A new approach of Design for the Academic Timetabling problem through Genetic Algorithms

Soria-Alcaraz Jorge A., Carpio Martin, Puga Hector
Leon Institute of Technology
División de estudios de Posgrado e Investigación
León Guanajuato, México
soajorgea@gmail.com, jmcarpio61@hotmail.com, pugahector@yahoo.com

Abstract

The Academic timetabling problem is a common and hard problem inside every educative institution, this problem tries to coordinate Students, Teachers, Classrooms and Timeslots under certain constrains that dependent in many cases the policies of each educational institution. The Genetic Algorithm is a popular methodology used to find solutions in problems with a huge search space, this algorithm have been proved in wide range of discrete problems with encouraging results. This paper present a GA based method that solves the Academic Timetabling Design Problem using the API-Carpio Methodology. This GA uses a variable-length representation; which involves the complete encoding of a solution that is directly applicable to real world. this solution is achieved using several real instances of academic timetabling problem from Leon Institute of Technology (LIT) producing encouraging results for all the instances. Finally we analyze the quality of our solutions against the human expert and inside the context of a real academic timetabling process.

1. Introduction

The timetabling problem is one of the most difficult and common problems inside a company, where it must done some activities such time assignment to workers and machinery, this activities are examples of timetabling.

A correct timetabling is fundamental for a good performance of every company, to assign and utilize efficiently the resources that owned. The same problem can be seen into an academic environment where classrooms, teachers and students have a important role in the schedule design process.

An automatic educational timetabling generation provides great benefits not only decreasing the time required to produce it and by, optimizing human and physical resources, but also by improving the precision and quality of schedules eliminating the human factor, in this way, a lot of many errors and conflicts like two or more classes assigned at the same time to a student can be avoided.

The proposal is the creation of a computational module which allows permits academic timetabling design process automatically, saving time and resources for the academic institution where it applies.

The aim of this paper is to explore the automatization of a novel alternative in the academic timetabling process, through the application of Genetic Algorithm. the Genetic Algorithm scheme appears in 1975 when John Holland [10] described a new approach to the search for solutions in a problems with a huge search space.

GA uses concepts from Evolutive theory like Population, Crossover and Mutation in order to find good enough solutions in a reasonable time. In this paper we will describe our approach about these operators in the Academic Timetabling Design Problem.

The investigation in this article, presents a GA capable to find good solutions for the Academic Timetabling Design Problem. Our Procedure uses a direct representation of each subject into a timeslot called $vector$, each vector have a variable length. This approach uses real world instances form Leon Institute of Technology Mexico in order to measure the quality of a solution. The paper is organized as follows. Section 2 presents the timetabling design problem, the solution method proposed and its justification. Section 3 contains the experimental setup, the results, their analysis and discussion. Finally Section 4 include our conclusions and some ideas for future work.

2. Solution Approach

In the literature, one can see that Evolutionary Computation has been used in scheduling and timetabling problems

978-0-7695-4204-1/10 $26.00 © 2010 IEEE
DOI 10.1109/CERMA.2010.22

[5] [15] [14] [2]. Recently a novel methodology of schedule design has emerged [3], [17], this API-Carpio methodology has shown based on experience several great results in the educational timetabling design for regular and non-regular students, this methodology is capable to find good solutions minimizing the conflict for regular students defined as students who only have in their schedules subjects of the same semester or period, and non-regular students defined as students who have subjects from different semesters or periods in their schedules.[3], [17] Unfortunately, there still no exist an automated computational module that can perform this method.

Evolutionary Computation usually includes several types of evolutionary algorithms [16] one common example is Genetic Algorithms [16],[9] In this research we use a GA approach to resolve the schedule design problem using the API-Carpio methodology.

2.1 API-Carpio Metodology

The solution of academic timetabling (TTP) produces a table of times [13], in general, the TTP problem can be seen as a set of events e, a set of time spaces (slots) s and a set of constrains c between events and slots. Every timetabling process tries to assign every event e into a slot s without violating the constrains c.[13]. So basically our problem consist into achieved a series of elements e assigned into several slots s, how allow us to cover several academic necessities. Each element e have some properties:

$a)$ One student group is assigned to an event e.

