Indonesian Journal of Electrical Engineering and Computer Science
Vol. 13, No. 1, January 2019, pp. 72~76
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v13.i1.pp72-76

University course timetabling model using ant colony optimization algorithm approach

Munirah Mazlan, Mokhairi Makhtar, Ahmad Firdaus Khair Ahmad Khairi, Mohamad Afendee Mohamed
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Terengganu, Malaysia

Article Info

Article history:

Received Oct 1, 2018
Revised Nov 21, 2018
Accepted Dec 2, 2018

Keywords:

Ant Colony Optimization
Course Timetabling
Meta-heuristics
Optimization

ABSTRACT

Due to the increased number of students and regulations, all educational institutions have renewed their interest to appear in the number of complexity and flexibility since the resources and events are becoming more difficult to be scheduled. Timetabling is the type of problems where the events need to be organized into a number of timeslots to prevent the conflicts in using a given set of resources. Thus in the intervening decades, significant progress has been made in the course timetabling problem monitoring with meta-heuristic adjustment. In this study, ant colony optimization (ACO) algorithm approach has been developed for university course timetabling problem. ACO is believed to be a powerful solution approach for various combinatorial optimization problems. This approach is used according to the data set instances that have been collected. Its performance is presented using the appropriate algorithm. The results are arguably within the best results range from the literature. The performance assessment and results are used to determine whether they are reliable in preparing a qualifying course timetabling process.

Corresponding Author:

Munirah Mazlan,
Faculty of Informatics and Computing,
Universiti Sultan Zainal Abidin,
Terengganu, Malaysia.
Email: munimazlan@gmail.com

1. INTRODUCTION

Timetabling in educational institutions is an important activity to schedule courses or examination, which must be entirely assigned into appropriate timeslots for students, lecturers and rooms subject to constraints. Course timetabling is a common problem for every academic community and is often attempted by researchers [1]. Although various scientific and commercial works were made in this area, in many educational institutions, timetable is still scheduled manually and even if the computer is frequently used, it is only necessary to present data or checking constraints validation [2]. It is extremely difficult to solve a large course timetabling problems with manual approach which may need a group of academic staff to work for several days [3].

Thus, university course timetabling problem is considered as non-deterministic polynomial (NP) hard problem, which means the complexity of computational amount on time required increases exponentially with problem size [4]. The university timetabling problems are actually different from each other depending on the type of educational institution, the scheduled entities, and the constraints involved. The main idea of this problem is to assign a set of events into a limited number of timeslots subject satisfying a number of constraints. The set of constraints are usually divided into two particular types which are hard and soft constraints [6]. Therefore, the problem objective is to satisfy the hard constraints and minimize the

violation of the soft constraints. It is therefore necessary to use efficient approach to produce timetable that satisfies the constraints.

There have been numerous amounts of research on solving course and examination timetabling problems [7-10]. Nowadays with better computing technology, various methods are proposed to produce a better solution of the timetabling process [11]. This paper concentrates on the course timetabling problems. Metaheuristic or approximation optimization algorithms are becoming increasingly effective for solving course timetabling problem such as Genetic Algorithm (GA) [12,13], Tabu Search [14,15], Simulated annealing[16] and Ant Colony Optimization (ACO) [17]. These algorithms can produce high-quality solutions but do not guarantee optimality [6]. In recent years, ACO has been extensively studied for tackling this problem. It has been taken up by some scientists and mathematicians and has been exploited and widespread in the early 90's [18,19]. ACO is inspired by ant stigmergic behavior which simulates a set of agents that work together to find solutions to optimization problems by relying on uncomplicated communication.

In light of the above, this paper discusses the used of ant colony optimization in dealing with the course timetabling problem based on appropriate algorithms applied. This paper is organized as follows: Section 2 will summarize the problems definition of course timetabling and Section 3 elaborates the ACO approach and algorithm used. Section 4 presents the results, while Section 5 will highlight the overall conclusion of the paper.

