



A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem

O. ROSSI-DORIA, M. SAMPELS, M. CHIARANDINI,
J. KNOWLES, M. MANFRIN, M. MASTROLILLI,
L. PAQUETE and B. PAECHTER

Technical Report No.

TR/IRIDIA/2002-17
July 2002

Presented at the 4th International Conference for the Practice and Theory
of Automated Timetabling (PATAT 2002)

A comparison of the performance of different metaheuristics on the timetabling problem

Olivia Rossi-Doria¹, Michael Sampels², Marco Chiarandini³, Joshua Knowles³,
Max Manfrin³, Monaldo Mastrolilli⁴, Luis Paquete³, Ben Paechter¹

¹ School of Computing, Napier University,
10 Colinton Road, Edinburgh, EH10 5DT, Scotland
{o.rossi-doria, B.Paechter}@napier.ac.uk

² IRIDIA, Université Libre de Bruxelles, CP 194/6,
Av. Franklin D. Roosevelt 50, 1050 Bruxelles, Belgium
{msampels, jknowles, mmanfrin}@ulb.ac.be

³ Intellektik, Technische Universitaet Darmstadt,
Alexanderstr. 10, 64283 Darmstadt, Germany
{machud, lpaquete}@intellektik.informatik.tu-darmstadt.de

⁴ IDSIA, Galleria 2, 6928 Manno, Switzerland
monaldo@idsia.ch

1 Introduction

This work is part of the Metaheuristic Network⁵, a European Commission project undertaken jointly by five European institutes, whose aim is to empirically compare and analyse the performance of various metaheuristics on different combinatorial optimization problems including timetabling.

We attempt here an unbiased comparison of the performance of basic versions of different metaheuristics on a timetabling problem using a common search landscape for a fair and meaningful analysis. The timetabling problem considered is a reduction of a typical university timetabling problem. It is described in [1], where a direct solution representation, a neighborhood structure and a local search are also presented, and provide a common basis for all the metaheuristic implementations proposed here. More precisely, we used the local search in an evolutionary algorithm, an ant colony optimization algorithm, and an iterated local search. A simulated annealing, and a tabu search were restricted to the same neighborhood structure. All the algorithms use the same direct representation and are implemented in their basic components in a straightforward manner. The stress here is in the comparison of the different methods under a common framework, rather than in high performance. More freedom in the use of more efficient representations and more heuristic information may give different results. In the following we give a brief description of the general features

⁵ <http://www.metaheuristics.net/>

of each metaheuristic under consideration and of their implementations for the problem.

2 Evolutionary Computation

Evolutionary Algorithms (EA) are based on a computational model of the mechanisms of natural evolution. They operate on a population of potential solutions and comprise three major stages: selection, reproduction and replacement. EA's can use local search to enhance their performance and are then called memetic algorithms.

A simple implementation of a memetic algorithm for the timetabling problem is presented in the following: An initial population of candidate solutions is constructed according to a random distribution and local search is applied to all its members. At each generation only one couple of parent individuals reproduces in a steady-state evolution process. Each parent is selected with tournament selection, where a number of individuals is chosen randomly from the population and the best one wins the tournament. The reproductive operators crossover and mutation are applied to the selected parents yielding a new child individual. We define a uniform crossover on the given direct representation: each assignment of timeslot to event is copied with uniform distribution from one or the other parent, and the room assignment are taken care of by the matching algorithm described in [1]. Mutation is just a random move in an extension of the neighbourhood used by the local search. Each new child individual is improved by means of the local search, then it replaces the worst member of the population.

3 Ant Colony Optimization

Ant Colony Optimization (ACO) algorithms take inspiration from the foraging behavior of real ants. The basic ingredient of ACO is the use of a probabilistic solution construction mechanism based on stigmergy. The algorithm presented here is the first implementation of an ACO to a timetabling problem and follows the ACS version of ACO.

The basic principle of an ACS for tackling the timetabling problem is outlined in the following: At each iteration of the algorithm, each of m ants constructs, event by event, a complete assignment of the events to the timeslots. To make a single assignment of an event to a timeslot, an ant takes the next event from a pre-ordered list, and probabilistically chooses a timeslot for it, guided by two types of information: (1) heuristic information, which is an evaluation of the constraint violations caused by making the assignment, given the assignments already made, and (2) stigmergic information in the form of a 'pheromone' level, which is an estimate of the utility of making the assignment, as judged by previous iterations of the algorithm. The stigmergic information is represented by a matrix of 'pheromone' values $\tau : E \times T \rightarrow \mathbf{R}_{>0}$, where E is the set of events and T is the set of timeslots. These values are initialized to a parameter τ_0 , and then updated by local and global rules; generally, an event-timeslot pair which has

been part of good solutions in the past will have a high pheromone value, and consequently it will have a higher chance of being chosen again in the future. At the end of the iterative construction, an event-timeslot assignment is converted into a candidate solution (timetable) using the matching algorithm. This candidate solution is further improved by the local search routine. After all m ants have generated their candidate solution, a global update on the pheromone values is performed using the best solution found since the beginning. The whole construction phase is repeated, until the time limit is reached.

