A Novel Educational Timetabling Solution through Recursive Genetic Algorithms

Shara S. A. Alves
Telecommunications Engineering Dept.
Federal Institute of Ceará, IFCE
Fortaleza, Ceará
Email: shara.alves@gmail.com

Saulo A. F. Oliveira
Telecommunications Engineering Dept.
Federal Institute of Ceará, IFCE
Fortaleza, Ceará
Email: saulo.freitas.oliveira@gmail.com

Ajalmar R. Rocha Neto
Telecommunications Engineering Dept.
Federal Institute of Ceará, IFCE
Fortaleza, Ceará
Email: ajalmar@ifce.edu.br

***Abstract*—This paper addresses the Educational Timetabling Problem for multiple courses. This is a complex problem that basically involves a group of agents such as professors and lectures that must be weekly scheduled. The goal is to find solutions that satisfy the hard constraints and minimize the soft constraint violations. Moreover, universities often differ in terms of constraints and number of professors, courses and resources involved which increases the problem size and complexity. In this work we propose a simple, scalable and parameterized approach to solve Timetabling Problems for multiple courses by applying Genetic Algorithms recursively, which are efficient search methods used to achieve an optimal or near optimal timetable.**

***Index Terms*—Educational Timetabling; Course Timetabling; Scheduling; Genetic Algorithms; Heuristics;**

I. Introduction

Timetabling problems are present in several enterprises and institutions. The main idea of this kind of problem is to set events into a number of time slots. Each event involves agents and may require some resources. Moreover, assignments would not violate institutions constraints which are categorized in soft and hard constraints [1]. The goal is to find solutions that satisfy the hard constraints and minimize the soft constraints violations. Educational Timetabling Problem is the most popular timetabling problem and is an NP-hard problem, which means that the amount of computation required to find solutions increases exponentially with problem size [2]. Therefore, efficient search methods to achieve an optimal or near optimal timetable are highly desirable.

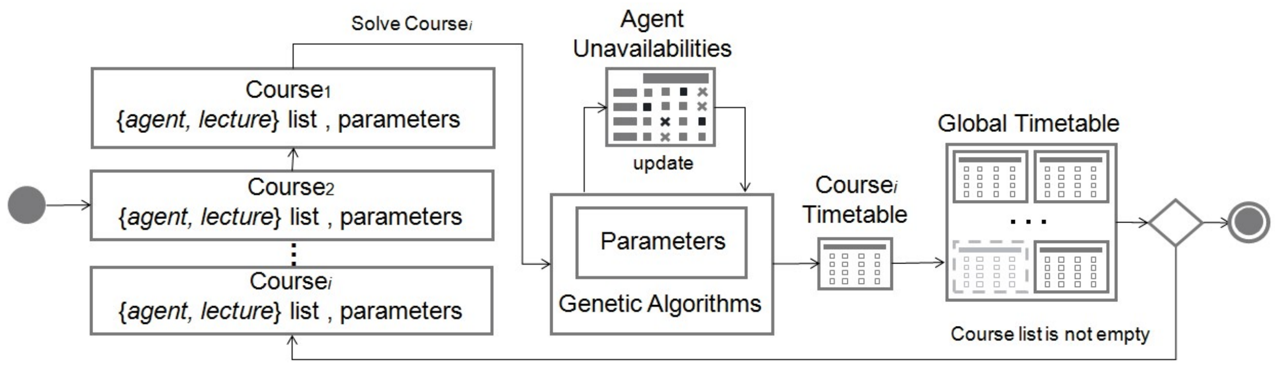
Educational Timetabling Problem is generally divided in three main types, to wit, (i) the school timetabling that is weekly for all school lessons and avoid agents meeting two lessons at the same time, (ii) the course timetabling that is weekly for all the lectures of a set of university courses and minimize overlapping lectures of courses having common students and (iii) the examination timetabling for the exams of a set of university courses, spreading them for the students as much as possible and avoiding overlap of course exams having common students [1].

Genetic Algorithms (GAs) are a search meta-heuristic method inspired by natural evolution, such as inheritance, mutation, natural selections and crossover [3]. This meta-heuristic can be used to generate useful solutions to optimization problems. Due to the characteristics of GAs methods, it is easier to solve few kind of problems by GAs than other mathematical methods which do have to rely on the assumption of linearity, differentiability, continuity, or convexity of the objective function [4].