2. COURSE TIMETABLING PROBLEM DESCRIPTION

The university course timetabling problem is an optimization problem where a set of events needs to be scheduled in timeslots for students, lecturers and located in the appropriate room while maintaining the constraints. The problem presents asset of N courses to be scheduled in 5 days of 9 periods each which time T is equal to 45 timeslots, a set of R rooms where each room has a set of F features and capacity, a set of M students and a set of features required by the courses associated [20]. The problem objective is to satisfy the hard constraints and to minimize the violation of the soft constraints.

The general constraints for course timetabling problem can be classified into two types which are hard constraints and soft constraints[6][21]. A practical and feasible timetable must satisfy all the hard constraints with no consideration while the soft constraints are not absolutely essential but the amount of related violation should be minimized to maximize the perfection the timetable.

The hard constraints considered in this problem are:

H1: No student can be assigned to more than one course at the same time.
H2: The room should satisfy the features required by the course.
H3: The number of students attending the course should be less than or equal to the capacity of the room.
H4: No more than one course is allowed at a timeslot in each room.

Besides satisfying the hard constraints, the violation of soft constraints can be considered as preferences that will fulfill the user requirements to improve the quality of timetable. Soft constraints are often confronted which include the following:

A student has to attend only one course in a day.
A student has to attend more than two courses consecutively.
A student has to attend a course in the last period in any day.

3. ANT COLONY OPTIMIZATION ALGORITHM

Ant colony optimization (ACO) is a technique that mimics the number of artificial ants moving on a graph that encode the problem itself proposed by Dorigo [22,23]. The original member of ACO Algorithm can be specifically classified as Ant Colony (AS). After few years, ACO is enhanced with two successful variants which are Ant Colony System (ACS) and Max-Mix Ant System (MMAS) with better searching performance. Nowadays, these approaches have been widely applied in solving the discrete optimization problem and other combinatorial problems for example travelling salesmen problem (TSP), graph coloring, scheduling problems and vehicle routing problems [24, 25].

ACO algorithm is inspired from the foraging behavior of real ant colonies. In ACO algorithm artificial ants successfully construct solution based on the global information (pheromones) and local information. The pheromone then acts as a probabilistic model for the construction of solutions and is constantly amplified by ants built with high quality solutions. Hence, pheromone evaporation surpasses premature convergence to a poor local optimum.

In ACO, the problem is actually dealt with by simulating some artificial ants that move on the graph that issues the problem itself. The traveling salesman problem (TSP) plays a major role in ACO as it is the

first problem to be attacked by ACO. Thus, TSP has been implemented using Ant System algorithms as shown in Figure 1. The figure explains the basic principles of the ACO algorithms.

3.1. The Strategy of Ant's Movements

The basic material of ACO is the use of a probabilistic solution construction mechanism based on the stigmergy. All l ants are initially placed randomly at node i. Each ant builds a complete timetable in each iteration algorithms in which all the hard constraints are satisfied at the highest level in every resulting solution. The ant k travels from node to node starting from the node i. The transition probability P_{ij}^{k} that the ant k, currently at node i will choose and move to the next node j is calculated using the random proportional rule given by,

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \, , j \in N_i^k \quad (1)$$

Where τ_{ij} is the pheromone trail value on the edge connecting to node i to node j meanwhile η_{ij} is the heuristic value of that edge and $\eta_{ij} = 1/d_{ij}$, α and β are the parameters that determine the relative influence of pheromone trail and the heuristic information and N_i^k is a set of all nodes that remains to be visited when the ant k is at node i. After iterating the process, each ant completes the tour.

