Applied Soft Computing 11 (2011) 337–344

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Solving timetabling problems using a cultural algorithm

Carlos Soza[a], Ricardo Landa Becerra[b,1], María Cristina Riff[a,*,2], Carlos A. Coello Coello[c,1]

[a] Universidad Técnica Federico Santa María, Departamento de Informática, Av. España No. 1680, Valparaíso, Chile
[b] CINVESTAV Tamaulipas, Laboratorio de Tecnologías de Información, Parque Científico y Tecnológico Tecnotam, Km. 5.5 carretera Victoria-Soto La Marina, Victoria Tamaulipas, 87130, Mexico
[c] CINVESTAV-IPN (Evolutionary Computation Group), Departamento de Computación, Av. Instituto Politécnico Nacional No. 2508, Col. San Pedro Zacatenco, México D.F. 07300, Mexico

ARTICLE INFO

Article history:
Received 29 September 2007
Received in revised form 5 September 2009
Accepted 17 November 2009
Available online 24 November 2009

Keywords:
Cultural algorithms
Metaheuristics
Parameter control

ABSTRACT

This paper addresses the solution of timetabling problems using cultural algorithms. The core idea is to extract problem domain information during the evolutionary search, and then combine it with some previously proposed operators, in order to improve performance. The proposed approach is validated using a benchmark of 20 instances, and its results are compared with respect to three other approaches: two evolutionary algorithms and simulated annealing, all of which have been previously adopted to solve timetabling problems.

1. Introduction

The timetabling problem is a combinatorial optimization problem that consists of assigning schedules to several workers or students, which also require some resources. In order to make a feasible timetable, a set of hard constraints must be satisfied (most of them technical constraints); moreover, a good timetable must satisfy some soft constraints (frequently, comfort-related constraints), and if all soft constraints are met, we can consider the solution as optimal. From this point of view, the timetabling problem can be considered as an optimization problem, when trying to minimize the violations of the existing soft constraints.

This NP-hard problem presents several variants, such as the employee, exam and university timetabling problems. In 2002, the Metaheuristics Network organized a competition on the University Course Timetabling Problem (UCTP), and published a set of instances of the problem, in order to make easier the comparisons of different algorithms. However, in that competition no evolutionary algorithms were proposed, even when evolutionary algorithms have shown to be very effective techniques for solving combinatorial problems.

Cultural algorithms [21] are a particular class of evolutionary algorithm that use domain knowledge extracted during the evolutionary process in order to improve the performance of the search engine (i.e. the evolutionary algorithm) adopted. What we explore in this paper is the use of a combination of knowledge extracted during the evolutionary search with some knowledge that is inserted *a priori* because it has been previously found that it is useful when solving combinatorial optimization problems. The main hypothesis in this regard was that the incorporation of knowledge into an evolutionary algorithm would increase its performance as to make it competitive with other approaches whose computational cost is significantly higher.

Several heuristics have been used for different types of timetabling problems: evolutionary algorithms [18], memetic algorithms [25] tabu search [6], simulated annealing [17], the ant system [27], and hybrid algorithms [12,3,2], among others.

Note however, that this paper presents the first attempt (to the authors' best knowledge) to use cultural algorithms to solve timetabling problems.

The proposed approach is compared with respect to an evolutionary algorithm with specialized crossover operators [18], a recently published memetic algorithm [25], and a simulated annealing approach [17] that won the competition of the Metaheuristics Network, in all the test cases adopted for that competition. The obtained results indicate that the proposed approach is a viable alternative for solving, efficiently, timetabling problems.

The remainder of this paper is organized as follows: in Section 2 a brief description of the statement of the problem is provided. Section 3 contains an introduction to cultural algorithms which includes a description of their main components and the main

* Corresponding author. Tel.: +56 32 2654000.
E-mail addresses: csoza@inf.utfsm.cl (C. Soza), rlanda@tamps.cinvestav.mx (R.L. Becerra), mcriff@inf.utfsm.cl (M.C. Riff), ccoello@cs.cinvestav.mx (C.A. Coello Coello).
1 Tel.: +52 55 5061 3800x6564.
2 She is partially supported by Centro Científico-Tecnológico de Valparaíso (CCT-Val), NºFB0821 and Fondecyt 1080110.

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.11.024

motivation to use them. Section 4 contains the details of the proposed approach to solve university course timetabling problems using a cultural algorithm. As part of this section, a description of the representation of solutions adopted in this work is included, as well as the mechanisms implemented to add domain knowledge to the evolutionary algorithm both before and during the search process. Section 5 provides a comparative study. Finally, Section 6 presents the general conclusions and some possible paths for future research.

2. Problem statement

The variant of the problem tackled here was proposed by Ben Paechter for the International Timetabling Competition organized by the Metaheuristics Network [16]. It is referred to in the following as the University Course Timetabling Problem (UCTP). Lecture must be scheduled in 45 timeslots (5 days of 9 h each) and a number of rooms, with varying facilities and student capacities, so that the following hard constraints are satisfied:

- $H1$: lectures having students in common cannot take place at the same time;
- $H2$: lectures must take place in a room suitable for them in terms of facilities and student capacity; and
- $H3$: no two lectures can take place at the same time in the same room.

