2015 IEEE 14th International Conference on Machine Learning and Applications

Solving the Academic Timetable Problem Thinking on Student Needs

Maria Weslane Sousa Almeida[†], João Paulo Souza Medeiros[*], and Patrícia Rufino Oliveira[‡]
[‡]Escola de Artes, Ciências e Humanidades (EACH)
[†‡]Universidade de São Paulo (USP)
[*]Department of Computing and Technology (DCT)
[*]Universidade Federal do Rio Grande do Norte (UFRN)

***Abstract*—The assembling process of an academic timetable structure is one of hardest scholar planning tasks. A well done schedule requires considerable time, because of many factors involved (e.g. professor availability, courses and their workload, students and classrooms). Based on related researches, it is possible to claim that in the most of the institutions, the solution is done by manual work, which requires more time and effort. Although the search for the best solution is infeasible, it is possible to find near optimal solutions using heuristics. Most current solutions provide answers optimizing administrative factors, i.e. considering mainly the factors related to the disciplines, classrooms and professors, not considering students needs, e.g. reducing time gaps between non-consecutive classes. In this work, we focus on the timetable improvement for the students and assess our results using real data from a Brazilian university.**

I. Introduction

The timetable problem is one of the most important works in the school calendar planning. Also, it became an important and classic problem in the constrained optimization literature [1]–[5]. The school or academic timetable problem is to arrange, in a determinate period of time, a set of classes in a timetable, which must meet the rules associated to the curriculum of the course and the staff's agenda.

To elaborate an academic timetable it is necessary to consider a set of factors: (i) the available schedule of each professor, (ii) the disciplines that will be offered, (iii) the student needs, (iv) how many classes the course will have in a week, and (v) classrooms availability [6].

Although the solution could be produced by manual work (using some heuristics and by trial and error), the effort is considerable and the feasibility is impractical as the problem input scales. Moreover, computational solutions to find the best structure are also impractical, since the problem was proved to be NP-hard [7]. Thereafter, the problem should be solved using heuristics ans stochastic search procedures. From the many possible techniques, in this work we used a Genetic Algorithm (GA). It is important to note that an heuristic does not make sure that the best solution will be found, but the algorithm probably will find a reasonable good solution [8].

Genetic algorithms, from evolutionary computing, use heuristics based on Darwin's evolution theory [9] with genetic concepts as sexual reproduction, inheritance and mutation [8]. Genetic algorithms have an outstanding performance in finding an acceptable solution to this kind of problem [10].

The objective of this work is to propose a genetic algorithm to solve the timetable problem considering the students needs. This differs from other works (e.g. [10]) in the finding of an acceptable solution that take into account various aspects of students productivity. This could be achieved, for example, grouping classes, which can minimize students spare time and improve their performance [11].

This paper is organized as follows. The concepts about the genetic algorithm are presented in Section II. In Section III, we explain the proposed algorithm. The experiments and results are described in Section IV. We present our conclusion in Section V.

II. Genetic Algorithm

The studies about artificial intelligence began with the Allan Turing work about learning machines. Turing affirmed that the machine works like the human thinking (behavior) [12].

Years later, the genetic algorithm was proposed by Holland, who explained how it works and what is necessary to an algorithm to be considered a genetic one. In his work [13], Holland adapted the Darwin's theory and genetics to describe the algorithm behavior.

A genetic algorithm is a heuristic technique that looks for acceptable solutions to a problem domain. In genetic algorithms there is a population, composed by a number of individuals. Those individuals are selected for reproduction according to their aptitude, creating new individuals with inherit their parents characteristics. This produces a new population which can replace at some level the old one. This process is repeated until some stopping criteria is meet.

The genetic algorithm nomenclature is analogue to the natural one. For example, a chromosome represents the individual proper characteristics, composed by genes. The genes are the information about the problem in a data structure (e.g. a vector or a binary word) which represents a chromosome. Each chromosome can be a feasible solution to the problem and has an adaptation value associated to it, called fitness.

