool chromosomes' fitness values are calculating for selection and generation new generation. Only chromosomes that have better fitness value from chromosomes in the population are transferred to the new population. The algorithm process begins again until reaching

the threshold fitness value or maximum iteration limit.

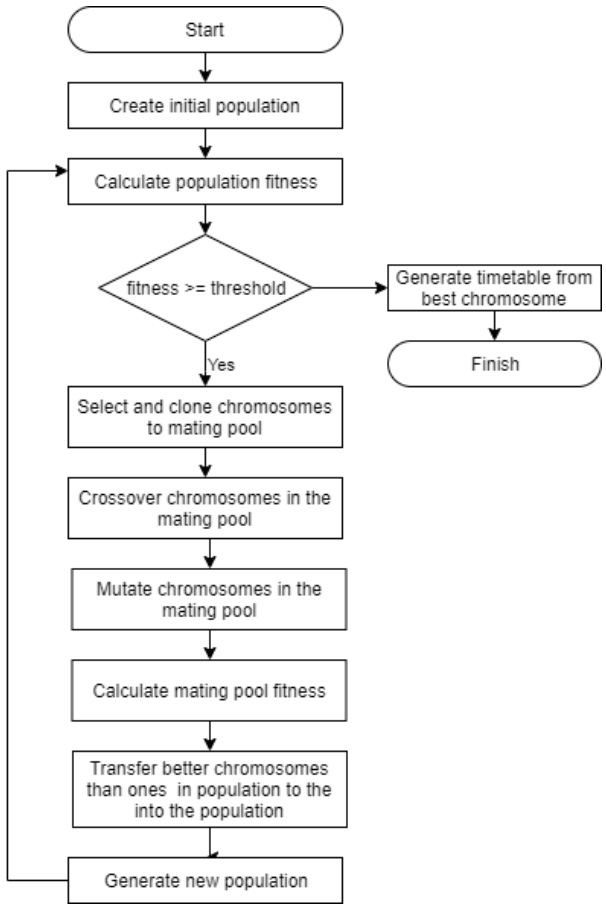

Fig. 2. *Flowchart of overall genetic calculations*

After reaching the fitness threshold value or iteration limit, the chromosome that has the best fitness value is selecting for the solution and converted to the exam timetable then saved as an HTML formatted file.

EXPERIMENTS

In the experimental phase of this study, all of the genetic algorithm functionalities are coded in Python. Software classes have been used as data structures to manage data in algorithms. Each of the lecturers, classes, exams, classrooms have been storing within individual objects with their related data. These objects and object fields have been shown in figure 3.

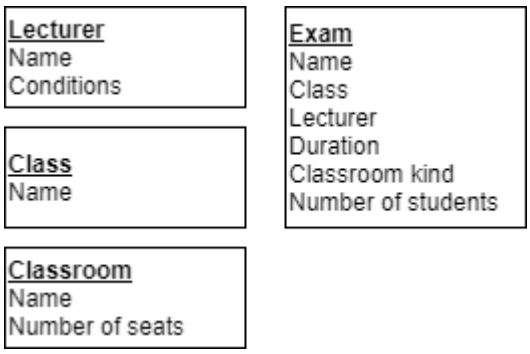

Fig. 3. *Classes of data structures*

All objects are created from stored text databases automatically. All needed data has been entered manually from a real-world exam timetable had been used in Ipsala Vocational School. The experimental timetable information is shown in table 1.

Table. 1. *Exam timetable information*

Property	**Value**
Number of exams	106
Number of lecturers	20
Number of classrooms	12
Number of classes	12

RESULTS

Some test results with parametric variations are shown in table 2.

Table. 2. *Results with parametric variations*

	Experiments					
	1	**2**	**3**	**4**	**5**	**6**
Population Size	30	30	30	40	60	100
Mating Poll Size	10	10	10	20	20	50
Crossover Rate (%)	1.89	5.66	11.32	1.89	1.89	1.89
Mutating Rate (%)	0.94	1.89	3.77	0.94	0.94	0.94
Working Duration	3:01	3:01	3:00	5:20	5:20	12:18
Fitness	0.180	0.180	0.277	0.178	0.180	0.179

In the first three experiments, population and mating pool sizes are fixed while crossover and mutation rates have been increased. In experiments 4, 5, 6 crossover and mutation rates are fixed while population and mating pool sizes have been increased.