3.2. Pheromone Update

The pheromone trail value is updated at the end of each iteration algorithm. The ants with shorter route will leave more pheromone trail than those with longer route. The pheromone update rule defines the way in which good solutions are reinforced in the pheromone trail by adding a quantity $\Delta\tau_{ij}$ which is high on the best solution arcs. Therefore, the amount of pheromone in each route will be adjusted by the pheromone trail decay equation (2). The trail levels are updated as on every route each ant leaves pheromone quantity given by Q/L_k, where Q is constant and L_k is the length of its best tour respectively. On the other hand, the evaporation of pheromone trail is applied at the end of each iteration algorithm. Thus, the rule for updating trails could be defined as follows:

$$\tau_{ij}(t+1) \leftarrow (\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (2)$$

$$\Delta\tau_{ij} = \sum_{k=1}^{l} \Delta\tau_{ij}^{k} \quad (3)$$

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L_k} \, , \text{if ant k travels on edge } (i,j) \\ 0 \text{ otherwise} \end{cases} \quad (4)$$

Where t is the iteration counter, p is the pheromone evaporation rate, $\Delta\tau_{ij}$ is the amount of increase in the trail level on the edge (i,j) and $\Delta\tau_{ij}^{k}$ is the increased levels of trail at the edge (i,j) caused by the ants k respectively. The algorithm needs to be applied with the pheromone evaporation process to avoid the unlimited pheromone collection and initial convergence toward a suboptimal solution region. The next iteration $t+1$ will start after updating the pheromone trail process. Figure 1 is the basic principle of the ACO algorithm.

ACO Algorithms

- Initialize pheromone trails
- Do while (Stop condition/if criteria are not satisfied) - loop
Generate solution
Local Search solution
Update pheromone trails
- End Do
- End

Figure 1. The basic Principles ACO Algorithm

4. EXPERIMENTAL RESULT

The ACO approach is constructed and tested in a web-based computer system of the UniSZA Course Timetabling System supported by the CPU Intel Core i7-3610QM with 2.3 GHz and RAM 6GB under window 7. To evaluate the proposed approach, we conducted several experiments. A priority is given on several tests to determine the allocation of course timetabling for a different large number of students to be scheduled into the prescribed time.

The experiment test has been carried out on the course timetabling where applied using ACO approach to compute the course timetabling results with the test case from Faculty of Informatics and Computing (FIC) datasets as shown in Table 1. The result of the test case in Table 2 indicated that the FIC_A1 which is without priority has produced 0.29 of standard deviation. On the other hand, the result of the test case in Table 3 indicated that FIC_A2 which is priority given has produced 0.38 of standard deviation. The results indicated that the course timetabling was effectively scheduled and obtained better result rather than the other test case results.

Table 1. The FIC Datasets

Test Case	Subjects	Enrolled Students	Timeslots	Priority (P) / No Priority
FIC_A1	32	1650	10	P
FIC_B1	32	1415	10	P
FIC_A2	32	1650	10	NP
FIC_B2	32	1415	10	NP

4.1. Experiment test without the priority

The Table 2 shows the ACO approach performance without priority on timetabling problem instances.

Table 2. ACO Approach performance without priority

Test Case	ACO Approach			
	Density Conflict (%)	mean	var.	st.dev(σ)
FIC_A1	1.93	0.74	0.08	0.29
FIC_B1	2.26	0.71	0.15	0.39

4.2. Experiment test with the priority

The Table 3 shows the ACO approach performance with priority on timetabling problem instances.

Table 3.ACO Approach performance with priority

Test Case	ACO Approach			
	Density Conflict (%)	mean	var.	st.dev(σ)
FIC_A2	1.93	0.39	0.14	0.38
FIC_B2	2.26	0.45	0.20	0.45

5. CONCLUSION

This paper has introduced the ant colony optimization algorithm in solving the university course timetabling problem. This study has resulted in a feasible approach for course timetable using the generated ACO approach. The implemented ACO approach generated comparable results like other high performance results presented in the literature. The presented experiment results are good and indicated a reliable approach for university course timetabling problems. We believed that better results can be obtained based on how dataset problems are fixed. In addition, the algorithms implemented can be enhanced depending on the adjustment results. Future work will aim to improve the ACO approach and applied the proposed approach on real-world courses timetabling.