To generate the new population, the reproduction is sexual, and two chromosomes are selected according to their adaptation value. The chosen ones have their genes combined (crossover) to create a new chromosome (offspring). Then, the chromosome may mutate according to a specified mutation rate. This process is repeated until a new population is completely generated.

978-1-5090-0287-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ICMLA.2015.184

The algorithm designer has to choose a data structure that represents the chromosome, which has to describe a feasible solution to the problem. Then, he can previously choose the mutation rate, the number of generations (number of iterations as a stopping criteria), and how the population substitution will happen (e.g. with elitism or a substitution rate). These aspects will directly influence the quality of solutions and the performance of the algorithm. In addition to these general rules, the algorithm designer could do some adjustments. This is justified by the fact that the concerned problem may have specific requirements which, for example, may not be meet using standard crossover and mutation operators [14].

III. Proposed Solution

Our proposed genetic algorithm was designed using information retrieved from interviews with an coordinator [6] and a researcher graduated in pedagogy [11]. This was done to understand the specific requirements and design a genetic algorithm that uses as much information as possible. The case study was applied to data from the course of Bachelor's degree in Information Systems of the Federal University of Rio Grande do Norte (UFRN).

From the interviews we extracted the following representative information: (i) the curricular structure of the course; (ii) the set of professors and their respective availability; (iii) the disciplines which will be offered; (iv) the current semester (the year is divided into two periods of six months), (v) the classes schedule; and (vi) the number of classes that each discipline will have per week. In addition, a discipline can be offered to more than one set of students, which can be consider as a team (class) to differ one from other. Some information was ignored because they were not relevant for the case. For example, the number of available rooms was always sufficient and the number of students for each discipline was always less than the capacity of all rooms.

A. Chromosome representation

Considering the understandings gained based on interviews we designed the GA solution representation, the chromosome. In our case, the course is daytime and it is composed by eight semesters (half year periods). Each semester has specific pre-defined disciplines. Sometimes, these disciplines has to be offered to students that delayed their progress in the course. According to the politics adopted by the professors set, the disciplines offered to the students who were not approved will be placed at the morning. The regular classes will be placed at the afternoon blocks.

Each chromosome has to describe the structure of a set of periods. Each period has information about the team (or class, i.e. a discipline instance) and its schedule. Each team has information about the discipline, the current semester, the professor, and how many time it will happen in a week.

To better arrange all this information, the chromosome structure is organized in other data structures. Therefore, a chromosome is a vector, which each position represents a period description. A period description is a matrix which represents the week's day (column), and blocks to allocate the classes (line). The vector structure is depicted in Figure 1.

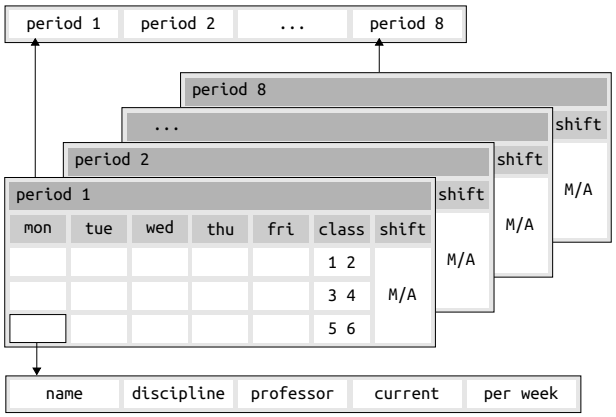

Fig. 1. Chromosome structure.

The "M/A" information in the "shift" column of each period characterize if the period will be offered in the morning or afternoon. Each shift has six time slots which cloud be used to place an discipline block of two class hours.

B. Search space

In a timetable problem, there is a set of restrictions and a number of classes, allocated in blocks, for each period. The chromosome is composed of eight periods (a vector). If we equate these variables to describe the size of the search space S, we will have

$$|S| = \sum_{p=1}^{n} \frac{e!}{(e - b_p)!}, \tag{1}$$

where S is the search space; p is the period index; n is the total number of periods (between 4 and 8, in our case study); b_p is the number of classes that will have in each week of a period p (between 1 and 15, in our case); and e is the size of the data structure of each period (in our case 15 blocks).

