Solving a Scholar Timetabling Problem Using a Genetic Algorithm - Study Case: Instituto Tecnologico de Zitacuaro

Noel Rodriguez, Jose Martinez, Juan J. Flores, Mario Graff
{nrodriguez, jmartinez}@dep.fie.umich.mx, juanf@umich.mx, mgraffg@dep.fie.umich.mx
Universidad Michoacana de San Nicolas de Hidalgo,
Av. Francisco J. Mugica S/N Ciudad Universitaria, Morelia, Michoacan, Mexico

***Abstract*—The Scholar Timetabling Problem consists of fixing a sequence of meetings between lecturers, classrooms and schedule to a set of groups and courses in a given period of time, satisfying a set of different constraints, where each course, lecturer, classroom, and time have special features; this problem is known to be NP-hard. Given the impossibility to solve this problem optimally, traditional and metaheuristic methods have been proposed to provide near-optimal solutions. This paper shows the implementation of a Genetic Algorithm (GA) using a real coding to solve the Scholar Timetabling Problem. A naive representation for chromosomes in a population-based heuristic search leads to high probabilities of violation of the problem constraints. To convert solutions that violate constraints (unfeasible solutions) into ones that do not (feasible solutions), we propose a repair mechanism. Based on the proposed mechanism, we present a possible solution to the Scholar Timetabling Problem applied to a real school (Instituto Tecnologico de Zitacuaro). Here we present experimental results based on different types of GA configurations to solve this problem and present the best GA configuration to solve the study case.**

Keywords*—Scholar Timetabling Problem, Genetic Algorithms, Optimization

I. Introduction

The timetabling problem, which has an important role in education, is a special version of optimization problems found in real life situations. The timetabling problem has always been solved by leveraging human resources in educational institutions. During the process, numerous aspects have to be taken into consideration. Almost a week of work of an experienced person is needed to produce a timetable for an average institution and the result is often not satisfactory; it does not meet all the requirements. Furthermore, when the constraints change, the whole work becomes unusable, and has to be restarted from scratch. The problem; as almost all optimization problems, is computationally NP-hard. Therefore, only the important conditions can be considered during the manual arrangement process, but it is still extremely complex to find the optimal solution.

Thus, a good timetable generator software that would take into consideration not only the essential conditions necessary for a usable timetable, but also other important didactic and organizational requirements would be very useful. This possibility became a reality with the tremendous growth of computing capacity. In optimization problems, as the problem described in this paper, Genetic Algorithms proved to be a suitable solution means.

As an optimization problem, Scholar Timetabling Problem can be defined mathematically using a set of constraints. In the timetabling literature, constraints are usually categorized into two types: hard constraints and soft constraints.

1) Hard Constraints.: These constraints cannot be violated. For instance, a class or lecturer can not be at different locations at the same time. A time table assignment which satisfies all of the hard constraints is called a feasible solution.

2) Soft Constraints.: These constraint are desirable to maintain but are not absolutely critical. These ones can be violated but violations must be minimized. In practice some of them may be violated because it is usually impossible to find feasible solutions that satisfy all of them.

Most Genetic Algorithms start with a random population, and then successively apply genetic operators to members of the population until the best possible solution is reached. When generating new solutions, both randomly or by genetic operators, the probability of violating Hard Constraints is high. Two methods have been extensively explored and used to enable metaheuristic optimization to produce solutions to Hard Constraints. One is to apply a high penalty to individuals that violate Hard Constraints, so they are discarded in the evolutionary process. The second approach is known as repairing, which uses a function that takes an individual that violates the Hard Constraints and returns a different individual that does not violate them. This is the approach we use in this paper. In this article, we present an approach to get a population-based solution to the timetabling problem, specifically related to the study case Instituto Tecnologico de Zitacuaro. The proposal solution uses feasible solutions through GA applying a repair function. First we test a set of GA parameters, and select the best one. The best set of GA parameters is used to evaluate the performance of a solution. Experimental results show that our approach has a good performance.

