Indonesian Journal of Electrical Engineering and Computer Science
Vol. 12, No. 1, October 2018, pp. 303~309
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v12.i1.pp303-309

Genetic Algorithm with Elitist-Tournament for Clashes-Free Slots of Lecturer Timetabling Problem

Marina Yusoff[1], Anis Amalina Othman[2]
[1]Advanced Analytic Engineering Center (AAEC), Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia
[2]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia

Article Info

Article history:

Received May 6, 2018
Revised Jun 27, 2018
Accepted Jul 17, 2018

Keywords:

Evolutionary optimization
Genetic algorithm
Penalty measure
Tournament elitism

ABSTRACT

Genetic algorithm (GA) approach is one of an evolutionary optimization technique relies on natural selection. The employment of GA still popular and it was applied to many real-world problems, especially in many combinatorial optimization solutions. Lecturer Timetabling Problem (LTP) has been researched for a few decades and produced good solutions. Although, some of LTP offers good results, the criteria and constraints of each LTP however are different from other universities. The LTP appears to be a tiresome job to the scheduler that involves scheduling of students, classes, lecturers and rooms at specific time-slots while satisfying all the necessary requirements to build a feasible timetable. This paper addresses the employment and evaluation of GA to overcome the biggest challenge in LTP to find clashes-free slots for lecturer based on a case study in the Faculty of Computer and Mathematical Sciences, University Technologi MARA, Malaysia. Hence, the performance of the GA is evaluated based on selection, mutation and crossover using different number of population size. A comparison of performance between simple GA with Tournament Selection scheme combined with Elitism (TE) and a GA with Tournament (T) selection scheme. The findings demonstrate that the embedded penalty measures and elitism composition provide good performance that satisfies all the constraints in producing timetables to lecturers.

Corresponding Author:

Marina Yusoff,
Advanced Analytic Engineering Center (AAEC),
Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA,
Shah Alam, Selangor, Malaysia.
Email: marinay@tmsk.uitm.edu.my

1. INTRODUCTION

The timetabling problem appears to be a tiresome job that involves scheduling of students, classes, lecturers and rooms at specific time-slots while satisfying all the necessary requirements to build a practical timetable. The normal problem of creating a practical timetable is a scheduling lectures, classes and rooms into fixed timeslots while ensuring that there are no clashes between lecturers and rooms in the same period. Therefore, the resources become difficult to schedule, thus the choice of this project topic to optimize timetable scheduling under given constraints.

The problem considers the allocation of the resources such as lecturers, students and classrooms in timeslots while satisfying the constraints of existing facilities. Traditionally, timetable scheduling was done manually by the education center staffs which require a lot of times and efforts. Numerous aspects should be taken into considerations and it is often difficult to meet all the requirements and satisfy the constraints. One of the biggest problems with the manual timetabling is to deal with finding clashes-free slot [1]. Any changes require the team to undo previous allocation and find a new allocation resulting in a series of

backtracks. In short, timetable scheduling is one of the hardest areas that proven to be NP-Complete problem [2]. Educational institutions usually grow in number and its complexity. A general timetable scheduling is considered as a multi-dimensional problem in which it needs to assign a set of students, classrooms, subjects, and staff to timeslots according to required constraints [3].

To overcome these problems, automated and practical timetable generator is needed in which it can help provide the path to find an optimal or near-optimal solution to the problem within a short period of time using different techniques of optimization. A feasible timetable is a schedule which essentially must satisfy several constraints. Constraints are almost universally employed by people dealing with timetable scheduling problem [4]. Although many researchers involved in solving the timetabling problem, it is impossible to perfectly solve it because of the variety of constraints in each problem [5]. There are two categories of constraints which are soft and hard constraints. Hard constraints are constraints that cannot be violated and any violation is unacceptable. For example, a lecturer cannot be in two places at the same time, two subjects cannot have common students scheduled in the same time slot and courses cannot be assigned to the same room at the same time. If the hard constraint is violated, then such a schedule will be considered as a failure. While soft constraints are constraints that can be broken, but must be minimized. Although it fails to satisfy these constraints, it is said to be legal if it satisfies all hard constraints [6]. For example, the travel distance of lecturers and students between classrooms should be minimized. It is very difficult to get a solution to solve timetabling problem with all the constraints.