Educational Institutions environments differ in terms of constraints, classes (group of students), rooms, concurrent courses, number of agents and their unavailabilities. According to the Institutions, some constraints are more important than the others. The related works often focus on one of the three main types of problem and on events involving some specific constraints, as an example the student constraints. The student constraints are commonly contemplated because students can not attend to different events at the same time [5], [6], [7]. On the other hand, some works presuppose classes to be disjoint [8]. Nowadays, a promising constraint is the agent unavailabilities since it is common the agents work at different places [9], [10].

In this work, we proposed a simple, scalable and parameterized model with GAs to solve timetabling problem for multiple courses. Such suitable model is capable to deal with different courses since universities differ from each other in number of courses and often courses lessons may change depending on the shift. The constraints are embed in fitness function and can be added or removed easily. The soft and hard constraints considered in this work were defined after an environment analysis held at Federal Institute of Ceará. The soft constraints are related to (i) adjacent lectures: more than 3 lessons in a row and (ii) agents unavailabilities: agents unavailable due other activities. The hard constraints comprehend (i) agents matches when agents are assigned to concurrent events and (ii) same course semester lectures overlapping, which minimize overlapping lectures of courses having common students. Our model solves course timetabling problems through GAs recursive executions, and as a result a global timetable is obtained.

The remaining part of this paper is organized as follows. In the Section II, we expose some relevant related works. Then, we present our proposal in Section III. After that, in Section IV, we describe the experiments carried out. Finally, some conclusions remarks and future works are represented in Section V.



Solve Course$_i$
Agent Unavailabilities
update
Course$_1$ {agent, lecture} list , parameters
Course$_2$ {agent, lecture} list , parameters
Course$_i$ {agent, lecture} list , parameters
Parameters
Genetic Algorithms
Course$_i$ Timetable
Global Timetable
Course list is not empty

Fig. 1: Overview of our Model.

II. Related work

A large number of approaches have been proposed for solving Educational Timetabling Problems [11]. Most related approaches work on school and course timetabling. Spreading exams is another interesting problem to be solved, but not often the focus in most educational institutions. We found solutions with Simulated Annealing ([8], [7]), Meta-heuristic methods [12], Memetic Algorithms ([13], [6]), Genetic Algorithms ([14],[9],[10]) and Graph-based ([5], [15]).

As an example, Borges's approach [9] addressed to the university timetabling problem with 8 classes and 33 agents. In such work, a mechanism for avoiding stagnation was employed, but the population size was 1000 and the time to achieve the best solution was not informed. In addition, the same timetabling problem was also covered by Ramos [10] with 14 classes, 21 agents and 10 rooms. We highlight the number of generations and runtime necessary to carry out such model, which was 15000 generations and 35 up to 90 minutes, respectively. We draw attention to the best fitness achieved by Ramos' model, which was 0.5 in $[0, 1]$. Furthermore, both considered agents unavailabilities constraint.

III. Proposal

We aim to propose a simple scalable and parameterized model using Genetic Algorithms (GAs) to solve timetabling problem for multiple courses. Our model suits the amount of courses, number of semesters, days and lessons per day. The global timetable is the solution, which is obtained by applying GAs recursively so that each execution solves one course. We present our proposal from this point on.

First, the agents inform their unavailabilities in terms of period of time. Each course contains number of days and lessons per shift and a 2-tuple list $[lecture, agent]$. The model receives as input a list of courses, the agent unavailabilities and the GAs parameters, to wit, number of generations, mutation and crossover rate. The global timetable is achieved by recursive executions. As stated before, each GAs execution solves one course timetable and generates new assignments. These new assignments are used to update the agent unavailabilities for the next execution. Thus, after having several executions, we achieve a global timetable solution, which takes into account each and every agent unavailabilities obtained as depicted in Fig. 1.

A. Genetic Representation

In Brazil, a course is divided in semesters, for example, a undergraduated course that lasts 4 years will have 8 semesters. Thus, our individual is composed by m chromosomes, which each chromosome represents a course timetable. Each timetable is divided into semesters which is a group of daily time slots as depicted in Fig. 2.

