Hybrid Algorithm to Solve Timetabling Problem

Wesam Ali Algasm
Kerbala Education Directorate, Iraq, E-mail: wesamalhusainy@gmail.com

Abstract

The university course timetabling problem is a well-known highly-constrained difficult optimization problem. The problem seeks the best allocation of courses to time slots and rooms while ensuring all related constraints are satisfied. Due to the limited resources (rooms and time slots), finding an optimal, or even a high quality, timetable is a challenging task that every university encounters every semester. Many metaheuristic algorithms have been proposed for university course timetabling problem. Genetic algorithm is a class of metaheuristic and has shown very good results for many real-world problems. However, for university course timetabling problems, a traditional genetic algorithm is not usually considered as an efficient solver because it is very hard to maintain the solution feasibility. In this research, we propose a new hybrid algorithm that combines genetic algorithm with simulated annealing to find good solutions for university course timetabling problems. The proposed hybrid algorithm uses simulated annealing in adaptive manner to rectify solutions and to improve the quality of the generated solution by genetic algorithm. The proposed algorithm is tested over Socha dataset from the scientific literature and compared with the state of the art methods.

Key Words: Genetic Algorithm, Simulated Annealing, Hybrid Genetic Algorithm, Timetabling, University Course Timetabling.

1- Introduction

This study focuses on University Course Timetabling Problem (UCTP). UCTP is usually occurring at the beginning of each semester of the new academic year in educational institutions. UCTP can be defined as the process of assigning a set of events include courses, teachers, and students to rooms and fixed time-slots while respecting the pre-defined constraints. The constraints are divided into hard and soft constraints. The hard constraints must be satisfied under any circumstances. The timetable that meet all hard constraints is called feasible timetable. Examples of hard constraints are: no one student has two courses at same time, no two courses allocated to the same room at the same time, the room capacity should equal or greater than the number of students registered in the allocated course. On the other hand, soft constraints should be satisfied as much as possible. The quality of the timetable is measured based on how much the soft constraints are satisfied. Example of soft constraints are: no student should
be required to attend an event in the last time slot of a day and no student should sit more than two events in a row. The main role of the optimization method is to generate timetable that meet all hard constraints and satisfy soft constraints as much as possible.

2-Related works

2-1- Constraint-Based Methods
A constraint satisfaction problem model was developed by Zhang and Lau [4] to be used for the UCTP. By using the ILOG scheduler and ILOG solver, they implemented a constraint satisfaction approach. In addition, they performed several operations using ILOG in order to improve performance. The results showed a slight improvement in performance. However, a simple case study was used in their work.

2-2- Sequential Heuristics
Many works have been published in literature on such methods. Most of these heuristics methods are generated by heuristics processes to solve the graph coloring problem. The main idea behind graph coloring heuristics is to schedule events to timeslots sequentially [11, 12]. Thus, graph coloring heuristics considered to be as sequential heuristics.

2-3- Meta-heuristic Approaches
Meta-heuristics approaches are usually based on two main methods namely the local-area based searches and population-based searches. Local-area based searches improves the current solution by exploring the solution space. The tabu search and simulating annealing search technique are fall in this category. These methods are based on iterative processes in order to reach the optimal solution. Research techniques that rely on population perform the search operations by maintaining a population of candidate solutions. So, what distinguishes these methods from others is that they reveal neighborhood of a whole population instead of covering a single solution in the search process [5]. The following section will summarize the most used meta-heuristic approaches applied to course timetabling.

2-4- Hybrid Meta-heuristics
Among the many researchers who worked on the hybrid approach Kostuch [42] which offers a three-stage approaches, which combines graph colouring and Simulated Annealing algorithm for University Course Timetabling Problem. In the first stage he used the graph colouring approach to generate feasible timetable and then in the second stage developed this work using SA algorithm. In the last stage he applied further improvement using local search guided by simulated Annealing. This method had achieved high results, where it has been able to provide 13 best solutions out of 20 instances.

2-5- Hyper-heuristic Approaches
Burke et al. [9] proposed Tabu Search hyper-heuristic approach for University Course Timetabling Problem. In this proposed algorithm a set of six low-level heuristics contend with each other. The approach based on functions, a set of low-level heuristics, and tabu list. If there is a need to use
heuristics, the cost function is used in this case, where the change in the value of this function is noted between the old solution and the current solution. If there is an improvement in the cost function, the case of the tabu list of the heuristics will be changed from active to non-active. Experimental results showed that this technique is able to produce good quality solutions for UCTPs as well as different types for optimization problems.

