A Simulated Annealing Hyper-heuristic for University Course Timetabling

Ruibin Bai[1], Edmund K. Burke[1], Graham Kendall[1]
Barry McCollum[2]

[1] Automated Scheduling, Optimisation and Planning (ASAP) Research Group
School of Computer Science & IT, University of Nottingham
Nottingham, NG8 1BB, UK
{rzb, ekb, gxk}@cs.nott.ac.uk
[2] Department of Computer Science, Queen's University Belfast
Belfast, BT7 1NN, UK
b.mccollum@qub.ac.uk

1 Introduction

The university course timetabling problem involves assigning a given number of events (including lectures, seminars, labs, tutorials, etc) into a limited number of timeslots and rooms subject to given set of constraints. Two primary hard constraints are that no student should be assigned two events in one timeslot and that capacity and features of rooms should satisfy the requirement of the event. Other constraints can be different from one university to another. For example, some universities might want the timetable to be constructed so that there is a good separation between the courses that a student attends, while other universities may prefer to have consecutive courses.

Timetabling is a well-known difficult combinatorial problem. Several techniques have been used to automatically generate university timetabling problems, including graph colouring heuristics (Burke et al., 2004), tabu search (Costa, 1994; Schaerf, 1996), simulated annealing (Thompson and Dowsland, 1996; Kostuch, 2004), evolutionary algorithms (Burke et al., 1998) and case-based reasoning (Burke et al., 2006a). Hyper-heuristic approaches have recently been applied to timetabling problems (Burke et al., 2003; Burke et al., 2006b). In (Burke et al. 2003), a tabu search based hyper-heuristic was applied to both a nurse rostering problem and a university course timetabling problem to demonstrate the increased level of generality of the method. In their approach, the hyper-heuristic dynamically ranks a set of heuristics according to their performance in the search history. A tabu list was incorporated to prevent the selection of some heuristics at certain points in the search. At each iteration, the hyper-heuristic keeps applying the highest "non-tabu" heuristic to the current solution until the stopping criterion is met. Competitive results have been obtained on both problems when compared with other state-of-the-art techniques. In (Burke et al., 2006b), a case-based reasoning hyper-heuristic system was proposed for the course timetabling problem. The system differs from other case-based reasoning systems in that case-based reasoning was used to predict the best heuristic methods that can be

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 345–350. ISBN 80-210-3726-1.

used to produce a good quality solution rather than finding a solution for the problem directly. The experimental results showed that the system can perform intelligently and effectively in the production of automated timetables.

In this research, we propose a simulated annealing hyper-heuristic approach (see fig. 1) for the university course timetabling problem. The simulated annealing hyper-heuristic manages a set of neighbourhood functions or heuristics and dynamically bias the selection of these heuristics. This approach has been successfully applied to a shelf space allocation problem (Bai and Kendall, 2005). It is hoped that the algorithm will either produce better results than the current proposed approaches or will reduce the computational time while generating good quality solutions.

2 Problem Description

In this research, we will study a course timetabling problem that was introduced in (Socha et al., 2002). The problem can be described as follows:

Given a set of events e_i (i = 0,...,n) and a number of rooms r_j (j = 0, ...,m) with each room having f types of features. Each event is attended by a given number of students and the total number of students is K. The aim of the problem is to assign every event e_i to a timeslot t_k (k = 1, ..., 45) and a room r_j so that the following *hard constraints* are satisfied:

1. No student should be assigned to more than one event at a timeslot;
2. The room assigned to an event should have sufficient capacity and all the features required by the given event;
3. No more than two events can be scheduled in one room at a timeslot.

The objective of the problem is to minimise the number of students involved in the following soft constraint violations:

1. A event is scheduled at the last timeslot of the day;
2. A student has only one event in a day;
3. A student has more than two consecutive events.

In reality, the course timetabling problem is often much more complex (McCollum, 1998). Additional constraints will be considered in subsequent research to those presented here.

3 Simulated Annealing Hyper-heuristics

3.1 Hyper-heuristics

The aim of hyper-heuristics is to develop a reusable, generic optimisation approach for a range of different problems and different problem instances. Hyper-heuristics can be defined as “heuristics to choose heuristics”. Hyper-heuristics search the solution space indirectly by operating on heuristics. A hyper-heuristic makes use of a set of diverse domain-dependent heuristics or neighbourhood functions and strategically

changes their preferences during the local search in order to adapt to different situations and problem instances (Burke et al., 2003; Ross, 2005).

