A $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System for the University Course Timetabling Problem

Krzysztof Socha, Joshua Knowles, and Michael Sampels

IRIDIA, Université Libre de Bruxelles, CP 194/6,
Av. Franklin D. Roosevelt 50, 1050 Bruxelles, Belgium
{ksocha|jknowles|msampels}@ulb.ac.be
http://iridia.ulb.ac.be

Abstract. We consider a simplification of a typical university course timetabling problem involving three types of hard and three types of soft constraints. A $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System, which makes use of a separate local search routine, is proposed for tackling this problem. We devise an appropriate construction graph and pheromone matrix representation after considering alternatives. The resulting algorithm is tested over a set of eleven instances from three classes of the problem. The results demonstrate that the ant system is able to construct significantly better timetables than an algorithm that iterates the local search procedure from random starting solutions.

1   Introduction

Course timetabling problems are periodically faced by virtually every school, college and university in the world. In a basic problem, a set of times must be assigned to a set of events (e.g., classes, lectures, tutorials, etc.) in such a way that all of the students can attend all of their respective events. Some pairs of events are edge-constrained (e.g., some students must attend both events), so that they must be scheduled at different times, and this yields what is essentially a form of vertex colouring problem. In addition, real timetables must usually satisfy a large and diverse array of supplementary constraints which are difficult to describe in a generic manner. However, the general university course timetabling problem (UCTP) is known to be NP-hard, as are many of the subproblems associated with additional constraints [5, 9, 22]. Of course, the difficulty of any particular instance of the UCTP depends on many factors and, while little is known about how to estimate difficulty, it seems that the assignment of rooms makes the problem significantly harder than vertex colouring, in general.

Current methods for tackling timetabling problems include evolutionary algorithms [6], simulated annealing [13] and tabu search [14]. Many problem-specific heuristics also exist for timetabling and its associated sub-problems. These have been used within evolutionary methods and other generic search methods, either as 'hyper-heuristics' [1, 7], or to repair or decode indirect solution representations [16]. Local search has also been used successfully within a memetic algorithm to do real-world exam timetabling [3]. Although several ant colony

optimization (ACO) algorithms [2, 11, 21] have been previously proposed for other constraint satisfaction problems [18], including vertex-coloring [8], a full timetabling problem has not been tackled before using ACO.

The work presented here arises out of the Metaheuristics Network[1] (MN) – a European Commission project undertaken jointly by five European institutes – which seeks to compare metaheuristics on different combinatorial optimization problems. In the current phase of the four-year project, a university course timetabling problem is being considered. In this phase, five metaheuristics, including ACO, will be evaluated and compared on instances from three UCTP classes. As a potential entry for evaluation by the MN, we developed a $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MMAS}$) [21] algorithm for the UCTP. In this paper we describe this algorithm, discussing the selection of an appropriate construction graph and pheromone representation, the local search, the heuristic information, and other factors. We then report on experiments in which the performance of the $\mathcal{MMAS}$ algorithm is evaluated with respect to using the local search alone, in a random-restart algorithm.

The remainder of this paper is organized as follows: Section 2 defines the particular timetabling problem considered and specifies how instances of different classes were generated. A brief description of the local search is also included. Section 3 describes the design of the $\mathcal{MMAS}$ algorithm, focusing on the key aspects of representation and heuristics. Section 4 reports the experimental method and computational results of the comparison with the random restart local search. In Sect. 5, conclusions are drawn.

2 The University Course Timetabling Problem

The timetabling problem considered by the MN is similar to one initially presented by Paechter in [15]. It is a reduction of a typical university course timetabling problem [6, 16]. It consists of a set of n events E to be scheduled in a set of timeslots $T = \{t_1, \ldots, t_k\}$ ($k = 45$, 5 days of 9 hours each), a set of rooms R in which events can take place, a set of students S who attend the events, and a set of features F satisfied by rooms and required by events. Each student attends a number of events and each room has a maximum capacity. A feasible timetable is one in which all events have been assigned a timeslot and a room so that the following hard constraints are satisfied:

– no student attends more than one event at the same time;
– the room is big enough for all the attending students and satisfies all the features required by the event;
– only one event is in each room at any timeslot.

