Three heuristics to solve Timetabling

JUAN FRAUSTO-SOLÍS, FEDERICO ALONSO-PECINA, MONICA LARRE
Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Cuernavaca

CINTHIA M. GONZÁLEZ-SEGURA
Universidad Autónoma de Yucatán
gsegura@uady.mx
JOSÉ LUIS GÓMEZ-RAMOS
Universidad Juárez Autónoma de Tabasco

Abstract: - This work deals with one relevant problem in nowadays: the Educational Timetabling (ET). This problem consists in schedule meetings among teachers, and students, in a fixed time. Our work deals the problem with three heuristics very well known: Genetic Algorithms, Tabu Search and Simulated Annealing. This paper presents Timetabling’s results using instances of the benchmark of PATAT, and an analysis about the behavior of the three methods is presented; the experimentation shows that Simulated Annealing and then Tabu Search Outperforms the implementation of Genetic Algorithms.

Key-Words: - Timetabling, Tabu Search, Simulated Annealing, Genetic Algorithm, optimization

1 Introduction
Educational Timetabling Problem (ETTP) consists in fixing a sequence of meetings between teachers and students in a prefixed period of time (typically a week), satisfying a set of different classes of constraints [10]. In the literature, there are many variants of the timetabling problem; these variants are as many as the number of the school’s different requirements. Significant events as PATAT [9] show an increasing interest for the ETTP problem; instances of PATAT are usually taken as a challenge for ETTP research area. ETTP has been treated by several methods like the transformation of ETTP into graph coloring problem [6; 7], Genetic Algorithms [3], Tabu Search [4], Simulated Annealing [1], Constraint Satisfaction [12; 5], and many others. The manual solution of the timetabling problem usually requires considerably human effort, and the solution can be far away of the optimal solution.

The schools differ among them in their educational politics; in consequence, ETTP has many variants, classified in three categories [10]:

• School Timetabling: The objective is the weekly scheduling for all the classes of a high school, avoiding teachers meeting two classes in the same time, and vice versa.
• Course University Timetabling: This problem deals with the weekly scheduling for all the lectures of a set of university courses, minimizing the amount of overlaps among lectures and courses having common students.
• Examination Timetabling: The objective is to find the schedule for the exams of a set of university courses, avoiding overlapping exams of courses having common students, and spreading the exams of the students as much as possible.

For the ETTP, there are some constraints that should be completely satisfied to get a practical solution; these are known as hard constraint (hc). A solution satisfying absolutely all the hc of an ETTP is named a feasible solution. Additionally, some constraints are desirable to be satisfied, but they are not completely necessary to get a feasible timetable; these constraints are called as “soft constraints” [11].

To solve the ETTP problem known as Course University Timetabling, algorithms based on the common metaheuristics Simulated Annealing (SA), Tabu Search (TS) and Genetic Algorithms (GA) are
explored. The algorithms implemented are tested with PATAT instances. The paper is organized as follows: in section 2 the problem is described. In section three, the instances to test the algorithms are presented. Section four shows the implementations and the results gotten with them are given in section five. Finally, in section six the conclusions and future work are discussed.

2 Problem Description
For Timetabling problems, as for any other NP problem, it is not efficient to apply exhaustive and/or deterministic methods due to their exponential complexity [14]; also, it is useful to create automatic methods to generate the best possible solutions. To assure the validity of the results obtained with the algorithms tested here, instances taken from PATAT’s benchmark (Practice and Theory of Automated Timetabling) are used [8]. The selected problem consists of: i) a set of events (E) to be scheduling in 45 periods of time (5 days of 9 intervals every one), ii) a set of classrooms (R) that will hold the events, iii) a set of students (U) that attend the events, and iv) a set of features (F), satisfied by rooms and required by events. Every student attends some events and every room has its capacity.