3) Study Case. : The Instituto Tecnologico de Zitacuaro (ITZ) is a public university located at east of state of Michoacan, Mexico. The ITZ offers 8 academic programs: Civil Eng., Computer Systems Eng., Industrial Eng., Enterprise Management Eng., Electromecs Eng., Informatics Eng., Public Accounting, and Management. Due to the growing size of the university population, every cycle (semester) the ITZ has

978-1-4799-9900-2/15 $31.00 © 2015 IEEE
DOI 10.1109/MICAI.2014.36

a major complexity in the establishment of the academic processes.

In the academic processes the most important one is the generation of an academic schedule for students of the different academic programs. The complexity of academic schedules is determined by variables like number of lecturers in the cycle, number of available classrooms, available time, offered courses, etc.

A normal academic schedule, contains the following processes: for each academic program a) courses are offered according to an academic program, other courses are offered according to students needs, b) each course is assigned a lecturer, the lecturer is assigned according to their profiles and expertise, c) each course-lecturer is assigned to a classroom, d) each course-lecturer-classroom is scheduled in two-time sessions per day (some courses need more than one day, even one time per day), and finally e) subsets of course-lecturer-classroom-times conform groups.

These variables included in each process conform a set of values of 8 academic programs, 181 courses, 69 lecturers, 34 classrooms, 24 different times and 36 groups. These yield a total of $8 \times 181 \times 69 \times 34 \times 24 \times 36 = 2,935,014,912$ different forms of combine the academic variables. This process, when made by humans, requires around 100 hours and three workers.

This paper is organized as follows: Section 2 presents the problem statement and formulation of the scholar timetabling problem. Section 3 presents work related to the scholar timetabling problem. Section 4 describes the proposed population-based solution. Experimental results are presented in Section 5. Finally Section 6 presents our conclusions and further research directions.

II. Problem statement

The Scholar Timetabling Problem consists of a set of tasks and a set of resources. The tasks are the set of courses clustered by groups, every cycle there are different fixed courses by semester; the scholar year is divided into two semesters, each of which represents a set of available courses for each academic program. The resources are represented by lecturers, classrooms, and available times, these have a constant length, availability, and special characteristics.

The Scholar Timetabling Problem can be defined as follows. Given the following preliminary definitions:

- a set $\mathbb{C}$ of $N_{\mathbb{C}}$ courses; each course $c_i \in \mathbb{C}$, $i \in [1, 2, ..., N_{\mathbb{C}}]$,
- a set $\mathbb{L}$ of $N_{\mathbb{L}}$ lecturers; each lecturer $l_j \in \mathbb{L}$, $j \in [1, 2, ..., N_{\mathbb{L}}]$,
- a set $\mathbb{R}$ of $N_{\mathbb{R}}$ classrooms; each classroom $r_k \in \mathbb{R}$, $k \in [1, 2, ..., N_{\mathbb{R}}]$,
- a set $\mathbb{T}_1$ of $N_{\mathbb{T}_1}$ available times in the week (from monday to thursday); each available time $a_l \in \mathbb{T}_1$, $l \in [1, 2, ..., N_{\mathbb{T}_1}]$ and
- a set $\mathbb{T}_2$ of $N_{\mathbb{T}_2}$ available times on friday; each available time $b_m \in \mathbb{T}_2$, $[m \in 0, 1, ..., N_{\mathbb{T}_2}]$.
- a target matrix $\mathcal{M} = [X_{l,r,t_1,t_2}]_{N_L \times N_R \times (N_{T_1} + N_{T_2})}$.

All the indices start at 1, except the index for $\mathbb{T}_2$ which starts at 0, to simplify the cases when a course does not have time in a given day.

To allocate the taks-resources it is necessary the use a 3D matrix called *target matrix* ($\mathcal{M}$), this matrix is used to find suitable time slots for scheduling tasks [6]. $\mathcal{M}$ is a $||\mathbb{L}|| \times ||\mathbb{R}|| \times (||\mathbb{T}_1|| + ||\mathbb{T}_2||)$ matrix, where the dimensions are: times (x-axis), classrooms (y-axis) and lecturers (z-axis). The combination of $(l_i, r_j, t1_k, t2_l)$, are represented for two cells of target matrix: $\mathcal{M}_{l,r,t_1}$ and $\mathcal{M}_{l,r,t_2}$, where $1 \leq t_1 \leq 12$ and $13 \leq t_2 \leq 24$, for each $(lecturer, classroom, time_1, time_2)$ combination.