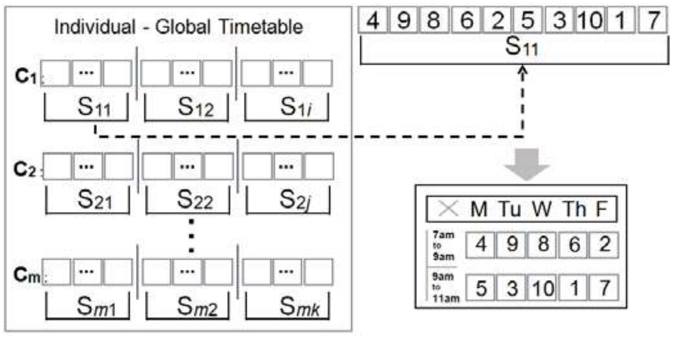


Fig. 2: Genetic Representation.

In our representation we adopted integer genes to compose the chromosome and the chromosome size is defined by the number of semesters and daily time slots. Each time slot, which is a single gene, comprehends 2 periods of time since most lectures has two lessons at least. The gene value is unique and refers to a 2-tuple $[lecture, agent]$ or a free time in timetable.

We designed this genetic representation due its simplicity and easy verification of further restrictions, such as, agents matches. Moreover, it already avoids lectures of a same semester overlap since each gene position is related to an unique time slot.

B. Genetic Operators

The genetic representation does not allow repeated gene values, thus a crossover operator that follows this restriction is appropriate. Among the popular crossover operators, the Single Point [16], Multi-Point [3] and Uniform Crossover [17] operators violate this prerogative. However, there are operators which change the order or arrangement of genes, such as OX [18], CX [19] and PMX [20]. This type of operator is called permutation operator and generates feasible individuals.

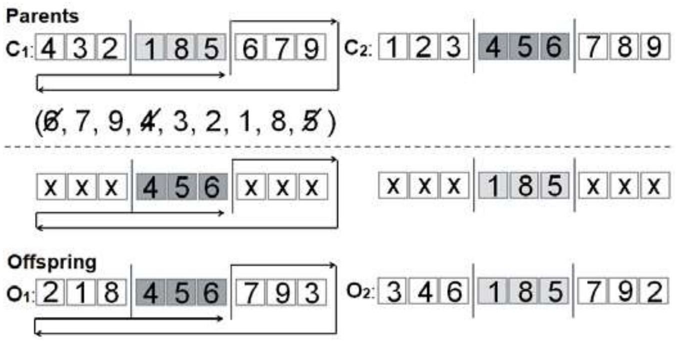

Parents
C1: 4 3 2 | 1 8 5 | 6 7 9
C2: 1 2 3 | 4 5 6 | 7 8 9
(6, 7, 9, 4, 3, 2, 1, 8, 5)
x x x | 4 5 6 | x x x
x x x | 1 8 5 | x x x
Offspring
O1: 2 1 8 | 4 5 6 | 7 9 3
O2: 3 4 6 | 1 8 5 | 7 9 2

Fig. 3: OX Crossover Operator Default Implementation.

In a performance comparison it was found that OX works more effectively than the others for producing feasible course timetables [21]. We present how OX operator works in Fig. 3. Nevertheless, for our genetic representation, the default OX implementation generates invalid individuals. Due to the operator to be applied to the whole individual, it would blend $[lecture, agent]$ tuples from different semesters. To overcome this drawback, we employed a slight modification by only crossing over semester block pairs, so now it works properly. Our modification is depicted in Fig. 4.

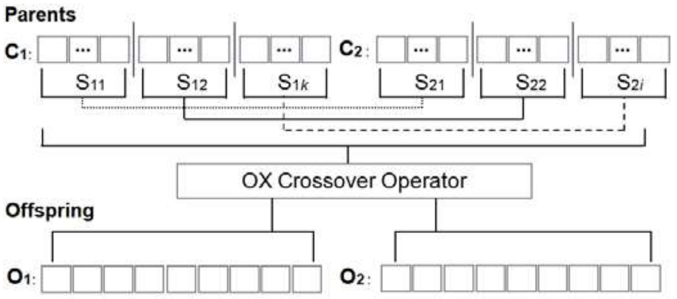


Fig. 4: OX Crossover Operator Modified.

As for the mutation operator, we carried out some modifications on the Swap Mutation Operator [22]. This modified operator mutates only a random semester block, swapping one gene to another as depicted in Fig. 5. Such behavior also avoids mixture of $[lectures, agent]$ tuples from different semesters.