A feasible timetable is one in which all events have been assigned in a timeslot to a classroom, so that the following hard constraints are fulfilled:

- No student attends more than one event at the same time.
- The classroom is big enough to house all the attending students and satisfies all the technical features required by the event.
- Only one event is in each classroom at any timeslot.

In addition, satisfying the next soft constraint is desirable:

- A student should not have a class in the last slot of the day
- A student should not have more than two classes consecutively
- A student should not have a single class on a day.

The PATAT’s criteria to determine the winner of the contest was based on the equation (1) [8]:

\[ F_i = \frac{1}{(w_i - b_i)} \]  

Where \( i \) is the number of the instance (1 ≤ \( i \) ≤ 20), \( x \) is the number of soft constraints violations for the contestant; \( b \) is the number of soft constraints violations of the best participant on this instance, and \( w \) the number of soft constraint violations of the worst participant on this instance.

3 Problem’s Instances
In [11], a generator to produce instances from different seeds is used. All the instances generated have at less one perfect solution, that is, zero violated constraint (hard and soft). In the table, the features of PATAT’s instances are presented, and table contains the criteria to classify them. The small cases are called S1, S2, S3, S4 and S5. The medium instances are call M1, M2, M3, M4 and M5, and the large instances are L1 and L2. For the PATAT, every instance Competitionx is called Cx, (e.g., Competition01 is C01, and so on).

<table>
<thead>
<tr>
<th>Instance</th>
<th>Events (E)</th>
<th>Rooms (R)</th>
<th>Features (F)</th>
<th>Students (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C01</td>
<td>400</td>
<td>10</td>
<td>10</td>
<td>200</td>
</tr>
<tr>
<td>C02</td>
<td>400</td>
<td>10</td>
<td>10</td>
<td>200</td>
</tr>
<tr>
<td>C03</td>
<td>400</td>
<td>10</td>
<td>10</td>
<td>200</td>
</tr>
<tr>
<td>C04</td>
<td>400</td>
<td>10</td>
<td>5</td>
<td>300</td>
</tr>
<tr>
<td>C05</td>
<td>350</td>
<td>10</td>
<td>10</td>
<td>300</td>
</tr>
<tr>
<td>C06</td>
<td>350</td>
<td>10</td>
<td>5</td>
<td>300</td>
</tr>
<tr>
<td>C07</td>
<td>350</td>
<td>10</td>
<td>5</td>
<td>350</td>
</tr>
<tr>
<td>C08</td>
<td>400</td>
<td>10</td>
<td>5</td>
<td>250</td>
</tr>
<tr>
<td>C09</td>
<td>440</td>
<td>11</td>
<td>6</td>
<td>220</td>
</tr>
<tr>
<td>C10</td>
<td>400</td>
<td>10</td>
<td>5</td>
<td>200</td>
</tr>
<tr>
<td>C11</td>
<td>400</td>
<td>10</td>
<td>6</td>
<td>220</td>
</tr>
<tr>
<td>C12</td>
<td>400</td>
<td>10</td>
<td>5</td>
<td>200</td>
</tr>
<tr>
<td>C13</td>
<td>400</td>
<td>10</td>
<td>6</td>
<td>250</td>
</tr>
<tr>
<td>C14</td>
<td>350</td>
<td>10</td>
<td>5</td>
<td>350</td>
</tr>
<tr>
<td>C15</td>
<td>350</td>
<td>10</td>
<td>10</td>
<td>300</td>
</tr>
<tr>
<td>C16</td>
<td>440</td>
<td>11</td>
<td>6</td>
<td>220</td>
</tr>
<tr>
<td>C17</td>
<td>350</td>
<td>10</td>
<td>10</td>
<td>300</td>
</tr>
<tr>
<td>C18</td>
<td>400</td>
<td>10</td>
<td>10</td>
<td>200</td>
</tr>
<tr>
<td>C19</td>
<td>400</td>
<td>10</td>
<td>5</td>
<td>300</td>
</tr>
<tr>
<td>C20</td>
<td>350</td>
<td>10</td>
<td>5</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>Events (E)</th>
<th>Rooms (R)</th>
<th>Features (F)</th>
<th>Students (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>100</td>
<td>5</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>medium</td>
<td>400</td>
<td>10</td>
<td>5</td>
<td>200</td>
</tr>
<tr>
<td>large</td>
<td>400</td>
<td>10</td>
<td>10</td>
<td>400</td>
</tr>
</tbody>
</table>

