Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 126 (2018) 899–906

Procedia Computer Science

www.elsevier.com/locate/procedia

22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem

Maram Assi, and Bahia Halawi, and Ramzi A. Haraty*

Department of Computer Science, Lebanese American University, Beirut, Lebanon
Email: {maram.assi@lau.edu, bahia.halawi@lau.edu, and rharaty@lau.edu.lb)

Abstract

The Timetable Problem is one of the complex problems faced in any university in the world. It is a highly-constrained combinatorial problem that seeks to find a possible scheduling for the university course offerings. There are many algorithms and approaches adopted to solve this problem, but one of the effective approaches to solve it is the use of meta-heuristics. Genetic algorithms were successfully useful to solve many optimization problems including the university Timetable Problem. In this paper, we analyse the Genetic Algorithm approach for graph colouring corresponding to the timetable problem. The GA method is implemented in java, and the improvement of the initial solution is exhibited by the results of the experiments based on the specified constraints and requirements.

Keywords: University Timetable Problem, m eta-heuristics, genetic algorithms, and graph coloring.

1. INTRODUCTION

The university timetable problem is composed of a set of courses offered during the academic semester, a set of instructors teaching specific courses, a fixed number of room location and a list of students registered to a number of courses.

* Corresponding author. Tel. *E-mail address:* rharaty@lau.edu.lb

1877-0509
10.1016/j.procs.2018.08.024

The problem is about assigning all of this input into available timeslots. However, the timetable must satisfy a set of requirements and the solution must be clash-free where all the offered courses are scheduled. This paper discusses this problem and proposes an approach to find a near optimal solution that minimizes the clashes in a proposed timetable.

The University Timetabling problem is known to be NP and a hard optimization problem [1]. It cannot be solved in polynomial time by a deterministic algorithm. The input is large and it includes the list of students registered in the courses, the list of instructors teaching the offered courses, a list of the physical rooms with its capacity, the available timeslots and all the courses. In addition, there are a number of hard and soft constraints to be satisfied. Hard constraints are very sensitive in the sense that the absence of one such constraint can lead to the failure of the scheduler. Whereas soft constraints come at a rather more flexible level and can be treated with some tolerance at the time of failure. Our objective is to satisfy every single hard constraint while minimizing the violations to the soft ones as much as possible. A feasible solution is any possible assignment of courses (each with its instructor and student identifiers) to a possible timing and location without violating any of the hard constraints [2].

Our goal is to have the near optimal solution. Scheduling a timetable can be represented as a graph colouring instance. Each course will be represented by a node. Any conflict between two courses is represented by an edge connecting the nodes corresponding to the classes in the graph. So if two nodes are connected then the courses they stand for are conflicting and cannot be assigned to the same timeslot. Two classes are conflicting if a student is registered in both of the classes, or if the classes are co-requisites and should be registered together in the same semester by a student.

The list of hard constraints includes:

- No student can attend more than one lecture at the same time *(penalty of 10000)*.
- No lecturer can teach more than one lecture at the same time *(penalty of 10000)*.
- No room can be assigned a lecture with more students than its capacity *(penalty of 10000)*.
- Some courses are co-requisite for others; therefore, they must take place in different slots *(penalty of 10000)*.

The list of soft constraints includes:

- For each student, the major and minor required courses should be scheduled in non-overlapping slots *(penalty of 20)*.
- A student does not have has a single course on a day *(penalty of 30)*.
- Each instructor must teach at most three classes in a day *(penalty of 40)*.

The problem data consists of courses with the set of registered students and to be taught by a specific instructor, classrooms with a capacity, and timeslot with a fixed number of study days in a week (usually five from Monday till Friday). Each day is equally split into a fixed number of times slots (usually eight).

The rest of the paper is organized as follows: section II presents the different possible approaches to solve the course offering problem. In section III, we illustrate the proposed algorithm and representation, and we provide an interpretation of the results in section IV. Finally, section V sums up the main ideas discussed in this paper, and presents future direction.

2. LITERATURE REVIEW

Different approaches have been proposed to solve the university timetable problem. These methods include cluster algorithms [3], sequential algorithms [4], constraint-based [5, 6] approaches, and meta- heuristics [7, 8, 9].