$\mathcal{H}(\cdot)$ is a function that returns the Hard Constraints penalization, the Hard Constraints are represented by:

1) A classroom cannot be assigned to more than one course in the same time/day.
2) A lecturer cannot be assigned to more than one course in the same time/day.
3) The number of hours assigned to each lecturer, cannot exceed his/her availability.

$\mathcal{S}(\cdot)$ is a function that returns the Soft Constraints penalization, the Soft Constraints are represented by:

1) Check the suitable classroom (Theory or Practice)
2) Check lecturer profile.
3) Check the times preference by lecturers.
4) Check that the assigned classrooms have the needs of course capacity.
5) Classrooms must be assigned consecutively (no holes in schedule).
6) The number of weekly hours assigned to a course must match the course's needs.

The objective is to minimize the penalization of Hard constraints ($\mathcal{H}$) and Soft constraints ($\mathcal{S}$); the Hard constraints must be fullfiled and the Soft constraints could be minimized [1]. A mathematical formulation can represented by:

Determine assignment $[X_{l,r,t_1,t_2}]_{N_L \times N_R \times (N_{T_1} + N_{T_2})}$ that minimizes [5]:

$$f(X) = \sum_{i=1}^{3} \mathcal{H}(X) + \sum_{j=1}^{6} \mathcal{S}(X) \quad (1)$$

$Subject\ to$

$\forall j \in [1, ..., N_L], \forall k \in [1, ..., N_R], \forall l \in [1, ..., N_{T_1}],$
$\forall m \in [1, ..., N_{T_2}], x_{j,k,l,m}$ must be assigned to a free slot from $\mathcal{M}_{N_L \times N_C \times (N_{T_1} + N_{T_2})}$. (2)

$$\forall j \in [1, ..., N_L], \forall k \in [1, ..., N_R], \forall l \in [1, ..., N_{T_1}],$$
$$\forall m \in [1, ..., N_{T_2}], \sum_{c}^{n_C} \mathcal{H}(x_{j,k,l,m}) <= 0 \quad (3)$$

$$\forall j \in [1, ..., N_L], \forall k \in [1, ..., N_R], \forall l \in [1, ..., N_{T_1}],$$
$$\forall m \in [1, ..., N_{T_2}], \sum_{c}^{n_C} \mathcal{S}(x_{j,k,l,m}) >= 0 \quad (4)$$

where $\mathcal{H}(X)$ is a function that penalizes the violation of Hard Constraints, and $\mathcal{S}(X)$ is a function that penalizes the violation of Soft Constraints. $f(X)$ assesses the total penalization for a given X. Constraint 2 prevents the overlaps between lecturers assignment, classrooms, and times. Constraint 3 prevent Hard Constraint violations, and Constraint 4 are soft constraints that can be violated.

III. Related Work

The timetabling problem is one of the scheduling problems that have been extensively studied. The solution approaches range from graph coloring to heuristic algorithms, including mathematical programming models and metaheuristics as well. A number of artificial intelligence based approaches using metaheuristic techniques are considered. Among metaheuristic techniques, GA is of special interest because automated timetabling has been one of the main optimization problems in GA applications [2]. The concept uses hybrid approaches by which an evolutionary algorithm performs a search in the domain for candidate solutions to mutate. The initial timetable is supported by a specific mutation operator to generate a final timetable. GA is capable of producing examination timetables that fulfill both the hard and soft constraints [8].

Burke et al. [10] propose a GA in which a chromosome is represented as an ordered sequence of exams. They introduce a hybrid crossover operator; only scheduled exams in both parents are chosen for the next period.

Colorni et al. [3] propose a direct GA that uses a matrix to represent the timetable. Each element of this matrix i.a. a gene. An alphabet of characters is used to represent attributes of events. An advantage of this method is to use a filtering algorithm to generate feasible solutions.

