An hyperheuristic approach to course timetabling problem using an evolutionary algorithm

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. In this paper we present an evolutionary algorithm which evolves the choice of the heuristics to be used at each step of the building process of a timetable. The original idea was developed at the EvoNet Summer School 2001, and published in [1]. Here we propose an improved algorithm with a different representation, and new sets of heuristics.

1 Introduction

Since their first applications to timetabling problems at the beginning of the 1990s, evolutionary algorithms have been extensively and successfully used to tackle the problem. Most approaches in the literature use a direct encoding. However it is often difficult to apply traditional crossover operators and some kind of repair mechanism is needed. Also, EAs fail to coordinate in solving different parts of the problem as shown in [7]. B. Paechter et al. [6] successfully use an indirect representation to tackle a real university timetabling problem. A different indirect approach is taken by Terashima-Marín et al. in [10], where they evolve different strategies to tackle examination timetabling using well-established problem-specific heuristics. We attempt here a similar approach for university course timetabling, evolving which heuristic is best used at each step of the building process of a timetable. In fact domain specific heuristics have been used in timetabling ever since the first attempts to automation [3], [2]. In particular sequential methods use heuristics to estimate how difficult an event will be to schedule, so to order the events by decreasing difficulty and first schedule the most problematic ones, which are most likely to cause problems if left until the end of the building process. Heuristic sequential methods generally produce acceptable results, and are computationally inexpensive. However they lack optimisation capabilities. That is why we think that they are best used in the context of a hyperheuristic approach. Hyperheuristics are heuristics which guide the choice of simpler problem-specific heuristics. They have proven successful in a number of applications, *e.g* scheduling [4], [5] and bin-packing [8]. The EA presented mix-and-matches heuristics to avoid the weakness of any single simple algorithm, evolving a strategy for solving the problem, rather than a specific solution.

2 University Course Timetabling Problem

The timetabling problem considered here is a reduction of a typical university timetabling problem. A problem instance generator produces problem instances with different characteristics for different values of given parameters. The problem consists of a set of events E to be scheduled in 45 timeslots (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 size. 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 candidate timetable is penalised 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 a single class on a day.

3 The solution representation

The solution representation is indirect and a timetable builder is used to decode the instructions encoded in the chromosome to produce a timetable.

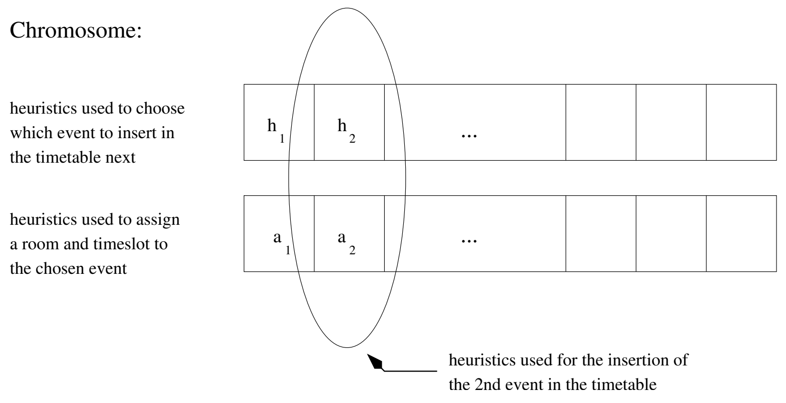
Chromosome:
heuristics used to choose which event to insert in the timetable next
h1
h2
...
heuristics used to assign a room and timeslot to the chosen event
a1
a2
...
heuristics used for the insertion of the 2nd event in the timetable

Fig. 1. Chromosome structure as a 2-row matrix.

A chromosome consists of two rows of integers of length $|E|$ representing the different heuristics to use at each step of the building process, as sketched in

Figure 1. The integer h_i in position i of the first row tells the builder how to choose the i-th event to insert in the timetable, while the integer a_i in position i of the second row is used to make an assignment of room and timeslot to the chosen event. In a previous work [1] two distinct rules were tried for each assignment of a chosen event, one to decide in which room the event should take place, and the other one to decide in which timeslot. It is quite intuitive though that the two choices are interconnected and it does not make much sense to decide to put an event in a certain room without knowing in which timeslot and vice-versa. Therefore we decided for using just one rule to make the complete assignment, and this seems to give better performance. This is also due to the fact that in this way it is easier to assess constraint violations of the whole assignment, and we do that to help building feasible good timetables.

Each chromosome represents a unique timetable when it undergoes the deterministic step-by-step sequential building process outlined below.

1. Let E_i be the set of all the unscheduled events at step i;
2. choose an event $e_i \in E_i$ according to heuristic h_i;
3. let H_i be the set of all possible room/timeslot assignments for e_i causing minimal hard constraint violations;
4. let S_i be the subset of the assignments in H_i causing minimal soft constraint violations;
5. choose an assignment for e_i in S_i according to heuristic a_i;
6. repeat until $i = |E|$ and all events have been inserted in the timetable.