Cluster approach to the course timetabling problem dates back to 1978 where it was modeled by Desroches et al [10]. The solution usually can be decomposed into three phases: grouping the timeslots to come up with a feasible solution, evaluating existing permutations of timeslots in an attempt to reduce conflicts of order two, and then trying further improvement to the quality of the solution obtained so far.

Although simple, and seem to be effective in reaching out for feasible solutions, sequential algorithms are hybridized with other approaches in order to improve their efficiency. Among the most widely adopted sequential methods is the graph coloring approach. Graph coloring has been among the widest approaches to solving the course timetabling problem due to the similarity in modeling between the two [11]. Basically, the vertices or nodes stand for courses and the colors assigned to each represents the timeslot chosen. Obviously, edges between nodes represent the existence of clashes or conflicts. The allocation criterion is to exploit coloring to remove conflicts between allocated courses. Broder (1964) is considered to be the oldest in that field where he worked on minimizing violations rather than eliminating them completely so once a color caused a conflict, the color would be chosen to minimize the penalty and not eliminating [12]. Border, remarkably used the "largest degree first coloring method". Later on, Wood [13] used matrices to improve the coloring approach and he was followed by Matula, Marble and Isaacson [14]. With time, improvements emerged from different scientists and researchers. The problem; however with sequential heuristics, is that they do not lead to high quality solutions and again researchers tried improving this issue by using hybridization [15, 16].

A wide range of scientists adopted this method in their implementation where the problem is treated as a collection of variables within a limited containing domain. Then values or numbers are associated with values relative to their violations to constraints. It is important to note that the solutions attained by constraint-based approaches alone are usually hard to improve. That is why they are used in hybridization methods along with other local search approaches. Merlot et al (2003) [16] used constraint programming to generate a feasible timetable and then they combined it with local search.

Meta heuristics are usually inspired by natural phenomena and they are remarkably gaining interest in solving optimization problems as they are promising in terms of finding good solutions without intensively exerting computation efforts. When it comes to the university timetabling problem, different approaches have been proposed in order to solve the problem. Usually, the general approach is to generate an initial solution using a regular heuristic and then use meta heuristics methods to improve the solution. Authors of [17] used swarm optimization to satisfy the hard constraints and minimize violations to soft ones. Nonetheless, they tried to hybridize their work by using Particular Swarm Optimization (PSO) local search and hybrid PSO constraint based reasoning. In another attempt, Al Milli [18] used genetic algorithms and simulated annealing together in solving the problem whereas Bhatt and Sahajpal [19] relied on genetic algorithm hybridization only. Many other researchers tried using different approaches in genetic algorithms such fuzzy genetic algorithm and combinations of genetic algorithms with other search methods [20].

3. GENETIC ALGORITHMS AND GRAPH COLORING

Genetic algorithms (GA) are optimization approaches inspired by the biological evolution. They are very effective in solving complex problems. The main idea behind GA is to start with an initial population and to generate a new population using genetic operators like the selection, crossover and mutation. In this paper, we will be using genetic algorithms to serve as the metaheuristic for approaching the university Timetable Problem. The general algorithm is:

```
population = randomly generated chromosomes; while (terminating condition is not reached) {
runGA();
}

// a single run of a genetic algorithm function runGA() {
parents = getParents();
child = crossover(parents);
child = mutate(child); population.add(child);
}
```

In our algorithm, we start from infeasible timetables, and try to get feasible ones. The initial population is a random assignment of timeslots (day and hour) to each class. Room assignment is done in such a way where the room capacity fits the total number of registered students in a class.

The main goal behind GA is to improve the overall fitness of the population. A generation is usually composed of a population or collection of solutions. Moreover, each single individual is a particular possible solution in the search space. Solutions are supposed to go through a phase of evaluation and the fittest is to survive. Usually, the fitness value of the parents is an indicator to how probable it is to obtain offspring from them. The successive generation will become more suited to their environment. This can be done by selecting the best parent among the population and let them undergo a crossover mutation that will lead to better offspring. The above process is repeated until a termination condition is met or upon a certain number of generations reached.