$b)$ One teacher is assigned to an event e.

$c)$ One subject is assigned to an event e.

$d)$ Each event e is assigned to a Slot s.

$e)$ One classroom is assigned to an event e.

then our Timetabling primary objective is to assign these events e to timeslots s tending to several constrains c like:

$a)$ Normally the events are repeated each week.

$b)$ There may be special schedules per Teacher.

$c)$ There are limited number of classrooms.

$d)$ Each teacher can only teach some types of subjects.

The API methodology[5] presents a mathematical description of this behavior, we have:

$$f(x) = FA(x) + FP(x) + FI(x) \tag{1}$$

where:
$FA(x)$=number of students conflicts into timetablig x.
$FP(x)$=number of teacher conflicts into timetablig x.
$FI(x)$=number of classrooms conflicts into timetablig x.

With:

$$FA(x) = \sum_{i=1}^{k} FA_{s_i} \tag{2}$$

$$FP(x) = \sum_{i=1}^{k} FP_{s_i} \tag{3}$$

$$FI(x) = \sum_{i=1}^{k} FI_{s_i} \tag{4}$$

Where:

FA_{s_i}=Number of student conflicts into Slot i of our current timetabling.

FP_{s_i}=Number of teacher conflicts into Slot i of our current timetabling.

FI_{s_i}=Number of classroom conflicts into Slot i of our current timetabling.

K = Number of timeslots available for our timetabling.

Then:

$$FA_{V_j} = \sum_{s=1}^{(M_{V_j})-1} \sum_{l=1}^{M_{V_j}-s} (A_{j,s} \wedge A_{j,s+l}) \tag{5}$$

$$FP_{V_j} = \sum_{s=1}^{(M_{V_j})-1} \sum_{l=1}^{M_{V_j}-s} (P_{j,s} \wedge P_{j,s+l}) \tag{6}$$

$$FI_{V_j} = \sum_{s=1}^{(M_{V_j})-1} \sum_{l=1}^{M_{V_j}-s} (I_{j,s} \wedge I_{j,s+l}) \tag{7}$$

With:

$A_{j,s} \wedge A_{j,s+l}$ = number of students who demand the concurrent enrollment of subjects $M_{j,s} \wedge M_{j,s+l}$.

$P_{j,s} \wedge P_{j,s+l}$ = number of teachers who demand the concurrent teach of subjects ,$M_{j,s} \wedge M_{j,s+l}$.

$I_{j,s} \wedge I_{j,s+l}$ = number of classrooms who serves concurrent the subjects $M_{j,s} \wedge M_{j,s+l}$.

M_{V_j}=Number of diferent subects in a vector. $\eta(V_j) = M_{V_j}$

In this paper we focused into the first part of the methodology, the design of a timetabling only considering the first variable of API-Carpio strategy, Students.

Now our problem is to choose a way of combining subjects of an academic career into several timeslots in order to have the less possible student conflicts. A student conflict is defined as a student who have two or more subjects of the same semester into the same timeslot [3].

2.2 API-Carpio Educational Timetabling Student Model

This model considers timeslots as vectors monitoring students, teachers, classrooms and laboratories simultaneously. As stated previously we only perform the first step into the API-Carpio methodology, the minimization of Student conflicts into a timetabling. Because, we do not perform a complete timetabling we consider our work as a design of a timetabling with a great characteristic, this timetabling will be offer the less student conflict so it is a excellent start point to the timetabling complete process.[3]

Table 1 shows the general structure of a academic career from the point of view API-Carpio methodology

where:
n : Number of semesters, or periods of the career.