Therefore, many researchers have come up with several techniques to solve hard constraints while minimize the soft constraints even it is difficult to find the best feasible timetable. It has been research few decades ago in various domains related to timetable including assignment and scheduling [7]-[16]. Varies optimization algorithms had been applied and enhanced to support the timetabling problems as such the combination of standard genetic algorithm and hill climbing algorithm that utilize a meme encoded score as a performance indicator to create new candidate solutions during the process of choosing operators [17],

In addition, local optimization with hill climbing algorithm [2], ant colony optimization strategy [18], Harmony search and a modified harmony search algorithm [19], Tabu Search [20], Hyper-heuristics search the space of heuristics and use the limited problem specific information to control the search process can be seen as an adaptive version of iterated local search strategy combining some move operators. In short, this approach consists of number of move operators of different strengths and characteristics combined into an adaptive hyper-heuristic approach to produce better results [15]. However, clashes-free slot is still not address well when considering the numbers of lecturers and the number of classes, although, one of the solution for clashes-free slots have been addressed focusing on an examination timetable based on small data set and still need improvement in its solution [2], [21].

This paper addresses the Lecturers' timetabling solution in Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia as a case study. The interest is on the LTP students and lecturers to fixed timeslots within numbers of constraints. Assigning times and places to lecturers are considered hard problems faced in every university. There are two types of constraints, which are the hard constraint and soft constraint. Hard constraint cannot be violated such as all lecturers must be scheduled and assigned to a distinct room at specified periods. A practical timetable must satisfy hard constraints as it is strictly imposed. While soft constraints are desirable, but they are not essential. They can be violated, but they must be minimized. The remainder of this article is organized as follows. Section 2 explains the LTP and its fitness. The GA is discussed in Section 3. Section 4 presents computational results for the tournament (T) and tournament elitism (TE) selection. Section 5 discussion of the findings and Section 6 concludes this paper.

2. LECTURER TIMETABLING PROBLEM

Lecturer Timetabling Problem (LTP) emphasizes on the assignment of each of lecturers to each of the classes. Lectures of different capacity load for lecturing must accommodate the class requirement for every week. The slot for each lecturer must be determined and satisfied the allocation of time-slot. The following mathematical formulation considers this requirement as an objective function. The LTP can be linearly defined as follows. The LTP consists of a set of lectures, l, a set of subjects, a set of t time slot, a set of r classrooms, and a set of g student groups. The mathematical model formulation is presented in following section. It was adapted from [22]. The notations and parameters used in the model are as follows:

Let:

$l = \{l_1, l_2, l_3, l_4, \ldots . l_i\}$ is a set of lecturers,

$s = \{s_1, s_2, s_3, s_4, \ldots s_j\}$ is a set of subjects,

$t = \{t_1, t_2, t_3, t_4, \ldots t_k\}$ is a set of time slots

$c = \{c_1, c_2, c_3, c_4, \ldots c_n\}$ is a set of classrooms,

$g = \{g_1, g_2, g_3, g_4, \ldots g_p\}$ is a set of student groups.
Cr = $\{cr_1, cr_2, cr_3, crs_4, \ldots cr_j\}$ the credit hour for each lecturer
P = $\{P_1, P_2, P_3, \ldots P_q\}$ is a set of generated timetables

The objective is to minimize the overall summation of all penalties from the generated timetable. Higher penalties will be given for hard constraints violation while lower penalties will be given for soft constraints violation. The penalties will be summed up. The calculation of fitness is shown in Equation. (1) The model is as follows:

$$\text{Minimize } Z = \frac{1}{\sum_1^q P + 1} \quad (1)$$

Subject to:

$$l_i = s_j = t_k \quad \forall_{i,j,k} \quad (2)$$

$$c_n = l_i = t_k \quad \forall_{n,i,k} \quad (3)$$

$$c_n <= g_p \quad \forall_{n,p} \quad (4)$$

$$l_i <\neq cr_i \quad \forall_i \quad (5)$$

A constraint in Equation. (2) ensures that lecturer cannot teach more than one course at the same time. Equation (3) ensures that no room can occupy more than one lecture at the same time. Constraint in Equation (4) considers no student can attend more than one lecture at the same time. Constraints in Equation (5) and Equation (6) ensure the capacity of classrooms should match with student size and lecturer cannot teach less than given credit hour, respectively. Table 1 shows the violation and its penalties.

