Analytically Tuned Parameters of Simulated Annealing for the Timetabling Problem

JUAN FRAUSTO-SOLÍS, FEDERICO ALONSO-PECINA
Instituto Tecnológico y de Estudios Superiores de Monterrey, Cuernavaca, Morelos, México

CINHTIA GONZALEZ-SEGURA
Universidad Autónoma de Yucatán

Abstract: - University Timetabling problem (UTT) has a computational complexity that grows exponentially as the size of the problem augments, then random algorithms become indispensable to solve it; among these algorithms, Simulated Annealing (SA) is one of the most efficient algorithms. However, SA obtains the optimal solution or a very good approximation one, but only when SA parameters are well tuned. SA requires an initial solution for solving UTT. Besides, analytical tuning strategies for SA in UTT have not been explored. In this paper a SA Markov tuning strategy and a heuristic to generate a feasible solution are proposed. This strategy improves the performance of SA algorithms for ETT as is shown with experimental instances taken from PATAT benchmark.

Key-Words: - Timetabling, Simulated Annealing, Optimization

1 Introduction

This paper proposes a solution to an Educational TimeTabling problem proposed in PATAT [1], and named University Timetabling (UTT) which has a computational complexity that grows exponentially as the size of the problem is increased. UTT has been solved with many heuristics as Simulated Annealing (SA) [2] [3],[4] Tabu Search [5][6], Ant Systems [7], and so on. A common approach to solve UTT is finding an initial feasible solution with some heuristics specially developed for that purpose; then a random algorithm derived from well known metaheuristics [6], [8], [9] is commonly used to improve this solution. SA not always obtains good results for UTT [5], because SA should have a good neighborhood and be well tuned. In this paper we propose to solve UTT using two particular neighborhoods, and a Markov tuning strategy. This strategy allows a best performance of SA algorithms for UTT as is shown with experimental instances taken from PATAT benchmark [1].

The paper is organized as follows. Section 2 includes the PATAT problem description and the mathematical model. Section 3 presents the implementations of SA tuned experimentally. Section 4 describes the analytical tuned parameters and its improvements. Finally, Section 5 and Section 6 include the results and the conclusions respectively.

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 [8]; also, it is useful to create automatic methods to generate the best feasible solution. 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 [1]. The selected problem consists of: i) a set E of events to be scheduled in 45 periods of time (5 days of 9 intervals every one), ii) a set of classrooms (R) that hold the events, iii) a set of students (U) that attend the events, and iv) a set of features (F), that should be satisfied by the rooms because the requirements of the events. Every student will attend some events and every room has a capacity.

A feasible timetable is one in which all the events have been assigned in a specific timeslot, in a specific classroom, and all of the following hard constraints must be completely fulfilled:

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

In addition, is desirable that the next soft constraints be satisfied:

- 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 is based on the equation [1]:

$$\mathrm{F_i} = {(x_i - b_i)}\Big/{(w_i - b_i)} \quad (1)$$

Where i is the number of the instance ($1 \leq i \leq 20$), x is the number of soft constraints violated for the contestant; b is the number of soft constraint violated for the best participant on this instance, and w is the number of soft constraints violated for the worst participant on this instance. For the PATAT problem we have [1]:

- n Events: $E = \{e_1, e_2, \ldots, e_n\}$
- m Students: $U = \{u_1, u_2, \ldots, u_m\}$
- 45 Periods: $P = \{p1, p2, \ldots, p45\}$
- r Rooms: $A = \{a1, a2, \ldots, ar\}$
- r Room sizes: $C = \{c1, c2, \ldots, cr\}$
- t Features: $F = \{f1, f2, \ldots, ft\}$

We have three binary matrixes too [1]:

- Matrix student/event: D_{nxm}

$$D_{nxm} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}$$

Where $d_{il} = 1$ if the student l attend the event i, and 0 otherwise.

- Matrix room/feature: S_{txr}

$$S_{txr} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1t} \\ s_{21} & s_{22} & \cdots & s_{2t} \\ \cdots & \cdots & \cdots & \cdots \\ s_{r1} & s_{r2} & \cdots & s_{rt} \end{bmatrix}$$

where $s_{jf} = 1$ if the room j satisfy the feature f, and 0 otherwise.

- Matrix event/feature: Q_{txn}

$$Q_{txn} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1t} \\ q_{21} & q_{22} & \cdots & q_{2t} \\ \cdots & \cdots & \cdots & \cdots \\ q_{n1} & q_{n2} & \cdots & q_{nt} \end{bmatrix}$$

where $q_{if} = 1$ if the event i requires the feature f, and 0 otherwise.

