An opinion algorithm for university course timetabling

Olivia Rossi-Doria and Ben Paechter

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

Abstract. We present here some ideas for a new constructive approach to the university course timetabling problem. The main motivation of this work is the observation that in timetabling it is the relative position of events that plays a major role in determining the quality of a solution, not the particular timeslot they are in. Therefore, when inserting a new event into the timetable, we want to take into account the "opinions" of other events on where it should be placed with respect to their own position, e.g. in the same timeslot, in a different day, or in an adjacent timeslot. The opinions here play a role similar to that of the pheromone in ant algorithms. A feasible timeslot for the event is chosen according to a probability based on this information and some additional heuristic information.
Some preliminary tests suggested that it is better to use the opinions information in the constructive decision process only some of the time with a certain diversity rate, while copying the best timetable so far in the other cases. Moreover at the moment it seems best to concentrate just on whether events want to be in the same timeslot, leaving 'adjacent timeslots' and 'same day' information out. We present promising initial results on the International Timetabling Competition benchmark instances.

1 Introduction

University course timetabling is the problem of producing a weekly timetable for the lectures of a university. A general problem consists in assigning a set of events (lectures, tutorials, etc.) into a limited number of timeslots, so that a set of constraints are satisfied. Constraints are usually classified as hard or soft. Hard constraints are constraints that must not be violated under any circumstances, while soft constraints should preferably be satisfied, but can be accepted with a penalty if necessary. Different versions of the problem can arise at different universities, depending on different requirements which are translated into constraints.

The general course timetabling problem is NP-hard. A considerable amount of research has dealt with the problem and comprehensive reviews can be found in [2, 8].

The version of the problem we focus on here is a reduction of a typical university timetabling problem, which was proposed by the Metaheuristics Network for the International Timetabling Competition [4]. The problem is described in Section 2.

In this paper we propose a new general approach to the university course timetabling problem. This is based on the observation that in timetabling what is important is not for any event to be in a specific timeslot and room, but it is the relative position of events with respect to each other that ultimately determines the quality of a solution in terms of hard and soft constraints. This is also true in particular for the variant of the problem we tackle here.

We therefore propose a constructive approach to the problem that takes into account the "opinions" of other events when a new event has to be placed into the timetable. This information plays a role similar to the one of the pheromone in ant algorithms, as explained in Section 3.1.

Since preliminary experiments suggest that the probabilistic element of the algorithm based on the opinions information leaves too much change in the constructive process, we tried to restrict its use only to a certain percentage of the decisions on where events have to be inserted. In the other cases the event will just be placed where it was in the best solution found so far, if the position is still available, otherwise the algorithm will resort again to the opinions system. We also noted that the algorithm works much better when it uses just one of the opinion lists at a time, in particular the opinion on whether events want to be in the same timeslot as other events gives the most promising results. As a result the algorithm seems to be promising. We describe it in more details in Section 3.

A very effective local search [3, 9] is used to further improve the solution timetables obtained by the algorithm at each iteration and it is described in Section 4.

Promising results on the International Timetabling Competition benchmark instances are presented and discussed in Section 5. Finally conclusions are drawn in Section 6.

2 Problem description

The problem consists of producing a weekly timetable for a university. Events, or lectures, have to be scheduled in 45 timeslots (5 days of 9 hours each) and a number of rooms, with varying facilities and student capacities, so that the following hard constraints are satisfied:

- **H1:** lectures having students in common cannot take place at the same time;
- **H2:** lectures must take place in a room suitable for them in terms of facilities and student capacity;
- **H3:** and no two lectures can take place at the same time in the same room.

We consider as well the following soft constraints:

- **S1:** students should not attend more than two lectures in a row;

– **S2:** they should not have to attend lectures in the last timeslot of the day;
– **S3:** and they should not have only one lecture in any given day.

Note that the given soft constraints are representative of three different types of constraints: constraint **S2** can be checked without knowledge of the rest of the timetable; **S1** can be checked while building a timetable; and **S3** can only be checked when the timetable is complete and all lectures have been assigned a timeslot.

A timetable in which all lectures have been assigned a timeslot and a room so that no hard constraint is violated is said to be feasible. The aim of the problem is to find a feasible solution with minimal soft constraint violations.