We consider as well the following soft constraints:

- $S1$: students should not have to attend lectures in the last timeslot of the day;
- $S2$: they should not attend more than two lectures in a row; and
- $S3$: they should not have only one lecture in any given day.

Note that the S1 constraints can be checked without knowledge of the rest of the timetable; S2 constraints can be checked while building a timetable is complete and all lectures have been assigned a timeslot. A timetable in which all lectures have been assigned to a timeslot and a room so that no hard constraints are violated, is said to be feasible. The aim of the problem is to find a feasible solution with minimal soft constraint violations.

2.1. Model

2.1.1. Parameters

The parameters of the problem are the following:

- R: Set of Rooms
- E: Set of Events
- F: Set of Features
- S: Set of Students
- T: 45 timeslots (5 days of 9 slots per day).

2.1.2. Variables

Let: a, indicate students attendance:

$$a_{(s_i,e_j)} \begin{cases} 1 & \text{if student } s_i \text{ attends event } e_j \\ 0 & \text{otherwise} \end{cases}$$

g, indicate which rooms are suitable for which events (i.e. the room provides all the features that the event requires and has the right size) e_j:

$$g_{(r_l,e_j)} \begin{cases} 1 & \text{if room } r_i \text{ is suitable for the event } e_j \\ 0 & \text{otherwise} \end{cases}$$

p, indicate the placement of events:

$$p_{(e_j,t_k,r_l)} \begin{cases} 1 & \text{if event } e_j \text{ is placed in timeslot } t_k \text{ and room } r_l \\ 0 & \text{otherwise} \end{cases}$$

2.1.3. Hard constraints

- Lectures having students in common cannot take place at the same time.

$$H1: \forall_{s \in S} \forall_{t \in T} \sum_{j=0}^{|E|} \sum_{t=0}^{|R|} a_{(s,e_j)} \times p_{(e_j,t,r_l)} \leq 1 \tag{1}$$

- Lectures must take place in a room suitable for them in terms of facilities and student capacity.

$$H2: \forall_{e \in E} \sum_{k=0}^{|T|} \sum_{t=0}^{|R|} p_{(e,t_k,r_l)} \times g_{(r_l,e)} = 1 \tag{2}$$

- No two lectures can take place at the same time in the same room.

$$H3: \forall_{t \in T} \forall_{r \in R} \sum_{j=0}^{|E|} p_{(e_j,t,r)} \leq 1 \tag{3}$$

2.1.4. Soft constraints

- Students should not have to attend lectures in the last timeslot of the day.

$$S1: \sum_{d=0}^{4} \sum_{l=0}^{|R|} \sum_{j=0}^{|E|} \sum_{i=0}^{|S|} p_{(e_j,t_{9d+8},r_l)} \times a_{(s,e_j)} \tag{4}$$

- They should not attend more than two lectures in a row.

$$S2: \sum_{d=0}^{4} \sum_{i=0}^{|S|} \sum_{k=9d}^{9d+7} h(), \tag{5}$$

$$h(\,) = \begin{cases} 1 & \text{si} \sum_{n=0}^{2} \sum_{t=0}^{|R|} \sum_{j=0}^{|E|} p_{(e_j,t_k,r_l)} \times a_{(s_i,e_j)} = 3 \\ 0 & \text{otherwise} \end{cases}$$

- They should not have only one lecture in any given day.

$$S3: \sum_{d=0}^{4} \sum_{i=0}^{|S|} q(\,), \tag{6}$$

$$q(\,) = \begin{cases} 1 & \text{si} \sum_{k=9d}^{9d+9} \sum_{t=0}^{|R|} \sum_{j=0}^{|E|} p_{(e_j,t_k,r_l)} \times a_{(s_i,e_j)} = 1 \\ 0 & \text{otherwise} \end{cases}$$

2.1.5. Objective function

$$Minimize\{S1 + S2 + S3\} \tag{7}$$

The goal of solving the problem formulated in such a way is to minimize the number of soft constraint violations.

3. Cultural algorithms

Cultural algorithms were developed by Reynolds [21] as a complement to the metaphor used by evolutionary algorithms [10], which had focused mainly on genetic and natural selection concepts.

Cultural algorithms are based on some theories originated in sociology and archaeology which try to model cultural evolution (see for example [7]). Such theories indicate that cultural evolution can be seen as an inheritance process operating at two levels: (1) a micro-evolutionary level, which consists of the genetic material that an offspring inherits from its parents, and (2) a macro-evolutionary level, which consists of the knowledge acquired by individuals through generations. This knowledge, once encoded and stored, is used to guide the behavior of the individuals that belong to a certain population.

Culture can be seen as a set of ideological phenomena shared by a population [23]. Through these phenomena, an individual can interpret its experiences and decide its behavior. In these models, it can be clearly appreciated the part of the system that is shared by the population: the knowledge, acquired by members of a society, but encoded in such a way that such knowledge can be accessed by every other member of the society. And then there is an individual part, which consists of the interpretation of such knowledge encoded in the form of symbols. This interpretation will produce new behaviors as a consequence of the assimilation of the corresponding knowledge acquired, combined with the information encoded in the ancestors' genes.

Reynolds [21] attempts to capture this double inheritance phenomenon through his proposal of cultural algorithms. The main goal of such algorithms is to increase the learning or convergence rates of an evolutionary algorithm such that the system can respond better to a wide variety of problems [11].