Let be $x_{ijk} = 1$ if the event i, is assigned to the room j in the period k, and 0 otherwise.

Therefore, the problem can be formulated as follows:

$$\begin{aligned} &Min\ z = \sum scv \\ &subject\ to \\ &Every\ hcv\ is\ Fulfilled \end{aligned}$$

That means UTT consists on minimizing the objective function z = Number of soft constraints violated (scv), subject to the following hard constraints (hcv):

Hard Constraints:

1. **No student attends more than one event at the same time**

This constraint establish that in any period k, at most one event $i \in Q_e$ can be programmed; Q_e is the set of events that include the event e and all events that have conflicts with this event e. This constraint is written as:

$$\sum_{i \in Q_e} x_{ijk} \leq 1 \quad j = 1,\ldots,r \quad k = 1,\ldots,45 \quad e = 1,\ldots,n \quad (2)$$

Where i, j and k represent the number of event, room and period respectively.

2. **The room is big enough for all the attending students and satisfies all the features required by the event**

This constraint can be divided in two:

- A) The room is big enough for all the attending students, and
- B) The room satisfies all the features required by the event.

The constraint A can be also stated as follows: For every room j where the event i is programmed, the capacity of the room j (c_j) will be higher or equal to the number of students that attend the event i. That is equivalent to the next relation among the number of participants b_i of the event i :

$$b_i = \sum_{l=1}^{m} d_{li} \quad i = 1,\ldots,n \quad (3)$$

Where the student l attends the event i, d_{li} is 1 if the student l attend the event i.

The constraint A can be stated like:

$$\forall x_{ijk} = 1, \quad b_i \le c_j, \qquad i = 1,...,n \quad j = 1,...,r \quad k = 1,...,45 \tag{4}$$

The constraint B can be stated as follows: For each k period, the room j must satisfy all the features required by every event i programmed in this room. This constraint can be state like:

$$x_{ijk} = 1 \Rightarrow \forall f \; q_{if} \le s_{jf}, \qquad i = 1,...n \quad j = 1,...,r \quad k = 1,...,45 \tag{5}$$

Where q_{if} represents the feature f associated to the event i, and s_{jf} represents the feature f satisfied by the room j.

3. **Only one event is in each room at any timeslot.**

This constraint means that in a period k, any room j can hold at most one event. This constraint can be stated like:

$$\sum_{j=1}^{r} x_{ijk} \le 1 \quad i = 1,...,n \quad k = 1,...,45 \tag{6}$$

Besides, there is another very important hard constraint implicit in the problem's definition:

- All the events must be programmed in some period.

That means that during all the 45 periods, every event i must be programmed exactly once. Therefore this constraint can be written as:

$$\sum_{k=1}^{45} \sum_{j=1}^{r} x_{ijk} = 1 \quad i = 1,...,n \tag{7}$$

Soft constraints

The soft constraints are the following:

1. **Any student should not have a class in the last slot of the day**

Let be a set V that contains the last periods of the days: $V = \{k \mid k \bmod NUMPER = 0\}$. Where k = 1, 2, ... , 45 and *NUMPER* = Number of periods for day (9, in this case). Besides, this constraint can be realized in this way: "some event i should not be programmed in any room j, in any period $k \in V$". Therefore, this constraint can be written as:

$$\forall k \in V \quad \sum_{i=1}^{n} \sum_{j=1}^{r} x_{ijk} = 0 \tag{8}$$

2. **Any student should not have more than two classes consecutively**

That means that, any student l should not have 3 or more events programmed in a row. Let be S_l the set of events of the student l so this constraint is written as:

$$\forall \, v \in V \qquad \sum_{k=v-NUMPER+1}^{v-2} \; \sum_{i \in S_l} \sum_{j=1}^{r} x_{ij(k+1)} x_{ij(k+2)} x_{ij(k+3)} = 0$$

$$l = 1,2,...m \tag{9}$$

3. **Any student should not have a single class on any day.**

This constraint establishes that all the students should have programmed zero or more that one event per day; that means:

$$\forall v \in V \qquad \sum_{k=v-NUMPER+1}^{v-2} \; \sum_{i \in S_l} \sum_{j=1}^{r} x_{ijk} \ne 1 \qquad l = 1,...,m \tag{10}$$

3 Algorithm Description

Simulated Annealing (SA) is one of the metaheuristics used with more success to solve the timetabling problem. The algorithm allows some "wrongs" movements in order to escape of local optimum with the purpose of reach the global optimum. SA has showed that it is a powerful tool to solve many problems of combinatorial optimization.

The most common cooling scheme of SA and used in this paper is a geometric scheme: $T(k+1)=\alpha T(k)$, where k is the temperature number and $0<\alpha<1$. The algorithm's parameters are: the initial temperature T_0, the final temperature T_F , the parameter alpha α and the length of Markov chain L. Also, in SA a feasible initial solution S_0 is required; to find this initial solution for the PATAT's benchmark, two different methods can be used. The first method uses a heuristic that provides a feasible solution to SA and it begins to improve it. The heuristic that finds this initial solution uses the concept of "more constrained event" [8]. The second method uses SA to find the initial feasible solution. In spite of the previous negative results of SA to find feasible solutions in large instances [5], now it is possible to use it in an efficient way and find an initial solution. This is done by restricting the movements and exchanges of events, in such a way that, they do not introduce any new hard restriction to the solution. The procedure is the following: all the events are initialized to the first hour in the first room. Then, a random event is chosen. Its feasible neighborhood is calculated and a random feasible neighbor is chosen. In most of the cases, the procedure reaches a feasible solution and when it is achieved, the procedure continues improving its objective function until the system is frozen. The next implementations of SA were developed:

3.1 First Neighborhood

For the SA implementation a first neighborhood (SA01), was established with the next parameters: The Markov chain length was set to 10000 [8]. A geometric cooling scheme was used ($T_n = \alpha T_{n-1}$). The experimentation was done with different alpha values: 0.70, 0.75, 0.85, 0.90 and 0.95. The initial temperature was obtained by $470m + n$, where m is the number of students and n is the number of events. This SA implementation starts from a feasible solution obtained for other heuristic [8]. This first neighborhood used to generate new solutions is as follows. First, two random periods are selected and after that, two random classrooms are chosen. If an interchange of events is feasible, this interchange is accepted. Otherwise, two news random periods and two new random rooms are selected, and this process goes on.

3.2 Second Neighborhood

This implementation, called SA second neighborhood (SA02), is similar to SA01: First, a random event is chosen, second if it is feasible several interchanges are calculated (over the period and/or over the room or even over other event) taking care that any hard constraint is not violated. Finally, a random change is chosen.

3.3 SA without feasible initial solution

SA without feasible initial solution was label SA03. Initially, any feasible solution is given to the algorithm, all the events are set in the timeslot 1 and the room is set to 1. Using this solution, the algorithm starts working until it reaches a feasible solution. The percentage of not feasible solutions obtained with this approach is less to 2%. The neighborhood using by this implementation is the same as the Second Neighborhood.

4 Analytically tuned parameters of Simulated Annealing

A scheme of tuned parameters published by Sanvicente et. al. [10] was used. This scheme was used successfully in other problems [11] [12]. The T_0 parameter is obtained in function of the maximum possible deterioration of the objective function that can be accepted in a current solution with this temperature, T_f parameter is obtained in function of the minimum possible deterioration of the objective function. The maximum possible deterioration, with the proposed neighborhood is: K*(maximum number of students per event). Where is easy to note that in this case K is equals to eight

T_0 and T_f are calculated with the next formulas:

$$T_0 = \frac{-\Delta Z_{V\max}}{\ln\left(P^A(\Delta Z_{V\max})\right)} \qquad \textbf{(11)}$$

$$T_f = \frac{-\Delta Z_{V\min}}{\ln\left(P^A(\Delta Z_{V\min})\right)} \qquad \textbf{(12)}$$

The Markov chains length L of the metropolis loop (the inner one) are also tuned dynamically. In [10], an analytical method to determine the longitude L_i for i iteration is presented. In this method, L_i is determined establishing the relationship between the cooling function and the Markov chain length. In the first Metropolis cycle the *L-parameter is* 1 and it increases according to a β-parameter, until, in the last cycle of Metropolis, it reaches the L_{max},:

$$L_{max} = \beta^n L_1 \qquad \textbf{(13)}$$

Where

$$n = \frac{\ln L_{\max} - \ln L_1}{\ln \alpha} \qquad \textbf{(14)}$$

$$\beta = \exp\frac{\ln L_{\max} - \ln L_1}{n} \qquad \textbf{(15)}$$

With this analytical method, T_f will be 0; however the stochastic equilibrium is verified here since 0.01. L_{max} takes the same value for all implementations (10000 iterations). Implementations with these tuned parameters are next described: The implementation of the analytical tuned parameters of SA first neighborhood (SA01) was labeled SA04. The implementation of the analytical tuned parameters of SA second neighborhood (SA02) was called SA05

Usually, using the analytical tuned approach is possible to obtain a SA algorithm which has a similar quality of the experimental tuned version of the same algorithm, but the former may save until fifty percent of computational time [11], [12].

