A memetic 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@napier.ac.uk, b.paechter@napier.ac.uk

Abstract

We present here a metaheuristic approach to the university course timetabling problem using a memetic algorithm with a very effective local search. The algorithm was tested on the International Timetabling Competition benchmark instances. Results are encouraging and show that evolutionary computation with the help of local search can compete with succesful algorithms on the problem.

1 Introduction

University course timetabling is the problem of producing a weekly timetable for a university. Lectures have to take place in a given number of timeslots and rooms, so that a number of constraints are satisfied.

Different versions of the problem arise at different universities. In general two lectures having students in common cannot take place at the same time. This is a hard constraint which means that it must not be violated under any circumstances. Other types of hard constraints are usually included. A number of soft constraints are also usually considered. These are constraints that should be preferably obeyed, but, if necessary, can be broken at a certain cost. The aim of the problem is to minimize this cost while all hard constraints are satisfied. The general course timetabling problem is NP-hard [6, 5].

Comprehensive reviews on the timetabling problem and a number of research works to tackle it can be found in [3, 14].

The size and complexity of modern university timetabling problems has encouraged research in metaheuristic techniques, such as evolutionary computation, tabu search and simulated annealing. Metaheuristics are stochastic methods used when the size of the search space becomes unmanageble for exact methods and no effective algorithm capable of finding an optimal solution is available. They provide a framework to guide the search in order to find good solutions in an acceptable amount of time. There is no guarantee of finding the optimal solution, and due to the stochastic nature of the algorithms different solutions can be found in different runs. Nonetheless metaheuristics have shown to be highly effective and provide state-of-the-art solutions to many different problems.

In this paper we present a memetic algorithm for the university course timetabling problem. Memetic algorithms [10] are metaheuristics combining population-based global search as evolutionary computation with local search made by each of the individuals. While evolutionary computation [7, 9] takes inspiration from biological evolution of species, memetic algorithms mimic cultural evolution. The term *meme* denotes the idea of a cultural unit of knowledge transmitted after re-interpretation and improvement, that in the context of combinatorial optimization corresponds to local serach. A population of candidate solutions is created. Individuals can be chosen randomly or with the help of heuristic information. After that, local search is applied to each individual to improve them before they interact with each other. The population is evolved through three major stages to generate more

fit individuals: selection of better individuals, their reproduction through recombination and mutation, and a replacement strategy to include into the population the offspring after improvement by local search.

In Section 2 we describe the version of the problem we address here. Section 3 accounts for the memetic algorithm used to solve it. A very effective local search [4, 15] is used to further improve the solution timetables at each generation. Section 4 gives information on the International Timetabling Competition benchmark instances used to test the algorithm. Promising results are presented and discussed in Section 5, proving that evolutionary computation in combination with a good local search can deal with the problem and compete with other succesful algorithms. Finally conclusions are drawn in Section 6.

2 The University Course Timetabling Problem

The version of the problem tackled here was proposed by Ben Paechter for the International Timetabling Competition [8] organized by the Metaheuristics Network. It is referred to in the following as the University Course Timetabling Problem (UCTP-C), where the C stands for competition.

Lectures must 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 have to attend lectures in the last timeslot of the day;
- **S2:** they should not attend more than two lectures in a row;
- **S3:** and they should not have an only lecture in any given day.

Note that the given soft constraints are representative of three different types of constraints: constraint **S1** can be checked without knowledge of the rest of the timetable; **S2** 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 A memetic algorithm for the UCTP-C

We have implemented a memetic algorithm which makes use of the representation and local search, due to Socha and Chiarandini [4, 15], that proved very good for the International Timetabling Competition. We briefly recall them in Section 3.1 and 3.2. Constructive heuristics used by the algorithm are reported in Section 3.3 and the evolutionary operators are described in Section 3.4. The algorithm is outlined in Algorithm 1.

Algorithm 1 The memetic algorithm.

input: A problem instance I
for $i = 1$ **to** $n/2$ **do**
  $s_i \leftarrow$ `Initial_Heuristic_Assignment(`h_i`)`
  $s_i \leftarrow$ `Local_Search()`
  $s_i \leftarrow$ `Local_Search_Further_Improvements()`
end for
for $i = n/2$ **to** n **do**
  $s_i \leftarrow$ `Initial_Random_Assignment()`
  $s_i \leftarrow$ `Local_Search()`
  $s_i \leftarrow$ `Local_Search_Further_Improvements()`
end for
sort population by fitness
while time limit not reached **do**
  $p_1 \leftarrow$ `Tournament_Selection()`
  $p_2 \leftarrow$ `Tournament_Selection()`
  $c \leftarrow$ `Uniform_Crossover(`p_1, p_2, H`)`
  $c \leftarrow$ `Adaptive_Mutation(`p_1, p_2, H`)`
  $c \leftarrow$ `Local_Search()`
  if $c \neq p_1$ **and** $c \neq p_2$ **then**
    $s_n \leftarrow c$
    sort population by fitness
    $s_{best} \leftarrow s_1$
  end if
end while
output: An optimized timetable solution s_{best} for I

3.1 Representation

As suggested in [15], the timetable is represented in the form of an integer matrix T with t rows and r columns, t and r being respectively the number of timeslots and the number of rooms. The value of entry $T_{i,j}$ is the label of the lecture that takes place in timeslot i and room j. If no lecture is placed in the position of the timetable corresponding to timeslot i and room j then $T_{i,j}$ takes the value -1.