Sigl et al. [9] propose a GA and use 3D cubes corresponding to rooms, days, and timeslots to model the timetable. In this method, every gene in an individual represents one class and each individual represents one timetable. The algorithm starts from an unfeasible timetable, and tries to get a feasible one. This algorithm was tested on small and large instances of the problem.

Wilke et al. [11] propose a direct GA where genes are organized in a nested model. This method applies some heuristic mechanisms and hybrid operators to avoid exploring the whole search space and improve the quality of individuals. The standard GA continues until the number of generations without significant improvement reaches a predefined value.

IV. Population-based Solution

A population is a set of individuals where each individual represents a prospect solution to the problem. At the beginning a population-based solution explores a large search space; a randomly picked initial population of individuals is scaltered throughout the search space. This initial population exhibits a large variance in their characteristics. During later iterations the variance in population members is expected to be reduced due to the population converging near an optimum.

To solve Scholar Timetabling Problem using a population-based metaheuristic, we need to explore the search space and determine those individuals that do not violate Hard Constraints and produce the best possible fitness. Individuals that violate Hard Constraints produce unfeasible solutions, so to make them feasible we apply a repair process. A repaired solution substitutes the unfeasible solutions and can be used in the search process to make them feasible candidates. See Figure 1.

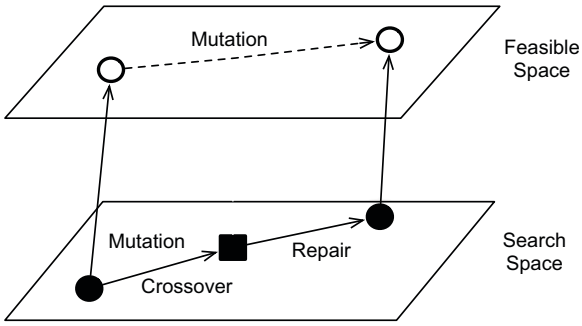

Fig. 1: Feasible Space

The success of Genetic Algorithms lies on its ability to exploit the information accumulated in an initial search space, and subsequent adaptation [4].

This section details the different configurations and GA parameters, to finally describe the procedure to solve the study case.

A. Genetic Algorithm Configuration

Our approach uses a real-coded GA chromosome configuration. This representation allow us to solve the problem directly. The GA has the ability to recombine, by means of explotation and exploration, the individuals of the population. The chromosome is a vector of real values; each one represents the distinct combination of resources assigned to a task.

$$C = [l, r, t_1, t_2] \quad (5)$$

where $l \in [1, \ldots, 69]$, $r \in [1, \ldots, 34]$, $t_1 \in [1, \ldots, 12]$, and $t_2 \in [0, \ldots, 12]$ are the different genes to be mutated (see definition in section II).

1) Selection operator.: We use tournament selection, where the selection favors the better individuals in the population for the mating pool.

2) Mutation operator.: Perturbing the genes is based on multiplying a scalar δ by a random uniform value called $range$.

$$offspring = offspring + \delta \times range$$

where $\delta \in [0, 1]$ varies with each generations, and $range$ is a uniform random number, its boundaries are according to each $gene$ (see above, gene boundaries).

3) Crossover operator.: The arithmetical crossover (AC) [7] is used to make the explotation on the search space. We select AC because the resulting offspring are feasible with respect to linear constraints and bounds.

$$offspring = \alpha * parent_1 + (1 - \alpha) * parent_2.$$

where α is a random uniform number between $[0, 1]$, and $parent_1$ and $parent_2$ are the selected parents.

B. Solution GA.

Algorithm 1 shows the procedure called *ITZP*, which takes the different characteristics of the study case, these characteristics are passed to a GA optimization process to finally get a solution to the problem.