4 Implementation and results
The three algorithms (Simulated Annealing, Tabu Search and Genetic Algorithms) begin in a feasible
solution. This initial solution is gotten for a heuristic for which the concept of “more constrained event” is used. Of course, the availability of events is updated, every time the algorithm assigns one event. One concept used in the proposed algorithm is named Conflict between two events (conflict in short); a period is other concept related with the former. Both of these concepts are defined:

DEFINITION 1: The event \( E_i \) has a conflict with an event \( E_j \) if both of them have at least a common student.

DEFINITION 2: A period is an interval where it is possible to schedule an event.

To choose the “more constrained event” the saturation concept is used [15]. In order to do that a measure for each event and its related constraint is introduced as “constraint level” as follows:

DEFINITION 3: An event \( E_1 \) has a lower (higher) constraint level than an event \( E_2 \) if \( E_1 \) has lower (higher) constraint violation of its constraints than \( E_2 \).

In order to have a measure of constraint level some enchanted rules are proposed in this paper. An event \( E_i \) will be higher (lower) constraint level that an event \( E_j \) according with the next Constraint Level Rules (CLR):

- CLR No. 1: If the number of periods available for the event \( E_i \) will be lower (higher) than the number of periods available for \( E_j \), then the event \( E_j \) has a lower (higher) constraint violation of its constraints than \( E_i \).
- CLR No. 2: The event \( E_i \) has higher (lower) constrained level that the event \( E_j \) if \( E_i \) has less (more) classrooms availability than \( E_j \). Otherwise, in case of a tie, the third CLR is applied.
- CLR No. 3: The event \( E_i \) has higher (lower) constraint level that the event \( E_j \) if \( E_i \) has more (less) conflicts with others events \( E_k \) \((k\neq2)\). Otherwise, in case of a tie, the lexicographic rule is applied.

The process to make the schedule, once an event \( E_i \) is chosen using CLRs is as follows:

1. Use the first room \( a_1 \) that hold the event \( E_i \). At the beginning, the first periods of the day should be intended. If there is not any room available for these periods of that day, then try the first periods of the next day, and so on. The last periods of the day are assigned with low priority and only if the event can not be scheduled in first periods.
2. Once an event of a classroom and its period are assigned it is necessary to update the availability of the rest of the unscheduled events and the availability of the classrooms.
3. Step 1 and 2 are repeated taken other event as \( E_i \) using for that CLRs.

The algorithm proposed here to find an initial feasible solution for Timetabling uses CLRs and is now presented:

```
Begin
Itera = 0;
Lista_conflictos = \emptyset;
Do
Cont = 0;
L = all the events;
While Lista_conflictos <> \emptyset
    E = First element of Lista_conflictos
    Assign classroom and period to E
    Update the list L
    Lista_conflictos = Lista_conflictos - E
    Cont = Cont + 1;
End_While
While Cont < number_of_events
    Sort the events unscheduled according its level of constrained
    Choose the first higher event (E)
    L = L - E
    if exist some classroom and period available to assign the event and preserve the feasibility then
        Choose the first one
    else
        ban_feasible = False
        Assign to the event E a classroom and a period available (even though violate hard constraints)
        Lista_conflictos = Lista_conflictos + E;
    End_if
    Cont = Cont + 1;
End_while
Itera = Itera + 1;
While Itera < 10 and ban_feasible == False
End
```

In this pseudocode all the variables are defined except the following:

- \( L \) is the list of unscheduled events.
- \( Lista_conflictos \) is a list that has events that can not be assigned without break some hard constraint.
Tabu Search
For the Tabu algorithm proposed here, two neighborhoods called TS1 and TS2 are implemented:

- TS1 neighborhood: For a given solution, two steps to get a neighbor of it should be done. Both of these steps take care of do not violate any hard constraint and the entire neighborhood is considered. These steps are: a) Two events are interchanged and then b) A movement of an event to an available place is done.
- TS2 neighborhood: A neighbor is found using the two steps of TS1 neighborhood and then applying the next step: c) Interchange all the events of two periods \( k_1 \) and \( k_2 \) where \( k_1 \neq k_2 \).

The pseudocode of the Tabu algorithm proposed here is the next:

```plaintext
Begin
  x* = initial solution
  x_actual = x*
  t = tenure of the tabu list
  LCandi = ∅;
  LTabu = ∅;
  Cont=0;
repeat
  LCandi = list of candidate solutions take of the neighborhood of x_actual
  x = the best solution of LCandi
  if f(x) < f(x*) then
    x* = x
    x_actual = x
    cont=0;
  else
    cont=cont+1;
    if f(x)<f(x_actual) y x ≠ Ltabu then
      x_actual = x
    else
      x = one random solution of LCandi
      x_actual = x
  end_if
  LTabu = LTabu ∪ (x, t);
  for every e e LTabu decrease the tenure of e
  if expire the tenure of e then
    LTabu = LTabu - (e, t)
  end_if
end_repeat
until Cont == 400 || x*==0
End
```

The tenure in the tabu list is a random number \( b \in \mathbb{Z}^+ \) between a [1, t] where t is tuned by experimentation. Taken in account the instances of benchmark used, twenty is the best value found for t.

In the Tabu algorithm proposed her, the stop criterion is as follows: a) The optimum solution (zero hard constraints violated and zero soft constraints violated) is found or b) Maximum number of iterations without improve the last best solution is reach. (400 iterations were used in this work).

Simulated Annealing
In Simulated Annealing implemented, for the timetabling problem are considered: the search space, the criterion of neighborhood for find new solutions, the objective function (cost function), the initial and final temperature \( T_0 \) and \( T_f \) respectively, and the cooling scheme temperature. The last one is modeled by a cooling function \( T_k = \alpha T_{k-1} \), where \( k \) is the iteration number, and \( \alpha \) is a parameter in (0.7, 1), that can be tuned in an experimental or analytical way [16].

The search space is restricted to only feasible solutions. Once a feasible solution is gotten, the algorithm starts working only with feasible solutions. This algorithm also uses the initial solution using CLR's algorithm described before.

The neighborhood used to generate new solutions is the next one:

- Select two random periods and two random classrooms
- If the interchange of events in these places is feasible, the interchange is accepted.
  - Otherwise if the interchange is not feasible, we select two new random periods and two random rooms and so on.
- If the new solution is better that the last one, then the new solution is accepted
  - Otherwise if the interchange produces a worse solution, then the new solution is accepted with an exponential probability.

The objective function evaluates the energy of a solution as the sum the hard constraints violated multiply by a constant (1000 in the case of PATAT) plus the number of soft constraints violated. However in the proposed algorithm the number of hard constraints violated is constraint to be zero, and so the algorithm proposed here always try to minimize the number of soft constraints violated.