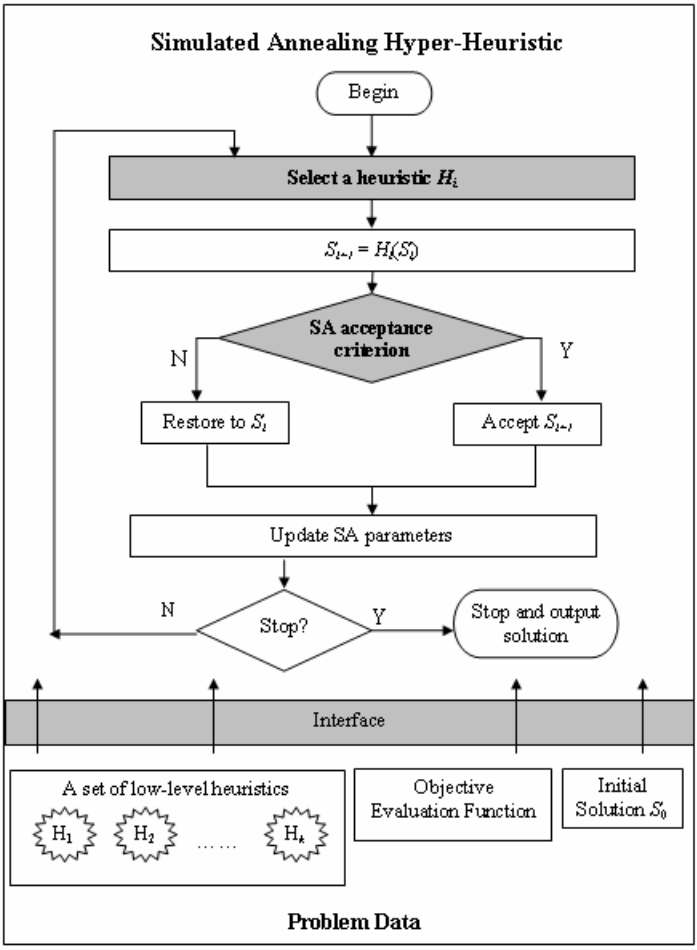

Fig. 1. The framework of simulated annealing hyper-heuristics for a maximisation problem

3.2 Simulated Annealing Hyper-heuristics

The proposed simulated annealing hyper-heuristic algorithm is a modified version of the algorithm in (Bai and Kendall, 2005), and is based on the following assumptions and observations.

1. A heuristic is selected probabilistically rather than deterministically. The reason for this is based on the empirical conclusion that probabilistically biasing the candidate solutions in SA is more beneficial than using a deterministic method (Tovey, 1988). A deterministic heuristic selection approach may be not suitable

for the simulated annealing hyper-heuristics due to the probabilistic characteristic of simulated annealing.

Set initial temperature t_s, stopping temperature t_f and total iterations K;
Generate an initial solution s_0; $t=t_s$;
Define a set of heuristic H_i (i=0, …, n), assign appropriate weight w_i to each heuristic H_i;
Do
  Select a heuristic (H_i) based on probability $p_i = w_i / \sum_{i=1}^{n} w_i$;
  Generate a candidate solution using heuristic H_i;
  Let δ_i stand for the difference in the evaluation function between s and s';
  If $\delta_i > 0$
    $s=s'$; $w_i = w_i + k$;
  else if $\delta_i = 0$ and a new solution is created
    $s=s'$; $w_i = w_i + \varepsilon$
  else if $\delta_i = 0$ and no new solution is created
    $w_i = w_i - \varepsilon$
  else if $\delta_i < 0$ and $\exp(\delta / t) < random(0,1)$
    $w_i = w_i - k$;
  if $w_i > w_{\max}$
    $w_i = w_{\max}$
  if $w_i < w_{\min}$,
    $w_i = w_{\min}$
Loop until stopping criteria are met