In the design process, we used the classic genetic algorithm with some adjustments. These adjustments were mainly applied to the genetic operators in order to satisfy restrictions the should be meet for feasible solutions. The genetic operators used were crossover and mutation.

The initial population was randomly generated. Specifically, we randomly generated a chromosome and apply permutations to create other chromosomes until complete the population size.

C. Crossover operator

Due to the problem characteristic, the crossover operator used was the crossover in order. This crossover operator uses the concepts of sexual reproduction of two chromosomes to produce a permutation using data from both parents. To get the parent information, it was used an auxiliary structure, called gene selection matrix, which is a random binary matrix. When the value is 1, the son will inherit the gene from the first parent chromosome. Later, the other genes are also inherit from the first parent chromosome, but using information presented in the second. The structure of this matrix is presented in Figure 2.

mon	tue	wed	thu	fri
1	1	0	0	1
0	1	1	0	0
1	0	0	1	0

Fig. 2. Gene selection matrix.

The process of block selection using the gene matrix and the first parent chromosome is illustrated in the left of Figure 3, resulting in the partially filled chromosome (1).

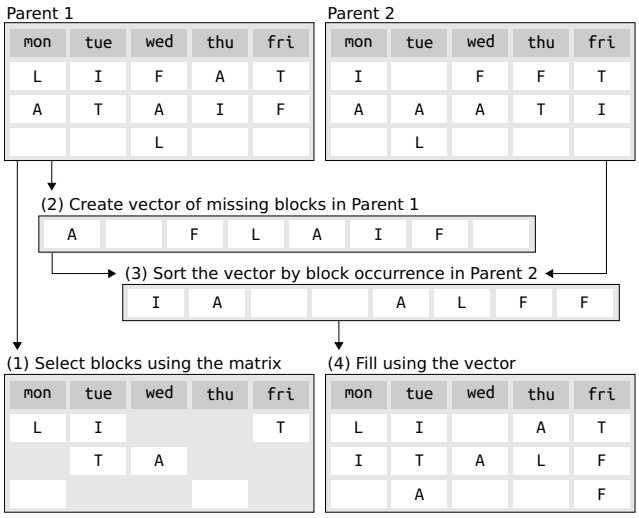

Fig. 3. In order crossover. Empty blocks represents vacancies and the letters stands for discipline names: (L)ogic, (A)lgorithms, (I)ntroduction to informatics, (T)heory of administration and (F)undamentals of mathematics.

The blocks not selected by the gene selection matrix are chronologically placed in a vector, the missing blocks vector (2) in Figure 3. This vector is sorted using the order of blocks sequence in the second parent chromosome, resulting in the ordered vector (3) in Figure 3. Finally, we use the sequence of blocks in the ordered vector (3) to fill in chronological order the missing slots of the offspring chromosome (1), resulting in the full filled offspring chromosome (4) in Figure 3.

This operation was done for each period in the full chromosome (composed by at least four periods). The crossover process was repeated in the population until a new generation has been created with the same size of the old population. In the experiments, all chromosomes from the old population were substituted by the new ones.

D. Mutation operator

Because of some restrictions, sometimes the crossover operator does not produce feasible solutions. To solve it, our mutation operator has a slight different purpose. It was used as a corrector, making small changes in the chromosome in order to meet the restrictions.

This operator tries to meet all restrictions. The mutation operator is responsible to check if there are conflicts is classes or professors availability. If a conflict is detected, the chromosome is submitted to a permutation. The same is done for all the population individuals in each generation. For example, individuals (professors or students) can not be in different classes at the same time. The mutation does a permutation in conflicting blocks aiming to eliminate the schedule shock.

E. Fitness function

The fitness function is responsible to give to the chromosome its adaptation value in accordance with the rules of the problem. In this work, this function is a sum of punishments. When a chromosome does not meet a requirement, or have undesired characteristics, some value must be added to its adaptation value. Therefore, our problem should be optimized by the minimization of the fitness function. The smaller is the adaptation value bigger is the chance of the chromosome be selected to reproduce.