3 Algorithm description

The main idea of the algorithm is to take into account the "opinions" events might have on each other's relative positions. This information, together with some additional heuristic information, is used in a constructive process that at each step chooses probabilistically where to place any new event, in a way very similar to Ant Colony Optimization (ACO) [5].

The events are inserted into the timetable in a fixed order, determined according to a heuristic which considers how many other events clash with them, where an event is said to clash with another one if it shares at least one student with it. We also take into account which events have to go in just one particular room as suggested in Socha [9]. The resulting order is indicated with the simbol $\prec$, where $e_i \prec e_j$ if according to it event e_i comes before event e_j.

3.1 The opinion lists

When an event is being considered for insertion into the timetable we want other events to give their opinions on where that event might be placed. Each of the other events can have an opinion, for example, on how much they want it be in the same or a different timeslot; how much they want it be on the same or a different day; or how much they want it to be in an adjacent or non-adjacent timeslot.

This global information is kept in 6 lists of events, one for each type of opinion for each event. To each event in the list of a given event there is an associated value. The higher the value the greater is the weight of that event's opinion on where the given event has to be placed. These lists play a role similar to that of the pheromone in ACO, in that they help making a probabilistic choice during the constructive process, and in the fact that they are updated at each iteration with global and local information on the built solutions. The difference with respect to ACO is that there can be more than one pheromone information, one for each type of opinion, and for this reason, as the pheromone information becomes larger, it is worthwhile to have partial graphs, not complete ones with edges between each pair of events. This feature also expresses the fact that an

event might not have something to say about all of the other events, but only few relationships might be critical, and it is the task of the algorithm to make them emerge, as the opinion lists of events for each event are updated. We considered lists of length 6, i.e. each event will take into account the opinion of 6 other events for each opinion type.

The opinion lists are initialised by a heuristic and/or the information coming from an initial solution built randomly and improved by means of local search. In the case of the 'same timeslot' list we place in the list for a given event 6 events sharing the timeslot with it in the initial solution, giving precedence to the events that have the greater number of clashing events in common with it. If there are not enough events in the timeslot to fill the list, then the remaining events are chosen according to this latter heuristic. The opinion values for all the events are initialised to the maximum value allowed, that we chose to be 1.

3.2 The voting mechanism

At each constructive step the events in the lists of the event under consideration vote, according to their opinion value, on how much they want it to go into a particular timeslot. Feasible timeslots only are considered for each event, that is timeslots in which the event will not cause hard constraint violations. If no feasible timeslot is available the number of timeslots limit is temporarily relaxed to more than 45, and the local search will then take care of reducing this number back to 45 or less.

Suppose an event in the 'same timeslot' list of the event under consideration is assigned to a certain timeslot, then only that timeslot, if feasible, gets the corresponding vote. On the other hand an event in the 'different day' list gives its vote to all feasible timeslots that are not in the day it is in. If an event in an opinion list has not yet been placed then its vote is split between various timeslots according to where the event has been in previous iterations. The idea is that that event might have a higher probability of going into a timeslot where it has been it the past.

The votes from different events and opinions add up, and timeslots are chosen probabilistically according to the share of the votes they get. If more than one suitable room is available for the event in the chosen timeslot, then the room is chosen probabilistically. The probability of a room been chosen is inversely proportional to the number of events the room is suitable for.

An effective local search described in Section 4 is used to improve the quality of the built solutions.

3.3 Lists management

The opinion lists are then updated, after each iteration of the algorithm, based on global and local information on the quality of the resulting timetables.

The 'same timeslot' lists for example are updated by rewarding the events on the list of a given event which are actually in the same timeslot as that event

in the best solution found so far. The reward is proportional to the number of rooms filled in that particular timeslot for the best found timetable.

When the value of an event in a list becomes lower than a certain threshold, or minimum value, the event is deleted from the list and replaced by other promising events that might have something to say on the event owner of the list. That is events that are now in its same timeslot in the best solution found so far, with ties broken according to the greater number of clashing events in common. On the other hand the opinion values are always kept to be not greater than a certain maximum value.