Different representation and data structures can be used to represent a timetable. Each of these representations can have its own advantages and disadvantages in terms of memory and performance of GA. A timetable is a chromosome. One possible representation could be a matrix where each cell corresponds to a room and a timeslot, and classes are assigned to each slot. This type of representation introduces the possibility of assigning the same class to two different timeslot. In their paper, Sigl, Golub, and Mornar [21], explain how they use a three-dimensional structure to represent the timetable where days (x-axis), timeslots (y-axis) and rooms (z-axis) are the dimension of the timetable. In our proposed algorithm, the timetable is an array of classes, where each class stores information about the instructor, the room and the timeslot. This is depicted in figure 1. This representation ensures that each course will appear only once. Hence, for each cell of the array is a gene. Consequently, each gene corresponds to a specific course that should be scheduled. Thus, an object having instruction id, room id and a pair of day and timeslot is assigned to each class. By choosing this representation, we reduced one of the important violations that is scheduling a single course in different slots.

Under this representation each course can be assigned one of the R available rooms and one of the T slots. We mean by T the product of number of day and the number of times available. For example, if classes can be schedules from Monday till Friday and from 9 am till 4 pm then the number of available slots would be $5 \times 8 = 40$ different slots. Since we have R rooms, then there are R different courses schedules at slot t at the same time.

Having N courses, the total search space is: N x R x T

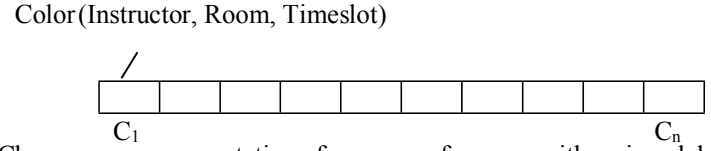

Figure 1: Chromosome representation of an array of courses with assigned data.

Moreover, the use of graphs will make the conflict detection easier. For this purpose, we deployed the idea of graph coloring for checking out for conflicts. We used a graph to represent the classes and the corresponding conflicts. Classes represent the nodes, and edges stand for the conflicts between classes. A conflict could be a student registered in two classes or an instructor giving two classes (so two nodes connected by an edge cannot be scheduled at the same time). Detection of hard and soft constraints can be easily done by checking if two adjacent nodes are scheduled at the same timeslot.

The adjacency matrix corresponding to the graph, as depicted in figure 2, will carry three possible values at each entry:

0: represents the absence of conflicts.

1: at least one student's allocation is conflicting. 2: at least one instructor's allocation is conflicting.

3: at least one instructor and one student are allocated in conflict with the rules.

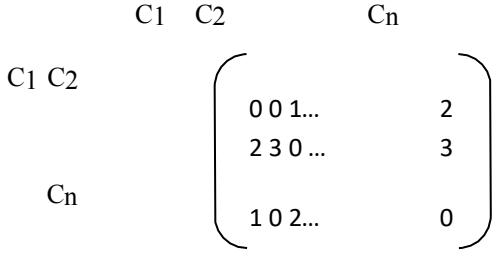

Figure 2: Adjacency matrix representing different conflicts between courses

This representation will serve in detecting the conflict type in order to evaluate the fitness value of each individual - timetable- efficiently.

The fitness function evaluates each individual with the corresponding scheduling data and returns a penalty value for that chromosome for each constraint's violations. The scheduling data is simply an object that contains information about the instructor of a given course, the rooms, and the timeslot assigned. In other words, the fitness function is penalty-based. Thus, given a solution with k types of constraints, let $w_i$be the associated weight for that relative constraint. Let $v_i(T)$ be the collection of all the violations attributed to type i constraints. As a result, let the quality of this timetabling solution be $P(T) = \sum_{i=0}^{k} w_i\, v_i$. The fitness function is $F(T) = 1/P(T)$.

As for the selection operator, we will be using the fitness proportionate selection known as the roulette wheel selection. This operation is used to select potential parents that are useful for combination and are most probably going to produce better offspring. The quality of the individuals is used to map a probability of selection

for each individual. Even though candidate solutions with a higher fitness will have lesser probability to be disregarded, there is still a chance that they might not be selected.