Table 1. Violation and its Penalty Value

No	Violation	Penalty Value
1.	Lecturer cannot teach more than one course at the same time	50
2.	No room can occupy more than one lecture at the same time	50
3.	No student can attend more than one lecture at the same time	50
4.	The capacity of classrooms should match with student size.	20
5.	Lecturer cannot teach less than given credit hour	20

3. CONSTRUCTION OF GENETIC ALGORITHM

3.1. Solution Mapping

The development GA requires a representation of the problem. We represent using a discrete value and it addresses the time slot, room and subject. Figure 1 is the chromosome representation for GA. The range is based on the datasets obtained for FSKM. The range of range of 1-18 for time slots, 1-43 for the number rooms and 1-34 for the number subjects are considered.

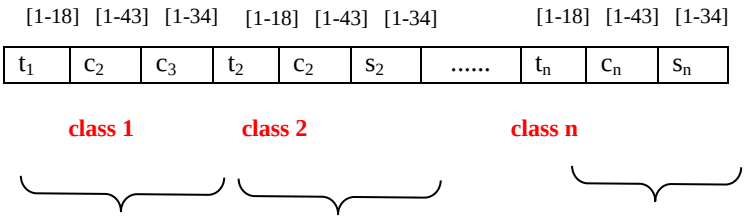

Figure 1. Solution Mapping for the LTP

Where,
t_n = represent the number of timeslots
c_n = represent the number of rooms
s_n = representation the number subjects

Each of the subjects can have more than one class. Hence, the chromosome length would depend on the number of the classes. Sometimes class can accommodate for one group only and can be more depending on the number students. The length of chromosomes in the population will be as follows:
Chromosome length = number of classes * 3 (genes)

3.2. GA Steps for the LTP Solution

Basically, the genetic algorithm process for solving the LTP is as follows:

Step 1: Represent the problem as strings of chromosomes of a fixed length, with a population size of N, as depicted in Figure 1.
Population size is the number of individuals that represent the solution. If there are too many chromosomes, GA tends to slow down. However, GA has very few possibilities to perform crossover and can explore only a small part of the search space if there are too few chromosomes.
Step 2: Define a fitness function to measure the fitness of each chromosome for selecting chromosomes as parents to mate during the reproduction process to produce new off springs. The fitness function stated in Eq. (1) and all constraints are considered.
Step 3: Determine GA parameters (crossover probability, mutation probability and maximum number of generations) and set the initial best fitness equal to 1.
Step 4: Randomly generate an initial population of size N:

$X_1, X_2, \ldots, X_N$

The implementation of the customized random function is as follows:

1. The program will generate classes for every group and the subjects taken to get number of classes that need to be scheduled.
2. The program will randomly generate timeslot, room, and lecturer for each of the classes
3. The chromosome length depends on the number of the classes. The length of chromosomes in the population will be as follows:

   Chromosome length = number of classes * 3 (genes)

Step 5: Calculate the fitness for each chromosome in the population using the formula in (2).
Step 6: Start the first generation.
Step 7: Compute fitness and do selection.

Select parent from the current population for crossover using Tournament selection method
The Tournament selection algorithm is as follows:

1. Randomly choose individuals from the whole population.
2. Compare fitness and choose the fittest individual to be the parent

Step 8: Do crossover and mutation

Create offspring chromosomes by applying crossover and mutation operators according to their probabilities and put the newly created offspring in the new population. As for the crossover, uniform crossover scheme is used where individual bits in the chromosomes are compared between two parents. One of the parents is the parent chosen during the selection stage. The bits are swapped with a fixed probability, 0.5. Mutation is used to maintain genetic diversity and avoid local minima. The program will create random individual to swap genes with the individuals in the current population.

Step 9: Evaluate current population (based on selected population from step 7).
Step 10: Update generation.
Step 11: If the number of generation has reached its termination criterions, go to Step 12.
Step 12: The algorithm is finished. The best solution found when the fitness is recorded as the best fitness.