It should be noted that this representation does not allow to encode all possible assignment of lectures to timeslots and rooms. In particular no two lectures can share the same position in the timeslot. Such an assignment, however, is not feasible by definition, and the chosen representation is able to encode any feasible assignment. Furthermore, no assignment that would cause hard constraint violations is allowed by the algorithm. This means that a lecture will only be placed in a suitable room, and that no lecture will ever be placed in a timeslot where there is already another lecture sharing students with it. To make this possible the maximum number of timeslots used might be temporarily relaxed to more than 45 in order to accomodate all lectures that wouldn't fit anywhere without breaking hard constraints. Of course a timetable that uses more than 45 timeslots cannot be regarded as feasible as it does not fulfil the requirements in the problem definition. The local search is usually able to reduce the number of timeslots used and find a feasible timetable.

3.2 Local search

We use a very effective local search consisting 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 a lecture to a different suitable place, and swaping timeslots and rooms of two lectures so that they still are in suitable places. A suitable place for a lecture here means a timeslot and room pair where the lecture 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 and of Kempe chain interchanges. The matching algorithm is used to re-assign rooms to lectures within a timeslot every time that a lecture is moved into it. Kempe chain interchanges consists in exchanging all connected lectures in two given timeslots and re-assigning rooms via the matching algorithm. Lectures are connected when they cannot be in the same timeslot because of student sharing or because they both require the same single room.

3.3 Heuristics

Constructive heuristics have been used in the literature [2, 16] to solve graph colouring problems or related timetabling problems. The most well known for graph colouring is related to the number of edges incident to a vertex of the graph, that in the corresponding timetabling problem is the number of correlations of a lecture, or the number of other lectures sharing students with it. Here they are used to initialise the population in a constructive process as well as for a crossover repair mechanism, when lectures remain unscheduled.

In particular we use two types of heuristics which were originally designed for a hyperheuristic approach to the problem [13]. The first type is used to choose which lecture should be inserted next into the timetable and include the following heuristics: choose the lecture with maximum number of correlations, the lecture with maximum weighted number of correlations, with maximum number of students, with minimum number of possible rooms, with maximum number of required facilities, with room suitable for most lectures. The second type of heuristics are used to choose into which timeslot and room the lecture should be placed, and include: choose the timeslot with most parallel lectures, the first or last available timeslot, the smallest possible room, the room suitable for least lectures.

3.4 Evolution process

We use the Steady-State evolution model, proposed by Withley [17], where only one child solution is generated from two parents at each generation.

Half of the initial population of 10 individuals is generated with the help of costructive heuristics described in Section 3.3, where the assignments of timeslots and rooms are always chosen among feasible ones, i.e. assignments that do not cause hard constraint violations. The other half is generated in a semi-random manner to ensure diversity. This means that the assignment of each lecture is determined randomly among the feasible ones. Every individual is improved by means of the local search before evolution starts.

Selection: Both parents are selected using tournament selection of size 2, that is two individuals are chosen at random from the population and compared, and the best one of the two is selected to be parent.

Recombination operator: The two selected parents are recombined using a simple crossover where the child's lectures inherit their position uniformly from both parents. If at any time a position for a lecture is already taken then the lecture is inserted in a list of not yet scheduled lectures. Lectures from this list are then placed into the timetable with the help of constructive heuristics described in Section 3.3.

Mutation operator: Mutation is adaptive with respect to parents' diversity, with a maximum mutation rate of 0.8 when parents are identical. It consists of random moves in the local search neighbourhood, extended by 3-opt. It tries at random one of the following:

- pick at random a starting lecture and try to put it in a feasible timeslot;
- try to swap two lectures taken at random within the feasible region;
- try to exchange the positions of three lectures picked at random within the feasible region.

All three types of move are tried until a feasible move can be performed or all moves of that type have been tried;

A directed mutation re-scheduling the lectures causing most problems according to heuristic information was tested as well, also in combination with random mutation, but it was discarded as it was not effective.

Local search: Only the basic local search described in Section 3.2 is applied to improve every child, as the local search extensions using the matching algorithm and Kempe chains proved not useful and quite expensive at this level. Applying the local search extensions to the best known solution every so often doesn't give any improvement either.

Replacement strategy: At each generation the child replaces the worst member of the population, provided that it is not identical to either parent in order to avoid early convergence.