Table 1. Structure of an academic career

General structure of an academic career						
Relative subjects position						
1	...	ζ_1	...	$\left(\sum_{i=1}^{n-1} \zeta_i\right) + 1$	...	L
(semester 1)			...	(semester n)		
a_{11}	...	$a_{1_{\zeta_1}}$	...	a_{n1}	...	$a_{n\zeta_n}$

ζ_i: Number of different subjects for semester $i; i = 1, 2, 3, ..., n$.
L = Total of subjects of a especific career.

$$L = \sum_{i=1}^{n} \zeta_i = \zeta_1 + \zeta_2 + ... + \zeta_n \qquad (8)$$

a_{ij} = Subject index inside into a chromosome.

The vectorization is performed by assigning subjects to different vectors. a hard constrain in API-Carpio methodology says : **Never assing two o more subjects from a same semester to a same vector**. This rule ensures that there are no subjects of the same semester at the same timeslot, so a regular student , student who have in her schedule only subjects of the same semester or period, never will have a two o more subjects assigned at the same time.These vectors have a number which are directly related to the number of timeslots offered by the academic institution. An example of this timeslots can be seen in Table 2

Table 2. Timeslots ID offered by LIT

Timeslot	Monday & wednesday	Tuesday & Thursday
7:00 - 8:40	1	2
8:45 - 10:25	3	4
10:30 - 12:10	5	6
12:15 - 13:55	7	8

API-Carpio Hard constrain can be achieved in the following way: in the first step all the subjects are arranged by semester or period as show in Table 1, then to each subject from first semester is assigned randomly different vector number between 1 and k, where k is determined by the maximun number of timeslot offered by the institution, for our example we use number from 1 to 8, after we proceed similarly with subjects of second, third and N-semester.

Once assigned this number we perform an ordering of all the subjects by this vector number, leaving a partition of the career in independent vectors, where each vector contains subjects of differents semesters or periods. An example of how is constructed this Vector shown in Table 3.

Now we can read our schedule design on the following way, if one subject of the first semester or period was assigned the number 5 as vector number, this subject will be offered by the institution at 10:30-12:10 on monday and wednesday. All the subjects have now assigned a specific timeslot.

Table 3. Construction of a vector

Id_Subject	Semester or period	Vector number
ACM0403	1	2
SCM0414	1	4
SCEB042	1	6
ACG0415	1	8
AFD0984	1	5

2.3. Genetic Algorithm Approach

As noted, until now we have seen how to model the schedule design by API-Carpio methodology, this methodology can be automatized by an Genetic Algorithm (GA) approach. GA basic principles were porposed in the 70's [10] and they have been applied for the academic timetabling problem for the last two decades[5] [15] [14] [2]. These algorithms are based on the natural evolution of species (natural selection) and offer many outstanding features over traditional optimization methods.[13]. For instance, their ability to search non-convex solution spaces with multiple ans insolated maxima, global convergence, and inherent parallel search capability, among others characteristics [8]. Nowadays is possible to define a basic template for the GA, this template have the common operators, in the real world application in necessary to adapt these operator to our problems. this basic frame can be seen on Algorithm (1).

Algorithm 1 A Genetic Algorithm Common Template

```
Require: Nothing.
1:  $t = 0$
2:  $P$ = initial_population($t$)
3:  Evaluate($P$)
4:  while termination condition not satisfed do
5:      $t \leftarrow t + 1$
6:      $P' \leftarrow$ Select_parents ($P$)
7:      $P' \leftarrow$ Crossover_parents ($P$)
8:      if Mutation condition satisfied then
9:          Apply_mutation($P'$)
10:     end if
11:     Evaluate ($P'$)
12:     $P \leftarrow P'$
13: end while
14: return Best_individual ($P$)
```

2.4. Combining Genetic Algorithm with Academic Timetabling Design problem Using API-Carpio Methodology.

GA's are population based techniques which sample the solution space randomly by using several candidate solutions. These solutions are evaluated and those that perform better are selected to compose and generate the population in the next generation. After several generations, these solutions improve the objective function value as they explore the solution space for an optimal value. In the context of Academic Timetabling Problem, GAs optimaze the cost function, in other words minimizes the number of conflicts for the students trough the design of a timetable for subjects inside a semester or academic period.