In addition, a feasible candidate timetable is penalized equally for each occurrence of the following soft constraint violations:

– a student has a class in the last slot of the day;

– a student has more than two classes in a row;
– a student has exactly one class on a day.

Feasible solutions are always considered to be superior to infeasible solutions, independently of the numbers of soft constraint violations. In fact, in any comparison, all infeasible solutions are to be considered equally worthless. The objective is to minimize the number of soft constraint violations (#scv) in a feasible solution.

2.1 Problem Instances

Instances of the UCTP are constructed using a generator written by Paechter[2]. The generator makes instances for which a perfect solution exists, that is, a timetable having no hard or soft constraint violations. The generator is called with eight command line parameters that allow various aspects of the instance to be specified, plus a random seed. For the comparison being carried out by the MN, three classes of instance have been chosen, reflecting realistic timetabling problems of varying sizes. These classes are defined by the values of the input parameters to the generator, and different instances of the class can be generated by changing the random seed value. The parameter values defining the classes are given in Tab. 1.

Table 1. Parameter values for the three UCTP classes.

Class	small	medium	large
Num_events	100	400	400
Num_rooms	5	10	10
Num_features	5	5	10
Approx_features_per_room	3	3	5
Percent_feature_use	70	80	90
Num_students	80	200	400
Max_events_per_student	20	20	20
Max_students_per_event	20	50	100

For each class of problem, a time limit for producing a timetable has been determined. The time limits for the problem classes `small`, `medium`, and `large` are respectively 90, 900, and 9000 seconds. These limits were derived experimentally.

2.2 Solution Representation and Local Search

To make comparison and evaluation meaningful, all metaheuristics developed for the MN project, including the algorithm described here, employ the same

---

[2] `http://www.dcs.napier.ac.uk/~benp` .

solution representation, neighborhood structure, and local search routine (where applicable), as described fully in [19].

A solution vector is an element of T^E and represents an assignment of events to timeslots. The assignment of each event-timeslot pair to a room is not under the direct influence of the metaheuristics (or local search routine). Instead, the room assignment is carried out using a deterministic network flow algorithm. The local search routine (LS) is a first-improvement search based on two move operators. The first operator moves a single event to a different timeslot. The second swaps the timeslots of two events. The LS is deterministic and ends when a true local optimum is reached. The LS makes extensive use of delta evaluation of solutions so that many neighboring timetables can be considered in a short time.

3 Design of an $\mathcal{MMAS}$ for Timetabling

Given the constraints on the representation discussed in the last section, we can now consider the choices open to us, in designing an effective $\mathcal{MMAS}$ for the UCTP. We must first decide how to transform the assignment problem (assigning events to timeslots) into an optimal path problem which the ants can tackle. To do this we must select an appropriate *construction graph* for the ants to follow. We must then decide on an appropriate pheromone matrix and heuristic information to influence the paths the ants will take through the graph.

3.1 Construction Graph

One of the cardinal elements of the ACO metaheuristic is the mapping of the problem onto a *construction graph* [10, 12], so that a path through the graph represents a solution to the problem. In our formulation of the UCTP we are required to assign each of $|E|$ events to one of $|T|$ timeslots. In the most direct representation the construction graph is given by $E \times T$; given this graph we can then decide whether the ants move along a list of the timeslots, and choose events to be placed in them, or move along a list of the events and place them in the timeslots. Fig. 1 and Fig. 2 depict, respectively, these construction graphs.

