International Timetabling Competition
A Hybrid Approach

Marco Chiarandini
Intellektik, Technische Universität Darmstadt,
Hochschulstr. 10, 64289 Darmstadt, Germany
machud@intellektik.informatik.tu-darmstadt.de

Krzysztof Socha, Mauro Birattari
IRIDIA, Université Libre de Bruxelles, CP 194/6,
Av. Franklin D. Roosevelt 50, 1050 Bruxelles, Belgium
{socha|mbiro}@ulb.ac.be

Olivia Rossi Doria
School of Computing, Napier University,
10 Colinton Road, Edinburgh, EH10 5DT, Scotland
o.rossi-doria@napier.ac.uk

31st March 2003

1 Introduction

This work arises from the research of the Metaheuristics Network[1] on the University Course Timetabling Problem. In a first phase straightforward implementations of different metaheuristics were compared on a common local search framework [4]. In a second phase problem specific knoweldge was exploited and several alternatives for local search and metaheuristics were considered. Many possibilities to combine different components in a hybrid algorithm arose and a large experimental analysis that involved about 1185 configurations identified the most promising algorithm. This report describes this algorithm which was submitted to the International Timetabling Competition and the experimental methodology used to select it among many alternatives.

2 Local Search

The problem is solved in two distinct phases. In a first phase, only hard constraints are considered and solved. When a feasible assignment is reached, that is, no hard constraint is violated anymore, a second phase, that tries to reduce the number of soft constraint violations without breaking any hard constraint, begins.

2.1 The assignment representation

As suggested in the work of Socha [5], an assignment of events to rooms and timeslots can be defined by a rectangular matrix $X_{r\times t}$, such that rows represent rooms and columns represent timeslots, where r and t are respectively the number of rooms and of timeslots in the problem. To each cell of the table is assigned a number $k \in \{-1, 0, \ldots, e\}$ corresponding to an event.

[1] More information on the Metaheuristics Network research programme is available at `http://www.metaheuristics.org`

Specifically if $x_{i,j} = k$, event E_k is assigned to room R_i in timeslot T_j, if $x_{i,j} = -1$, the room R_i in timeslot T_j is free. With this representation we assure that there will be no more than one event assigned to each room in any timeslot, which means one of the hard constraints will always been satisfied.

The room assignment can be done by some heuristic search or by a matching algorithm. The matching algorithm used is the one implemented by Rossi-Doria [3]. For every timeslot it is considered the list of events taking place in it and a preprocessed list of possible rooms to which these events can be assigned. The matching algorithm gives a maximum cardinality matching between these two sets using a deterministic network flow algorithm. In the following, however, we assume that the room assignment is left to the specific local search or metaheuristic unless where explicitly stated.

2.2 Neighborhoods

Different neighborhood schemes are used in the two phases of the search process. For the first phase, in which only hard constraints are considered, we define two neighborhoods:

1. the neighborhood $\mathcal{N}_1$ is defined by the operator that moves a single event to a different timeslot. Formally, given an event E_k assigned to R_i in T_j and a location in the assignment matrix, $X_{l,m} = -1$, the assignment obtained by this operator is the current assignment with $X_{i,j} = -1$ and $X_{l,m} = k$.

2. the neighborhood $\mathcal{N}_2$ is defined by the operator that swaps the timeslots and rooms of two events. Formally, given two events E_h and E_k assigned respectively to R_l in T_m and to R_i in T_j, the assignment obtained by this operator is the current assignment with $X_{l,m} = k$ and $X_{i,j} = h$.

With respect to the soft constraint phase, we define the neighborhoods $\mathcal{N}'_1 \subset \mathcal{N}_1$, $\mathcal{N}'_2 \subset \mathcal{N}_2$, obtained by limiting the moves to only those neighbors that do not introduce violations of the hard constraints and two other neighborhoods:

3. the neighborhood $\mathcal{N}_3$ is defined by the operator that swaps events assigned to two timeslots. Formally, given two timeslots T_j and T_m the assignment obtained by this operator is the current assignment X with the two columns i and m swapped.