Several studies before [5] [15] [14] [2] uses also a GA, but in all this cases they try to solve the entire Academic Design Process at once, Recently a Novel methodology has emerged [3][16], this methodology view the Academic Design Process in parts, each part have an order, the first part works only with Students, the second with Teachers and the last one with classrooms and laboratories, so in perspective this methodology has the opportunity to design schedules only for Students before to work with teachers and classrooms.

In order to achieve it, we use a GA with several adaptations in its operators, the basic concept is to search for a timetable to minimize the conflict for *non regular* students once that this timetable is achieved we can perform the next part of the API-Carpio Mehtodology.

Our Genetic Algorithm Uses direct representation, in other words, each individual of the GA is itself a timetable for a carrier from Leon Institute of Technology. Also each individual have several divisions called Genes, each Gene represents an Slot of time offered by the institution, but each Gene have variable Leght. The GA also uses a novel generator of random numbers this generator allows us to create chains of numbers (with any leght) without repetitions respecting the randomness uniformity. [4]

We will use several test instances, this instances are a real historical ones from the leon Institute of Technology from 2003 to 2006. each instance have 52 subjects and nearly 400-450 students. The general procedure consist in solve each instance with the proposed GA method, at the end will have a timetable for each instance with the minimum student conflict achieved, it is important to say that our match point will be the expert because there is no another computational module that uses API-Carpio Methodology to solve timetabling design problem. Now we can define each operator used in our investigation.

Representation : Each indivual inside our GA have a complete representation of schedule for a career, at the same time we have $k = 8$ Genes or Vectors for the APi-Carpio methodology, these vectors are the same that Timeslots offered by Leon Institute of Technology. An example of a Gen or Vector can be seen on figure 2.4.

$$V_i = \left\{ \begin{array}{l} Subject \\ Position \\ ID \end{array} \right\} \left\{ \begin{array}{ccccc} a_{11} & a_{34} & a_{45} & a_{53} & a_{65} \\ 1 & 18 & 23 & 27 & 36 \\ acm & scc & scm & sca & acs \end{array} \right\}$$

Figure 1. Example of a Single Vector

Each individual have 8 Vectors. We construct them with the random assignation of each subject from a semester to a vector, in this way, we use a chain of random numbers without repetition in order to assure to there is no exists a subject of a same semester in the same vector.

Fitness Function : Once we have all our subjects into vectors we need to calculate the number of possible students who can be affected with this schedule, this information can be obtained if we sum the numbers of conflicts in each vector , the fitness equation can be seen as follows.

$$FV_j = \sum_{p=1}^{R_j-1} \sum_{q=p}^{R_j-1} (a_{jp} \otimes a_{jp+1}) \; for \; j = 1, 2, \ldots, K \tag{9}$$

where:

$\otimes$: Represents number of students who can take the subjects a_{jp} and a_{jp+1}

R_j =Cardinality of V_j.

Finally our fitness fuction:

$$FC_i = \sum_{j=1}^{K} FV_j, \; for \; i = 1, 2, \ldots, N. \tag{10}$$

Elitism : Simply we pass directly a percentage of our best current solutions to the new generation or iteration.

Selection : To select individuals inside our algorithm for the crossover, we use the next acumulative probability distribution formula:

$$PC_i = \sum_{j=1}^{i} P_j \; para \; i = 1, 2, \ldots, N \tag{11}$$

Where:

$$P_j = \frac{1 - (FC_j/S)}{1 - N} \tag{12}$$

With:

$$S = \sum_{i=1}^{n} FC_i \qquad (13)$$

And N is the number of chromosomes in the current population.

This calculates an accumulated distribution , next we perform a roulette process to make a choice.

Crossover :After selected two parents we chose randomly semester and then we interchange the numbers of vector in a semester, this operation never assign two subjects of the same semester into a same vector.

Mutation : Randomly selected and individual of the population, also randomly take one Gen and regenerated it.