Fig. 2. Pseudo-code of the simulated annealing hyper-heuristics

2. To bias the selection of heuristics, each heuristic is associated with a weight that reflects their importance at the current stage. During the search, these weights are dynamically changed based on the performance of their corresponding heuristics.
3. The mechanism to change the weights of heuristics is a *penalty-reward* strategy. That is, the weight of a heuristic is increased if it produces a better solution and decreased otherwise. However, for those heuristics that cannot improve the evaluation function, we distinguish between the heuristics that generate new solutions and those that do not. We observe that during the search, although some heuristics cannot improve the solution directly, they are still useful in creating some intermediate situations, from which the optimal solution (or a good quality solution) could be reached. Hence in this system, we give a minor positive score to those heuristics which could transfer the state of the solution but could not

improve the objective value. Meanwhile, we penalise those heuristics which could neither improve the current solution nor generate a new solution.

The pseudo-code of the algorithm can be described in figure 2.

4 Application

To apply the proposed simulated annealing hyper-heuristics, we need to design a set of problem-dependent heuristics. We shall use the heuristics that were used in (Burke et al., 2003; Abdullah et al., 2005). The algorithm will then be tested on the benchmark problems that was introduced by (Socha et al., 2002).

References

1 Abdullah, S., Burke, E. K. and McCollum, B., Using a Randomised Iterative Improvement Algorithm with Composite Neigjhbourhood Structures for University Course Timetabling. In the Proceedings of MIC 05: The 6th Meta-heuristic International Conference, Vienna, Austria, 22-26 Aug, 2005.

2 Bai, R. and Kendall, G., "An Investigation of Automated Planograms Using a Simulated Annealing Based Hyper-heuristics," in Ibaraki, T., Nonobe, K., and Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solvers - (Operations Research/Computer Science Interfaces Series, Vol. 32) Berlin, Heidelberg, New York: Springer, pp. 87-108, 2005.

3 Burke, E. K., Causemacker, P. D., and Vanden Berghe, G., "Applications to Timetabling," in Gross, J. and Yellen, J. (eds.) Handbook of Graph Theory Chapman Hall/CRC Press, pp. 445-474, 2004.

4 Burke, E. K., MacCarthy, B. L., Petrovic, S. and Qu, R., Multiple-retrieval Case-based Reasoning for Course Timetabling Problems. Journal of Operations Research Society, 57(2): 148-162, 2006a.

5 Burke, E. K., Newall, J. P. and Weare, R. F., Initialisation Strategies and Diversity in Evolutionary Timetabling. Evolutionary Computation Journal (special issue on Scheduling), 6.1: 81-103, 1998.

6 Burke, E. K., Kendall, G. and Soubeiga, E., A Tabu-Search Hyperheuristic for Timetabling and Rostering. Journal of Heuristics, 9: 451-470, 2003.

7 Burke, E. K., Petrovic, S. and Qu, R., Case Based Heuristic Selection for Timetabling Problems. Journal of Scheduling, 9(2): 99-113, 2006b.

8 Costa, D., A Tabu Search Algorithm for Computing an Operational Timetable. European Journal of Operational Research, 76: 98-110, 1994.

9 Kostuch, Philipp, "The University Course Timetabling Problem with a 3-Phase Approach," in Burke, E. K. and Trick, M. (eds.) The Practice and Theory of Automated Timetabling V - Lecture Notes in Computer Science, Vol. 3616 Springer-Verlag, pp. 109-125, 2004.

10 McCollum, B., "The Implementation of a Centrally Computerised Timetabling System in a Large British Civic University," in Burke, E. K. and Carter, M. W. (eds.) Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT97), Lecture Notes in Computer Science Vol. 1408 Springer-Verlag, pp. 237-253, 1998.

11 Ross, Peter, "Hyper-Heuristics," in Burke, E. K. and Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques Springer, pp. 529-556, 2005.
12 Schaerf, A., "Tabu Search Techniques for Large High-school Timetabling Problems," in Franz, A. and Kitano, H. (eds.) Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI Press, pp. 363-368, 1996.
13 Socha, K., Knowles, J. and Samples, M., A Max-min Ant System for the University Course Timetabling Problem. Proceedings of the 3rd International Workshop on Ant Algorithm, ANTS 2002, Lecture Notes in Computer Science 2463: 1-13, 2002.
14 Thompson, G. L. and Dowsland, K. A., Variants of Simulated Annealing for the Examination Timetabling Problem. Annals of Operations Research, 63: 105-128, 1996.