Cultural algorithms operate in two spaces. First, there is the population space, which consists of (as in all evolutionary algorithms) a set of individuals. Each individual has a set of independent features that are used to determine its fitness. Through time, such individuals can be replaced by some of their descendants, which are obtained through the application of a set of operators from the population.

The second space is the belief space, which is where the knowledge, acquired by individuals through generations, is stored. The information contained in this space must be accessible to each individual, so that they can use it to modify their behavior. In order to join the two spaces, it is necessary to provide a communication link, which dictates the rules regarding the type of information that must be exchanged between the two spaces.

Algortihm 1 shows the pseudo-code of a cultural algorithm.

Algorithm 1. Pseudo-code of a cultural algorithm.

```
Generate the initial population
Initialize the belief space
Evaluate the initial population
repeat
  Update the belief space (with the individuals accepted)
  Apply the variation operators (under the influence of the belief space)
  Evaluate each child
  Perform selection
until the end condition is satisfied
```

Most of the steps of a cultural algorithm correspond with the steps of a traditional evolutionary algorithm. It can be clearly seen that the main difference lies in the fact that cultural algorithms use a belief space. In the main loop of the algorithm, the belief space must be updated. It is at this point in which the belief space incorporates the individual experiences of a select group of members of the population. Such a group is obtained with the function *accept*, which is applied to the entire population.

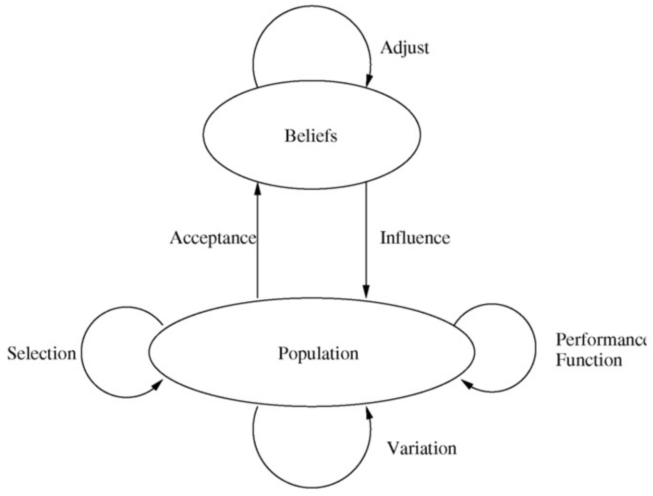


Fig. 1. Spaces of a cultural algorithm.

On the other hand, the variation operators (such as recombination or mutation) are modified by the function *influence*. This function applies some pressure such that the children resulting from the variation operators can exhibit behaviors closer to the desirable ones and farther away from the undesirable ones, according to the information stored in the belief space.

These two functions (*accept* and *influence*) constitute the communication link between the population space and the belief space. Such interactions can be appreciated in Fig. 1[22]. The implementation details for these functions in the current proposal are given in the next section.

In [21], it is proposed the use of genetic algorithms [14] to model the micro-evolutionary process, and Version Spaces [20] to model the macro-evolutionary process of a cultural algorithm. This sort of algorithm was called the *Version Space guided Genetic Algorithm* (VGA). The main idea behind this approach is to preserve beliefs that are socially accepted and discard (or prune) unacceptable beliefs. Therefore, if a cultural algorithm for global optimization is applied, the acceptable beliefs can be seen as constraints that direct the population at the micro-evolutionary level [19].

In genetic algorithms' theory, there is an expression, called *schema theorem* [15] that represents a bound on the speed at which the best schemata of the population are propagated. Reynolds [21] provided a brief discussion regarding how the belief space could affect the schema theorem. His conclusion was that, by adding a belief space to an evolutionary algorithm, the performance of such algorithm can be improved by increasing its convergence rate. That constitutes the main motivation to use cultural algorithms. Despite the lack of a formal mathematical proof of this efficiency improvement, there is empirical evidence of such performance gains reported in the literature (see for example [4,5]).

4. Proposed approach

The approach proposed in this paper uses, in its population space, a population based on the evolutionary algorithm originally proposed in [18]. A pseudo-code with the main steps of the proposed cultural algorithm is shown in Algorithm 2.

In our algorithm, we have considered three types of knowledge: situational, normative and domain knowledge. Also, we are using other three variation operators, designed to add the exploration component of the algorithm (interchange, sequencing and simple mutation). It is worth mentioning that only one of the exploration operators is applied to each individual.

4.1. Representation

The representation adopted to encode the solutions plays a very important role when applying an evolutionary computation technique [26,24]. In this case, a matrix representation was adopted, where columns represent slots of time, and rows represent rooms for the events. The entries of the matrix are the events themselves.

This encoding was chosen because it can represent any feasible timetable, and is easier to analyze the violation of some hard constraints, considering only one column at a time.