4 Iterated Local Search

Iterated Local Search (ILS) is based on the simple yet powerful idea of improving a local search procedure by providing new starting solutions which are obtained from perturbations of the current solution, leading to far better solutions than if using repeated random trials of the local search. The local search is applied to the perturbed solution and a locally optimal solution is reached. If it passes an acceptance criterion, it becomes the new current solution; otherwise, one returns to the previous current solution. The perturbation must be sufficiently strong to allow the local search to explore new solutions, but also weak enough so that not all the good information gained in the previous search is lost.

We considered different kinds of perturbations and acceptance criteria. Tuning for the best configuration of different parameters has been done automatically with the racing algorithm proposed by Birattari et al. [2]. This method is based on the statistical test of Friedman and tries to eliminate configurations that appear to be significantly worse than the current champion configuration. We propose for comparison with the other metaheuristics the best performing ILS configuration. It uses as perturbation a sequence of a certain number n of random moves, where a move consists of a choice of a different timeslot for a randomly chosen event. The moves considered are a subset of the neighbourhood used by the local search. The acceptance criterion is a simulated annealing type acceptance criterion.

5 Simulated Annealing

Simulated Annealing (SA) is a variant of local search that allows some uphill moves to avoid becoming trapped in a local optimum. This is done as follows: an improving local search move is always accepted while a worsening local search move is accepted according to a probability which depends on the relative deterioration in the evaluation function value, such that the worse a move is, the less likely it is to accept it. Formally a move is accepted according to the following probability distribution, known as the Metropolis distribution:

$$p_{accept}(T, s, s') = \begin{cases} 1 & \text{if } f(s') < f(s) \\ \exp\left(-\frac{f(s')-f(s)}{T}\right) & \text{otherwise} \end{cases}$$

where s is the current solution, s' is a neighbour solution and $f(s)$ is the evaluation function. The temperature parameter T , which controls the acceptance probability, is allowed to vary over the course of the search process.

Different variants and different parameters for neighbourhood exploration strategy, initial temperature, temperature length and cooling schedule have been object of an automated tuning for finding the best configuration as in Sect. 4. We describe in the following the resulting final implementation: The neighbourhood is explored in a random way. The procedure is divided in two separate phases: a first loop for solving feasibility and a second loop for improving soft constraints, when going back from a feasible region to an infeasible one is not allowed. The initial temperature is obtained by multiplying a given factor by the average of the variation in the evaluation function produced by 100 neighbours of a randomly generated initial solution. 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} in iteration $n + 1$ by multiplying the temperature in iteration n , T_n , with a constant factor α (*cooling rate*):

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

A sort of adaptation to the behavior of the search process is included by re-heating the temperature when the search seems to be stagnating. Indeed, according to an *acceptance ratio* given by the number of moves rejected on the number of moves tested, the temperature is increased to a value equal to the initial one when the ratio exceed a given limit. This inspection is done every fixed number of iterations, in our case three times the temperature length. The number of iterations at each temperature is kept proportional to the size of the neighborhood.

6 Tabu Search

Tabu search (TS) is a local search metaheuristic which relies on specialized memory structures to avoid entrapment in local optima and achieve an effective balance of intensification and diversification. More precisely, TS allows the search to explore solutions that do not decrease the objective function value only in those cases where these solutions are not forbidden. This is usually obtained by keeping track of the last solutions in term of the move used to transform one solution to the next. When a move is performed it is considered *tabu* for the next T iterations, where T is the tabu status length. A solution is forbidden if it is obtained by applying a tabu move to the current solution.

According to the given neighbourhood, a move for the timetabling problem is defined by moving one event or by swapping two events. We forbid a move if at least one of the event involved has been moved less than T steps before. The tabu status length T is set to the number of events divided by a suitable constant k (we set $k = 100$). With the aim of decreasing the probability of generating cycles, we consider a variable neighbourhood set: every move is a neighbour with

probability 0.1. Moreover, in order to explore the search space in a more efficient way, tabu search is usually augmented with some aspiration criteria. The latter are used to accept a move even if it has been marked tabu. We perform a tabu move if it improves the best known solution. To summarize, the proposed TS considers a variable set of neighbours and performs the best move that improves the best known solution, otherwise performs the best non tabu move chosen among those belonging to the current variable neighbourhood set.

7 Results

The presentation and full paper will include the results of the study and insights into why certain metaheuristics perform better with problem instances with different characteristics. The analysis carried out should give us some guidelines for the design of more sophisticated, and possibly hybrid, algorithms.

Acknowledgments: This work was supported by the “Metaheuristics Network”, a Research Training Network funded by the Improving Human Potential program of the CEC, grant HPRN-CT-1999-00106. 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. O. Rossi-Doria, C. Blum, J. Knowles, M. Sampels, K. Socha, B. Paechter, *A local search for the timetabling problem*, extended abstract submitted to PATAT 2002.
2. M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, *A racing algorithm for configuring metaheuristics*, Technical report, Intellektik, Technische Universität Darmstadt, Darmstadt, Germany, 2002.