Algorithm 1 $ITZP$ ($\mathbb{C}$, $\mathbb{L}$, $\mathbb{R}$, $\mathbb{T}_1$, $\mathbb{T}_2$)

1: $courses\_solutions$ = {}
2: $\Omega \leftarrow \mathbb{L} \cup \mathbb{R} \cup \mathbb{T}_1 \cup \mathbb{T}_2$
3: **for** $c \in \mathbb{C}$ **do**
4: i = 0;
5: *initialize* P(i, Ω);
6: *evaluate* P(i);
7: **while** i $\leq$ *number-of-generations* **do**
8: i = i + 1;
9: *select* P(i) *from* P(i-1);
10: *recombine* P(i);
11: ***Repair*** P(i);
12: *evaluate* P(i);
13: **end while**
14: **add** $courses\_solutions \leftarrow best\_individual_{l,r,t_1,t_2}$
15: **mark** $best\_individual_{l,r,t_1,t_2}$ in $\mathcal{M}$ matrix
16: **end for**
17: return $courses\_solutions$

The sets $\mathbb{C}$, $\mathbb{L}$, $\mathbb{R}$, $\mathbb{T}_1$, $\mathbb{T}_2$ (courses, lecturers, classrooms, times 1 and times 2 respectively) are passed as parameters to procedure $ITZP$, the set $courses\_solutions$ is initialized (line 1), the search space Ω is created (line 2), initialize the iterative process for each course from $\mathbb{C}$ (line 3), initialize the **GA process** (line 4), from Ω at time i, the initial population P is generated (line 5), each individual from P is evaluated according to equation (1) (line 6). Once the initialization process ends, the iterative process is started while the number of generations is not reached (line 7). Time is increased (line 8). The selection of individuals from population P at time i is performed (line 9), the mutation-crossover processes is performed to the population P at time i (line 10), the infeasible individuals (those that violated Hard Constraints) are repaired (line 11); the reparation consists, for each infeasible individual (individual that violates hard constraints) find best values (values that does not violate hard constraints) in each gene into individual's chromosome. The evaluation of population P at time i is performed (line 12). The $best\_individual_{l,r,t_1,t_2}$ from the **GA process** is added to the set $courses\_solutions$ (line 14), the cells $\mathcal{M}_{l,r,t_1}$ and $\mathcal{M}_{l,r,t_2}$ are marked in matrix $\mathcal{M}$ as unavailable; the cells correspond to the tuple $best\_individual_{l,r,t_1,t_2}$ (line 15). The process continues until the courses are covered. Finally the set $courses\_solutions$ is returned (line 17).

Algorithm 2 $Repair$ ($Individual$)

1: **if** $\mathcal{H}(Individual_{l,r,t_1,t_2}) > 0$ **then**
2: **if** violation is in tuple $\mathcal{M}_{Individual_{l,r,t_1}}$ **then**
3: $new\_individual$ = from available resources, find values to genes t_1 or r or l
4: **end if**
5: **if** violation is in tuple $\mathcal{M}_{Individual_{l,r,t_2}}$ **then**
6: $new\_individual$ = from available resources, find values to genes t_2 or r or l
7: **end if**
8: **end if**
9: return $new\_individual$

Algorithm 2 shows the Repair process, it has as input parameter $Individual$ which is a possible infeasible individual. Checks if $Individual$ has hard constraints (line 1), if so: checks if the violations are in genes $Individual_{l,r,t_1}$ (line 2), then from a pool of avaiable resources, assigns a new value to gene t_1 if solves the hard constraint violation, $new\_individual$ is created replacing gene t_1, otherwise, it assigns a new value to gene r, if it solves the hard constraint violation, the $new\_individual$ is created replacing the gene r, otherwise, it assigns a new gene value lecturer l, if it solve the hard constraint violation, the $new\_individual$ is created replacing the gene l (line 3), the same process is in the case of violations are in genes $Individual_{l,r,t_2}$ (line 6). Finally a $new\_individual$ is returned (line 9).