By the limit of 500 iterations, the best fitness value is about 0.180 has been obtained. When the results are examined, it seems that increasing rates of crossover and mutation worsen fitness. Also, increasing the population and mating pool sizes did not affect the fitness value, but the duration of the study increased proportionally. As a result of evaluating experiment results, experiments 1, 2, 4, 6 have the best fitness values. But when evaluating for the working duration, experiments 1 and 2 have got minimum working durations.

The fitness graph that has been drawing during the calculating generations for experiment 1 is shown in figure 4.

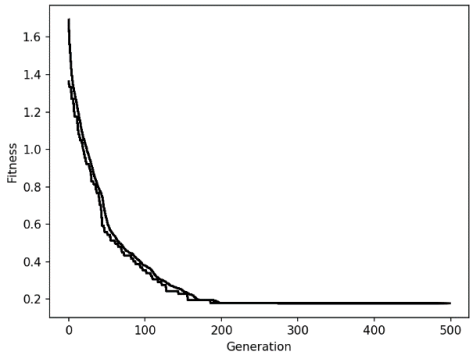

Fig. 4. *Fitness graph of experiment 1*

There are two curves in figure 4. The upper curve showing the mean fitness values in population over generations. The lower curve showing the best fitness values of the population over generations. The gap between curves shows the deviation between the mean and best fitness value. The narrow gap as in figure 4 means that all fitness values of chromosomes in the population are not far from the best chromosome fitness value.

When the generated timetable is examined, it seems that all lecturers' happiness coefficients have been fulfilled. Also, lecturers' workloads as observers for exams has been distributed smoothly. There are only ±2 differences in workloads between observers' tasks. Classes' exams distribution along days in the timetable has not been good as expected. It was observed that most of the classes had one spare day on the timetable, while there is more than one exam on the other days. Classroom fitnesses have been mostly filled by 73% to 97% fullness rates. But the number of usages of classrooms is unbalanced. One classroom has been assigned 19 times but another one only one. It may be considered to change the weight value of the classroom in fitness calculation or to rearrange the classroom fitness formula.

CONCLUSION

Experiments are showing that an automatically generated exam timetable with real-world data is almost ready for usage by some modification by hand. Also, these exam timetables have been generated within a relatively very short time such as under 5 minutes. With the improvement of the genetic algorithm, it has been observed that it is possible to obtain outputs very close to the manually placed timeline.

REFERENCE

[1] Al- Hawari, F., et al., *A practical three- phase ILP approach for solving the examination timetabling problem.* International Transactions in Operational Research, 2020. **27**(2): p. 924.

[2] Febrita, R.E. and W.F. Mahmudy. *Modified genetic algorithm for high school time-table scheduling with fuzzy time window*. in *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*. 2017. IEEE.

[3] Rahime Sancar, E. and E. Emrah Bünyamin, *Gerçek bir sınav çizelgeleme problemi için iki aşamalı çözüm yaklaşımı.* A two-phase solution approach for a real-life examination timetabling problem, 2020. **25**(1): p. 71-81.

[4] Huang, W., G. Yi, and S. He, *Memetic algorithm and its application to the arrangement of exam timetable.* Statistics, Optimization and Information Computing, 2016. **4**(2): p. 147-153.

[5] Burke, E., R. Qu, and A. Soghier, *Adaptive selection of heuristics for improving exam timetables.* Annals of Operations Research, 2014. **218**(1): p. 129-145.

[6] Soghier, A. and R. Qu, *Adaptive selection of heuristics for assigning time slots and rooms in exam timetables.* Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, 2013. **39**(2): p. 438.

[7] Mansour, N., V. Isahakian, and I. Ghalayini, *Scatter search technique for exam timetabling.* Applied Intelligence, 2011. **34**(2): p. 299.

[8] Komar, M., D. Grbic, and M. Cupic, *Solving exam timetabling using distributed evolutionary computation*. 2011. p. 301-306.