C. Fitness Function

The fitness function evaluates a course timetable based on its soft or hard constraints violations and it is defined as follows

$$F(C) = 1 - \frac{AM_C + AU_C + AL_C}{AM_{wc} + AU_{wc} + AL_{wc}}, \tag{1}$$

where the numerator (AM_C, AU_C and AL_C) indicates the constraints violated by C and the denominator is the worst case

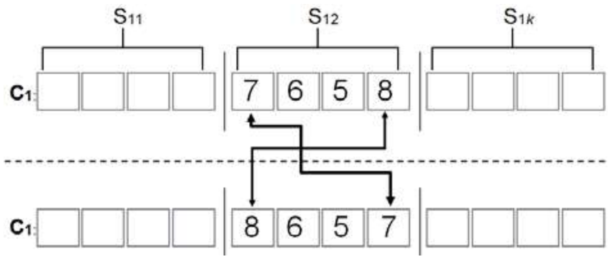


Fig. 5: Swap Mutation Operator.

of each constraint (AM_{wc}, AU_{wc} and AL_{wc}). AM stands for the number of agents matches, AU for agents unavailabilities and AL for the adjacent lectures. The fitness value is $\in [0, 1]$, in which values close to 1 represent good solutions.

D. Algorithm

In this subsection, we present two algorithm versions of our proposal. The first one is an iterative version, named SOLVE-ITERATIVE-TP(.) and the second version is a recursive one, called SOLVE-RECURSIVE-TP(.).

The algorithm parameters for the SOLVE-ITERATIVE-TP(.) are the list of courses (LC), the GAs parameters (GAP) and the agent unavailabilities (U). As a result, we have a global timetable solution (T). In the beginning, a course C from LC is selected and then solved by the GA. After that, we have a partial solution S and the new agent assignments. These assignments are used to update the agent unavailabilities, as well as the partial solution S is used to update the global timetable T. These steps are executed until LC being empty.

```
SOLVE-ITERATIVE-TP($LC, GAP, U$)
    $LC$ :      list of courses
    $GAP$ :     parameters for GENETIC-ALGORITHM
    $U$ :       agents unavailabilities

1  $T \leftarrow empty$ ▷ the global timetable solution
2  for $j \leftarrow 1$ to LENGTH($LC$)
3       do
4           $C \leftarrow LC[j]$ ▷ course to be solved
5           $S \leftarrow empty$ ▷ timetable solution for $C$
6           $S \leftarrow$ GENETIC-ALGORITHM($C, U, GAP$)
7           $U \leftarrow$ EXTRACT-ASSIGNMENTS($S$) $\cup U$
8           $T \leftarrow T \cup S$ ▷ adds the found solution $S$ to $T$
9  return $T$
```

The parameters for the SOLVE-RECURSIVE-TP(.) are the list of courses (LC), the GAs parameters (GAP), the agent unavailabilities (U) and the global timetable (T). As a result, we have a global timetable solution (T) filled with the partial solutions S after each recursive execution. These recursive executions still until LC being empty.

```
SOLVE-RECURSIVE-TP($LC, GAP, U, T$)
    $LC$ :      list of courses
    $GAP$ :     parameters for GENETIC-ALGORITHM
    $U$ :       agents unavailabilities
    $T$ :       the global timetable solution

1  if EMPTY($LC$)
2      return $T$
3  $C \leftarrow$ POP($LC$) $\triangleright$ course to be solved
4  $S \leftarrow empty$ $\triangleright$ timetable solution for $C$
5  $S \leftarrow$ GENETIC-ALGORITHM($C, U, GAP$)
6  $U \leftarrow$ EXTRACT-ASSIGNMENTS($S$) $\cup U$
7  $T \leftarrow$ SOLVE-RECURSIVE-TP(LC,GAP,U,T)$\cup S$
8  return $T$
```

The order the courses are selected is based on its complexity in terms of how many tuples it has. Thus, courses having more tuples are solved first.

IV. Experimental Results

A. Experiment Setup

To further validation of our proposal, we carried out some simulations by using two different environments from Federal Institute of Ceará, as presented in Table I. The environments are divided in two cases, namely, complex and simple. The difference is that in the simple case the agents unavailabilities were not informed.

TABLE I: Environment: courses, shifts, number of semesters, agents and unavailabilities in Complex and Simple cases.