Algorithm 2. Pseudo-code of the cultural algorithm adopted.

```
Generate s random schedules (initial population)
Compute the fitness of each individual in the initial population
Initialize the belief space (copying the best individual to the situational belief
  space and create the normative matrix)
repeat
  for each individual in current population do
    Apply cultural mutation operator
    switch (exploration operator)
      case Interchange:
        Apply Interchange Operator
      case Sequencing:
        Apply Sequencing Operator
      case SimpleMutation:
        Apply Simple Mutation Operator
    end switch
    Apply repair operator (with domain knowledge)
  end for
  Selection process
  Update the belief space (with the individuals accepted)
until the end condition is satisfied
```

4.2. Mutation operators with cultural influence

The operator begins selecting an event e and a position (r, t) to move it. This is done through different types of cultural influence.

4.2.1. Situational influence

With the situational influence each individual tries to follow a leader. Such a leader is the best individual found, and is stored in the situational belief space. The key idea is that the individual to be mutated becomes more similar to the leader after the mutation process. The mutation operator randomly selects an event e from the leader, and tries to inherit its position (r, t) to the individual.

The situational belief space is updated at each generation. If the best individual of the current generation is better than the leader in the situational belief space, then the leader is replaced by that individual.

4.2.2. Normative influence

This type of influence is more complex. At each generation, the above average individuals are selected. The idea is to influence the individual to be mutated to inherit some of their characteristics. Before describing the procedure, we need the following definitions: We define a *ranking of events* as the set of all the events ordered by the number of events with shared students among them. Thus, the event most connected with other events is the first in the ranking. Given a population $P(gen)$ of the generation gen and the set S_{gen} composed by the best individuals, we define the matrix M where each element m_{ij} is the timeslot assigned to the event i in the individual j which belongs to S_{gen}.

The operator proceeds as follows. The room r is fixed. The event is chosen from the ranking of events using a roulette wheel procedure which is biased to the most interconnected events. The new timeslot in the same room r is randomly selected from the matrix M, thus the most common timeslot t of the event e in M has the largest probability of being selected. The hardest event to be assigned, from the constraints point of view, is the event that shares students with the largest number of events.

The matrix M is updated at each generation, after the selection of the set S_{gen} (the above average individuals).

Once an event e and the position (r, t) have been selected (by any of the cultural influences mentioned), the process of mutation continues as shown in Algorithm 3. If the new position selected (r, t) is empty and if it is feasible to place e there (from the hard constraints point of view), the current position of event e is modified to (r, t). If another event e_m is in (r, t), the operator makes swapping moves to change e_m to another position, in order to release (r, t).

Algorithm 3. Mutation(e, (r, t)) procedure, which implements mutation after the influence of cultural selection.

```
$mutation\_finished = FALSE$
while $mutation\_finished \neq TRUE$ or $maxtries < 1000$ do
  if the position $(r, t)$ of the chromosome is empty then
    try to move the event $e$ from its original position, to $(r, t)$, satisfying the
  hard constraints
  else
    try a swapping move of the event $e_m$ in $(r, t)$
  end if
  if the position of $e$ has changed then
    $mutation\_finished = TRUE$
  end if
end while
```

4.3. Domain knowledge

Our algorithm makes a post-processing procedure which uses the domain knowledge to modify individuals. In the timetabling problem, it is known that the best solution does not include events in the last timeslots of each day. Thus, the purpose of the repair operator is to try to move the events located in the last timeslots to the earliest ones, always satisfying the hard constraints.

4.4. Exploration operators

The exploration operators are those that allow to maintain diversity of the population. They are listed next.

The sequencing operator is similar to the one in [17], and its intention is to generate a large change in the individual since it interchanges two timeslots (this operator is the most destructive one used here).

The interchange operator of [18], interchanges two events, and its purpose is to modify the individuals when the problems have in their feasible solutions the same number of places available and events to assign.

The simple mutation operator changes the place of an event, and it is useful when the problems have more places available in their feasible solutions (without considering the last periods of the day) than events to assign. The last two operators make use of the matching algorithm [13] to increase their rate of success.

4.4.1. Parameter control for the application of exploration operators

The parameter control is a process, concurrent to the search of solutions, that allows values of the parameters to change during this process [8]. We use a mechanism of parameter control in order to select the exploration operator (interchange, sequencing or simple mutation) to apply during the mutation process, using a roulette wheel and based on the success rate of each operator.

This mechanism consists of updating the probability of each operator to be applied, following some simple rules.

If the application of the operator number i results in an improvement of the fitness of the generated individual (with respect to his parent) ($f_{cur} < f_{prev}$), the update of the probabilities is made as follows:

$$operator[i] = operator[i] + \Delta variation$$

where $operator[\,]$ is the array that contains the probabilities of the operators to be applied, $\Delta variation = (f_{prev} - f_{cur})/(f_{prev} + f_{cur})$, and $\forall j \in \{1, \ldots, NumOper\}$ and $i \neq j$, $operator[j] = operator[j] - (\Delta variation/(NumOper - 1))$, with $NumOper = 3$ in this case, because we have three operators.

This technique to update the operator rates was taken from the techniques to adapt the population size proposed by Eiben et al. [9] and Michalewicz et al. [1].

When an operator i is applied and the present solution gets worse ($f_{cur} > f_{prev}$); the updating of the probabilities is made as follows:

$$operator[i] = operator[i] - \Delta variation \times \alpha$$

where $\alpha = PresentTime/TotalTime$, and $\forall j \in \{1, \ldots, NumOper\}$ and $i \neq j$, $operator[j] = operator[j] + (\Delta variation \times \alpha/(NumOper - 1))$.

