University Course Timetabling Using Genetic Algorithms

Conference Paper · November 2020

CITATIONS
2

READS
1,145

1 author:

Ozan Akı
Trakya University
26 PUBLICATIONS **58** CITATIONS

SEE PROFILE

UNIVERSITY COURSE TIMETABLING USING GENETIC ALGORITHMS

Ozan Akı[1]
[1] *Trakya University*

Abstract

Course timetabling problems are one of the optimization problems and these are also known as NP-Hard problems. A Course timetabling problem consists of many parametric optimizations to be considered. Due to the complexity of optimization, many timetabling problems are done manually by one of an experienced person. In this paper, solving the timetabling problem for universities has been discussed. Genetic algorithms have been used for the solution. Both hard and soft constraints have been taken into account. Hard constraints are ensured for conflicts while soft constraints are satisfied lecturers, classes, classrooms. Results have been discussed and compared with hand-made course time tables previously done with the same constraints.

Keywords: courses, timetable, genetic algorithm

INTRODUCTION

Timetables are used in planning the events in various organizations. The difficulty of generating timetables varies depending on the number of facts used in the timetables and the needs of the organizations. Generating schools or universities course time tables are one of these problems. These types of problems knowns as optimization problems and these are classified as NP-hard [1-3]. It cannot be solved through linear programming. Testing all possibilities takes nearly infinite trials and takes excessive time [4]. For this reason, timetables are generating either manually by humans or by heuristic programming methods.

Most of the timetables in universities are generating manually because each organization has its unique requirements. Or using specialized custom software (most of its own software) for doing it. There is no general-purpose timetabling application in the market that will meet all the requirements of all universities.

There are many studies about generating course timetables with different approaches. Genetic algorithms [2, 5-7], genetic and ant colony algorithms [8], genetic and simulated annealing algorithm [9, 10], genetic and hill-climbing algorithm [1, 11], quantum genetic evolutionary algorithm [12], adapted flower pollination algorithm [13], population-based incremental learning algorithm [14], localized island model genetic algorithm [15] are algorithms used in studies about solving university timetabling problem. Genetic algorithms and their derivatives have been widely used to solve the time scheduling problem.

This paper aims to generate real-world timetables automatically using genetic algorithms. Experiments on this paper have the same constraints as real-world timetable constraints and the performance of the algorithm has been evaluated with manually generated one with having the same constraints. The real-world test data and constraints were obtained from Ipsala Vocational School, Trakya University in Turkey. Experimental results were promising, with placements fairly close to the manual placement timetables.

PROBLEM DESCRIPTION

Timetabling the courses in universities has many constraints. Courses, lecturers, classes, classrooms, and timeslots are the main actors of these constraints. These must not be in the same timeslot more than once. In another saying, only one course of class with one teacher in a room must be in a single timeslot. [4, 9]. These constraints are also called as hard constraints and cannot be violated [4, 16].

Constraints other than hard constraints are non-essential and are called as *soft constraints [4]*. These are like smooth distribution of class's courses, some free periods for classes and lecturers, used number of classrooms and effective usage of classrooms' capacity and labs, classroom changes between courses, wishes of lecturers, or even walking distances or considering daylight effects related to the orientation of school building and location of classrooms. The last twos are extreme examples of soft constraints. Soft constraints are entirely dependent on school-specific requirements.

When hard constraints are once met, soft constraints will increase the satisfaction of the timetable in respect of defined conditions [4].

In this paper, four simple hard constraints are defined as followed

- Course duration must fit in same day periods. Cannot split between days.
- Courses of the same class cannot be in the same timeslot
- A lecturer cannot take more than one lesson in the same timeslot unless there is a combined lesson.
- A course cannot be in a timeslot when there is no available classroom met the number of students in this course.

Soft constraints are defined as follows:

- Selecting the fittest (chose classroom that the number of seats in a classroom should closer to the number of students) classroom if possible.
- 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 and then maximize overall soft constraints' coefficiencies.

METHODOLOGY

Genetic Algorithms has been used to generate possible optimal timetable solution. Core elements of genetic algorithms are chromosomes that have all information about one candidate solution. Each chromosome consists of genomes that smallest unit of the chromosome. A population has 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 courses. Each genome is consists of a timeslot and classroom tuple. A section of the chromosome is shown in figure 1.

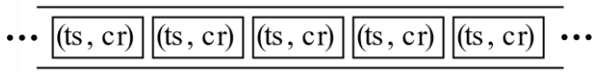

Fig. 1. *Representation of chromosome*

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

$$ts = d \cdot Np + p \qquad (1)$$

In formula 1, N_p is the number of periods per day. By using this form of representation, a 1-dimensional chromosome was obtained for a 2-dimensional table. Thus, genetic operations and fitness calculations on a chromosome have been simplified.

In this study, fitness value is calculated by sums of weighted fitnesses of lecturers L_f, classes C_f, and the classrooms R_f. Chromosome fitness value K_f is calculating by formula 2.