The initial temperature of the algorithm can be obtained with the formula \( 470n + e \), where \( n \) is the number of students and \( e \) is the number of events of the problem. This formula was obtained considered the three hard constraints and the three soft constraints of the PATAT's problem. The final
temperature gotten after tune’s experimentation is 0.01. In every iteration, the temperature is decrease geometrically, \( T_n = \alpha T_{n-1} \), with \( \alpha = 0.9 \). Finally the Markov Chain in the Metropolis cycle, was set to 10,000. The algorithm implemented is:

\[
\text{Begin}
\]
\[
\begin{align*}
&x = \text{initial solution} \\
&\text{BestCost} = f(x) \\
&T = 470 \times \text{num\_students} + \text{num\_events} \\
&\text{END\_TEMP} = 0.01; \\
&\text{ALPHA} = 0.09; \\
&\text{iter}=0; \\
\text{While} (T > \text{END\_TEMP}) \\
&\text{While} (\text{iter} < \text{MAXNUMITER}) \\
&x_{\text{new}} = \text{perturb}(x); \\
&\text{costNew} = f(x_{\text{new}}); \\
&\text{costDiff} = \text{costNew} - \text{costIni}; \\
&\text{if} (\text{costNew} < 0) \text{ then} \\
&\text{costIni} = \text{costNew}; \\
&x = x_{\text{new}}; \\
&\text{else} \\
&r = \text{rand}(); \\
&\text{if} (r < \exp(-\text{costDiff}/T)) \text{ then} \\
&\text{costIni} = \text{costNew}; \\
&x = x_{\text{new}}; \\
&\text{End\_if} \\
&\text{End\_if} \\
&\text{if} (\text{BestCost} > \text{costIni}) \text{ then} \\
&x^{*} = x_l; \\
&\text{BestCost} = \text{costIni}; \\
&\text{End\_if} \\
&\text{iter} = \text{iter} + 1 \\
\text{End\_While} \\
&T = T \times \text{ALPHA} \\
\text{End\_While} \\
\text{End}
\]

Genetic Algorithms

In the implementation of Genetic algorithms for the timetabling problem, the chromosome is constituted for alleles, corresponding to a subset of the numbers \( Z^+ \cup 0 \). Here the allele is the event that we want to schedule, and its locus points both, the period and the room assigned to the event. The chromosome’s size is obtained of the multiply the number of rooms by the number of periods. The stop criterion is not found an improve in 50 generations.

The chromosome is generated in a determinist way starting from the initial solution using CLRs explained before. All the others elements of the population is generated by the next mutation operator.

```
Fig 1. Example of a chromosome
```

```
13 0 5 2 10 14 1 3 8 4 6 12 13 7 9
```

```
Fig 2 Chromosome’s representation
```

```
2 10 14
1 3 8
4 6 12
11 7 9
```

5 Results obtained

The algorithms implemented described before, was run in a PC with SO Windows XP, in a Toshiba Satellite, with processor Intel Celeron with 2.4 GHz and 512 MB of Memory RAM.

In [2] is done a comparison between the hyperheuristic HH, and the metaheuristics ANT, and their results are columns HH and ANT in table 3. The implementation of ANT was made in [11]. The results obtained by the algorithms Tabu Search, Simulated Annealing and Genetic algorithms appear also en table 3. TS1 and TS2 neighborhoods appear in columns of the same name in table 3. While Simulated Annealing and Genetic Algorithms results appear in columns SA and GA respectively. As we can notice from table 3, SA has better performance work (in quality of solutions) than all the others.
In the next two tables, the minor number of soft constraints violated for every algorithm HH, ANT, TS, SA and GA is given. The best result for every instance is in boldface. In table 4, the best results are italicized.

From the experimentation presented in this paper, it can be notice that the best solutions of Simulated Annealing or Tabu Search were obtained.

6 Conclusions

In this works, a novel implementation of Simulated Annealing, Tabu Search and Genetic Algorithms using CRLs to get a feasible initial solution is presented. CRLs are simple rules to move into a feasible space. Neighborhood rules for both SA and TS algorithms are also proposed. After the experimentation, good results are gotten if they are compared against those of the literature. All the three algorithms have improved notably the initial solution. Like can be observed in the results presented, Simulated Annealing implementation has a better performance for large and medium instances while Tabu Search is best performed for small ones.

The performance of algorithms tuned by an analytical will be presented in a future work.

References:


