Solving the Timetabling problem with three heuristics

JUAN FRAUSTO-SOLÍS, FEDERICO ALONSO-PECINA, MONICA LARRE

Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Cuernavaca

{juan.frausto, A01125296 }@itesm.mx

CINHTIA GONZALEZ-SEGURA

Universidad Autónoma de Yucatán

gsegura@uady.mx

JOSÉ LUIS GÓMEZ-RAMOS

Universidad Juárez Autónoma de Tabasco

jose.gomez@dais.ujat.mx

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]:

$$F_i = (x_i - b_i) / (w_i - b_i) \quad (1)$$

Where *i* is the number of the instance ($1 \leq i \leq 20$), *x* is the number of soft constraints violations for the contestant; *b* is the number of soft constraint 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 1, the features of PATAT's instances are presented, and table 2 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 1 PATAT´s Benchmark

Instance	Events	Rooms	Features	Students
C01	400	10	10	200
C02	400	10	10	200
C03	400	10	10	200
C04	400	10	5	300
C05	350	10	10	300
C06	350	10	5	300
C07	350	10	5	350
C08	400	10	5	250
C09	440	11	6	220
C10	400	10	5	200
C11	400	10	6	220
C12	400	10	5	200
C13	400	10	6	250
C14	350	10	5	350
C15	350	10	10	300
C16	440	11	6	220
C17	350	10	10	300
C18	400	10	10	200
C19	400	10	5	300
C20	350	10	5	300

Table 2 Socha's classification

Instance	Events	Rooms	Features	Students
small	100	5	5	80
medium	400	10	5	200
large	400	10	10	400

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 that 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 enchained rules are proposed in this paper. An event E_1 will be higher (lower) constraint level that an event E_2 according with the next Constraint Level Rules (CLR):

- CLR No. 1: If the number of periods available for the event E_1 will be lower (higher) than the number of periods available for E_2, then the event E_1 has a lower (higher) constraint level than E_2. Otherwise, in case of a tie, CLR number 2 is applied.
- CLR No. 2: The event E_1 has higher (lower) constrained level that the event E_2 if E_1 has less (more) classrooms availability than E_2. Otherwise, in case of a tie, the third CLR is applied:
- CLR No. 3: The event E_1 has higher (lower) constraint level that the event E_2 if E_1 has more (less) conflicts with others events Ek ($k \neq 2$). Otherwise, in the case of a tie, CLR No 4 is applied.
- CLR No. 4. The event E_1 will be more constrained that E_2 if E_1 has more students than E_2. In the case of a tie, a lexicographic rule is applied.

Using CLRs the events are sorted from the higher to lower constrained level, chosen in this way the first event E_1 to be scheduled.

The process to make the schedule, once an event E_1 is chosen using CLRs is as follows:

1. Use the first room a_1 that hold the event E_1. 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_1 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 = Φ;
Do
  Cont = 0;
  L = all the events;
  While List_conflicts <> Φ
      E = First element of List_conflicts
      Assign classroom and period to E
      Update the list L
      List_conflicts = List_conflicts – 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)
            List_conflicts= List_conflicts + 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.
- *List_conflicts* 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:

```
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
    end_if
    LTabu = LTabu  ∪ (x, t);
    for every e ∈ LTabu decrease the tenure
        of e
        if expire the tenure of e then
            LTabu = LTabu – (e, t)
        end_if
    end_for
 until cont == 400 || x*==0
End
```

The tenure in the tabu list is a random number b $\in Z^+$ 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 α 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 CLRs 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 news 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 a 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:

```
Begin
 x = initial solution
 BestCost = costIni = f(x)
 T = 470 * num_students + num_events
 END_TEMP = 0.01;
 ALPHA = 0.09;
 Iter=0;
 While (T > END_TEMP)
    While (iter < MAXNUMITER)
       x_new = perturb(x);
       costNew = f(x_new);
       costDif = costNew - costIni;
       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
       End_if
       if (BestCost > costoIni) then
          x* = xl;
          BestCost = costoIni;
        End_if
       iter = iter + 1
     End_While
     T = T * ALPHA
 End_While
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 elemens of the population is generated by the next mutation operator.

13	0	5	2	10	14	1	3	8	4	6	12	11	7	9

Fig 1. Example of a chromosome

13	0	5

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

Fig 2 Chromosome's representation

In the figure 2, every column corresponds to one room and every row to a period to the event to be programmed (the value of the cell).

The genetic operators that were implemented in this paper are the following:

- Partially Mapped Crossover [13]. The algorithm is:

```
Partially mapped crossover (PMX)
   1. Selection of a subset
   2. Establish relations between
      the genes
   3. Change the genes according to
      the relations
   4. Legalize cromosoma
```

- Mutation operator

```
Mutation
   1. Select the alleles for the
      interchange
   2. Interchange the aleles
```

The genetic algorithm implemented is the next:

```
1. Create population P0 starting with a
   seed
2. Repeat
      a. Evaluate P0
      b. Combine P0
    Until stop criterion
3. End
```

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.

Table 3 Instances used by Burke.

Instances	HH	ANT	TS1	TS2	SA	GA
S1	**1**	**1**	4	**1**	**1**	6
S2	**2**	3	3	**2**	10	13
S3	**0**	1	**0**	**0**	1	17
S4	**1**	**1**	**1**	**1**	**1**	14
S5	**0**	**0**	1	2	82	14
M1	146	195	214	195	**126**	414
M2	173	184	231	183	**161**	350
M3	267	248	304	250	**149**	510
M4	169	164.5	138	164	**105**	334
M5	303	219.5	221	121	**72**	383
L1	1166	851.5	994	691	**753**	1305
L2	*	*	939	757	**870**	1342

* We did not found Published results

Table 4 Instances of the PATAT´s

Instances	TS1	TS2	SA	GA
C01	224	220	*203*	436
C02	231	187	*152*	380
C03	320	308	*233*	440
C04	543	*452*	463	853
C05	588	*406*	511	746
C06	388	270	*245*	642
C07	303	*145*	294	469
C08	300	185	*64*	358
C09	241	202	*126*	440
C10	249	200	*169*	438
C11	307	258	*158*	398
C12	365	269	*229*	474
C13	378	308	*217*	513
C14	441	248	*247*	567
C15	347	*209*	310	521
C16	219	203	*137*	324
C17	511	*386*	427	738
C18	239	209	*135*	398
C19	323	391	*147*	573
C20	211	329	*196*	460

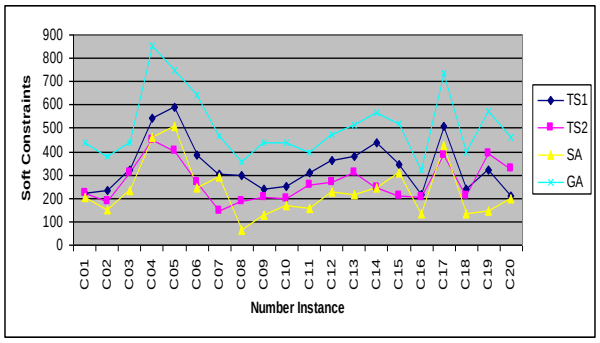

Fig 3 Results of the algorithms in the benchmark of PATAT

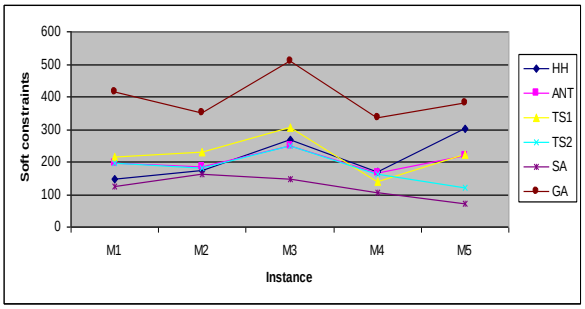

Fig 4 Results of the algorithms in the medium instances

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:

[1] Abramson, D. Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms. *Management Science*, Vol 37, No 1, Jan 1991, pp 98 - 113.

[2] Burke, E.K.; Kendall, G.; Soubeiga, E. A Tabu-Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics* Vol 9, 2003, pp 451-470.

[3] Colorni, A. Dorigo, M. & Maniezzo. A genetic algorithm to solve the timetable problem. Tech. rep. 90-060 revised, Politecnico di Milano, Italy. Submitted to *Computational Optimization and Applications*, 1992

[4] Hertz, A. Tabu search for large scale timetabling problems. *European Journal of Operational Research*, Vol 54, 1991, pp 39-47.

[5] Marte, Michael. Models and Algorithms for School Timetabling – A constraint-Programming Approach., Dissertation thesis, an der Fakult¨at f¨ur Mathematik, Informatik und Statistik der Ludwig-Maximilians-Universit ¨atM¨unchen 2002.

[6] Sara K. Miner, Saleh Elmohamed, Hon W. Yau. Optimizing Timetabling Solutions Using Graph Coloring, *Syracuse University*, NPAC REU program, 1995.

[7] Neufeld GA y Tartar j, Graph coloring conditions for the existence of solutions to the timetable problem. *Communications of ACM*, Vol 17, Issue 8, pp 450-453, 1974

[8] International Timetabling Competition, URL: http://www.idsia.ch/Files/ttcomp2002/ Consultant date: Wednesday , May 31 of 2006

[9] Patat Conferences, URL: http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml, Consultant date: Wednesday , May 31 of 2006.

[10] Schaerf, Andrea. A Survey of Automated Timetabling. *Artificial Intelligence Review*, Vol. 3 No 2. Springer Netherlands. April, 1999 pp 87-127.

[11] 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.

[12] Yoshikawa, M., Kaneko, K., Nomura, Y., & Watanabe, M. A constraint-based approach to high-school timetabling problems: a case study. In Proceedings of the twelfth national conference on Artificial intelligence (AAAI-94), Vol 2, 1994 pp 1111-1116.

[13] Yu, Enzhe; Sunga, Ki-Seok. A genetic algorithm for a university weekly courses timetabling problem. *In International transactions in operational research*, Vol 9 Kangnung National University, Korea. 2002. pp 703-717

[14] Garey, M. R., & Johnson, D.S. *Computers and Intractability – A guide to NP-completeness.* W.H. Freeman and Company, San Francisco. 1979

[15] Brelaz, D. (1979). New Methods to color the Vertices of a Graph *Communications A.C.M*. 7, 494-498.

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