The goal of incorporating the α factor is to maintain controlled the level of decrement, with the objective of not disturbing those operators whose decreasing ranks are much greater, like the sequencing operator.

Initially, the three operators in competition start with the same probability of being chosen: $\forall i \in \{1, \ldots, NumOper\}$, $operator[i] = 1/NumOper$. In order to assure that all operators always have a probability $\neq 0$ of being chosen, all values in $operator[\,]$ remain between $MinProb = 0.1$ and $MaxProb = 0.8$.

5. Comparison of results

The cultural algorithm (CA) is compared with respect to three different approaches: a simulated annealing (SA) that was the winner of the competition [17], a recent version of a memetic algorithm (MA) [25] and the evolutionary algorithm (EA) on which this work is based [18]. These references were chosen because they are representative of the state-of-the-art and very competitive on the timetabling problem. The comparison with another EA shows the improvement obtained with the incorporation of culture. The SA approach is not an EA, but it was adopted in our comparative study because it remains as the best approach known so far for the timetabling problem.

The benchmark adopted to make the tests and comparisons are the 20 instances of UCTP from the timetabling competition [16]. Those problems are characterized for being of varied difficulty, they consider the individual satisfaction of the students (which allows to consider them individually, not in classes nor groups), and have at least one solution that fulfills both types of restrictions.

The proposed approach was implemented in the C++ programming language and was compiled using the GNU g++ compiler in the operating system Debian 3.1. Also, the matching algorithm found in the LEDA library [13] was used.

The cultural algorithm spent a CPU time of 360 s per run, as was required by the benchmarking rules of the timetabling competition, for our system configuration. The population size was set to 5, allowing the execution of a larger number of generations; also, this value seemed to be a good choice for all the problems after numerous experiments. Regarding the application rates for the operators, they were adapted with the parameter control mechanism previously described, so they don't require additional parameters.

5.1. Performance of the operators with cultural influence

As an initial experiment, we should want to assess the performance of the operators with cultural influence, to ensure that the overall performance of the algorithm is due to the synergic improvements of all the operators, and not only those adopted from previous works.

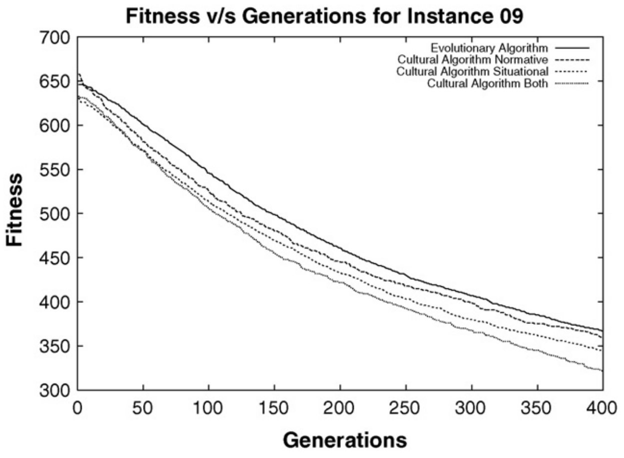


Fig. 2. Best performance of the algorithm with two influences.

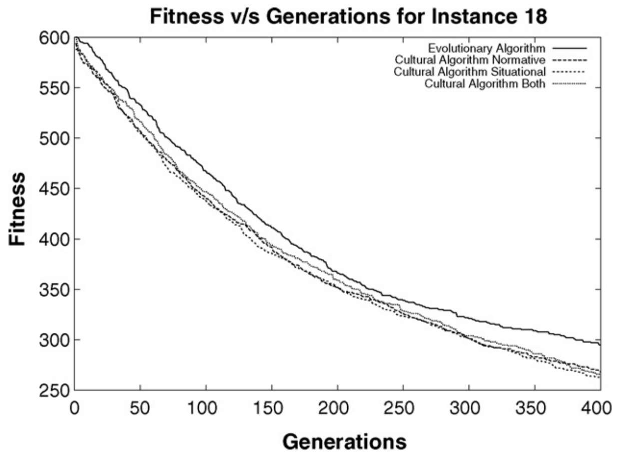


Fig. 3. Best performance of the algorithm with situational influence.

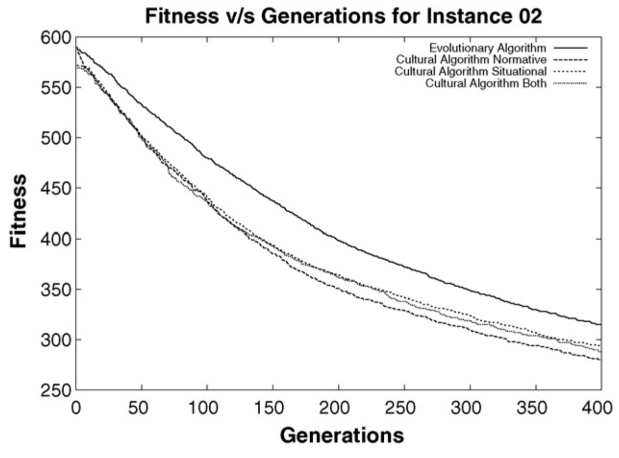


Fig. 4. Best performance of the algorithm with normative influence.