ACKNOWLEGEMENT

This work is partially supported by UniSZA (Grant RR008 CRIM/2016).

REFERENCES

[1] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," *Comput. Ind. Eng.*, vol. 86, pp. 43–59, 2015.

[2] W. Legierski, "System for solving timetabling problems," *Proc. Symp. Methods Artif. Intell.*, pp. 76–77, 2003.

[3] S. a. MirHassani, "A computational approach to enhancing course timetabling with integer programming," *Appl. Math. Comput.*, vol. 175, pp. 814–822, 2006.

[4] K. Socha, M. Sampels, and M. Manfrin, "Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art," *Appl. Evol. Comput.*, pp. 334–345, 2003.

[5] S. Abdullah and H. Turabieh, "Generating university course timetable using Genetic Algorithms and local search," in *Proceedings - 3rd International Conference on Convergence and Hybrid Information Technology, ICCIT 2008*, 2008, vol. 1, pp. 254–260.

[6] R. Lewis, "A survey of metaheuristic-based techniques for University Timetabling problems," *OR Spectr.*, vol. 30, no. 1, pp. 167–190, 2008.

[7] M. W. C. I and G. Laporte, "Recent Developments in Practical Course Timetabling," *Pract. Theory Autom. Timetabling II*, vol. 1408, pp. 3–19, 1998.

[8] M. W. Carter and G. Laporte, "Recent developments in practical examination timetabling," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1996, vol. 1153, pp. 3–21.

[9] A. Schaerf, "Survey of automated timetabling," *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 87–127, 1999.

[10] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *Eur. J. Oper. Res.*, vol. 140, no. 2, pp. 266–280, 2002.

[11] S. A. Mirhassani and F. Habibi, "Solution approaches to the course timetabling problem," *Artif. Intell. Rev.*, vol. 39, no. 2, pp. 133–149, 2013.

[12] S. Yang and S. N. Jat, "Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling," *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 41, no. 1, pp. 93–106, 2011.

[13] S. Gyori, Z. Petres, and A. Várkonyi-Kóczy, "Genetic Algorithms in Timetabling. A New Approach.," *Budapest Univ. Technol. Econ.*, 2001.

[14] E. P. Fod, "Theory and Methodology Tabu search for large scale timetabling problems," vol. 54, pp. 39–47, 1991.

[15] A. Hertz, "Tabu search for large scale timetabling problems," *Eur. J. Oper. Res.*, vol. 54, no. 1, pp. 39–47, 1991.

[16] S. Abdullah, K. Shaker, B. Mccollum, and P. Mcmullan, "Dual sequence simulated annealing with round-robin approach for university course timetabling," *Eur. Conf. Evol. Comput. Comb. Optim.*, pp. 1–10, 2010.

[17] G. Molnar and M. Cupi, "University Course Timetabling Using AGO: A Case Study on Laboratory Exercises," *Knowledge-Based Intell. Inf. Eng. Syst. Pt I*, vol. 6276, no. 36, pp. 100–110, 2010.

[18] M. Dorigo, V. Maniezzo, A. Colorni, and M. Dorigo, "Positive Feedback as a Search Strategy," *Tech. Rep. 91-016*, no. June, pp. 1–20, 1991.

[19] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.

[20] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria, "An effective hybrid algorithm for university course timetabling," *J. Sched.*, vol. 9, no. 5, pp. 403–432, 2006.

[21] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *Eur. J. Oper. Res.*, vol. 176, no. 1, pp. 177–192, 2007.

[22] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the First European Conference of Artificial Life*, 1991, pp. 134–142.

[23] A. Colorni, M. Dorigo, and V. Maniezzo, "An Investigation of some Properties of an``Ant Algorithm''.," *Ppsn*, no. Ppsn 92, pp. 2–7, 1992.

[24] Z. Chi, S. Su, and M. A. Khine, "An Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem," vol. 16, pp. 54–59, 2011.

[25] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, 2005.