Course	Shift	#Semesters	#Agents	#Complex	#Simple
Comp. Sci.	M	8	21	28	0
Comp. Net.	M	1	5	8	0
Comp. Sci.	A	1	4	12	0
Informatic	A	4	10	13	0
Comp. Net.	N	2	5	2	0

In order to evaluate the GAs performance and find the GAs parameters, it was carried out 10 executions of 15 tests in each environment by changing GAs parameters. Also, we highlight that the stop criteria were: find fitness equals to 1 and stall time limit of 10 minutes. Furthermore, we analyze the fitness values over the generations and the final population on the more complex scenario we have, the Computer Science course (morning).

B. Discussion

Observing the results in Table II, for all tests carried out with larger populations (600) on complex case, not all of their 10 executions could achieve the best solution (see column $\#fitness = 1$). Besides that, the average runtime was no less than 5 minutes for those who achieved it. Thus, we support that such parameters are expensive. Among the others tests, the test numbered 10 achieved the best solution in all executions with a runtime average of ≈ 3.7 minutes. Others results achieved best solution in less runtime average, but they did not succeed in most of the executions of Table II. Therefore, we support the test numbered 10 is more reliable, hence its parameters were defined to GAs.

As for the tests carried out in simple case, they outperformed those ones in complex case. The achieved results took less than 1 minute in average runtime, so we opt to not present all of them. Through these results, we are convinced of the unavailabilities previously informed increase the complexity of the problem.

The effectiveness of our model is proved by its higher initial fitness average value and its fast convergence as presented in Table III. Moreover, the initial population of the most complex scenario solved has already an individual with fitness value of ≈ 0.81 and after only 30 generations the higher fitness is increased to 0.98, see Fig. 6. Such model behavior is a consequence of the GAs configuration that was chosen and mainly due TO our genetic representation because it generates valid individuals that only violates some constraints.

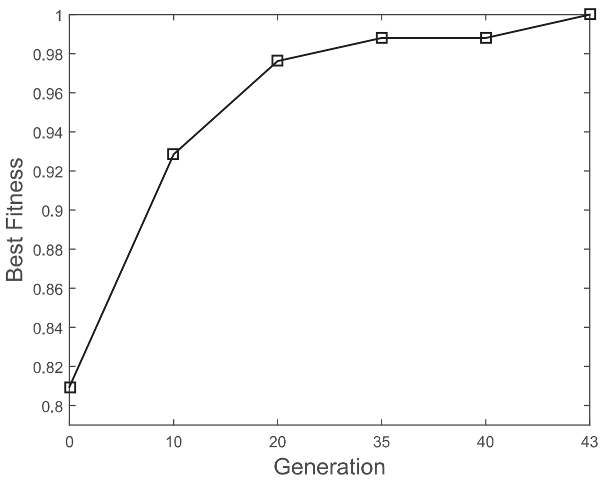


Fig. 6: Best Fitness Over The Generations.

Furthermore, by investigating the final population, we found 5 individuals with fitness equals 1 and the average fitness of each 5 ascendant ordered individuals is ≥ 0.98, as shown in Fig. 7. Actually, our model not only solves the problem, but it offers more than one feasible solution.

It is difficult to compare our results with others due to environment complexity divergence. However, we found two similar works in which the divergences in terms of number of courses, classes, agents and constraints are minimal [9], [10]. Both compared works solved complex and real course timetabling problem, taking into account agents unavailabilities too. They also defined the GAs parameters after carrying out some tests. We present such comparison in Table IV.

Based on Table IV, we summarize the advantages and disadvantages of such works. Concerning the population size, we reinforce that larger populations are very expensive. This is confirmed through our small population size. Also, we highlight that our crossover operator helped the fast convergence against Ramos'. Usually the number of generations is used

TABLE II: Tests results: mutation rate, crossover rate, population size, average of generations, number of executions which achieved fitness value equals 1 and the average runtime

