Noname manuscript No.
(will be inserted by the editor)

HySST: Hyper-heuristic Search Strategies and Timetabling

Ahmed Kheiri · Ender Özcan · Andrew J. Parkes

Received: date / Accepted: date

1 Introduction

High school timetabling (HST) is a well-known real-world combinatorial optimisation problem. It requires the scheduling of events and resources, such as courses, classes of students, teachers, rooms and more within a fixed number of time slots subject to a set of constraints. In a standard fashion, constraints are separated into *'hard'*[1] and *soft*. The hard constraints must be satisfied in order to achieve *feasibility*, whereas the soft constraints represent preferences and a solution for a given problem; solutions are expected to satisfy all hard constraints and as many soft constraints as possible. The HST problem is known to be NP-complete [2] even in simplified forms. For a recent survey of HST see [4]. Also, see [5] for a description of the specific HST version studied here, and also of the third international timetabling competition, ITC2011. Briefly, the ITC2011 problem instances contain 15 types of constraints and a candidate solution is evaluated in terms of two components: feasibility and preferences. The evaluation function computes the weighted hard and soft constraint violations for a given solution as *infeasibility* and *objective* values, respectively. For the comparison of algorithms, a solution is considered to be better than another if it has a smaller infeasibility value, or an equal infeasibility value and a smaller objective value.

Our approach to solving the HST is based on development of a hyper-heuristic (see [1] for a recent survey) in order to intelligently control the application of a range of neighbourhood move operators. Hyper-heuristics explore the space of (meta-)heuristics, as opposed to directly searching the space of solutions, and generally, split into one of two types: *selection* hyper-heuristics that select between existing low-level heuristics, and *generation*

University of Nottingham, School of Computer Science
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
{ axk, exo, ajp }@cs.nott.ac.uk

[1] In the competition, such constraints are not strictly hard but are simply much more heavily penalised than the 'soft' constraints.

Algorithm HySST_Solver_ITC2011

$S = create\_initial\_solution()$; // takes t_{init} time
$t_{remaining} = t_{overall} - t_{init}$
$S_{best} = S$;
while ($t_{remaining}$ $notExceeded$)
  $S_{stage\_best} = S$;
  $S_{stage\_start} = S$;
  while (t_{stage} $notExceeded$) // stage entry
    $LLH = SelectRandomlyFrom(MutationalHeuristics)$;
    $S' = ApplyHeuristic(LLH, S)$;
    if (S' $isBetterThan$ S_{best}) **then** $S_{best} = S'$;
    if (S' $isBetterThan$ S_{stage_best}) **then** $S_{stage\_best} = S'$;
    $S = MoveAcceptance(S, S', \epsilon)$; // can accept with factor $(1 + \epsilon)$ worse
  if (S_{stage_best} $isNotBetterThan$ S_{stage_start}) **then**
    $S = ApplyHillClimbing(S)$; // uses the hill climbing heuristics
return S_{best};

Fig. 1 Pseudocode of the implemented hyper-heuristic. Here ϵ is a small (tunable) parameter that controls the level of worsening moves that are acceptable.

hyper-heuristics that build the decision procedures used with the heuristics. A selection hyper-heuristic generally includes both heuristic selection and move acceptance within a single point search framework (that is, without the use of populations of solutions): At each iteration, a candidate new solution is produced by selecting and applying a heuristic (neighbourhood operator) from a set of low level heuristics; A 'move acceptance' component then decides whether or not the candidate should replace the incumbent solution.

In this work, we develop and exploit a generalised selective hyper-heuristic (though envisage that future work could well exploit generative methods). We build on a previous study [3] that demonstrated the effectiveness of a generalised version of the iterated local search approach. Specifically, our hyper-heuristic uses a structured and staged application of multiple perturbative and hill climbing operators as opposed to simple selection from a single pool of all operators.

2 Hyper-heuristic Search for High School Timetabling

The search algorithm is implemented as a time contract algorithm and completes its execution in $t_{overall}$ time. It consists of an initial construction phase followed by an extended improvement; see Figure 1 for the pseudocode. The initial construction of a complete solution is performed using the general solver implemented by Jeff Kingston in the KHE library[2]. The improvement phase uses the remaining time left ($t_{remaining}$) after the construction of the initial solution which takes t_{init} time. Subsequently, a hyper-heuristic is used to control and mix a set of 11 low-level domain-specific heuristics and that are (mostly)

[2] `http://sydney.edu.au/engineering/it/~jeff/khe/`

fairly simple moves such as moving a task to a different resource, or swaps of events. They are divided into two sets; 9 mutational operators that do a randomised move, and 2 hill climbing operators that search their neighbourhoods for better solutions. Note that the construction phase often gives a solution in which hard constraints are violated, and so the improvement phase also needs to improve the hard constraints. The hyper-heuristic divides the search into stages where each stage takes a prefixed amount of time (t_{stage}). In each stage, firstly and repeatedly, one of the 9 mutational heuristics is applied and the move is accepted if it is improving or is not worsening by more than a small amount. If no improvement is achieved during a stage, a hill climbing phase is applied using the 2 hill climbing heuristics. Unlike most of the mutational operators, these 2 hill-climbing heuristics are capable of making quite large changes to a solution. The hill climbing phase is itself also slightly non-standard. One of the operators is designed using neighbourhood structures based on ejection chains while the other operator is a type of first improvement hill climbing operator. Both hill climbing operators attempt to make moves which respect a particular constraint type while hoping to improve upon the other types of constraint violations but might have a net worsening of the objective, however, then such worsening moves are rejected. A hill climbing step is always non-worsening and so can be repeatedly applied in standard fashion until a local minimum is reached. A notable difference from standard methods (such as in memetic algorithms) is that we found that performance is better if the hill climbing is not applied if the mutational operators managed to improve the best solution. We suspect that excessive use of the hill climbing somehow gives over-optimised local solutions that afterwards lead to restricted movement within the search space.

In summary, we have developed and applied a hyper-heuristic to intelligently and effectively exploit a suite of neighbourhood move operators. Since the aim of hyper-heuristics is to separate adaptive search control from the details of the specific domain, we naturally also envisage our hyper-heuristic could be applied to other PATAT-related problems, such as to the timetabling of university examinations and courses.

References

1. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society (to appear)
2. Even, S., Itai, A., Shamir, A.: On the Complexity of Timetable and Multicommodity Flow Problems. SIAM Journal on Computing **5**(4), 691–703 (1976)
3. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis **12**(1), 3–23 (2008)
4. Pillay, N.: An overview of school timetabling research. In: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), pp. 321–335 (2010)
5. Post, G., Gaspero, L.D., Kingston, J.H., McCollum, B., Schaerf, A.: The third international timetabling competition. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012) (2012)