Clonation & Exploration : In order to improve the time and quality of our GA, we design a new strategy, Clonation percentage inform us about the number of individuals who have the same genetic material and the same fitness, once the user define a clonation umbral, and the current number of Clones is high or equal, the Algorithm perform an Exploration, this exploration is an neighborhood search, in this search we use a percentage of our worst individuals (Accord of fitness functions) then we mutated it. This strategy tries to jump out from an local minimum.

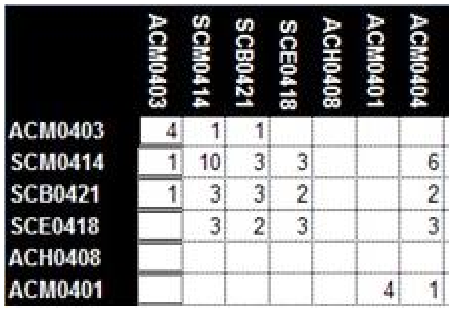
Stop condition : For the Stop conndition we use simply the number of generations.

3 Experiments and Results

This section presents the experiments carried out during the investigation and the results obtained. Our testing set is composed for several real instances from the career of Engineering in computer system at Leon Institute of Technology under the API-Carpio methodology from 2003 to 2008, this instances are composed by a matrix called MMA, this matrix have in its rows and columns subjects from a specific career, its cells contains the number of conflicts that the assignation of both subjects into a same timeslot could do. an example of this matrix is presented in Fig 2, characteristics of the instances used in this paper can be seen on Table 4.

3.1 Experiment Type 1

We perform several test using different initial populations sizes with the aim to taking the time that our AG uses to run 100 generations, The main idea is to analyze the resolution time thought Genetic Algorithm, our results are shown in Table 5.



	ACM0403	SCM0414	SCB0421	SCE0418	ACH0408	ACM0401	ACM0404
ACM0403	4	1	1				
SCM0414	1	10	3	3			6
SCB0421	1	3	3	2			2
SCE0418		3	2	3			3
ACH0408							
ACM0401						4	1

Figure 2. MMA Matrix

Table 4. Test Instances

Instance	Students	Subjects	Exp.Performance
MMASis93	440	52	162
MMASis93A	470	52	230
MMASis93B	445	52	410
MMASis93C	460	52	450

3.2 Experiment Type 2

In this experiment we look for the number of generations based in the number of population capable to achieve a fitness similar to our expert performance. Our results are shown in Table 6, we take information from our previous experiment and we use a population of 32 that have shown a reasonably time performance.

3.3 Experiment Type 3

In this experiment we search for the best Scheduling using the table 5, the best scheduling was found with the features presented in Table 6. The idea is to obtain a good schudule whit a good time and fitness.

3.4 Analysis of our best fitness

We can perform a deeper analysis of our results taking for example our first result in MMASis93 Instance, 72, this improves the quality of solution against the human expert who takes 1 week in order to achieve a schedule design solution without a proper conflict check . This 72 conflicts means 72 $non - regular$ students in the whole career who could have a conflict in theirs schedule, in a deeper study this career have 9 semester so it is possible to find the exact number of conflicts per semester, this number is 72/9 = 8 and each semester have near 6 subjects so the number of conflicts per subject is 8/6=1.3, finally we have 1 students affected per subject but this conflict could be easily corrected if we consider 2 or 3 classes per subjects. Also we have a schedule that is applicable to an real instance of

Table 5. Time for GA in 100 generations

Population	2	4	8	16	32	64
Time (s)	1.18	1.33	3.5	12.0	16.5	36.3

Table 6. Generations needed for achieve a fitness similar to our expert

Instance	Generations	Fitness	Expert
MMASis93	1,430	150	162
MMASis93A	1,600	235	230
MMASis93B	1,550	400	410
MMASis93C	1,600	440	450

computer systems engineering under API-Carpio Methodology.

4. Conclusions

Genetic Algorithms is a important method to resolve complex problems (in our case NP-Complete problem), in this paper we present some adaptations of each operator inside the GA to work with the Academic timetabling design problem. Also we propose a new Strategy, the Clonation and Exploration with encouraging results.