Test	Mutation rate (%)	OX Crossover rate (%)	Population Size	μ Generations	$\# fitness = 1$	μ Runtime (min)
COMPLEX CASE						
1	1/12	35	75	391.1	6/10	0.66
2	1/12	35	150	161.6	7/10	1.85
3	1/12	35	600	63.4	8/10	5.87
4	1/12	50	75	389.7	7/10	2.28
5	1/12	50	150	299.7	9/10	3.25
6	1/12	50	600	83.4	5/10	7.4
7	1/40	50	75	493	5/10	1.8
8	1/40	50	150	141.6	9/10	2.66
9	1/40	50	600	68, 9	6/10	6.33
10	**1/25**	**50**	**75**	**493**	**10/10**	**3.7**
11	1/25	50	150	222.2	8/10	4.5
12	1/25	50	600	78	6/10	7
13	1/25	60	75	510.1	6/10	4
14	1/25	60	150	191.3	8/10	3.25
15	1/25	60	600	75.5	6/10	6.16
SIMPLE CASE						
1	1/25	50	75	17.1	10/10	0.21
2	1/40	50	150	14.7	10/10	0.37

TABLE III: Fitness Average of Population

Generation	$\mu Fit\frac{1}{3}Pop$	$\mu Fit\frac{2}{3}Pop$	$\mu Fit\frac{3}{3}Pop$
Comp. Sci. M			
0	0.7614	0.7390	0.7136
10	0.9195	0.9138	0.9107
20	0.9590	0.9557	0.9536
30	0.9790	0.9776	0.9771
40	0.9880	0.9880	0.9880
43	0.9909	0.9895	0.9890
Comp. Net. M			
0	0.8577	0.7977	0.7303
1	0.96	0.9177	0.8711
Comp. Sci. A			
0	0.875	0.836	0.7875
2	0.9475	0.9425	0.9241
Informatic A			
0	0.8090	0.7799	0.7453
10	0.9677	0.9677	0.9677
13	0.9703	0.9690	0.9686
Comp. Net. N			
0	0.8799	0.8466	0.7955
1	0.9500	0.9166	0.8888

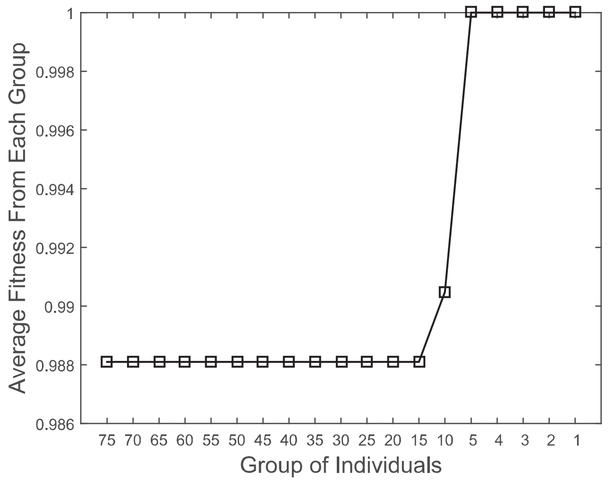


Fig. 7: Grouped Individuals Fitness Average of 43^{th} Population.

to measure speed and robustness of discovering an acceptable solution, such aspect implies directly on the runtime. Thus, we remark those that require a higher number of generations require more runtime as well. Even our model having a number of generations exceeding Borges', the contrast of population size confirms our effectiveness.

TABLE IV: Comparison with Related Work

Parameter	Ours	Borges' [9]	Ramos' [10]
Number of Classes	16	8	14
Number of Agents	21	33	21
Number of Rooms	$\times$	$\times$	10
Population Size	75	1000	100
Crossover Operator	OX	PMX	Single Point
Crossover Rate	50%	50%	70%
Mutation Rate	1/25	1/20	0.002
Selection Strategy	Random	Roulette Wheel	Tournament
Elitism	$\times$	10%	$\times$
Stagnation Monitor	$\times$	Yes	$\times$
Generations	493	201	≈ 15000
Runtime (in minutes)	≈ 3.7	$\times$	35-90
Fitness $[0, 1]$	1	1	0.5

V. Conclusion

This paper presented a simple, scalable and parameterized model to solve timetabling problems for multiple courses by applying Genetic Algorithms (GAs) recursively so that each execution solves one course. Universities often differ to another in number of courses and these to each other in terms of lessons per shift, for example. Such model is capable to deal with different number of courses and their features. It updates the agent unavailabilities with the new assignments after each execution. The model was compared to similar works and the results indicate that our model took less time than the others. Also worth mentioning the model repeatedly finding feasible solutions in the majority of the trials and not only, but find more than one feasible solution.