As shown in Fig. 1, the first construction graph must use a set of virtual timeslots $T' = \{t'_1, \ldots, t'_{|E|}\}$, because exactly $|E|$ assignments must be made in the construction of a timetable, but in general $|T| \ll |E|$. Each of the virtual timeslots maps to one of the actual timeslots. To use this representation then, requires us to define an injection $\iota : T' \to T$, designating how the virtual timeslots relate to the actual ones. One could use for example the injection $\iota : t'_g \mapsto t_h$ with $h = \left\lceil \frac{g \cdot |T|}{|E|} \right\rceil$. In this way, the timetable would be constructed sequentially through the week. However, for certain problems, giving equal numbers of events to each timeslot may be a long way from optimal. Other injection functions are also possible but may contain similar implicit biases.

The simpler representation (Fig. 2), where ants walk along a list of events, choosing a timeslot for each, does not require the additional complication of
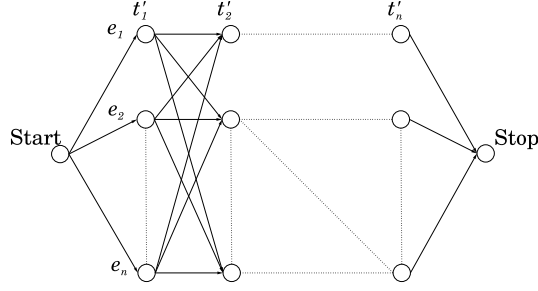
Fig. 1. Each ant follows a list of *virtual* timeslots, and for each such timeslot $t' \in T'$, it chooses an event $e \in E$ to be placed in this timeslot. At each step an ant can choose any possible transition
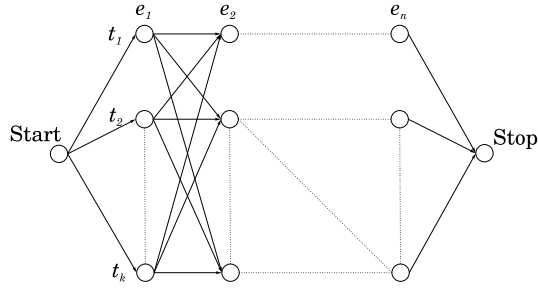

Fig. 2. Each ant follows a list of events, and for each event $e \in E$, an ant chooses a timeslot $t \in T$. Each event has to be put exactly once into a timeslot, and there may be more than one event in a timeslot, so at each step an ant can choose any possible transition

using virtual timeslots and does not seem to have any obvious disadvantages. In fact, it allows us the opportunity of using a heuristically ordered list of events. By carrying out some pre-calculation we should be able to order the events so that the most 'difficult' events are placed into the timetable first, when there are still many timeslots with few or no occupied rooms. For these reasons we choose to use this representation.

As the ants traverse our chosen construction graph, they construct partial assignments $A_i : E_i \rightarrow T$ for $i = 0, \ldots, |E|$, where $E_i = \{e_1, \ldots, e_i\}$. An ant starts with the empty assignment $A_0 = \emptyset$. After the construction of A_{i-1}, the assignment A_i is built probabilistically as $A_i = A_{i-1} \cup \{(e_i, t)\}$. The timeslot t is chosen randomly out of T according to probabilities $p_{e_i, t}$ that depend on the pheromone matrix $\tau(A_{i-1}) \in [\tau_{min}, \tau_{max}]^{E \times T}$ ($\tau_{min}, \tau_{max} \in \mathbf{R}$) and any heuristic information $\eta(A_{i-1})$ given by:

$$p_{e_i, t}(\tau(A_{i-1}), \eta(A_{i-1})) = \frac{(\tau_{(e_i, t)}(A_{i-1}))^\alpha \cdot (\eta_{(e_i, t)}(A_{i-1}))^\beta}{\sum_{\theta \in T}(\tau_{(e_i, \theta)}(A_{i-1}))^\alpha \cdot (\eta_{(e_i, \theta)}(A_{i-1}))^\beta} \ . \quad (1)$$

In this general form of the equation, both the pheromone information τ and the heuristic information η take as argument the partial assignment A_{i-1}.[3] The impact of the pheromone and the heuristic information can be weighted by parameters α and β. In the following sections, we consider different ways of implementing the pheromone and heuristic information.