3- Methodology

3-1- Genetic Algorithm

Ordinarily, in a simple GA one child is created through replication in each iteration (called a generation). It beginning with an initial population created randomly of possible solutions to a problem. The solutions are called an individual, a member of the population. The individuals are tested against a function specific to the given problem. This function is called the fitness function. Then comes a choosing (selection) phase, where when the evaluation is done, two suitable individuals are selected to be parents through a selection operator in order to continue the recombination method.

During the recombination process, two operators will be performed, the crossover as well as the mutation for creating children to explore new solutions. These new individuals replace the old ones, often the worst ones in the population. This iteration continues until a predefined stop condition is met. This condition may be a number of generation or a period of time.

3-1-1- Solution Representation

In GA, the chromosomes must be built in such a way to be effective for crossover, mutation, operators and fitness function.

In order to assign the rooms, we used a matching algorithm. For solving the problem, a direct representation method is applied. Every solution of population could have represented by a number of memes containing the data related to timeslots and rooms for a specific course.

In Table (1) a string of genes that compose the chromosomes’ representation is shown. The chromosome is representing a feasible solution and the genes show timeslots. As can be seen, e14, e17, and e1are taking place at timeslot T1.
Table 1- Chromosome Representation

| R1   | R2   | R3   |   | Rn   |
|------|------|------|---|------|
| T1   | e_{14}| e_{17}|   | e_{1}|
| T2   | e_{6} | e_{7}  | e_{2}|
| T3   | e_{3} | e_{5}  | e_{9}|
| T4   | e_{19}| e_{4}  | e_{15}| e_{8} | e_{10}|
| T5   | e_{7} | | | |
| T6   | | e_{13}| | |
| T7   | e_{16}| | | |
| T8   | e_{11}| e_{20} | e_{21}|
| T9   | e_{12}| e_{18} | | e_{22}|

3-1-2- Initial Population Generation

Figure (1) shows the pseudo code used for generating a large number of populations for feasible timetables in random way.

```
Generate a random ordering of courses;
    For all courses
        For all available room
            For all available timeslot
                Assign room and timeslot
                If no conflicts
                    Exit;
                End if
            End for
        End for
    End for
End;
```

3-2-1- Simulated Annealing Environment

SA, which is an algorithm for solving optimization of NP hard problems and has a historical use dating back to 1980s. For obtaining an optimum solution, these algorithms use neighborhood and fitness functions. A parameter for temperature controlling as well as cooling time programming is used by the SA in order to avoid a local optimum with the goal of achieving more optimum solution [5].
The materials’ annealing process inspired the idea of SA. In a liquid, the materials’ particles move around freely, while in solids, the particles have a minimum level of energy and are placed in a structured way. Here, the preferred state is the one with the highest possible temperature and with the appropriate speed of cooling. In other cases, the substance freezes stably but the perfect status is not achieved.

In SA Temperature plays the role of controlling parameter in SA, for imitating cooling and heating factors. In each iteration, cooling is enforced slowly. Sometimes heating can be used and it is possible to increase temperature rapidly. These changes are defined in cooling schedule that determines temperature changes throughout the time and affects the algorithm efficiency.

The cooling schedule can be defined by the following parameters:
• Initial temperature.
• Reduction in temperature function.
• Final temperature / criteria of Stopping.

A probability function is used with a number generated randomly between zero and one, are used to determine if a less fitness solution is allowed as the current solution or not. This function relies on the temperature as well as the fitness function (T and C, respectively) and for higher temperatures it produces higher probability while little fitness differences is generated for two solutions (i.e. C1 and C2).

The function of probability (1) relies on the annealing process and is matched with a random number that ranges between one and zero. When the probability is larger than the number generated randomly, the solution can be considered as acceptable. The fitness is produced using the fitness function. Fitness determines if the quality of a given solution is prefer than the current solution or not.

\[ P(\Delta C) = e^{-\frac{\Delta C}{T}} \] (1)

The difference between the current solution and a given solution (test solution) is shown by \( \Delta C \). \( \Delta C \) is positive since the better solution is instantly accepted and as a consequence, the probability function will not be runs. Since 1983 onwards, many optimization problems used the SA process. SA has been successful because of acceptable runtime, high quality of solution, easy application, as well as flexibility.

It is worth mentioning that although it can be easily implemented and it is flexible, creating an efficient algorithm for solving a problem is not a simple task all the times. Good neighboring function as well as cooling schedule are the factors affecting the SA efficiency [5].