4. the neighborhood $\mathcal{N}_4$ is defined by Kempe chain interchanges. If we focus only on events and timeslots, an assignment can be determined by a partition T of a graph G where nodes are events and edges connect events with students in common. A feasible assignment is, a partition of the graph in which no adjacent nodes are in the same class. A *Kempe chain* is the set of nodes that form a connected component in the subgraph G' of G induced by the nodes that belong to two timeslot classes T_i and T_j, $i \neq j$. A Kempe chain *interchange* produces a new feasible assignment by swapping the timeslot class labels assigned to the events (vertices) belonging to some specified Kempe chain. More formally: let K be a Kempe chain in the subgraph G', a Kempe chain interchange produces an assignment by replacing T_i with $(T_i \setminus K) \cup (T_j \cap K)$ and T_j with $(T_j \setminus K) \cup (T_i \cap K)$. We note that if $K = T_i \cup T_j$, the interchange would be simply a relabeling of the timeslots as done by neighborhood $\mathcal{N}_3$. As far as the room assignment is concerned, it is done deterministically after a Kempe chain interchange by the matching algorithm. If this is not possible the current Kempe chain interchange would require to break feasibility and therefore it is not allowed.

2.3 Search strategy and evaluation function

The local search in all cases is a stochastic first improvement local search. Depending on the phase that we are trying to solve, the objective function $f(s)$ is given by the sum of the hard constraint violations ($hcvs$) or the soft constraint violations ($scvs$). The contribution given by each violation

Procedure 1 The over-all procedure.

Input: A problem instance
1: create a population of assignments
2: **for** i=1 to *max_heur* **do**
3: $s_i \leftarrow$ build an assignment with Heuristic i
4: $s_i \leftarrow$ apply Hard Constraint Phase to s_i
5: $s_i \leftarrow$ assignment after applying LS_{SC}^{FAST} in $\mathcal{N}_1'$ to s
6: $s_i \leftarrow$ assignment after applying LS_{SC}^{FAST} in $\mathcal{N}_1' \cup \mathcal{N}_2'$ to s'
7: **end for**
8: $s_{best} \leftarrow$ select the best assignment for I in the population
9: $s_{best} \leftarrow$ apply Soft Constraint Phase to s_{best}
Output: An optimized assignment s_{best} for I

is one and soft constraints are not considered when solving hard constraints. The search strategy uses a list of events randomly ordered and goes through it trying the moves available in the given neighborhood until an improvement is found. Delta evaluation of neighbors for the evaluation function is extensively used for speeding up the search. When all the list is visited without finding any improvement the local search ends and a local optimum has been reached. Side walk moves, *i.e.* moves that do not change the evaluation function value, are performed with different criteria in the hard and soft constraint phase. In the case of hard constraints they are not accepted for $\mathcal{N}_2$ while for $\mathcal{N}_1$ they are accepted with a probability p given by:

$$p = \frac{g(R_j)}{g(R_i) \cdot g(R_j)}$$

where R_i and R_j are respectively the room of origin and of destination of the event considered for the move and g gives the number of events for which the room is suitable. The aim is to favor moves towards rooms that are suitable for many events since this should result in an easier assignment of the other events. In the soft constraint phase, instead, side walk moves are always accepted both in $\mathcal{N}_1'$ and $\mathcal{N}_2'$. In the feasible region, indeed, we have the impression that the fitness landscape presents large plateaus and therefore it appears better to move through there in order to discover new valleys.

3 The overall procedure

The overall procedure for solving the Course Timetabling Problem is outlined schematically in Procedure 1. At the beginning a population of assignments is considered. Each assignment is initialized with a different construction heuristic and a Hard Constraint Phase is applied to solve hard constraints. A fast soft constraint local search is then applied to each individual for having an indication of how promising it would be this assignment when soft constraints are tackled. LS_{SC}^{FAST} is a local search that involves only events which cause soft constraint violation.

As suggested by Rossi-Doria and Paechter [2] to build an assignment we use a sequential algorithm that inserts events into the timetable one at a time. Procedure 2 outlines this process.

A heuristic to chose which event to insert in the timeslot next and a heuristic to assign a room and a timeslot to the chosen event are needed. We considered 6 heuristics H to choose the event and 11 heuristics A to assign room and timeslot. Heuristics for H are the following:

- event with maximum number of correlated events (events that cannot stay in the same timeslot because they share students)
- event with the maximum number of correlated events weighted by the number of students shared
- among events with maximum number of students, one with maximum number of correlated events

Procedure 2 Construction procedure.