3.2 Pheromone Matrix

In a first representation, we let pheromones indicate the absolute position where events should be placed. With this representation the pheromone matrix is given by $\tau(A_i) = \tau, i = 1, \ldots, |E|$, i.e., the pheromone does not depend on the partial assignments A_i. Note that in this case the pheromone will be associated with *nodes* in the construction graph rather than *edges* between the nodes.

A disadvantage of this direct pheromone representation is that the absolute position of events in the timeslots does not matter very much in producing a good timetable. It is the relative placement of events which is important. For example, given a perfect timetable, it is usually possible to permute many groups of timeslots without affecting the quality of the timetable. As a result, this choice of representation can cause slower learning because during construction of solutions, an early assignment of an event to an 'undesirable' timeslot may cause conflicts with many supposedly desirable assignments downstream, leading to a poor timetable. This leads to a very noisy positive feedback signal.

In a second representation the pheromone values are indirectly defined. To do this we use an auxiliary matrix $\mu \in \mathbf{R}_+^{E \times E}$ to indicate which events should (or should not) be put together with other events in the same timeslot. Now, the values $\tau_{(e,t)}(A_i)$ can be expressed in terms of μ and A_i by

$$\tau_{(e,t)}(A_i) = \begin{cases} \tau_{max} & \text{if } A_i^{-1}(t) = \emptyset, \\ \min_{e' \in A_i^{-1}(t)} \mu(e, e') & \text{otherwise.} \end{cases}$$

Giving feedback to these values μ, the algorithm is able to learn which events should *not* go together in the same timeslot. This information can be learned without relation to the particular timeslot numbers. This representation looks promising because it allows the ants to learn something more directly useful to the construction of feasible timetables. However, it also has some disadvantages. For solving the soft constraints defined in Sect. 2, certain inter-timeslot relations between events matter, in addition to the intra-timeslot relations. This pheromone representation does not encode this extra information at all.

Some experimentation with the two different pheromone matrices indicated that the first one performed significantly better when the local search procedure was also used. Even though it is not ideal for the reasons stated above, it is capable of guiding the ants to construct timetables which meet the soft constraints as well as the hard ones. The problem of noisy feedback from this representation is also somewhat reduced when using the local search.

---

[3] For τ this is done to allow an indirect pheromone representation to be specified.

Clearly, other pheromone representations are possible, but with the variety of constraints which must be satisfied in the UCTP, it is difficult to design one that encodes all the relevant information in a simple manner. For the moment, the direct coding is the best compromise we have found.

3.3 Heuristic Information

We now consider possible methods for computing the heuristic information $\eta_{(e,t)}(A_{i-1})$. A simple method is the following:

$$\eta_{(e,t)}(A_{i-1}) = \frac{1.0}{1.0 + V_{(e,t)}(A_{i-1})}$$

where $V_{(e,t)}(A_{i-1})$ counts the additional number of violations caused by adding (e,t) to the partial assignment A_{i-1}. The function V may be a weighted sum of several or all of the soft and hard constraints. However, due to the nature of the UCTP, the computational cost of calculating some types of constraint violations can be rather high. We can choose to take advantage of significant heuristic information to guide the construction but only at the cost of being able to make fewer iterations of the algorithm in the given time limit. We conducted some investigations to assess the balance of this tradeoff and found that the use of heuristic information did not improve the quality of timetables constructed by the $\mathcal{MMAS}$ with local search. Without the use of LS, heuristic information does improve solution quality, but not to the same degree as LS.