3-2- Hybrid Approach

A hybrid GA employs the origin algorithm as genetic algorithm but takes advantage of other approaches in its scope to create a hybrid GA that yields a better performance compared to the standard genetic algorithm. In the standard genetic algorithm, we need to balance two goals, which are conflicting, where it exploits the most suitable solutions found so far, and meanwhile it explores for other promising solution spaces [6,7,8].
Through local search approaches inside a genetic algorithm, the search for a universal optimum speed up by means of adding new genes, and in case local information is used, then the genetic algorithm accelerates through spotting the most favorable search region.

In the genetic algorithm, the use of local search methods provides some benefits such as the solution quality enhancement, efficiency enhancement, making sure the feasible solutions are found, enabling the use of fitness function estimation, and making it possible to replace other operators instead of the standard ones. The way the solution quality improves through local search is that it is able to spot local optima very precisely. The improvement of efficiency takes place through a reduction in the time of obtaining a solution, and this is crucial in real problems, because the evaluation function take the most of time of an algorithm [9].

4-The Course Timetabling Datasets

We used Socha dataset in this research to evaluate our proposed approach.

- Socha, which has 11 instances (5 small, 5 medium and 1 large) are created by an algorithm proposed by Ben Paechter [8]. The time constraint is set to 90, 900, 9000 seconds for the small, medium, and large instances, respectively [10]. This is not an ideal solution though, since various machine characterization means that the running for 900 seconds, which is not a right comparison. Table 2 shows benchmark statistics [8].

| Instance | Small | Medium | Large |
|----------|-------|--------|-------|
| Event    | 100   | 400    | 400   |
| Room     | 5     | 10     | 10    |
| Feature  | 5     | 5      | 10    |
| Student  | 80    | 200    | 400   |

6-Conclusion

Genetic algorithm (GA) has proven as a powerful algorithm for many optimization problems. As a starting point of our research, we examined the behavior of a simple GA for the UCTP. The results and conclusions from the experiments are: For all datasets, while using genetic algorithms to solve the UCTP can produce promising results, these results are not better than those produced by approaches proposed in the literature. GA cannot produce feasible solution for all datasets. Regarding to dataset that have too many records in terms of students and courses are infeasible when the problem being solve by GA. This is mainly because solutions generated by GA did not always satisfy the hard constraints of UCTP. This is
caused by the crossover operator which mix the solutions in unguided way. Hence, a conventional GA needs to be enhanced to solve the UCTP. For these reasons we found that the ability of GA can be improved by many ways. We first modified the simple genetic algorithm by use roulette wheel selection method and a specialist crossover (PMX) to improve the resultant child in order to get feasible solutions through the GA. Finally, we hybridized GA with a local search algorithm SA. Hybrid algorithm GAISA have been successfully implemented to solve many problems. GAISA could achieve better performance than GA and SA by using the respective strengths of each algorithm better than if used independently and we could get a better result when we compared our results with most papers in the literature.

References

[1] M. W. Carter and G. Laporte. Recent developments in practical course timetabling. Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1408, pp. 3–19, 1998.
[2] A. Schearf. A survey of automated timetabling. Artificial Intelligence Review, 13(2): 87–127, 1999.
[3] S. Petrovic and E. K. Burke. University timetabling. Handbook of Scheduling: Algorithms, Models and Performance Analysis, Chapter 45, (Editor: J.Leung), CRC Press, 2004.
[4] L. Zhang and S. Lau. Constructing university timetable using constraint satisfaction programming approach. Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, pp. 1–5, 2005.
[5] Sørensen, Matias, and Thomas Riis Stidsen. "Hybridizing integer programming and metaheuristics for solving high school timetabling." 10th international conference on the practice and theory of automated timetabling, 2014.
[6] Akkan, Can, and Ayla Gülçü. "A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem." Computers & Operations Research 90 (2018): 22-32.
[7] Jaengchuea, S., & Lohpetch, D. (2015, October). A hybrid genetic algorithm with local search and tabu search approaches for solving the post enrolment based course timetabling problem: outperforming guided search genetic algorithm. In Information Technology and Electrical Engineering (ICTEEE), 2015 7th International Conference on (pp. 29-34). IEEE.
[8] Goh, Say Leng, Graham Kendall, and Nasser R. Sabar. "Improved local search approaches to solve the post enrolment course timetabling problem." European Journal of Operational Research 261.1 (2017): 17-29.
[9] E. K. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyper-heuristic for timetabling and rostering. Journal of Heuristics, 9(6): 451–470, 2003.
[10] Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. Journal of Scheduling, 9 (5), 403–432.