Input: A problem instance

1: **while** ∃ event unscheduled **do**
2: let S^E be the set of all the unscheduled events
3: choose an event E in S^E according to heuristic H
4: let S^{RT} be the set of all possible room/timeslot assignments for E causing minimal hard constraint violations
5: let S be the subset of the assignments in S^{RT} causing minimal soft constraint violations
6: choose an assignment for E in S according to heuristic A
7: **end while**

Output: An initialized assignment

- among events with minimum number of possible rooms, one with maximum number of correlated events
- among events with maximum number of required features, one with maximum number of correlated events
- among events with rooms suitable for most events, one with maximum number of correlated events

While, heuristics for A are:

- label order
- random order
- 3 pre-established timeslot order, keeping adjacent timeslots apart and rooms in label order
- smallest possible room, most parallel classes in a timeslot
- least used room and timeslot in label order
- last day, last timeslot, room suitable for least events
- room suitable for least events and last timeslot
- room suitable for least events and timeslot in label order
- among earliest timeslot during the day, one with most parallel classes, rooms in label order

All 66 possible combinations are considered since in a preliminary experimental analysis no evidence arose to prefer one among the other. Yet it appeared that different instances required different construction heuristics. A restriction to a group of promising ones is however one possible future development.

In the initial population we included also an assignment built assigning randomly a timeslot to each event according to an uniform distribution, and applying the matching algorithm for choosing rooms. If the matching algorithm cannot find a room for all the events assigned to a specific timeslot, the unplaced events are rescheduled into another timeslot and into a free room trying to minimize the hard constraint conflicts that their insertion would create.

To sum up, 67 possible ways to generate an initial assignment (max_heur) are considered in our approach.

Hard Constraint Phase

Procedure 3 outlines the hard constraint phase. First, the local search (LS_{HC}) is applied iteratively, then if after 10 local search iterations feasibility has not yet been reached, tabu search (TS_{HC}) is used to escape local optima and continue further the search. Tabu search is not applied since the beginning because from experimental observations we noticed that it would require much more time to find a feasible assignment. The fast descent provided by local search, instead, helps to get very close to a feasible assignment, then easily reached by tabu search. We remind also that local search can accept side walk moves and, therefore, a re-run can provide improvements.

Procedure 3 The Hard Constraint Phase.

Input: A problem instance I, an initialized assignment s

1: $loops \leftarrow 0$
2: **while** $hcvs > 0$ and time limit not reached **do**
3: $s \leftarrow$ assignment after applying LS_{HC} in $\mathcal{N}_1 \cup \mathcal{N}_2$ to s
4: **if** $hcvs > 0$ and $loops > 10$ **then**
5: $s \leftarrow$ assignment after applying TS_{HC} in $\mathcal{N}_1$ to s
6: **end if**
7: $loops \leftarrow loops + 1$
8: **end while**

Output: A feasible assignment s for I

Procedure 4 The Soft Constraint Phase.

Input: A problem instance I, a feasible assignment s

1: $s' \leftarrow s$
2: **repeat**
3: $s \leftarrow s'$
4: $s' \leftarrow$ assignment after applying LS_{SC} in $\mathcal{N}'_1$ to s
5: $s' \leftarrow$ assignment after applying LS_{SC} in $\mathcal{N}'_1 \cup \mathcal{N}'_2$ to s'
6: $s' \leftarrow$ assignment after applying LS^{MA}_{SC} in $\mathcal{N}'_1$ to s'
7: $s' \leftarrow$ assignment after applying LS^{MA}_{SC} in $\mathcal{N}'_1$ and $\mathcal{N}'_2$ to s'
8: $s' \leftarrow$ assignment after applying LS_{SC} in $\mathcal{N}'_3$ to s'
9: $s' \leftarrow$ assignment after applying LS_{SC} in $\mathcal{N}'_4$ to s'
10: **until** $f(s') = f(s)$ or time limit reached
11: $s_{best} \leftarrow$ best assignment found after applying SA^{MA}_{SC} in $\mathcal{N}_1$ and $\mathcal{N}_2$ to s

Output: An optimized assignment s_{best} for I

Local Search in $\mathcal{N}_1 \cup \mathcal{N}_2$ goes through the list of randomly ordered events and for each event it tries a move in a place chosen at random in the matrix X. If the destination place is occupied by another event then the move is a move in $\mathcal{N}_2$ otherwise if the place is free the move is in $\mathcal{N}_1$.