4. COMPUTATIONAL RESULTS AND DISUSSION

An in-depth analysis of the outputs produced by the GA is reported regarding its performance, on how different parameter tuning affect the efficiency of the GA in finding solutions for the LTP, and to test the robustness of each GA while satisfying all the constraints. A comparison of performance between simple GA with Tournament selection scheme combined with Elitism (TE) and a GA with Tournament (T) selection scheme is also performed

4.1. Parameter Setting

The selection of parameter was handling using trial and error due to GA is a problem dependent based [24]. The analysis of overall GA performance was done using the parameter ranges set as in Table 2.

Table 2. Parameter Setting

Parameter	Value
Number of Population	10, 20, 50, 100, 120, 180
Crossover Rate	0.7, 0.8, 0.9, 0.95
Mutation Rate	0.001, 0.002,0.005, 0.01
Generation	20, 100

4.2. Computational Experiments According to Population Size

The different number of population size of 20, 50, 120 and 180 are evaluated using the LTP datasets consist of a range of 1-18 for time slots, 1-43 for the number rooms and 1-34 for the number subjects. The constant value of crossover rate = 0.9, mutation rate= 0.001, maximum number of generation of 20, tournament size of 5 are used. Both TE and T are evaluated. The result was demonstrated in Table 3. The performance of GA based on population size with respect to the Tournament (T) and Tournament selection scheme combined with Elitism (TE). The optimal generation and optimal solution for each population size was finally obtained after executing many experiments using different parameters. The population size of 180 achieved fitness value of 1.0 at generation 17 while population size of 50 achieved the highest fitness at generation 18 when employed TE selection. The population size of 50 achieved fitness value of 1.0 at generation 17 when applying T selection.

Table 3. Performance of TE and T using Different Population Size

Population size	TE		T	
	Generation No	Fitness Value	Generation No	Fitness Value
20	20	0.0477	20	0.0123
50	**18**	**1.0**	**17**	**1.0**
100	20	0.047	20	0.0476
120	20	0.0244	20	0.0071
180	**17**	**1.0**	20	0.0010

4.3. Computational Experiments according to Probability of Crossover

This section reports the results for different value of crossover rates of 0.7, 0.8, 0.9 and 0.95 are tested. Other parameters are fixed; population size: 10, mutation rate: 0.00, maximum number of generations: 30, tournament size: 5. The optimal generation and optimal solution for each crossover rates were done by running a few tests to get the best results. To summarize the above figures, a summarization of the optimal generation and optimal solution by each selection scheme is exhibited in Table 4. Crossover rate of 0.9 achieved the highest fitness value, 1.0 at generation 6 when using TE. Meanwhile the crossover rate of 0.9 achieved fitness value of 1.0 at generation 9 for T.

Table 4. Best Results for TE and T using Different Crossover Rates

Crossover rates	TE		T	
	Generation No	Fitness Value	Generation No	Fitness Value
0.7	30	0.0010	30	0.0041
0.8	30	0.0245	30	0.0099
0.9	**6**	**1.0**	**9**	**1.0**
0.95	30	0.00826	30	0.0164

4.4. Computational Experiments According to Probability of Mutation

Table 5 demonstrates the results of mutation rates of 0.005, 0.002, 0.001 and 0. Table 4.11 shows the result of the experiment for TE based on different mutation rates. A mutation rate of 0.005 achieved fitness value of 1.0 at generation 10 while a mutation rate of 0.01 able to achieve the highest fitness at generation 30. Table 5 shows the result of the experiment for T based on different mutation rates. A mutation rate of 0.01 achieved fitness value of 1.0 at generation 33 while a mutation rate of 0.005 achieved the highest fitness at generation 49. Both T and TE indicate an optimal solution for mutation rates 0.005 and 0.01 at generation number of 10 and 30, respectively.

Table 5. Best Results for T and TE Selection using Different Mutation Rates

Mutation rates	TE		T	
	Generation No	Fitness Value	Generation No	Fitness Value
0.005	**10**	**1.0**	**49**	**1.0**
0.002	50	0.0062	50	0.0123
0.001	50	0.0050	50	0.0071
0.01	**30**	**1.0**	**33**	**1.0**

4.5. Computational Experiments based on Average using T and TE

The use of GA with T and TE is further experimented for 50 experiments and average of fitness and penalty values were calculated and evaluated. The experiments were resulted from the application of mutation rate of 0.05, crossover rate 0.9 and elitist value of 1 and 2. Overall, table 6 shows that with more generation number in this case 100 generations, the average of both fitness and penalty value gives better results compared to only 20 generations.