Future work will be aimed to enhance GAs processes, e.g., include a stagnation monitor, a heuristic for initializing population, constraints weighted, consider rooms and allow user interventions. Besides, we intend to improve our model to be capable to also solve school timetabling problems.

References

[1] A. Schaerf, “A survey of automated timetabling,” *Artificial Intelligence Review*, vol. 13, pp. 87–127, 1999.

[2] A. I. S. Even and A. Shamir., “On the complexity of timetabling and multicommodity flow problems.” *SIAM Journal of Computation*, pp. 691–703, 1976.

[3] D. E. Goldberg, *”Genetic Algorithms in Search, Optimization and Machine Learning”*. ”Hardcover”, 1989.

[4] L. Yu, H. Chen, S. Wang, and K. K. Lai, “Evolving least squares support vector machines for stock market trend mining,” *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 1, pp. 87–102, Feb 2009.

[5] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, “A graph-based hyper-heuristic for educational timetabling problems,” *European Journal of Operational Research*, vol. 176, no. 1, pp. 177–192, 2007.

[6] S. N. Jat and S. Yang, “A memetic algorithm for the university course timetabling problem,” in *Tools with Artificial Intelligence, 2008. ICTAI’08. 20th IEEE International Conference on*, vol. 1. IEEE, 2008, pp. 427–433.

[7] M. Tuga, R. Berretta, and A. Mendes, “A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem,” in *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*. IEEE, 2007, pp. 400–405.

[8] D. Abramson, “Constructing school timetables using simulated annealing: sequential and parallel algorithms,” *Management science*, vol. 37, no. 1, pp. 98–113, 1991.

[9] S. K. Borges, “Resolução de timetabling utilizando algoritmos genéticos e evolução cooperativa,” Master’s thesis, Universidade Federal do Paraná, 2003.

[10] P. de Siqueira Ramos, “Sistema automático de geração de horários para a ufla utilizando algoritmos genéticos,” 2002.

[11] R. Qu, E. K. Burke, B. McCollum, L. T. Merlot, and S. Y. Lee, “A survey of search methodologies and automated system development for examination timetabling,” *Journal of scheduling*, vol. 12, no. 1, pp. 55–89, 2009.

[12] R. H. de Jesus Alves, “Metaheurísticas aplicadas ao problema de horário escolar. dissertação,” Master’s thesis, Centro Federal de Educação Tecnológica de Minas Gerais, 2010.

[13] A. M. Coelho, “Uma abordagem via algoritmos memáticos para a solução do problema de horário escolar. dissertação,” Master’s thesis, Centro Federal de Educação Tecnológica de Minas Gerais, 2006.

[14] G. N. Beligiannis, C. Moschopoulos, and S. D. Likothanassis, “A genetic algorithm approach to school timetabling,” *Journal of the Operational Research Society*, vol. 60, no. 1, pp. 23–42, 2009.

[15] R. Qu and E. K. Burke, “Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems,” *Journal of the Operational Research Society*, vol. 60, no. 9, pp. 1273–1285, 2009.

[16] J. H. Holland, *”Adaptation in Natural and Artificial Systems”*, 1975.

[17] G. Syswerda, “Uniform crossover in genetic algorithms,” in *Proceedings of the Third International Conference on Genetic Algorithms*, J. D. Schaffer, Ed. Morgan Kaufmann, Jun. 1989, pp. 2–9.

[18] L. Davis, “Applying adaptive algorithms to epistatic domains.” in *IJCAI*, vol. 85, 1985, pp. 162–164.

[19] I. M. Oliver, D. J. Smith, and J. R. C. Holland, “A study of permutation crossover operators on the traveling salesman problem,” in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987, pp. 224–230. [Online]. Available: http://dl.acm.org/citation.cfm?id=42512.42542

[20] D. E. Goldberg and R. Lingle, “Alleles, loci, and the traveling salesman problem,” in *Proc. of the International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, 1985, pp. 154–159.

[21] W. Chinnasri, S. Krootjohn, and N. Sureerattanan, “Performance comparison of genetic algorithm’s crossover operators on university course timetabling problem,” in *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, vol. 2. IEEE, 2012, pp. 781–786.

[22] L. Davis, *”Handbook of genetic algorithms”*. ”Van Nostrand Reinhold”, 1991.