As an illustrative example, consider the following: as input parameters to $ITZP$, the tasks $\mathbb{C} = \{c_1, c_2, c_3, c_4\}$, and the resources $\mathbb{L} = \{L_1, ..., L_{69}\}$, $\mathbb{R} = \{r_1, ..., r_{34}\}$, $\mathbb{T}_1 = \{t1_1, .., t1_{12}\}$, $\mathbb{T}_2 = \{t2_0, ..., t2_{12}\}$ and individual's chromosome $CH = \{l, r, t1, t2\}$. The procedure initializes the population Ω and evaluates it, starting the GA's iterative process. Consider that in the GA process, the **Repair** process found a infeasible individual at time i (individual that violates hard constraints), its gene values before repairing process are $Individual_i = \{l_{23}, r_2, t1_1, t2_5\}$, the repair process checks the genes values and determines that the gene value $t1_1$ violates hard constraints, then, from a pool of available resources, the repair process finds a suitable value to this gene. After repairing process, the individual's gene values are $Individual_i = \{l_{23}, r_2, t1_3, t2_5\}$. The repairing process, repairs the gene $t1$, because this resource, before to be repaired, was marked in the target matrix $\mathcal{M}$, and the repairing process, finds a suitable value.

V. Experimental Results

The following experiments were performed to get the best GA configuration to the Algorithm (1), with and without the use of a repairing process; the parameter *population size* was varied within the range $100-500$, the *crossover rate* was varied in the range $0.1-1.0$ and the *mutation rate* was varied in the range $0.1-1.0$. Each experiment was repeated 10 times and the mean of performance was reported.

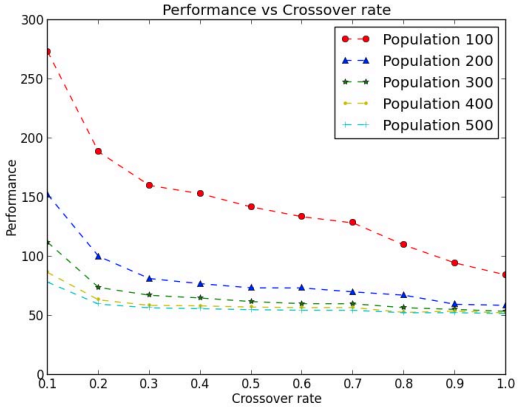

Fig. 2: Performance obtained from different crossover, without the use of repairing process.

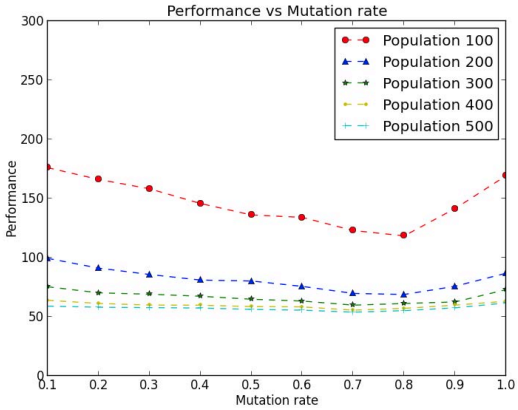

Fig. 3: Performance obtained from different mutation rate, without the use of repairing process.

Figures 2 and 3 show, how generally the use of a large number of population size gets the best performance, while with small population size the crossover and mutation rate are determinant to get the best performance.

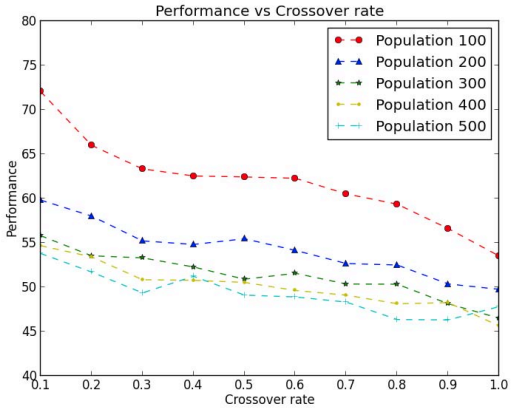

Fig. 4: Performance obtained from different crossover rates using the repairing process.

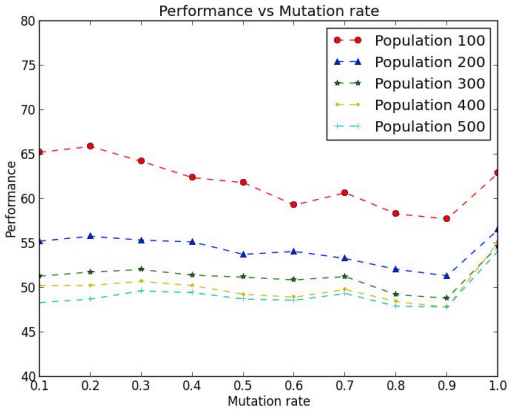

Fig. 5: Performance obtained from different mutation rates using the repairing process.