The crossover operation is a process in which two or more parents are combined to produce a better child. If the parents are good, it is more possible that their children will also be good. For this kind of problem, we choose a crossover that seeks to produce children that minimize the violations of constraints present for the parents. For instance, the algorithm picks two parents. It looks for two courses that conflicts and that are scheduled at the same time. Say course *n* and *m* conflict. The crossover exchanges the timeslot of the course *m* of the first parent and replaces it by the slot of course *m* of second parent. We apply the same operation to the second parent. In this way, we produced two offspring that minimize the violation of constraints compared to their parents. In fact, by recombining portions of good individuals, this process is likely to create even better individuals. However, this combination might create other type of violations. The solution to this problem will solved by the repair function mentioned in the upcoming section.

For the sake of maintaining diversity and preventing premature convergence, we apply the mutation operation with a low probability. We randomly select two genes of the chromosome - in our case two classes of a timetable and we swap the timeslots allocated to each.

Based on the representation of genes adopted in our algorithm, and after applying the explained crossover, some offspring that are outside the search space might be generated. In fact, each of the parents timetable should contain one instance of each course and a unique combination of timeslot room pair. A child produced by a crossover might violate these constraints leading to multiple assignments of same timeslot room pair to different courses. Such timetables are considered to be outside the search space. Thus, repair strategies could be used to remap the unacceptable offspring to the search space. In particular, the repair function will go through the list of courses and check if the pair of timeslot and room is uniquely assigned to the courses for a single individual. In other words, it ensures that no two rooms can be occupied by two different courses at the same time. The procedure will be applied to each timetable generated. The repair strategy will help our algorithm converge more quickly to the optimal solution.

4. INTERPRETATION OF THE RESULTS

For experimental purposes, we generated our own data including list of classes, list of students each registered in at most five courses, a list of major-minor courses, a list of co-requisites courses, and a list of rooms each with its capacity. The parameters used in the experiments are as explained in table 1.

Table 1. Size of the problem.

Population	100
Number of generations	150
Crossover rate	0.7
Mutation rate	0.02

The algorithm improves the solution iteratively. In each run – a generation - we keep track of the best local timetable having the best fitness value. And at the end of each run, we compare the global best to the local best solution of the generation and update the value of the global best. When the maximum number of iterations is reached, the global best will be the optimal solution for the problem. In fact, the solution is a timetable – a possible scheduling of the courses. The solution provides the list of courses along with the room allocated to each, the day, and the time assigned.

Starting with a random initial population, within 150 generations each made up of 100 individuals, our algorithm reduces the average cost of penalties by approximately 70%. While the average fitness value of the initial

randomly generated population was 2932300, the fitness is reduced to 846300 after 150 runs.

Genetic operators play an important role in improving the solution. From one side, selection alone makes the population become filled up with copies of the best individual from the population. Implementing both selection and crossover operators will cause convergence on a good but sub-optimal solution. Mutation alone is traversal for the search space; however, it does not provide a good improvement to the solution. Experiments showed that with mutation alone the average fitness value is improved by only 10% from the initial population after 150 generations.

5. CONLCUSION

Many approaches exist to solve the university Timetable Problem. In this paper, we used the genetic algorithm implementation approach. Moreover, the different operators including selection, mutation and crossover along with their effectiveness are discussed. In addition, the repair function is also explained to map each infeasible solution into a feasible one. The proposed algorithm starts with an infeasible random population and improves the solution.

For future work, we plan to apply the exchange of timeslot resulting from the crossover for each pair of conflicting courses for an individual instead of applying it to only one pair of conflicting courses. This way, the solution will be improved quickly where possibly the number of conflicts for an individual is reduced further in a single iteration. However, we cannot ignore the fact that crossover operator might resolve certain conflicts but can lead to other ones as well.

We also plan to introduce conditions on the mutation operation. If the mutation operation will lead to the violation of constraints, then the swap is reset and the process is repeated for another two genes until a mutation that does not introduce a violation occurs. In order to escape from an infinite loop, we can specify a number of trials for the mutation.

ACKNOWLEDGEMENTS

This work as supported by the Lebanese American University – Beirut, Lebanon.