For that end, we compared three modified algorithms: the first one uses only situational beliefs, the second one uses only normative beliefs and the third one uses both kind of beliefs applied with the same probability. We compare the three algorithms among them and also with the original evolutionary algorithm. Ten trials were carried out for the four algorithms for each of the 20

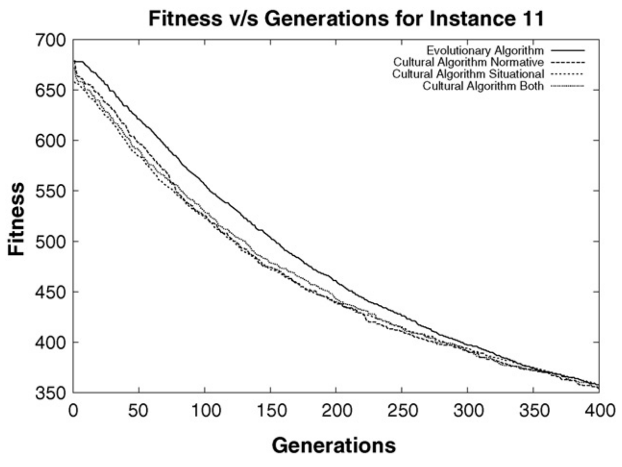


Fig. 5. Worst case results.

problems. The results are the average of these 10 runs for each of them.

We can identify that the cultural algorithm in all of its versions outperforms the original evolutionary algorithm. The algorithm is able to find better solutions, in the same amount of time, than the evolutionary algorithm. In Fig. 2 is shown the convergence graph for the instance 9, where the algorithm with both influences performs better; this is the case for the 45% of the problems. In 25% of the problems, the algorithm with situational influence performs better, as is the case of the instance 18, shown in Fig. 3. In the other 30% of the problems, the algorithm with normative influence performs better, for example in the instance 2, shown in Fig. 4.

Fig. 5 shows the behavior of the algorithms for the worst case, i.e., the lower improvement in performance among the evolutionary algorithm and the cultural ones.

5.2. Cultural algorithm and evolutionary algorithm

Now we know that the operators with cultural influence may improve the performance of the evolutionary algorithm, we should want to assess the performance of the whole algorithm, with all the operators described in previous sections, and also with the mechanisms of parameter control for the application of the operators.

The graphs of Fig. 6 show the best and the worst case of improvement of CA with respect to EA, in the 20 instances considered. The worst case Fig. 6(a)) and the best behavior (Fig. 6(b)) consider a significant improvement in the first stages which is reflected directly in the final result, in which the cultural algorithm has better results. These graphs show that, as expected, the incorporation of culture tends to accelerate the convergence of the algorithm and to improve the results.

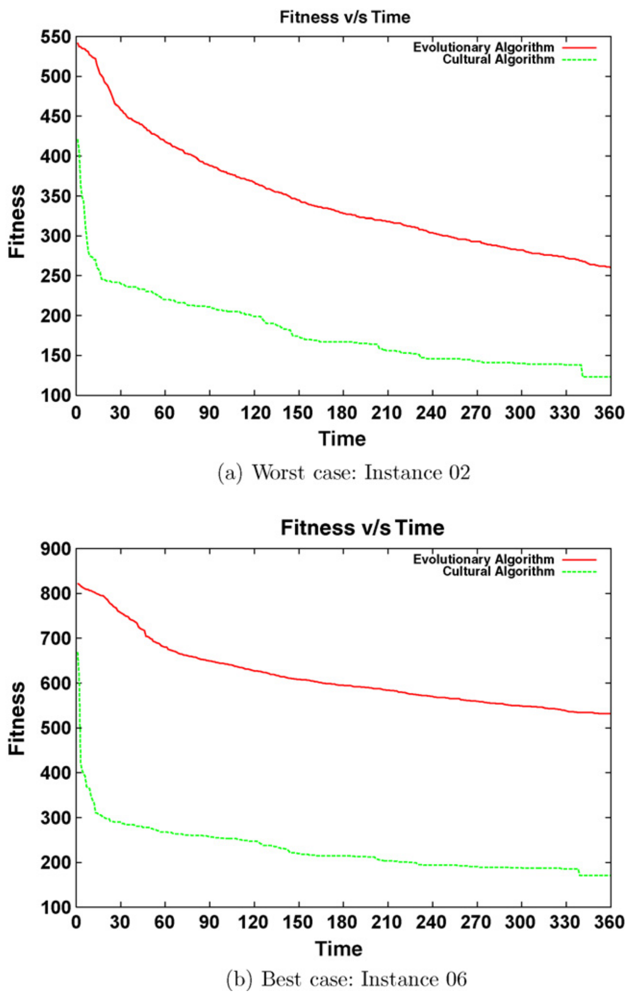


Fig. 6. Comparison in time

Table 1
Comparison of results.

Instance	EA	CA	MA	SA
1	288	140	104	45
2	260	123	91	25
3	322	149	126	65
4	679	330	189	115
5	557	306	212	102
6	532	171	90	13
7	430	159	127	44
8	305	133	94	29
9	283	101	78	17
10	311	147	113	61
11	328	120	90	44
12	350	187	138	107
13	420	233	185	78
14	469	267	187	52
15	400	204	120	24
16	302	102	74	22
17	521	311	182	86
18	254	100	75	31
19	550	296	224	44
20	424	159	60	7

Table 2
Improvement of the results of CA compared with the three other approaches.