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

Lecturers' fitness values are calculated by using coefficients for each timeslot. Each coefficient for a specific timeslot is representing the emotion of the timeslot if a course is assigned here. A typical coefficient is a real number that can have from -1.0 to +1.0 range and shown in formula 3

$$-1.0 \leq ts_c \leq 1.0, ts_c \in \mathbb{R} \qquad (3)$$

The negative values of coefficients represent the degree of unhappiness. Positive values of coefficients represent happiness degrees. Zero value indicates the neutral or doesn't care state.

These coefficient values have been entered in the timetable database for each lecturer for

each timeslot. For example, If a lecturer prefers to do his courses in early periods, coefficients in early periods can be set as positive values. Vise versa, If a lecturer wants to have spare periods or day, related coefficients can be set as negative values. Lecturers' happiness level calculation is shown in formula 4.

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

In this formula, L_h is the lecturer fitness value, N_{ts} is the number of timeslots in the course timetable, L_{tci} is the lecturer's happiness coefficient value for the assigned course in timeslot *i.*

The fitness value of a class is dependent on a balanced distribution of courses along days in the course timetable. The value of *Cf* is calculating as in formula 5.

$$C_f = 1 - \frac{H_{cb}}{\sum_{i=0}^{Nd} \left| X_{ni} - \frac{N_c}{N_d} \right|} \quad (5)$$

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

Classrooms' happiness is related to its fullness rate as shown in formula 6.

$$R_f = 1 - \frac{N_s}{R_c} \quad (6)$$

In this formula, *Rf* is a classroom fitness value. N_s is the number of students for the course. R_c is the capacity of the assigned classroom for the course.

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 maximize fitness value from negative values to the maximum possible positive value. 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.

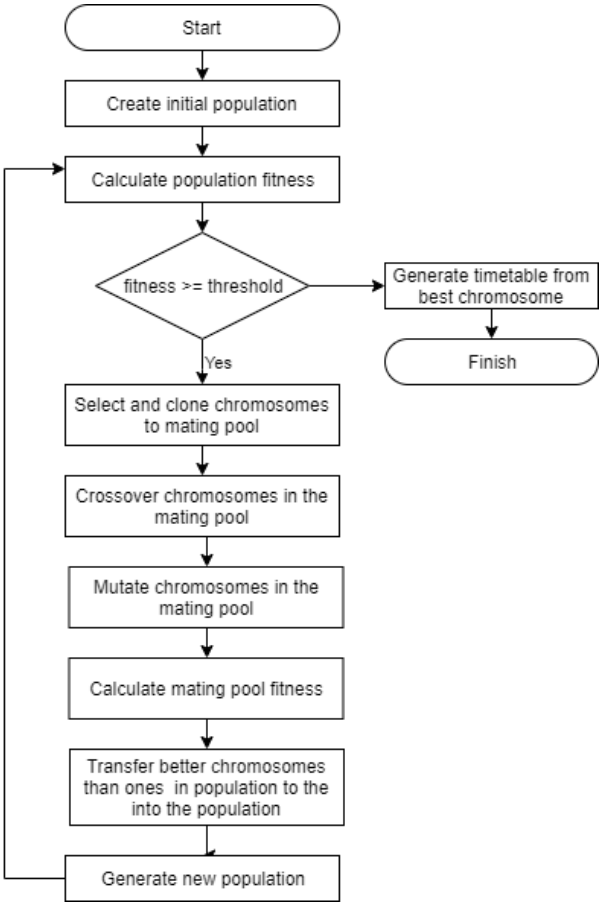

Fig. 2. *Flowchart of overall genetic calculations*

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 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, courses, classrooms have been storing within individual objects with their related data. These objects and object fields have been shown in figure 3.

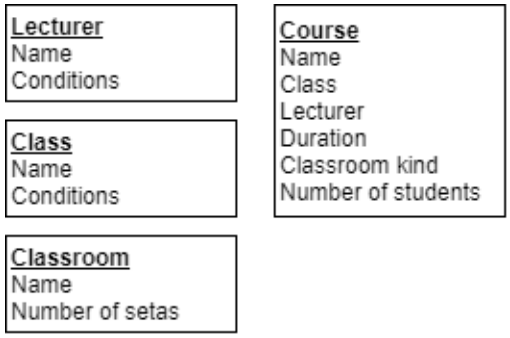

Fig. 3. *Classes of data structures*

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

Table. 1. *Timetable information*

Property	**Value**
Number of courses	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	5.66
Mutating Rate (%)	0.94	1.89	3.77	0.94	0.94	1.89
Working Duration	2:36	3:16	3:39	4:27	4:35	13:00
Fitness	0.278	0.497	0.858	0.250	0.334	0.369

Population and mating pool sizes remain unchanged while crossover and mutating rates have been changed in the experiments 1, 2, and 3. Population size and mating pool size are increased in experiments 4, 5, and 6. By inspecting the results, it has appeared that increasing crossover and mutating rates worsen the fitness. Increasing population and mating pool sizes with different rates have not significantly affected fitness value. Population and mating pool sizes have directly affected working duration. By evaluating fitness values along with working duration, there is no significant difference between better fitness values while working durations differ widely. As a result of evaluating experiment results, experiment 4's working parameters should be considered for generating timetables.