To create the function according to this objective, some criteria were adopted to get a schedule with grouped blocks, which will support the students productivity [11], [15].

- Avoid last day blocks: most of the students live in other cities and they depends on scholar buses that has specific time to come and go back to their cities. Sometimes, these buses should go near to the final of the last class hour. Therefore, it is desired to minimize the use of the last blocks in the schedule matrix.
- Gaps between classes: since not all students are engaged with other academic activities (e.g. research), mainly at the beginning of the course, they may be dispersed if there are long time gaps between classes.
- Same discipline classes in a day: spend long time doing the same thing is exhausting. This may happen if the students have to spend a long time studying or watching consecutive classes of the same discipline.
- Professors' preference: professors should coordinate teaching with other academic activities (e.g. research). Therefore, they could prefer some classes arrangements than others.
- Schedule shock: although the mutation operator tries to correct this case, it can happen again after the permutation mutation, because of this, if a shock is detected, the chromosome adaptation value is raised.

Given the criteria just described, we may define the chromosome fitness function as

$$F(c) = s(c) + \left[\sum_{p=1}^{n} a(c_p) + v(c_p) + u(c_p) + t(c_p)\right], \quad (2)$$

where c is the chromosome to be valued; $s(\cdot)$ is the number of shocks in the schedule; n is the total number of periods; p is the period index; $a(\cdot)$ is the number of gaps before regular classes; $v(\cdot)$ is the number of gaps between regular classes; $u(\cdot)$ is the number of classes offered in the last block of the day; and $t(\cdot)$ is the number of preferences that were not meet.

IV. Results

In this work we used data from the course of Bachelor's degree in Information Systems of the Federal University of Rio Grande do Norte (UFRN). The best solution proposed by our genetic algorithm is compared with the manual solution done by the current and previous coordinators.

We analyzed data from the first semester of 2012 (or 2012.1 for short) to the second semester of 2015 (or 2015.2 for short). For all simulations, the population size used was 100 and the stop criterion was the number of generations (until 2000).

For each simulation, we evaluated the algorithm performance and the fitness mean of the population. The results are presented in Figure 4 for the first semesters, and in Figure 5 for the second semesters.

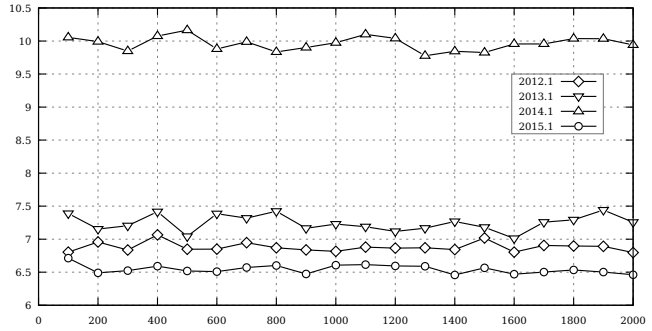

Fig. 4. Mean fitness evolution of the first semesters for the generations from 100 to 2000.

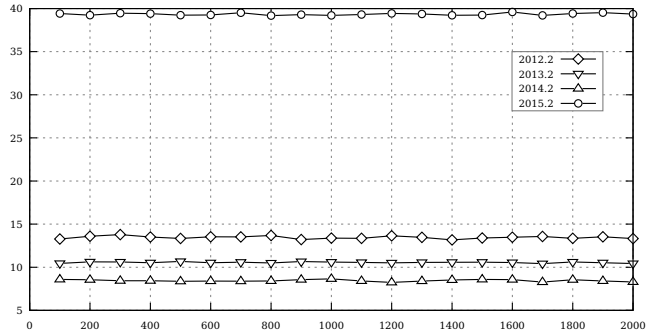

Fig. 5. Mean fitness evolution of the second semesters for the generations from 100 to 2000.