3.4 Algorithm Description

Our $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System for the UCTP is shown in Alg. 1. A colony of m ants is used and at each iteration, each ant constructs a complete event-timeslot assignment by placing events, one by one, into the timeslots. The events are taken in a prescribed order which is used by all ants. The order is calculated before the run based on edge constraints between the events. The choice of which timeslot to assign to each event is a biased random choice influenced by the pheromone level $\tau_{(e,t)}(A_i)$ as described in (1). The pheromone values are initialized to a parameter τ_{max}, and then updated by a global pheromone update rule. At the end of the iterative construction, an event-timeslot assignment is converted into a candidate solution (timetable) using the matching algorithm. After all m ants have generated their candidate solution, one solution is chosen based on a fitness function. This candidate solution is further improved by the local search routine. If the solution found is better than the previous global best solution, it is replaced by the new solution. Then the global update on the pheromone values is performed using the global best solution. The values of the pheromone corresponding to the global best solution are increased and then all the pheromone levels in the matrix are reduced according to the evaporation coefficient. Finally, some pheromone values are adjusted so that they all lie within

Algorithm 1 $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System for the UCTP

---

input: A problem instance I
$\tau_{max} \leftarrow \frac{1}{\rho}$
$\tau(e,t) \leftarrow \tau_{max} \; \forall \; (e,t) \in E \times T$
calculate $c(e,e') \; \forall \; (e,e') \in E^2$
calculate $d(e)$
sort E according to $\prec$, resulting in $e_1 \prec e_2 \prec \cdots \prec e_n$
while time limit not reached **do**
   for $a = 1$ **to** m **do**
      {construction process of ant a}
      $A_0 \leftarrow \emptyset$
      for $i = 1$ **to** $|E|$ **do**
         choose timeslot t randomly according to probabilities $p_{e_i,t}$ for event e_i
         $A_i \leftarrow A_{i-1} \cup \{(e_i, t)\}$
      end for
      $C \leftarrow$ solution after applying matching algorithm to A_n
      $C_{iteration\ best} \leftarrow$ best of C and $C_{iteration\ best}$
   end for
   $C_{iteration\ best} \leftarrow$ solution after applying local search to $C_{iteration\ best}$
   $C_{global\ best} \leftarrow$ best of $C_{iteration\ best}$ and $C_{global\ best}$
   global pheromone update for τ using $C_{global\ best}$, τ_{min}, and τ_{max}
end while
output: An optimized candidate solution $C_{global\ best}$ for I

---

the bounds defined by τ_{max} and τ_{min}. The whole process is repeated, until the time limit is reached.

Some parts of Alg. 1 are now described in more detail. In a pre-calculation for events $e, e' \in E$ the following parameters are determined:

$c(e,e') := 1$ if there are students following both e and e', 0 otherwise, and
$d(e) := |\{e' \in E \setminus \{e\} \mid c(e,e') \neq 0\}|$.

We define a total order $\prec$ on the events by

$$e \prec e' :\Leftrightarrow d(e) > d(e') \; \vee$$
$$d(e) = d(e') \wedge l(e) < l(e') \;.$$

Here, $l : E \rightarrow \mathbf{N}$ is an injective function that is only used to handle ties. We define $E_i := \{e_1, \ldots, e_i\}$ for the totally ordered events denoted as $e_1 \prec e_2 \prec \ldots \prec e_n$.

Only the solution that causes the fewest number of hard constraint violations is selected for improvement by the LS. Ties are broken randomly. The pheromone matrix is updated only once per iteration, and the global best solution is used for update. Let $A_{global\ best}$ be the assignment of the best candidate solution $C_{global\ best}$ found since the beginning. The following update rule is used:

$$\tau_{(e,t)} = \begin{cases} (1-\rho) \cdot \tau_{(e,t)} + 1 & \text{if } A_{global\ best}(e) = t, \\ (1-\rho) \cdot \tau_{(e,t)} & \text{otherwise,} \end{cases}$$

where $\rho \in [0, 1]$ is the evaporation rate. Pheromone update is completed using the following:

$$\tau_{(e,t)} \leftarrow \begin{cases} \tau_{min} & \text{if } \tau_{(e,t)} < \tau_{min}, \\ \tau_{max} & \text{if } \tau_{(e,t)} > \tau_{max}, \\ \tau_{(e,t)} & \text{otherwise.} \end{cases}$$

3.5 Parameters

The development of an effective $\mathcal{MM}$AS for an optimization problem also requires that appropriate parameters be chosen for typical problem instances. In our case, we consider as typical those problem instances made by the generator described in Sect. 2.1. We tested several configurations of our $\mathcal{MM}$AS on problem instances from the classes listed in Tab. 1. The best results were obtained using the parameters listed in Tab. 2.