The fitness graph that drew during calculating generations for experiment 4 is shown in figure 4.

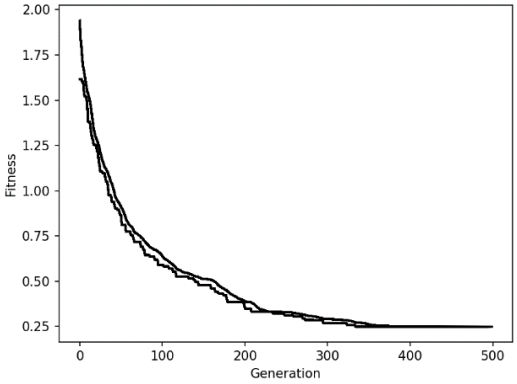

Fig. 4. *Fitness graph of experiment 4*

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 comparing the generated timetable and hand-made timetable with the same data and constraints show that, 44 of 106 courses are placed on the same day. 11 of 19 lecturers have a positive happiness score. But most lecturers have courses on the same day both in generated and hand-made timetables. 69 of 106 courses have a positive happiness score. By the overall consideration, the generated table can be used in real-world planning with some minor modifications by hand.

CONCLUSION

Experiments are showing that an automatically generated timetable with real-world data is almost ready for usage. Also, these timetables have been generated within a relatively very short time such as under 5 minutes. This genetic algorithm code can be converted to an application with some improvements and a user-friendly interface for supporting the person that making the timetables.

REFERENCE

[1] Yusoff, M. and N. Roslan. *Evaluation of Genetic Algorithm and Hybrid Genetic Algorithm-Hill Climbing with Elitist for Lecturer University Timetabling Problem*. in *International Conference on Swarm Intelligence*. 2019. Springer.

[2] Rezaeipanah, A., Z. Abshirini, and M.B. Zade, *Solving University Course Timetabling Problem Using Parallel Genetic Algorithm.* 2019.

[3] Jain, A., S. Jain, and P. Chande, *Formulation of genetic algorithm to generate good quality course timetable.* International Journal of Innovation, Management and Technology, 2010. **1**(3): p. 248.

[4] 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.

[5] Wiilams, K. and M. Ajinaja, *Automatic Timetable Generation Using Genetic Algorithm.* 2019.

[6] Modibbo, U., et al., *Genetic Algorithm for Solving University Timetabling Problem.* Amity Journal of Computational Sciences (AJCS) 3 (1), 43, 2019. **50**.

[7] Charan, S.S. and T. Nadu, *Timetable Generation–An optimal solution to the multi-constrained problem.* 2019.

[8] Mauluddin, S., I. Ikbal, and A. Nursikuwagus, *Complexity And Performance Comparison Of Genetic Algorithm And Ant Colony For Best Solution Timetable Class.* Journal of Engineering Science and Technology, 2020. **15**(1): p. 278-292.

[9] Hambali, A.M.O., Y.A. Dalhatu, M., *Automated University Lecture Timetable Using Heuristic Approach.* Nigerian Journal of Technology, 2020. **39**(1): p. 1-14.

[10] Gülcü, A. and C. Akkan, *Robust university course timetabling problem subject to single and multiple disruptions.* European Journal of Operational Research, 2020. **283**(2): p. 630-646.

[11] Roslan, N.B., *Lecturer Timetable Optimizer Using Genetic Algorithm with Hill Climbing Optimization Method (LETO 2.0).* 2019.

[12] Shuguang, L. and B. Lin. *Research on Complex Curriculum Arrangement Problem Based on Novel Quantum Genetic Evolutionary Algorithm*. in *2019 Chinese Control And Decision Conference (CCDC)*. 2019. IEEE.

[13] Sapul, M.S.C., R. Setthawong, and P. Setthawong. *Adapted Flower Pollination Algorithm for Lecturer-Class Assignment*. in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*. 2019. IEEE.

[14] Rao, M.S., K.L. Prasad, and P. Anusha, *Automatic Timetable Generation Using PBIL Algorithm.* i-Manager's Journal on Information Technology, 2019. **8**(2): p. 31.

[15] Gozali, A.A., et al., *Solving university course timetabling problem using localized island model genetic algorithm with dual dynamic migration policy.* IEEJ Transactions on Electrical and Electronic Engineering, 2019. **15**(3): p. 389-400.

[16] Boonyopakorn, P. and P. Meesad, *A Hybrid Immune Genetic Algorithm to Solve University Time Table Problems.* Walailak Journal of Science and Technology (WJST), 2017. **14**(10): p. 825-835.