Although an interesting saving of time was obtained with the analytical tuned parameters of SA, the quality of its solutions is lightly smaller to those obtained without formulae (11) to (15). The explanation is the following. The experimental method has obtained a very big T_0 value, which was bigger than the T_0 obtained with the analytical method (formulas 11-15); in fact the latter was relatively too small. The later result was obtained because both the acceptance probability and the exploratory capacity of the experimental implementation were very high. To solve this problem, some actions can be taken:

a) The initial temperature T_0 is set equal to that obtained with the experimental method, and its Markov chain length is set equal to 1 (L_1=1), as is established by the analytical method.
b) After each cycle of Metropolis, the temperature is decreased according to the geometrical cooling scheme, and the Markov chain length is increased by one, until is reach the T_0 obtained with the analytical method.
c) Starting from T_0 analytical the length Markov chain is increased according to the β parameter, until it reaches L_{max} and then, it stays constant until arriving to T_f.
This implementation is called, SATUNED and the pseudocode is the following one:

Pseudocode of the implementation SATUNED

```
Begin
x = initial_solution;
BestCost = costIni = f(x)
T=470*num_students+num_events;
T_analytical = DZmax / log
(Pacceptation)
$\beta$ = exp((log($L_{max}$)-log($L_1$))/n);
END_TEMP = 0.01; L=$L_1$; Iter=0;
While (T > END_TEMP)
   While (Iter < L)
      x_new = perturb(x);
      costNew = f(x_new);
      costDif = costNew – costIni;
      r = rand()
      if (costNew <= 0) then
         costIni = costNew;
         x = x_new;
      else
         r = rand()
         if (r < exp(-costDif/T)) then
            costIni = costNew;
            x= x_new;
         End_if
      if (BestCost > costoIni) then
         x* = xl; BestCost = costoIni;
      iter = iter + 1
   End_While
   T = T * ALPHA
   if(T<T_analytical) then L = L + 1
   else if(L<Lmax) then L = $\beta$ * L
End_While
End
```

5 Results

In table 1, and figures 1 and 2 are presented the results of quality of the different implementations of SA proposed in this paper. Graphical results for alpha 0.75and 0.95 used in the geometrical cooling scheme are shown. In most of the cases, the best results were obtained by a SA using the second neighborhood and SA without feasible initial solution or SATUNED (SA02, SA03 and SA06, respectively). Table 5 shows the execution time of the implementations with alpha 0.95; as can be noticed, the faster is the implementation SA05, but, its quality is not the best one. The best time was obtained with the implementations SA02, SA03 and SA06. For simplicity, only some alphas in this table and figures are presented. The results are shown in two categories: quality and time. The quality is measured considering the number of soft constraints violated. The time unit used is seconds. Every instance was run ten times with every alpha and an average is obtained:

Table 1 Quality results with alpha = 0.75

Alpha 0.75	SA01	SA02	SA03	SA05	SA06
s1	1	1.7	1.6	4.5	2.1
s2	10	1.9	3.5	4.3	3.5
s3	1	3.5	2.9	4.6	3.8
s4	1	4	2.6	4.2	5
s5	82	0.8	0.7	0.7	1.2
M1	126	115.5	105	112	103.9
M2	161	110.5	110.8	101.4	101.8
M3	149	153.5	145.3	147.1	141.4
M4	105	98.1	98.2	95.8	99
M5	72	79.9	72	83.8	70.9
h1	*	*	492.8	*	*
h2	*	*	432	*	*