Tabu Search. Tabu Search uses a best improvement strategy on the $\mathcal{N}_1$ neighborhood. In order to reduce the size of the neighborhood, only events which are involved in constraint violations are considered. We forbid a move if it tries to place an event in a timeslot to which it was assigned less than tl steps before. The tabu length tl is set equal to $ran(10) + \delta \cdot n_c$ where $ran(10)$ is a random number in $[0, 10]$, n_c is the number of events in the current assignment involved in at least one hard constraint violation and δ is a parameter to be decided. The aspiration criterion accepts a tabu move if it improves the best known assignment. The procedure ends when the best assignment is not improved for a given number of steps. We fixed this number to $0.4 \cdot t \cdot e \cdot r$ and δ to 0.6

Soft Constraint Phase

The Soft Constraint Phase is sketched in Procedure 4. It consists of two parts. First several local search operators (LS_{SC}) on different neighborhoods are applied until no further improvements can be found, then Simulated Annealing (SA^{MA}_{SC}) is applied until the end of the time available. The reason behind this is that usually SA gets good performance when long computational time are allowed. But the SA acceptance criterion is mainly useful at the end of the search for getting out from local optima. Therefore, trying to start SA from a local optimum should produce competitive results as it was shown by experimental observations.

The soft constraint local search of lines 4 and 5 uses the same search strategy as described for the hard constraint phase. The local search of lines 6 and 7, instead, does something different. It uses the matching algorithm (LS^{MA}_{SC}) to insert events in a timeslot. This introduces at each step larger modifications in the assignment matrix which are useful for a diversification of the search.

Furthermore, the use of the matching algorithm increases the possibilities to find a place for an event in a specific timeslot. LS_{SC}^{MA} on $\mathcal{N}_1$ simply scans the list of events trying to insert them into another timeslot. Running the matching algorithm, which is computationally expensive, is avoided when the timeslot has already a number of events assigned equal to the number of rooms. LS_{SC}^{MA} in $\mathcal{N}_1$ and $\mathcal{N}_2$ tries first for each event all the possible swaps with the events in a destination timeslot, then, if no feasible swap is found, it tries a simple insertion in the timeslot. Side moves are always accepted.

The procedures LS_{SC} in $\mathcal{N}_3$ and LS_{SC} in $\mathcal{N}_4$ work on the timeslots and all possible pairs of timeslots are scanned in random order. In $\mathcal{N}_3$ for each pair of timeslots all the possible Kempe chains are considered. Side walk moves are not accepted in this case due to the higher computational cost of performing Kempe chains; the aim, suggested by preliminary experiments, is, indeed, to reach the best possible assignment to pass to Simulated Annealing in the shortest time.

Simulated Annealing. In Simulated Annealing a move is accepted according to the following probability distribution, dependent on the virtual temperature T:

$$p_{accept}(T, s, s') = \begin{cases} 1 & \text{if} \quad f(s') \leq f(s) \\ e^{-\frac{(f(s')-f(s))}{T}} & \text{otherwise} \end{cases}$$

where s is the current assignment and s' is the neighbor assignment. The temperature parameter T, which controls the acceptance probability, is allowed to vary over the course of the search process.

We describe our implementation distinguishing the following components: neighborhood exploration strategy, initial temperature, temperature length and cooling schedule. A specific stopping criterion is not needed since we use the time limit imposed by the competition.

Neighborhood examination strategy: Events are randomly ordered in a list and in turn examined. For each event a timeslot is randomly chosen. In a first attempt the event is assigned to the timeslot and rooms are reassigned for all the events in the timeslot by means of the matching algorithm. If no feasible assignment of rooms can be found, the move is discarded, otherwise it is accepted according to the SA criterion. If the move is not accepted all the possible swaps with the events in the timeslot are tried. The matching algorithm is applied at each attempt and only moves that lead to a feasible assignment are considered.

Initial Temperature For determining the initial temperature a sample in $\mathcal{N}_1 \cup \mathcal{N}_2$ of 100 neighbors of the best assignment found so far is considered. The initial temperature is then determined by the average value of the variation in the evaluation function multiplied by a given factor, h ($h > 0$).

Temperature update We used a non monotonic temperature schedule realized by the interaction of two strategies: a standard geometric cooling and a temperature re-heating. The standard geometric cooling computes the temperature, T_{n+1} at step $n + 1$ by multiplying the temperature in iteration n, T_n, with a constant factor α (*cooling rate*):

$$T_{n+1} = \alpha \times T_n, \qquad 0 < \alpha < 1$$

When the search seems stagnating, the temperature is re-heated by adding to the current temperature the initial value. This is done when no improvement is found for a number of steps given by $reheat \times TL$ where TL is the temperature length and $reheat$ is a parameter to tune. The assignment that we consider for testing improvements is the best since the last reheating occurred.