4 Problem instances

We have tested the algorithm on the 20 instances proposed for the International Timetabling Competition. They can be found on the official website [8] together with the best results found by participants. They were produced by a random instance generator with the following parameters: number of lectures between 350 and 440, number of students between 200 and 300, and number of rooms around 10. Other parameters specified to the generator are the number of facilities, the approximate number of facilities per room, the percentage of facilities use, the maximum number of lectures per students, and the maximum number of students per lecture.

Perfect solutions, i.e. solutions with no constraint violations, hard or soft, exist for all the 20 instances.

5 Experimental results

Results are encouraging, giving great improvement on a previous version using a different local search [11, 4], and they are approaching best known results on the International Timetabling Competition instances.

Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
MHN	57	31	61	112	86	3	5	4	16	54	38	100	71	25	14	11	69	24	40	0
Winner	45	25	65	115	102	13	44	29	17	61	44	107	78	52	24	22	86	31	44	7
2nd	61	39	77	160	161	42	52	54	50	72	53	110	109	93	62	34	114	38	128	26
3rd	85	42	84	119	77	6	12	32	184	90	73	79	91	36	27	300	79	39	86	0
4th	63	46	96	166	203	92	118	66	51	81	65	119	160	197	114	38	212	40	185	17
MA best	88	67	**94**	169	**194**	**62**	**84**	**57**	61	97	82	**116**	**130**	**146**	**80**	53	**134**	53	**142**	28
MA average	104	91	126	189	212	90	127	94	78	113	90	138	185	187	120	74	182	75	224	60

Table 1: Our results compared with the first 4 best official results of the competition and the Metaheuristics Network submission. Reported for our algorithm are the best and average fitness values found on 10 runs per instance, within the time allowed for the competition (1800 secs per run on a 500MHz PC).

We run the algorithm 10 times on each instance for 1800 seconds, which is the benchmark time allowed by the competition on our 500MHz PC. Reported in Table 1 are the best and

average fitness values found for each instance, as well as the best values found by the best four participants to the competition and by the Metaheuristics Network team that was not allowed to officially enter the competition.

6 Conclusions

We presented an effective memetic algorithm for the UCTP-C, showing that evolutionary computation can deal succesfully with the problem. The use of the effective local search is an important element of its good performance. Best found solutions out of only 10 runs per instance are among the best 5 entries in the International Timetabling Competition.

Acknowledgements. We would like to thank Kryzstof Socha and Marco Chiarandini for sharing their code. Our 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. 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 and S. Y. Lee. Examination Timetabling: Algorithmic Strategies and Applications. Journal of the Operational Research Society, 47:373–383, 1996.

[3] M. W. Carter and G. Laporte. Recent developments in practical course timetabling. In [1]. 3–19, 1997.

[4] M. Chiarandini, M. Birattari, K. Socha, O. Rossi-Doria. An effective hybrid approach for the University Course Timetabling Problem. Submitted to the Journal of Scheduling.

[5] T. B. Cooper, J. H. Kingston. The complexity of timetable construction problems. The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag, Berlin, 283–295, 1996.

[6] S. Even, A. Itai, A. Shamir. On the complexity of timetabling and multicommodity flow problems. SIAM Journal of Computation, 5:4, 691–703, 1976.

[7] J. Holland. Adaption in natural and artificial systems, The University of Michigan Press, 1975.

[8] `http://www.idsia.ch/Files/ttcomp2002/` Metaheuristics Network International Timetabling Competition.

[9] Z. Michalewicz. Genetic Algorithms + Data Structure = Evolution Programs. Springer, 1996.

[10] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program, Report 790, 1989. Pasadena, CA.

[11] O. Rossi-Doria, C. Blum, J. Knowles, M. Samples, K. Socha, B. Paechter. A local search for the timetabling problem. Proceedings of the 4th international conference on the Practice And Theory of Automated Timetabling 2002, 124–127. Gent, Belgium, August 21-23, 2002.

[12] 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.

[13] O. Rossi-Doria, B. Paechter. An hyperheuristic approach to the course timetabling problem using an evolutionary algorithm. Technical report, Napier University, Edinburgh, UK, 2003.

[14] A. Schaerf. A survey of Automated Timetabling. Artificial Intelligence Review, 13:87–127, 1999.

[15] K. Socha. The influence of run-time limits on choosing Ant System parameters. In Proceedings of GECCO 2003. Lecture notes in Computer Science.

[16] H. Terashima-Marín and P. Ross and M. Valenzuela-Rendón. Evolution of Constraint Satisfaction Strategies in Examination Timetabling. Proceedings of the Genetic and Evolutionary Computation Conference GECCO, 635–642. Morgan Kauffmann, 1999.

[17] D. Whitley. GENITOR: A Different Genetic Algorithm. In Proceedings of the Rocky Mountain Conference on Artificial Intelligence. Denver, USA, 1988.