Instance	EA–CA (%)	MA–CA (%)	SA–CA (%)
1	51.4	−25.7	−67.9
2	52.7	−26	−79.7
3	53.7	−15.4	−56.4
4	51.4	−42.7	−65.2
5	45.1	−30.7	−66.7
6	67.9	−47.4	−92.4
7	63	−20.1	−72.3
8	56.4	−29.3	−78.2
9	64.3	−22.8	−83.2
10	52.7	−23.1	−58.5
11	63.4	−25	−63.3
12	46.6	−26.2	−42.8
13	44.5	−20.6	−66.5
14	43.1	−30	−80.5
15	49	−41.2	−88.2
16	66.2	−27.5	−78.4
17	40.3	−41.5	−72.3
18	60.6	−25	−69
19	46.2	−24.3	−85.1
20	62.5	−62.3	−95.6

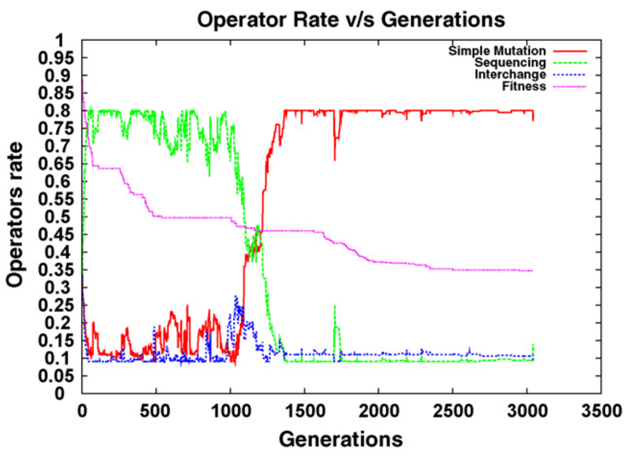
Table 3
Summary of results for all instances.

Algorithm	Average	Standard deviation
EA	399.25	119.46
CA	186.9	76.58
MA	127.95	50.72
SA	50.55	32.39

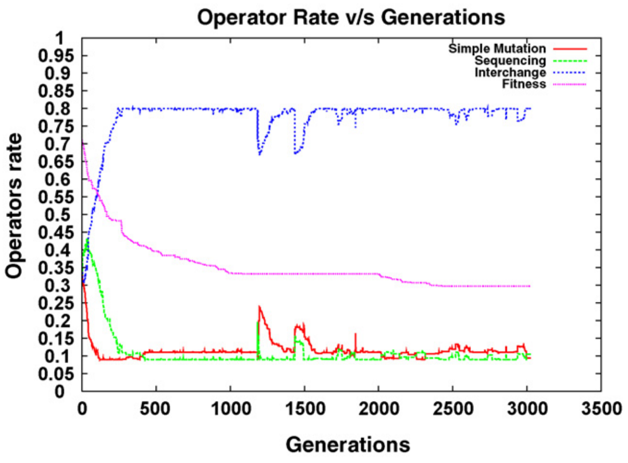
5.3. Cultural algorithm and other algorithms

Table 1 shows the results obtained by each algorithm in the 20 instances. Table 2 shows a comparison of distances of the cultural algorithm with respect to the other three algorithms. Table 3 shows a summary of the obtained results emphasizing that the CA improves all the results of the EA. The results of the CA are very close in quality from those of the MA. Finally, SA is still the most robust approach to solve timetabling problems, as can be seen in Table 3. It is important to indicate that SA has many special components especially used for the tests of the timetabling competition.

The most important one, as mentioned by its author, is that it incorporates some pre-processing tricks which discard some components of some instances that will never be used, because they cannot be part of a feasible solution. This significantly reduces the original search space. On the other hand, MA uses sophisticated heuristics coming from 3-color graphs in order to construct half of its initial population. Furthermore, the algorithm applies a local search procedure to the initial population of 10 individuals so that they are improved before starting the evolution. In MA, the specialized recombination also receives help from the heuristic adopted to construct the initial population. Both algorithms used many heuristics before starting their search process that helps them to give better results than our approach. In any case, the goal of this paper is to show that the use of a cultural algorithm can bring benefits, mainly through the achievement of better results. We believe that this can be clearly observed from the results reported in the paper.



(a) Instance 20, with more places than events



(b) Instance 09, with the same number of places and events

Fig. 7. Operators' rate of application

5.4. Adaptation on operators application rate

The incorporation of a mechanism to control the parameters of the cultural algorithm, during the selection of the operator to use, resulted in an improvement on the performance of every instance of the benchmark. The graphs of Fig. 7 show two representative instances of the UCTP. One of them is the instance number 20 (Fig. 7(a)) where 350 events in 400 places are considered; in such a case the simple mutation operator resulted useful because an important factor was the number of free places to assign events. On the other hand, instance number 09 (Fig. 7(b)) has less options to schedule an event, while it has 440 events and just 440 places; in this case, the interchange operator was more useful.