Before any feedback is added the opinion values are "evaporated", as the pheromone in ant algorithms, to mantain a good exploration rate. After preliminary trials the evaporation rate of the opinion lists values was set to 0.1, in combination with a minimum opinion value allowed of 0.5, and a maximum possible opinion value of 1.

3.4 Some preliminary observations

Preliminary experiments showed that using the complete system with all 6 different opinion lists, the votes on different timeslots might be averaged out to be the same value. Moreover the probabilistic element of the constructive algorithm leaves too much freedom on where events will be placed, especially at the beginning of the constructive process when lots of timeslots are still available and a different position for an event can have catastrophic effects on the rest of the timetable. This led us to try to narrow down some excessive change by allowing the above voting mechanism to make the decision on where to place an event just some of the times, while for the rest the algorithm just tries to place events where they were in the best solution found so far. If this is not possible then the decision is again made by using opinion information. The simplest choice of using just the 'same timeslot' list, together with a diversification rate of 0.2, that is 20% of the events are inserted according to the voting mechanism, seems to be the most effective.

The number of building agents used at each iteration seems to be best set to 1, taking the algorithm further from ACO and much closer to an Iterated Local Search algorithm [6], where the perturbation is given by the constructive process which is strongly relying on information on the current best solution.

The algorithm is outlined in Algorithm 1, where n indicates the number of events, $\prec$ is the fixed order in which events are considered for insertion, s_0 is the initial random solution, H is the set of heuristics used to initialise and update the opinion lists, A_i is the partial assignment when the first i events have been inserted into the timetable, the function $rand()$ produces a random number between 0 and 1, $v(A_{i-1}, t_j)$ is the vote that the events in the 'same timeslot' list of event e_i give to a suitable timeslot t_j, s is the current solution, and s_{best} is the best solution found so far.

Algorithm 1 The opinion algorithm.

input: A problem instance I
sort the events according to $\prec$, resulting in $e_1 \prec e_2 \prec \cdots \prec e_n$
$s_0 \leftarrow$ `Initial_Random_Assignment()`
$s_0 \leftarrow$ `Local_Search()`
`Initialise_Opinion_Lists(`s_0, H`)`
while time limit not reached **do**
  $A_0 \leftarrow \emptyset$
  for $i = 1$ **to** n **do**
    if $rand() < 0.8$ **and** position of e_i in the current solution is free **then**
      copy timeslot t and room r from the position of e_i in the current solution
    else
      compute available timeslots t_j;
      add up the votes $v(A_{i-1}, t_j)$ from the events in the 'same timeslot' list of event e_i for each available timeslot;
      choose among them the timeslot t and room r according to probability distribution P for event e_i resulting from the votes $v(A_{i-1}, t_j)$ and the least suitable room heuristic;
      $A_i \leftarrow A_{i-1} \cup (e_i, t, r)$.
    end if
  end for
  $s \leftarrow A_n$
  $s \leftarrow$`Local_Search()`
  $s_{best} \leftarrow$ best of s and s_{best}
  `Update_Opinion_Lists(`s_{best}, H`)`
end while
output: An optimized assignment s_{best} for I

4 The local search

The effective local search used is due to Socha and Chiarandini [3, 9]. It consists of a stochastic process in two phases: the first phase to improve an infeasible timetable so that it becomes feasible by reducing the number of timeslots used; and the second phase to increase the quality of a feasible timetable by reducing the number of soft constraint violations. Two basic moves are considered in both phases: moving an event to a different suitable place, and swaping timeslots and rooms of two events so that they still are in suitable places. A suitable place for an event here means a timeslot and room pair where the event will not violate any hard constraints. Note that in the case of the first phase solving feasibility this can mean a timeslot greater than the given limit of 45.

Some further improvements are obtained by means of a matching algorithm, to re-assign rooms to events every time that an event is moved into a timeslot, and by means of Kempe chain interchanges, that is by exchanging all the events in two given timeslots that are connected because of student sharing and re-assigning rooms via the matching algorithm.

5 Results

Results show that the opinion algorithm presented in the paper is promising.