Temperature length The number of iterations at each temperature is kept proportional to the size of the neighborhood as suggested by the majority of Simulated Annealing implementations. Hence, $TL = q \cdot e \cdot r \cdot t$, where q is a parameter to tune.

For selecting the best combination of parameters α, $reheat$, q, h we used the race described in Section 4. The outcome is the following: $h = 0.15$, $\alpha = 0.95$, $q = 10$, $reheat = 5$.

4 The race

The algorithm described above and the value of its parameters were selected among different candidates through a tuning procedure. Six different algorithms where considered: Ant Colony Optimization, $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System, Simulated Annealing, Tabu Search, Iterated Local Search and Evolutionary Algorithms. For each, different values of the parameters where tested giving raise to a total of 1185 candidates.

The procedure adopted for the selection is related to the *F-RACE* method [1]. In the *racing approach*, the initial set of candidates is sequentially evaluated on the available instance. During the race, poor candidates are discarded as soon as statistically sufficient evidence is gathered against them. The elimination of inferior candidates speeds up the procedure and allows a more thorough evaluation of the promising ones.

For the selection of the best configuration for the Timetabling Competition, a slightly modified version of the racing approach was adopted for matching the rules and characteristics of the competition. In the version we adopted, the elementary experimental unit is the *run*. A run consists in testing all surviving candidates on the available instances. At the end of each run a decision is made on which candidate should be discarded. Contrary to F-RACE, which is a completely automatic procedure where decisions are solely based on the outcome of the *Friedman two-way analysis of variance by ranks*, the method adopted in this work is of a more interactive nature and various statistical tests were used here only for supporting a human decision maker.

The first phase of the race (2 runs) was conducted on the first 10 instances made available by the organizers of the competition. When the second set of instances became available, the number of surviving candidates had already dropped to one half of the initial one. From then on, each run of the race consisted in running once each surviving candidate on each of the 20 available instances.

The whole procedure took about one month of alternating phases of computation, analysis, and human decision, and could rely on about 30 PC running Linux, with CPU speed in the range between 300 and 1800 MHz.

5 Results submitted

In Table 1 per each instance we report the results submitted to the competition. Values represent the best solution quality found on about 90 runs per instance.

Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Solution	57	31	61	112	86	3	5	4	16	54	38	100	71	25	14	11	69	24	40	0

Table 1: The quality of the solutions submitted to the International Competition.

6 Conclusions

We presented a hybrid algorithm for solving the Course Timetabling Problem. This algorithm is the outcome of a research in which several local search and metaheuristic approaches were tried, among them Ant Colony Optimization, $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System, Simulated Annealing, Tabu Search, Iterated Local Search, Iterated Greedy Search, and Evolutionary Algorithms. It has been developed assembling together different elements which showed promising performance and the experimental analysis confirmed that improvements can arise from hybridization. Nevertheless, we believe that further endeavors, spent to understand the components that play a fundamental role in the search, could reveal further improvements.

Acknowledgments The authors want to thanks Dr. Thomas Stützle for discussions and suggestions during the development of the algorithm. This work was supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106 and by a European Community Marie Curie Fellowship, contract HPMF-CT-2001. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

[1] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W.B. Langdon et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. Morgan Kaufmann, 2002.

[2] O. Rossi-Doria. An hyperheuristic approach to course timetabling problem using an evolutionary algorithm. submitted to MISTA The 1st Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2003).

[3] O. Rossi-Doria, Ben Paechter, C. Blum, K. Socha, and M. Samples. A local search for the timetabling problem. In *Proceedings of PATAT 2002: The 4th international conference on the Practice And Theory of Automated Timetabling*, pages 115–119, Gent, Belgium, August 2002.

[4] O. Rossi-Doria, M. Samples, M. Birattari, M. Chiarandini, J. Knowles, M. Manfrin, M. Mastrolilli, L. Paquete, B. Paechter, and T. Stützle. A comparison of the performance of different metaheuristics on the timetabling problem. In *Proceedings of PATAT 2002: The 4th international conference on the Practice And Theory of Automated Timetabling*, pages 115–119, Gent, Belgium, August 2002.

[5] K. Socha. The Influence of Run-Time Limits on Choosing Ant System Parameters. In Cantu-Paz et al., editor, *Proceedings of GECCO 2003 – Genetic and Evolutionary Computation Conference*, Lecture Notes in Computer Science. Springer, Berlin, Germany, Jul 2003.