The high mean value for semester 2015.2 in Figure 5 is due to conflicting professors' preference. In complement, the adaptation values of the best solutions found by the proposed algorithm, namely $c_{\text{algorithm}}$, were compared with the adaptation value from the manual structure used in the course, namely c_{manual}. This comparison is presented in Table I.

TABLE I. SOLUTIONS COMPARISON.

Semester	$F(c_{\text{manual}})$	$F(c_{\text{algorithm}})$
2012.1	19	0
2012.2	21	3
2013.1	23	0
2013.2	25	2
2014.1	23	2
2014.2	12	1
2015.1	24	1
2015.2	17	17

The results can be used to show that the proposed algorithm almost always found a much better solution. Only in the last analysed semester the solution proposed by the algorithm is as better as the manual one. Also, the behavior presented in Figures 4 and 5 support that the mean population fitness stabilizes in the first hundred iterations.

V. CONCLUSION

In this work we proposed a genetic algorithm to solve the academic timetable problem thinking on student needs. These needs are represented by a set of metrics extract from interviews with a course coordinator and a researcher in pedagogy. Using a genetic algorithm, we showed in a four years case study that the solutions found using automated timetabling are always better or equal to the manually produced schedules.

As a future work, we expect to consider the schedule planning before the registration period. The goal is to maximize the number of students in the classes using an estimative of students who are apt to apply to a discipline.

ACKNOWLEDGEMENT

The authors would like to thanks the staff of the Elements of Information Processing Laboratory (LabEPI) for the technical and theoretical support.

REFERENCES

[1] C. C. Gotlieb, "The construction of class-teacher time-tables," in *Proceedings of IFIP Congress*, C. M. Popplewell, Ed., 1962, pp. 73–77.

[2] R. Qu, E. K. Burke, B. McCollum, L. T. G. 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.

[3] S. A. Rahman, A. Bargiela, E. K. Burke, E. Özcan, B. McCollum, and P. McMullan, "Adaptive linear combination of heuristic orderings in constructing examination timetables," *European Journal of Operational Research*, vol. 232, no. 2, pp. 287–297, 2014.

[4] S. A. Rahman, E. K. Burke, A. Bargiela, B. McCollum, and E. Özcan, "A constructive approach to examination timetabling based on adaptive decomposition and ordering," *Annals of Operations Research*, vol. 218, no. 1, pp. 3–21, 2014.

[5] J. A. Soria-Alcaraz, G. Ochoa, J. Swan, J. M. Carpio, H. Puga, and E. K. Burke, "Effective learning hyper-heuristics for the course timetabling problem," *European Journal of Operational Research*, vol. 238, no. 1, pp. 77–86, 2014.

[6] G. G. da Silva, "Creating an academic timetable," personal communication, 2014.

[7] T. B. Cooper and J. H. Kingston, "The complexity of timetable construction problems," in *Practice and Theory of Automated Timetabling*, ser. Lecture Notes in Computer Science, E. Burke and P. Ross, Eds. Springer Berlin Heidelberg, 1996, vol. 1153, pp. 281–295.

[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Professional, 1989.

[9] C. Darwin, *On the origin of species*. New York, NY, USA: D. Appleton and Co., 1871.

[10] D. Woods and A. Trenaman, "Simultaneous satisfaction of hard and soft timetable constraints for a university department using evolutionary timetabling," in *Proceedings of the 10th Irish Conference on Artificial Intelligence and Cognitive Science*, University College Cork, Ireland, 1999.

[11] T. C. M. Garcia, "Improving students scholar performance," personal communication, 2014.

[12] A. M. Turing, "Computing machinery and intelligence," *Mind*, pp. 433–460, 1950.

[13] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[14] R. Linden, *Algoritmos Genéticos*, 3rd ed. Rio de Janeiro, RJ, Brazil: Ciência Moderna, 2012.

[15] M. W. S. Almeida, "Utilização de algoritmos genéticos para montagem de horários acadêmicos com foco na blocagem de horários," Caicó/RN, Brazil, 2015.