Table 2. Parameter configurations used in the comparison.

Parameter	small	medium	large
ρ	0.30	0.30	0.30
$\tau_{max} = \frac{1}{\rho}$	3.3	3.3	3.3
τ_{min}	0.0078	0.0019	0.0019
α	1.0	1.0	1.0
β	0.0	0.0	0.0
m	10	10	10

The values of τ_{min} were calculated so that at convergence (when one 'best' path exists with a pheromone value of τ_{max} on each of its constituent elements, and all other elements in the pheromone matrix have the value τ_{min}) a path constructed by an ant will be expected to differ from the best path in 20 % of its elements. The value 20 % was chosen to reflect the fact that a fairly large 'mutation' is needed to push the solution into a different basin of attraction for the local search.

4 Assessment of the Developed $\mathcal{MM}$AS

To assess the developed $\mathcal{MM}$AS, we consider whether the ants genuinely learn to build better timetables, as compared to a random restart local search (RRLS). This RRLS iterates the same LS as used by $\mathcal{MM}$AS from random starting solutions and stores the best solution found.

We tested both the developed $\mathcal{MM}$AS and the RRLS on previously unseen problem instances made by the generator mentioned in Sect. 2.1. For this test study, we generated eleven test instances: five small, five medium, and one large. For each of them, we ran our algorithms for 50, 40, and 10 independent trials,

Table 3. Median of the number of soft constraint violations observed in independent trials of $\mathcal{MMAS}$ and RRLS on different problem instances, together with the p-value for the null hypothesis that the distributions are equal. In the cases where greater than 50 % of runs resulted in no feasible solution the median cannot be calculated. Here, the fraction of unsuccessful runs is given. (In all other cases 100 % of the runs resulted in feasible solutions.) All infeasible solutions are given the symbolic value ∞. This is correctly handled by the Mann-Whitney test.

Instance	Median of #scv		p-value
	$\mathcal{MMAS}$	RRLS	
small1	1	8	$< 2 \cdot 10^{-16}$
small2	3	11	$< 2 \cdot 10^{-16}$
small3	1	8	$< 2 \cdot 10^{-16}$
small4	1	7	$< 2 \cdot 10^{-16}$
small5	0	5	$< 2 \cdot 10^{-16}$
medium1	195	199	0.017
medium2	184	202.5	$4.3 \cdot 10^{-6}$
medium3	248	(77.5 %)	$8.1 \cdot 10^{-12}$
medium4	164.5	177.5	0.017
medium5	219.5	(100 %)	$2.2 \cdot 10^{-16}$
large	851.5	(100 %)	$6.4 \cdot 10^{-5}$

giving each trial a time limit of 90, 900, and 9000 seconds, respectively. All the tests were run on a PC with an AMD Athlon 1100 Mhz CPU under Linux using the GNU C++ compiler gcc version 2.95.3.[4] As random number generator we used `ran0` from the Numerical Recipes [17]. For the reproducibility of the results on another architecture, we observed that on our architecture one step of the local search has an average running time of 0.45, 1.4, and 1.1 milliseconds, respectively.

Boxplots showing the distributions of the ranks of the obtained results are shown in Fig. 3. The Mann-Whitney test (see [4]) was used to test the hypothesis H_0 that the distribution functions of the solutions found by $\mathcal{MMAS}$ and RRLS were the same. The p-values for this test are given in Tab. 3, along with the median number of soft constraint violations obtained.

For each of the tested problem instances we got with very high statistical significance the result that $\mathcal{MMAS}$ performs better than RRLS. For some test instances of medium and large size some runs of RRLS resulted in infeasible solutions. In particular, the RRLS was unable to produce any feasible solution for the large problem instance.

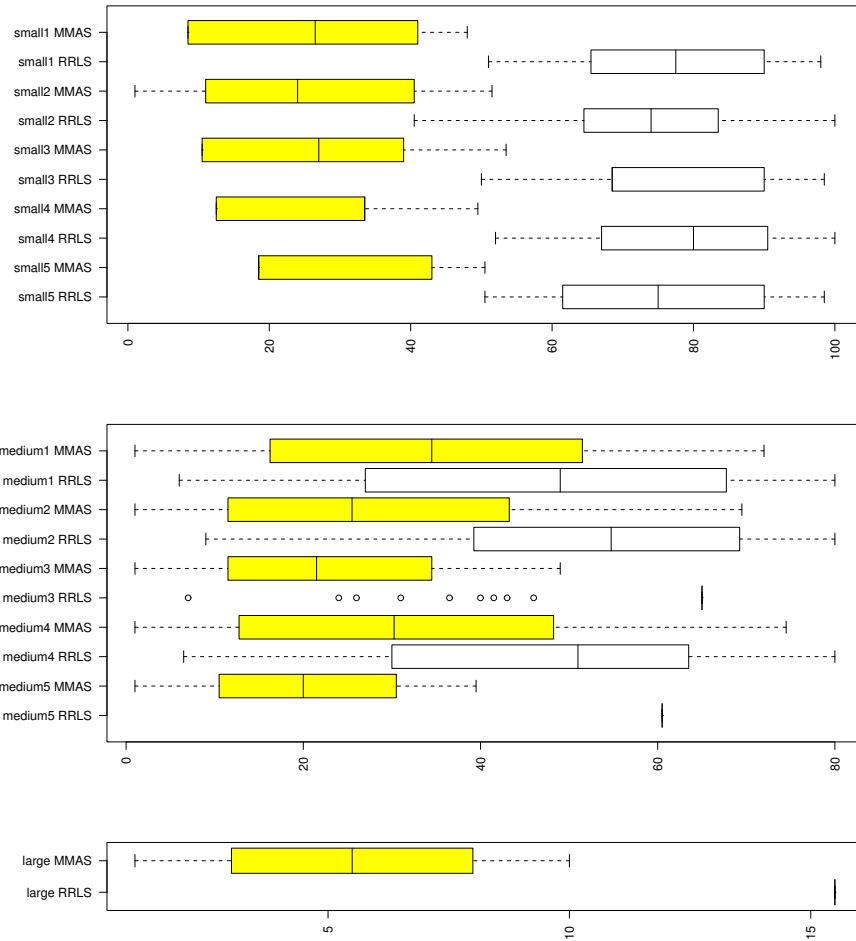[4] Both algorithms, the test instances and a detailed evaluation of the generated results can be found on `http://iridia.ulb.ac.be/~msampels/tt.data` .

Fig. 3. Boxplots showing the relative distribution of the number of soft constraint violations for $\mathcal{MMAS}$ (shadowed) and RRLS (white) on all test instances. This is the distribution of the ranks of the absolute values in an ordered list, where equal values are assigned to the mean of the covered ranks. A box shows the range between the 25 % and the 75 % quantile of the data. The median of the data is indicated by a bar. The whiskers extend to the most extreme data point which is no more than 1.5 times the interquantile range from the box. Outliers are indicated as circles

5 Conclusions

We devised a construction graph and a pheromone model appropriate for university course timetabling. Using these we were able to specify the first ACO algorithm for this problem. Compared to a random restart local search, it showed significantly better performance on a set of typical problem instances, indicating that it can guide the local search effectively. Our algorithm underlines the fact that ant systems are able to handle problems with multiple heterogeneous constraints. Even without using problem-specific heuristic information it is possible to generate good solutions. With the use of a basic first-improvement local search, we found that $\mathcal{MM}$AS permits a quite simple handling of timetabling problems. With an improved local search, exploiting more problem specific operators, we would expect a further improvement in performance.