The population size of 40 obtained the lowest average for penalty value and the highest average for fitness value. However, from 50 experiments, there are fitness value = 1.0 obtained from several experiments for all the three population sizes. The maximum generation of 100, revealed more frequent in achieving fitness value =1.0.

Table 6. Average of Computational Results based on 50 experiments

Maximum Generation	Elitism Value	Population size					
		30		40		50	
		Average Fitness Value	Average Penalty Value	Average Fitness Value	Average Penalty Value	Average Fitness Value	Average Penalty Value
100	1.0	0.48	**8.40**	**0.64**	**7.20**	**0.58**	**8.40**
20	1.0	0.42	20.40	0.27	17.04	0.40	16.86
20	2.0	0.29	18.40	0.27	19.62	0.25	29.20

The effect of different parameter tuning towards the performance of the GA in solving the Timetabling problem. In terms of minimization of sum of penalties to get the best fit for every solution, higher penalties will be given for hard constraint violation while lower penalties will be given for soft constraint violation for satisfying all the constraints. In terms of the best results produced by the GA, with respect to the parameter tuning and the selection schemes, findings showed that a GA with TE selection method achieved optimal generations faster than T selection method for the experiments based on population size, the probability of crossover and probability of mutation. However, for the experiment based on the number of generations, it achieved the best fitness at much later generation compared to the T selection method.

The findings also showed that a higher probability of crossover is better at achieving the optimal solutions while lower probability of mutation is highly encouraged to get the best solutions. It is also shown that the number of the population should not be very small or very high to achieve the optimal solutions. The results indicate that the selection schemes did not seem to affect the number of generations it converges since both selections are the Tournament selection except that one of them is combined with elitist. However, the results produced are not always constant as it is random and the GA tends not to converge. In addition, the termination criteria for the TE and T implementation are fixed to stop once the GA process reached maximum generation or the best fitness value, 1.0. GA is demonstrated at high potential optimization algorithm to solve LTP in a small and large scale.

5. CONCLUSIONS

The analysis of both selection schemes, T and TE and parameters tuning in terms of minimizing the sum of penalties were performed to see the effects of each selection scheme and different parameter tuning towards minimizing the overall sum of penalties to get the best fit for every solution in solving the timetabling problem. The finding has demonstrated that both T and TE implementation can offer a best fitness value, but the parameter must be carefully selected. The trial and error of using parameters as such mutation rate, crossover rates, population size and maximum number of generations. In future, a big dataset should be considered to see the capability of the GA of handling more complex optimization solutions.

ACKNOWLEDGEMENTS

The authors express a deep appreciation to Ministry of Education, Malayisia for the grant of 600-RMI/FRGS 5/3 (0002/2016), Institute of Research And Innovation, Universiti Teknologi MARA and the Information System Department, Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Malaysia for providing essential support and knowledge for the work.

REFERENCES

[1] Abdul-Rahman, S., Sobri, N. S., Omar, M. F., Benjamin, A. M., Ramli, R. *Graph coloring heuristics for solving examination timetabling problem at Universiti Utara Malaysia*. In *AIP Conference Proceedings*. 2014;1635(1):491-496. AIP.

[2] Babaei, H., Karimpour, J., Hadidi, A. A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 2015; 86: 43-59.

[3] Mahiba, A. A.,Durai, C. A. D. *Genetic algorithm with search bank strategies for university course timetabling problem*. Procedia Engineering; 2012. 38:253-263.

[4] Kumar, K., Sikander, R. S., & Mehta, K. Genetic Algorithm Approach to Automate University Timetable. *International Journal of Technical Research (IJTR*), 2012; 1(1): 47-51.

[5] Timilsina, S., Negi, R., Khurana, Y., & Seth, J. (2015). Genetically evolved solution to timetable scheduling problem. *International Journal of Computer Applications*, 2015; *114*(18): 12–17.