We tested the algorithm on the 20 benchmark instances of the International Timetabling Competition. They can be downloaded from the competition website [4], where results of the best participants to the competition can also be found. They were produced by a random instance generator, with size between 350 and 440 events, with between 200 and 300 students, and around 10 rooms. Other parameters needed by the generator to produce instances are the number of facilities, the approximate number of facilities per room, the percentage of facilities use, the maximum number of events per student, and the maximum number of students per event Perfect solutions, i.e. solutions with no constraint violations, exist for all of the 20 instances.

Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Best	98	57	145	201	160	50	31	80	86	139	116	140	143	66	94	76	152	69	180	20
Mean	110.6	77.6	150.6	208.4	185.7	62.6	43	97.2	110.6	173.4	130	167.2	160.8	74.8	101.2	102.2	176	81.8	192.8	37.8
Random	229	204	246	382	345	176	125	206	200	246	253	298	322	276	235	161	349	190	336	131

Table 1. Reported are the mean and the best value found by our algorithm on 5 runs per each instance. The last row of the table shows results of the random version of the algorithm on one run per each instance.

We ran 5 runs per instance for 20000 seconds on a group of 2.66 GHz Pentium 4 PCs, a much longer time than the one allowed for the competition. Indeed the algorithm manages to find good results. We report in Table 1 the best value found on the 5 runs for each instance, and the mean value of the 5 runs. We can

deduce that the algorithm appears to be reasonably stable, as the mean values are in general not too far from the best.

We also tested a version of the algorithm where, when not copying the current best solution, the decision of where to put an event are randomly taken, as opposed to probabilistically based on the opinion lists information. We show that the algorithm is indeed carrying important information for the building of good timetables as the random version is much worse. Results of one run per instance of the random version of the algorithm are reported in the last row labeled 'Random' in Table 1.

6 Conclusions

We presented some new ideas to tackle general timetabling problems, based on the fundamental observation that it is the relative position of events with respect to each other that determines the quality of a timetable. This led us to believe that it is important to take into account what other events might have to say on the placement of an event into the timetable. We therefore proposed a constructive approach that uses information on whether events want to be in the same timeslot as other events. We keep this information for each event in an opinion list of events that have something to say about being in the same timeslot as that event. The opinion lists play here a role very similar, yet different, to the role of pheromone in ant algorithms.

More work needs to be done to incorporate different types of opinions and different ways of rewarding events' opinions when they lead to good solutions. Nonetheless we showed that the direction of this research is promising, as the algorithm is carrying important information.

Acknowledgements. We would like to thank Marco Chiarandini and Kryzstof Socha for sharing the code of their local searches. Our work is supported by the *Metaheuristics Network,* a Research Training Network funded by the Improving Human Potential Programme 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. E. K. Burke, M. Carter (eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference. Lecture Notes in Computer Science 1408, Springer-Verlag, Berlin, 1997.
2. M. W. Carter and G. Laporte. Recent developments in practical course timetabling. In [1]. 3–19, 1997.
3. M. Chiarandini, M. Birattari, K. Socha, O. Rossi-Doria. An effective hybrid approach for the University Course Timetabling Problem. In preparation to be submitted to the Journal of Scheduling.

4. http://www.idsia.ch/Files/ttcomp2002/ Metaheuristics Network International Timetabling Competition.
5. 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.
6. H.R. Lourenço, O. Martin and T. Stützle Iterated Local Search In F. Glover and G. Kochenberger (eds), Handbook of Metaheuristics, Volume 57 of International Series in Operations Research & Management, 321–353, 2002, Kluwer Academic Publishers.
7. O. Rossi-Doria, M. Samples, M. Birattari, M. Chiarandini, J. Knowles, M. Manfrin, M. Mastrolilli, L. Paquete, B. Paechter, T. Stützle. A comparison of the performance of different metaheuristics on the timetabling problem. In PATAT 2002: The 4th international conference on the Practice And Theory of Automated Timetabling. Gent, Belgium, August 21-23, 2002. Lecture notes in Computer Science 2740, 329–351.
8. A. Schaerf. A survey of Automated Timetabling. Artificial Intelligence Review, 13:87–127, 1999.
9. K. Socha. The influence of run-time limits on choosing Ant System parameters. In Proceedings of GECCO 2003. Lecture notes in Computer Science.