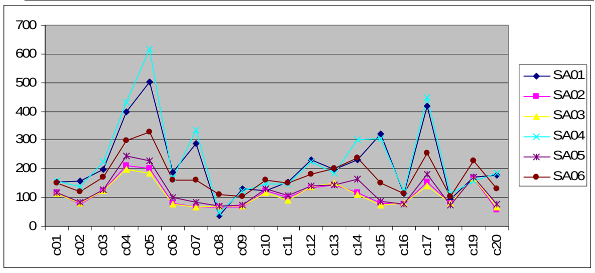
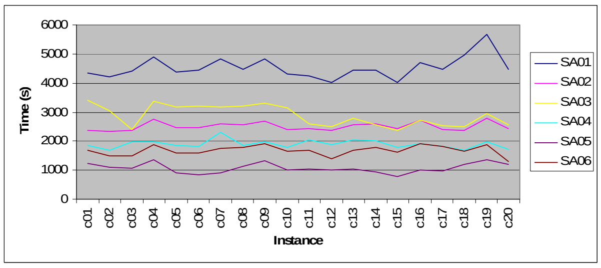


Fig. 1 Quality results using alpha = 0.95, The axe y is the number of soft constraints violate and the axe x is the instance.



* It was not possible to obtain enough feasible solutions to make an average.

Fig. 2 Time results using alpha = 0.95, the axe *y* are the seconds ant the axe *x* the instance

6 Conclusions

In this paper, several implementations of SA for UTT are presented. These implementations are able to find feasible solutions for hard instances. It represents an advance in relation with previous results [5]. The best results were obtained with SA02, SA03 and SA06; the fastest was SA06. SATUNED implementation (SA06) saves around 32% of the execution time wasted by SA02 or saves around 40% of the time used by SA03. Besides SA06 has a similar quality that other implementations. Therefore, SA with the analytical tuned method in the paper had a good performance and is relatively very simple to be implemented.

References:

[1] International Timetabling Competition, URL: http://www.idsia.ch/Files/ttcomp2002/ Consultant date: Wednesday , November 28 of 2007

[2] Cerny, V. Minimization of Continuous Functions by Simulated Annealing. *Research Institute for Theoretical Physics, University of Helsinki*, preprint No. HU-TFT-84-51, 1984.

[3] Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. *Optimization by Simulated Annealing, Science*, Vol 220, Number 4598, pages671-680, (1983).

[4] Kostuch, P.A. (2005). The University Course Timetabling Problem with 3-Phase approach. Practice and Theory of Automated Timetabling V, *Third International Conference on Practice and Theory of Automated Timetabling, Pittsburgh, PA, USA,* August 18-20, 2004. Lecture Notes in Computer Science 3616 Springer 2005, ISBN 3-540-30705-2. pp 251-266

[5] Rossi-Doria, O., Sampels, M., Biratrari, M., Chiarandini, M., Dorigo, M., Gambardella, L. M., Knowles, J., Manfrin, M., Mastrolilli, L., Paetcher, B., Paquete L., Stützle, T. A comparison of the performance of different metaheuristic on the timetabling problem. *Napier University, Université Libre de Bruxelles, Technische Universitaet Darmstadt.* (2002).

[6] Luca Di Gaspero and Andrea Schaerf. Timetabling Competition TTComp 2002: Solver Description, *Conference on Practice and Theory of Automated Timetabling, Pittsburgh, PA, USA,* August 31, 2004.

[7] Socha, K.; Knowles, J.; Sampels, M. "A MAX-MIN Ant System for the University Timetabling Problem". *In Proceedings of the 3rd International Workshop on Ant Algorithms*, ANTS 2002, Lecture Notes in Computer Science, Vol. 2463, Springer, (2002), pp. 1-13.

[8] Bykov, Y. The Description of the Algorithm for International Timetabling Competition. *Timetabling Competition of PATAT*, University of Nottingham, School of Computer Science & IT, Wollaton Road, August, 2004.

[9] Halvard Arntzen, Arne Lokketangen. A local search heuristic for a university timetabling problem, *Timetabling Competition of PATAT* August, 2004.

[10] Sanvicente-Sanchez, Hector y Frausto, Juan. (2004). Method to Establish the Cooling Scheme in Simulated Annealing Like Algorithms. International Conference, Assis, Italy. *ICCSA'2004. LNCS Vol. 3095*. 755-763.

[11] Hector Sanvicente-Sanchez, Juan Frausto-Solís, Froilan Imperial- Valenzuela, Solving SAT Problems with TA Algorithms Using Constant and Dynamic Markov Chains Length, *Algorithmic Applications in Management, Springer Verlag* LNCS, June (2005)

[12] Héctor Sanvicente-Sánchez, Metodología de Paralelización del Ciclo de Temperaturas en Algoritmos Tipo Recocido Simulado, PhD Thesis in Computer Science, *ITESM Campus Cuernavaca*, Octuber (2003).