4 The heuristics

Timetabling problems can be, and often are, seen as particular cases of graph colouring problems, where events are the nodes of a graph, timeslots are the colours to be assigned to each event, and clashes between any two events sharing one or more students corresponds to the edges between nodes. Sequential methods to solve graph colouring problems are extensively used and a number of well-established heuristics are known to tackle them. These same heuristics, opportunely adapted, can then be used to tackle timetabling: largest degree first; largest colour degree first; and least saturation degree first. Moreover, a number of heuristics have been and can be developed to better reflect the characteristics of the timetabling problem in consideration: maximum weighted number of event correlations (weighted largest degree first); maximum number of students; maximum number of features required by the events; minimum number of possible rooms; event with room suitable for most events; least saturation degree including room consideration. Of course these heuristics can be combined to break ties and generate more heuristics, and sometimes that appears to be very beneficial.

As for the assignment of room and timeslot to the chosen event we tried to look simply at timeslots in some previously intelligently arranged order, so to spread difficult events out for instance. But it appears to give better results, without being much more computationally expensive, to try to have a rule, or a combination of rules such as the following: most parallel timeslots; smallest possible room; room suitable for least events; least used room; latest or earliest timeslot in the day; latest or earliest day in the week. Again combinations of the previous rules to break ties seem to be most effective.

5 The evolution process

We implemented a Steady-State evolutionary algorithm with binary tournament selection. The initial population is seeded with single heuristic chromosomes and a number of randomly generated ones. We experimented with uniform and one-point crossover and results seem to be very similar, except that with one-point crossover the resulting simple algorithm is generally simpler to understand and describe, so we prefer this last choice. Random mutation is applied at a rate of $1/|E|$, and at each generation the offspring replace the worst member of the population.

6 Initial experiments and future work

The algorithm has been tested on randomly generated instances, mainly the five medium instances used in [9], giving very competitive results with respect to the algorithms described there for the two instances `medium02` and `medium05`. Not so good results are reached at the moment for the three other instances, maybe because we have not found the most appropriate heuristics to tackle them. Results are generally reasonable even before evolution, thanks to the seeded population of single heuristic strategies, but soft constraints do decrease meaning that better mixed strategies are found during the evolution process, up to a 50% on the best single heuristic strategy, generally around 25%.

Graph colouring heuristics seem not to work so well. This is probably due to the fact that they do not take into account students and rooms related issues, which are crucial to our problem. In facts, heuristics involving number of students and especially room related issues work much better. Moreover, graph colouring heuristics are more computationally expensive, since they imply re-evaluating the order of the list of events not yet scheduled at each insertion in the timetable. For this reasons it might be better to discard them from the set of available heuristics, at least on randomly generated instances produced by the problem instance generator presented in [9]. It is possible that they might work better on real data, where resources are generally less evenly distributed.

Further work has to be done to refine the set of heuristics and their combinations and to test the algorithm on different kind of instances. The evolution scheme might also be improved. Results of the improved algorithm on a variety of instances will be provided at the conference.

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. C. Blum, S. Correia, M. Dorigo, B. Paechter, O. Rossi-Doria, M. Snoek. A GA evolving instructions for a timetable builder. In Proceedings of the 4th international conference on the Practice And Theory of Automated Timetabling (PATAT 2002), Gent, Belgium, 120–123, 2002.
2. M. W. Carter, G. Laporte, S. Y. Lee. Examination Timetabling: Algorithmic Strategies and Applications. Journal of the Operational Research Society, 47:373–383, 1996.
3. A. J. Cole. The preparation of examination time-tables using a small-store computer. The Computer Journal, 7:117–121, 1964.
4. H. Fang, P. Ross, D. Corne. A promising hybrid ga/heuristic approach to open-shop scheduling problems. In A. Cohn ed., ECAI'94: Proceedings of the 11th European Conference on Artificial Intelligence, Wiley, 590–594, 1994.
5. E. Hart, P. Ross. A heuristic combination method for solving job-shop scheduling problems. Parallel Problem Solving from Nature (PPSN) V. Lecture Notes in Computer Science 1498, Springer-Verlag, Berlin, 845–854, 1998.
6. B. Paechter, R. C. Rankin, A. Cumming, T. C. Fogarty. Timetabling the classes of an entire university with an evolutionary algorithm. Parallel Problem Solving from Nature (PPSN) V. Lectures Notes in Computer Science 1498, Springer-Verlag, Berlin, 865–874, 1998.
7. P. Ross, E. Hart, D. Corne. Some observation about GA-based exam timetabling. In E. K. Burke, M. W. 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, 115–129, 1997.
8. P. Ross, S. Schulenburg, J. G. Marín-Blázquez, E. Hart. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002, Morgan Kauffmann, 942–948, 2002.
9. O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L. M. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, T. Stützle. A comparison of the performance of different metaheuristics on the timetabling problem. In Proceedings of the 4th international conference on the Practice And Theory of Automated Timetabling (PATAT 2002), Gent, Belgium, 115–119, 2002.
10. H. Terashima-Marín, P. Ross, M. Valenzuela-Rendón. Evolution of Constraint Satisfaction Strategies in Examination Timetabling. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO 1999, Morgan Kauffmann, 635–642, 1999.