REFERENCES

[1] D. de Werra., (1985) “An introduction to timetabling”, in European Journal of Operational Research, vol. 19, pp. 151-162.
[2] R. Qu, E. K. Burke, B. Mccollum, L. T. Merlot, S. Y. Lee, (2009) “A survey of search methodologies and automated system development for examination timetabling”, Journal of Scheduling, vol.12, no.1, pp.55-89.
[3] T. Arani and V. Lotfi., (1989) “A three phased approach to final exam scheduling,” IIE Trans. 21(1), pp. 86-96.
[4] M. W. Carter, (1986) “A survey of practical applications of examination timetabling algorithms,” Operations Research, vol. 34, pp. 193–202.
[5] P. Boizumault, Y. Delon and L. Peridy, (1996) “Constraint logic programming for examination timetabling,” Journal of Logic Programming 26(2), pp. 217–233.
[6] T. Elsaka, (2017) "Autonomous generation of conflict-free examination timetable using constraint satisfaction modelling," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, pp. 1-10.

[7] A. Colorni, M. Dorigo, and V. Maniezzo, (1990) “Genetic algorithms - a new approach to the timetable problem,” in Lecture Notes in Computer Science - NATO ASI Series, vol. F 82, pp. 235– 239, Combinatorial Optimization, (Akgul et al eds), SpringerVerlag.
[8] S. F. H. Irene, S. Deris, and S. Z. M. Hashim, (2009) “A study on PSO-based university course timetabling problem,” in Proc. Int. Conf. Adv. Comput. Control, pp. 648–651.
[9] K. Socha, J. Knowles, and M. Samples, (2002) “A max-min ant system for the university course timetabling problem,” in Proc. 3rd Int. Workshop Ant Algorithms (Lecture Notes in Computer Science), vol. 2463, pp. 1– 13.
[10] S. Desroches, G. Laporte and J.M. Rousseau, (1978) “HOREX: A computer program for the construction of examination”. INFOR, 16(3), pp. 294-298.
[11] B. Hussin, A. S. H. Basari, A. S. Shibghatullah, S. A. Asmai and N. S. Othman, (2011) "Exam timetabling using graph colouring approach," 2011 IEEE Conference on Open Systems, Langkawi, pp. 133-138.
[12] S. Broder, (1964) "Final Examination Scheduling," Communications of the ACM, vol. 7, no. 8.
[13] D. C. Wood, (1969) “A technique for colouring a graph applicable to large scale timetabling problems”. The Computer Journal, 12 (4), pp. 317-319.
[14] D. W. Matula, G. Marble, and D. Isaacson, (1972) "Graph Coloring Algorithms," in Graph Theory and Computing, R. Read (ed.), Academic Press, New York, pp. 109-122.
[15] R. Ilyas and Z. Iqbal, (2015) "Study of hybrid approaches used for university course timetable problem (UCTP)," 2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, pp. 696-701.
[16] L.T.G. Merlot, N. Boland, B.D. Hughes, P.J. Stuckey, (2003) “A hybrid algorithm for the examination timetabling problem ». In E.K. Burke and P. De Causmaecker (eds). (2003). Practice and Theory of Automated Timetabling. Selected Papers from the 4th International Conference. Springer Lecture Notes in Computer Science, 2740, pp. 207-231.
[17] I. S. F. Ho, D. Safaai and M. H. S. Zaiton, (2009) "A Combination of PSO and Local Search in University Course Timetabling Problem," 2009 International Conference on Computer Engineering and Technology, Singapore, pp. 492-495.
[18] N.R Al-Milli, (2010) “Hybrid Genetic Algorithms with Great Deluge for Course Timetabling”. International Journal of Computer Science and Network Security, vol.10 no.4, pp. 283-288.
[19] V. Bhatt and R. Sahajpal, (2004) "Lecture Timetabling Using Hybrid Genetic Algorithms",IEEE , International Conference on Intelligent Sensing and Information Processing Proceedings, vol. 21, pp. 24-34.
[20] A. Golabpour, H. M. Shirazi, A. Farahi, A. Z. M. Kootiani and H. Beigi, (2008) "A Fuzzy Solution Based on Memetic Algorithms for Timetabling," *2008 International Conference on MultiMedia and Information Technology*, Three Gorges, pp. 108-110.
[21] B. Sigl, M. Golub, and V. Mornar, (2003) "Solving timetable scheduling problem using genetic algorithms, " In Proceedings of the 25th International Conference on Information Technology Interfaces, (ITI' 2003), pp. 519-524.