Preliminary comparisons indicate that our $\mathcal{MM}$AS is competitive with the other metaheuristics developed in the Metaheuristics Network for the UCTP. Further results on the comparison of the metaheuristics will appear in [20].

References

1. C. Blum, M. Dorigo, S. Correia, O. Rossi-Doria, B. Paechter, and M. Snoek. A GA evolving instructions for a timetable builder. In *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling (PATAT 2002) (to appear)*, 2002.
2. E. Bonabeau, M. Dorigo, and G. Theraulaz. *From Natural to Artificial Swarm Intelligence.* Oxford University Press, 1999.
3. E. K. Burke, J. P. Newall, and R. F. Weare. A memetic algorithm for university exam timetabling. In *Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling (PATAT 1995)*, LNCS 1153, pages 241–251. Springer-Verlag, 1996.
4. W. J. Conover. *Practical Nonparametric Statistics.* John Wiley & Sons, 3rd edition, 1999.
5. T. B. Cooper and J. H. Kingston. The complexity of timetable construction problems. In *Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling (PATAT 1995)*, LNCS 1153, pages 283–295. Springer-Verlag, 1996.
6. D. Corne, P. Ross, and H.-L. Fang. Evolutionary timetabling: Practice, prospects and work in progress. In *Proceedings of the UK Planning and Scheduling SIG Workshop, Strathclyde*, 1994.

7. P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *Proceedings of the 3rd International Conference on Practice and Theory of Automated Timetabling (PATAT 2000)*, LNCS 2079, pages 176–190. Springer-Verlag, 2001.

8. D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48:295–305, 1997.

9. D. de Werra. The combinatorics of timetabling. *European Journal of Operational Research*, 96:504–513, 1997.

10. M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*. McGraw-Hill, 1999.

11. M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, 1999.

12. M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 26:29–41, 1996.

13. M. A. S. Elmohamed, P. Coddington, and G. Fox. A comparison of annealing techniques for academic course scheduling. In *Proceedings of the 2nd International Conference on Practice and Theory of Automated Timetabling (PATAT 1997)*, pages 92–115, 1998.

14. L. D. Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In *Proceedings of the 3rd International Conference on Practice and Theory of Automated Timetabling (PATAT 2000)*, LNCS 2079, pages 104–117. Springer-Verlag, 2001.

15. B. Paechter. Course timetabling. Evonet Summer School, 2001. http://evonet.dcs.napier.ac.uk/summerschool2001/problems.html .

16. B. Paechter, R. C. Rankin, A. Cumming, and T. C. Fogarty. Timetabling the classes of an entire university with an evolutionary algorithm. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN V)*, LNCS 1498, pages 865–874. Springer-Verlag, 1998.

17. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1993.

18. A. Roli, C. Blum, and M. Dorigo. ACO for maximal constraint satisfaction problems. In *Proceedings of the 4th Metaheuristics International Conference (MIC 2001)*, volume 1, pages 187–191, Porto, Portugal, 2001.

19. O. Rossi-Doria, C. Blum, J. Knowles, M. Sampels, K. Socha, and B. Paechter. A local search for the timetabling problem. In *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling (PATAT 2002) (to appear)*, 2002.

20. O. Rossi-Doria, M. Sampels, M. Chiarandini, J. Knowles, M. Manfrin, M. Mastrolilli, L. Paquete, and B. Paechter. A comparison of the performance of different metaheuristics on the timetabling problem. In *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling (PATAT 2002) (to appear)*, 2002.

21. T. Stützle and H. H. Hoos. $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

22. H. M. M. ten Eikelder and R. J. Willemen. Some complexity aspects of secondary school timetabling problems. In *Proceedings of the 3rd International Conference on Practice and Theory of Automated Timetabling (PATAT 2000)*, LNCS 2079, pages 18–29. Springer-Verlag, 2001.