Figures 4 and 5 show similar behaviors on the performance obtained in Figures 2 and 3. The main difference is the improved performance even with small populations.

According to Figures 2, 3 and 4, 5, we propose the next GA configuration showed in Table I.

TABLE I: Best GA configuration.

Parameter	Value
Population size	400
Selection operator	tournament selection between 10 individuals
Crossover rate	1.0
Mutation rate	0.9
Generations	100

Taking into consideration the GA configuration shown in table I, we perform 10 independent experiments. Table II shows the results.

TABLE II: Mean of hard and soft constraints and performance (computed according to Equation 1).

Constraints	Hard constraints violations
1	0
2	0
3	0
Constraints	**Soft constraints violations**
1	1.3
2	0
3	0
4	0
5	14.1
6	2.0
Total Performance	**49.5**

The table II shows the results of perform the best GA configuration; no hard constraints were violated, while approximately 17 soft contraints were violated. The total performance (sum individual performance to each task) 49.5 is in the performance boundaries showed in Figures 4 and 5, this result shows a good proposed solution, having a total of 17 constraints violated; we consider good results, taking in consideration, the large set of combinations (see section I-3).

VI. Conclusions and further research

This work presented a possible solution to a University timetabling problem using GA in a study case, in this case a real application, the Instituto Tecnologico de Zitacuaro.

The population-based solution is a procedure based on Genetic Algorithms that for every course (task) assigns the best solution (resources). The results show a possible GA configuration in the solution of the study case; a high number of population size combined with a high crossover and mutation rate, is a good initial GA configuration.

As future work we will address the study of timetabling starting with a partial solution and applying Hypermutation. Once a feasible timetabling configuration has been generated, it can be the starting point for a new search. Another interesting work is a possible solution using different Evolutionary Computation Algorithms.

References

[1] S. Abdullah, E. Burke, and B. McCollum. A hybrid evolutionary approach to the university course timetabling problem. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1764–1768, Sept 2007.

[2] M. Bufe, T. Fischer, H. Gubbels, C. Hacker, O. Haspirch, C. Scheibel, K. Weicker, N. Weicker, M. Wenig, and C. Wolfangel. Automated solution of a highly constrained school timetabling problem-preliminary results. *Computer Science*, 2037:431–440, 2001.

[3] A. Colorni, M. Dorigo, and V. Maniezzo. A genetic algorithm to solve the timetable problem. *Technical Report No. 90-060*, 1991.

[4] F. Herrera, M. Lozano, and J. L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12:265–319, 1998.

[5] C. Lara, J. Flores, and F. Calderon. Solving a school timetabling problem using a bee algorithm. In A. Gelbukh and E. Morales, editors, *MICAI 2008: Advances in Artificial Intelligence*, volume 5317 of *Lecture Notes in Computer Science*, pages 664–674. Springer Berlin Heidelberg, 2008.

[6] S. Lukas, A. Aribowo, and M. Muchri. Solving timetable problem by genetic algorithm and heuristic search case study: Universitas pelita harapan timetable. In O. Roeva, editor, *Real-World Applications of Genetic Algorithms*, volume 378, pages 303–316. InTech, 2012.

[7] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (2Nd, Extended Ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[8] J. J. Moreira. A system for automatic construction of exam timetable using genetic algorithms. *Tekhne*, 6(9):319–336, 2008.

[9] B. Sigl, M. Golub, and V. Mornar. Solving timetable scheduling problem by using genetic algorithms. *Information Technology Interfaces*, pages 519–524, 2003.

[10] R. Weare, E. Burke, and D. Elliman. A hybrid genetic algorithm for highly constrained timetabling problems. *Proc. 6th Int. Conf. on Genetic Algorithms*, pages 605–610, 1995.

[11] P. Wilke, M. Grobner, and N. Oster. A hybrid genetic algorithm for school. *Advances in Artificial Intelligence*, 2557:455–464, 2002.