In our experiments we obtain a better performance in time and quality of solution that a human expert with more that 10 years of experience in the academic timetabling process. This results are encouraging.

Our final results shown good advantages against our experts nearly of 30% and in our best result for the instance MMASis93 we achieve an advantage of 55the good quality of our results can be undestanding if we consider the mean characteristic of API-Carpio Methodology, the Design Phase, This phase allows us to construct a optimun schedule for the students, and in next steps of the methodology we use this design to assign teachers and classrooms.

For future work, we propose more investigation about or operators, maybe the test of new selection methods, new crossover methods and several adaptations of the clonation and exploration strategy, as well as posible use of Mahalonobis's distance.

[1]

References

[1] D. Abramson and J. Abela. Parallel genetic algorithm for solving the school timetabling problem. *Memories 15th Australian computer science conference*, page 11, 2002.

[2] L. Aparecido and H. Cesar. The school timetabling problems: a focus on elimination of open periods and isolated classes. *Sixth international conferences on hybrid intelligents systems*, 2006.

[3] M. Carpio. Modelo integral de asignacion optima de carga academica usando un algoritmo heuristico. In *Encuentro de Investigacion en Ingenieria Electrica*, 2006.

[4] M. Carpio and J. Soria-Alcaraz. Variable length number chains generation without repetitions. *International Seminar on Computational Intelligence 2010*, 2010.

[5] S. Chu and H. L. Fang. Genetic algorithm vs tabu search in timetable scheduling. In *Third International conference on Knowledge-Based Intelligent information*, 2003.

[6] B. Edmund, P. Newall, et al. Memetic algoritm for university exam timetabling. *Deparment of computer science*, 2004.

[7] W. Erber and J. Keppler. Genetic algorithm solving weekly course-timetabling problem. *Deparment of computer science*, 2004.

[8] C. R. Felix Calderon and Others. A constraint-handling genetic algorithm to power economic dispatch. *MICAI 2008*, pages 371–381, 2008.

[9] P. Gonzalo and S. Matilde. *Introduccion a la Inteligencia Artificial*, volume 1. AlfaOmega RAMA, 2006.

[10] J. Holland. Adaptation in natural and artificial systems. *University of Michigan Press*, 1975.

[11] M. Karova. Solving timetabling problems using genetic algorithms. *27th Spring Seminar on Electronics Technology*, 2004.

[12] Z. Lu and J. kao Hao. Solving the course timetabling problem with a hybrid heuristic algorithm. *AIMSA*, pages 262–273, 2008.

[13] R. Martinez and Q. Aguilera. Educational timetabling generation with genetic algorithms. *COMCEV*, pages 159–163, 2005.

[14] J. Mesa de Luna and F. Luna. Metodos evolutivos para la asignacion de horarios docentes en el cbtis 168 de aguascalientes. *Memorias COMCEV 07*, 2007.

[15] F. Padilla, C. Coello, et al. Generacion de horarios con algoritmos geneticos (educational timetabling). *COMCEV*, pages 159–163, 2006.

[16] C. Reeves. *Genetic Algorithms*, chapter 3, pages 55–82. Kluwer's International Series. kluwer academic, 2003.

[17] A. Rios and M. Carpio. Optimizacion y automatizacion en la asignacion de tareas aplicada en el area educativa. Master's thesis, INSTITUTO TECNOLOGICO DE LEON, Agosto 2009.

[18] P. Ross and E. Hart. Some observations about ga-based exam timetabling. *University of edinburg*, 2005.

[1]Authors thank the support received from the Consejo Nacional de Ciencia y Tecnologia (CONACYT).

Table 7. Best fitness achieved

Instance	Time (S)	Generations	Fitness	Expert
MMASis93	300.4	13800	72	162
MMASis93A	450.5	14800	160	230
MMASis93B	440.2	15600	315	410
MMASis93C	370.0	14400	307	450