[6] Ganguli, R., & Roy, S. A Study on Course Timetable Scheduling using Graph Coloring Approach. *International Journal of Computational and Applied Mathematics*, 2017;*12*(2): 469-485.

[7] Sigl, B., Golub, M., & Mornar, V. *Solving timetable scheduling problem using genetic algorithms.* In 25th International Conference Information Technology Interfaces, 2003: 519-524.

[8] Burke, E. K., Elliman, D. G., & Weare, R. *A genetic algorithm based university timetabling system*. In Proceedings of the 2nd east-west international conference on computer technologies in education, 1994;1:35-40.

[9] Chu, S. C., Chen, Y. T., & Ho, J. H. *Timetable scheduling using particle swarm optimization*. First International Conference on Innovative Computing, Information and Control, 2006. ICICIC'06, 2006;3:324-327

[10] Sun, H., Chen, S. P., Jin, C., & Guo, K. (2013). Research and simulation of task scheduling algorithm in cloud computing. *Indonesian Journal of Electrical Engineering and Computer Science*, *11*(11), 6664-6672.

[11] Razak, H. A., Ibrahim, Z., & Hussin, N. M. (2010, March). Bipartite graph edge coloring approach to course timetabling. In *Information Retrieval & Knowledge Management, (CAMP), 2010 International Conference on* (pp. 229-234). IEEE.

[12] Yusoff, M., Ariffin, J., & Mohamed, A. (2015). DPSO based on a min-max approach and clamping strategy for the evacuation vehicle assignment problem. *Neurocomputing*, *148*, 30-38.

[13] Kendall, G., & Hussin, N. M. (2004, August). A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA university of technology. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 270-293). Springer, Berlin, Heidelberg.

[14] Cowling, P., Kendall, G., & Hussin, N. M. (2002, August). A survey and case study of practical examination timetabling problems. In *PATAT* (pp. 258-261).

[15] Alwadood, Z., Shuib, A., & Hamid, N. A. (2013, August). Mathematical rescheduling models for railway services. In *Proceedings of World Academy of Science, Engineering and Technology* (No. 80, p. 620). World Academy of Science, Engineering and Technology (WASET).

[16] Aziz, R. W. A., Shuib, A., Aziz, W. N. H. W. A., Tawil, N. M., & Nawawi, A. H. M. (2013). Pareto analysis on budget allocation for different categories of faculties in higher education institution. *Procedia-social and behavioral sciences*, *90*, 686-694.

[17] Altıntas, C., Asta, S., Ozcan, E., & Yigit, T. *A self-generating memetic algorithm for examination timetabling*, In 10th International Conference of the Practice and Theory of Automated Timetabling PATAT, York, United Kingdom, 2014:434-437.

[18] Thepphakorn, T., Pongcharoen, P., & Hicks. An ant colony based timetabling tool. *International Journal of Production Economics*; 2014; 149:131-144.

[19] Al-Betar, M. A., & Khader, A. T. A harmony search algorithm for university course timetabling. *Annals of Operations Research*, 2012;*194*(1): 3-31.

[20] Islam, T., Shahriar, Z., Perves, M. A., & Hasan, M. University Timetable Generator Using Tabu Search. *Journal of Computer and Communications*, 2016; 4(16): 28

[21] Soria-Alcaraz, J. A., Özcan, E., Swan, J., Kendall, G., & Carpio, M. Iterated local search using an add and delete hyper-heuristic for university course timetabling. *Applied Soft Computing*, 2016; *40*(C):581-593.

[22] Ahmad, I. R., Sufahani, S., Ali, M., & Razali, S. N. A Heuristics Approach for Classroom Scheduling Using Genetic Algorithm Technique. *Journal of Physics.* 2018;995(1): 012050.

[23] Soyemi, J., Akinode, J., & Oloruntoba, S. Electronic Lecture Time-Table Scheduler Using Genetic Algorithm, DataCom-CyberSciTec.2017; 124: 710-715.

[24] Alves, S. S., Oliveira, S. A., & Neto, A. R. R. A Recursive Genetic Algorithm-Based Approach for Educational Timetabling Problems. In Designing with Computational Intelligence, Springer, Cham. 2017: 161-175.