6. Conclusions and future work

In this paper, we propose the use of domain knowledge, both *a priori* and extracted during the search, to improve the performance of an evolutionary algorithm when solving timetabling problems. The executed experiments provided very encouraging results.

As a future work it would be very interesting to analyze the mechanisms of the simulated annealing method, in order to incorporate them in an evolutionary algorithm or a cultural algorithm. Also, the development of a classification of instances, is a very interesting topic for future research, mainly to better understand the performance of different algorithms on different instances.

Acknowledgment

The fourth author acknowledges support from CONACYT/CONICYT collaboration project.

References

[1] J. Arabas, Z. Michalewicz, J. Mulawka, GAVaPS-A genetic algorithm with varying population size, in: Proceedings of the 1st IEEE Conference on Evolutionary Computation, 1994.
[2] H. Arntzen, A. Lkketangen, A local search heuristic for a university timetabling problem, International Timetabling Competition, 2003.
[3] M. Chiarandini, K. Socha, M. Birattari, O. Rossi-Doria, International timetabling competition. A hybrid approach. Technical Report AIDA-03-04, Intellectics Group, Computer Science Department, Darmstadt University of Technology, Darmstadt, Germany, March 2003.
[4] C.-J. Chung, R.G. Reynolds, CAEP: an evolution-based tool for real-valued function optimization using cultural algorithms, Journal on Artificial Intelligence Tools 7 (3) (1998) 239–292.
[5] C.A. Coello Coello, R. Landa Becerra, Adding knowledge and efficient data structures to evolutionary programming: a cultural algorithm for constrained optimization, in: W.B. Langdon, et al. (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002), San Francisco, California, July 2002. Morgan Kaufmann Publishers, pp. 201–209.
[6] J.F. Cordeau, B. Jaumard, R. Morales, Efficient timetabling solution with tabu search. International Timetabling Competition, 2003.
[7] W.H. Durham, Co-evolution: Genes, Culture and Human Diversity, Stanford University Press, Stanford, California, 1994.
[8] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation 3 (July 2) (2000) 124–141.
[9] A.E. Eiben, E. Marchiori, V.A. Valko. Evolutionary Algorithms with on-the-fly Population Size Adjustment. Parallel Problem Solving from Nature, PPSN VIII, vol. 3242, pp. 41–50, 2004.

[10] D.B. Fogel, Evolutionary Computation. Toward a New Philosophy of Machine Intelligence, The Institute of Electrical and Electronic Engineers, New York, 1995.
[11] B. Franklin, M. Bergerman, Cultural algorithms: concepts and experiments, in: Proceedings of the 2000 Congress on Evolutionary Computation, Piscataway, New Jersey. IEEE Service Center, 2000, pp. 1245–1251.
[12] L. Di Gaspero, A. Schaerf, Timetabling competition ttcomp 2002: solver description. International Timetabling Competition, 2003.
[13] Algorithmic Solutions Software GmbH. Leda 5.1. http://www.algorithmic-solutions.com/index.htm, 2006.
[14] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
[15] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, Michigan, 1975.
[16] IDSIA. International timetabling competition. http://www.idsia.ch/Files/ttcomp2002/, 2003.
[17] Philipp Kostuch. Timetabling compatition—sa-based heuristic. International Timetabling Competition, 2003.
[18] R. Lewis, B. Paechter, New crossover operators for timetabling with evolutionary algorithms, in: 5th International Conference on Recent Advances in Soft Computing (RASC 2004), vol. 5 of Lecture Notes in Computer Science, Nottingham, UK, 2004, pp. 189–195.
[19] Z. Michalewicz, A survey of constraint handling techniques in evolutionary computation methods, in: J.R. McDonnell, R.G. Reynolds, D.B. Fogel (Eds.), Proceedings of the 4th Annual Conference on Evolutionary Programming. The MIT Press, Cambridge, Massachusetts, 1995, pages 135–155.
[20] T. Mitchell, Version Spaces: An Approach to Concept Learning. PhD thesis, Computer Science Department, Stanford University, Stanford, California, 1978.
[21] R.G. Reynolds, An introduction to cultural algorithms, in: A.V. Sebald, L.J. Fogel (Eds.), Proceedings of the Third Annual Conference on Evolutionary Programming, World Scientific, River Edge, New Jersey, 1994, pp. 131–139.
[22] R.G. Reynolds, Cultural algorithms: theory and applications, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 367–377.
[23] J. Peter, R. Richerson, Boyd., Not By Genes Alone: How Culture Transformed Human Evolution, University Of Chicago Press, 2004, December.
[24] S. Ronald, Robust encodings in genetic algorithms, in: D. Dasgupta, Z. Michalewicz (Eds.), Evolutionary Algorithms in Engineering Applications, Springer-Verlag, 1997, pp. 30–44.
[25] O. Rossi-Doria, B. Paechter, A memetic algorithm for university course timetabling, in: Proceedings of Combinatorial Optimisation, CO, 2004.
[26] F. Rothlauf, Representations for Genetic and Evolutionary Algorithms, Physica-Verlag, New York, 2002.
[27] K. Socha, M. Sampels, M. Manfrin, Ant algorithms for the university course timetabling problem with regard to the state-of-the-art, in: Proceedings of EvoCOP 2003 – 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization, vol. 2611 of LNCS, Springer-Verlag, April 2003, pp. 334–345.