Information Sciences 314 (2015) 225–239

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems

Nasser R. Sabar [a,*], Graham Kendall [a,b]

[a] ASAP Research Group, The University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor, Malaysia
[b] ASAP Research Group, The University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, UK

ARTICLE INFO

Article history:
Received 3 March 2014
Received in revised form 5 October 2014
Accepted 12 October 2014
Available online 24 October 2014

Keywords:
Hyper-heuristic
Monte Carlo tree search
Timetabling
Personnel scheduling

ABSTRACT

Hyper-heuristics aim to automate the heuristic selection process in order to operate well across different problem instances, or even across different problem domains. A traditional hyper-heuristic framework has two levels, a high level strategy and a set of low level heuristics. The role of the high level strategy is to decide which low level heuristic should be executed at the current decision point. This paper proposes a Monte Carlo tree search hyper-heuristic framework. We model the search space of the low level heuristics as a tree and use Monte Carlo tree search to search through the tree in order to identify the best sequence of low level heuristics to be applied to the current state. To improve the effectiveness of the proposed framework, we couple it with a memory mechanism which contains a population of solutions, utilizing different population updating rules. The generality of the proposed framework is demonstrated using the six domains of the hyper-heuristic competition (CHeSC) test suite (boolean satisfiability (MAX-SAT), one dimensional bin packing, permutation flow shop, personnel scheduling, traveling salesman and vehicle routing with time windows). The results demonstrate that the proposed hyper-heuristic generalizes well over all six domains and obtains competitive, if not better results, when compared to the best known results that have previously been presented in the scientific literature.

1. Introduction

Meta-heuristic methodologies are used to address problems that cannot be solved by exact methods, due to the size/complexity of the problem. These problems are typically $\mathcal{NP}$-hard. Meta-heuristics have reported many success stories and have tackled problems that would remain intractable if these search methodologies were not available. Meta-heuristics are, and remain, an important addition to the tools available to those who tackle optimization problems, although we may have reached saturation point with regard to the number of meta-heuristic algorithms that we need [54].

A feature of meta-heuristics is that they can often only be used for the problem for which they have been developed. To achieve the results reported in the scientific literature often requires extensive parameter tuning. This is often worthwhile for a given problem instance but it does not provide a return on that investment if we wish to utilize the same meta-heuristic algorithm on another domain. Indeed, a meta-heuristic algorithm may not work well on different problem instances drawn from the same domain, without additional parameter tuning.

* Corresponding author.
E-mail addresses: Nasser.Sabar@nottingham.edu.my (N.R. Sabar), Graham.Kendall@nottingham.edu.my (G. Kendall).

http://dx.doi.org/10.1016/j.ins.2014.10.045
0020-0255/

Hyper-heuristics aim to raise the generality of search algorithms by attempting to design algorithms that are able to operate well across instances drawn from the same domain, as well as being able to operate well on different problem domains. The challenges in achieving this are (at least) twofold. Firstly, we need a software framework that is able to cope with different problem domains, without having to redevelop the software for each domain that we tackle. Secondly, we do not want to tune the algorithmic parameters for each instance/problem. That is, we either want to tune the parameters once (the approach we take in this paper) and then use those settings across all the instances/domains that we are tackling, or we want the hyper-heuristic to adapt to the new instance/problem and tune the parameters as the algorithm executes.

As hyper-heuristic research matures, this methodology could become important to the industrial community, especially smaller companies. They often do not have the expertise, or financial resources, to use complex packages, or have bespoke software developed specifically for them. If a hyper-heuristic package is able to work across several problem domains, or even a common problem domain such as staff scheduling, or vehicle routing, it could provide low cost alternatives to software products that are currently available. Companies are often interested in getting a good quality solution to their problem. They may not require, indeed may not recognize, an optimal solution, but prefer software that is easy to use and can adapt to their problem.

Hyper-heuristic research must be carried out in the context of the No Free Lunch theorem [58]. This says that a general algorithm, that is superior to every other algorithm, across all problems, is not possible. However, that does not stop us attempting to develop algorithms that are more general than those currently available.

In this paper, we propose a new hyper-heuristic framework that has some similarities with exiting frameworks in that there is a high level strategy and a set of low level heuristics. One of the main contributions of this paper is the use of Monte Carlo Tree Search (MCTS) as the high level selection strategy, and the use of a Monte Carlo acceptance criterion to decide if a solution should be accepted or not. We are motivated to use MCTS as it has shown recent, significant success in the game of Go [48] and MCTS is now seen as our current best option to create a Go player that is competitive with the best human players.

We test our proposed algorithm across six different domains, demonstrating that the proposed hyper-heuristic is able to generalize well across a diverse set of problems. The results we present show that this has been achieved, as well as being competitive with the state of the art algorithms and producing new best results.

The structure of the paper is as follows. In the next section we discuss related work before presenting our proposed framework in Section 3. In Section 4, we provide details of our experimental setup and report our results in Section 5. In Section 6, we discuss our results before concluding the paper in Section 7.

2. Related work

2.1. Hyper-heuristics related work

The first appearance of the term *hyper-heuristic* in the scientific literature is in a technical report from 1996 [21]. The term is used to refer to a method of combining artificial intelligence methods in automated theorem proving. A paper the following year [22], with the same title as the technical report, did not explicitly use the term *hyper-heuristic* although the paper is obviously based on the technical report. Both of these works were preceded by a technical report [20], which should be noted for completeness.

It was in 2000 that the term first appeared in a peer reviewed paper [16], in the way that it is now commonly used. However, the roots of what we now regard as hyper-heuristics can be traced back to the 1960s [25,18].

Hyper-heuristics aim to raise the level generality at which search methodologies operate. The holy grail is to have one algorithm that works well across a range of different problems. We know that we can never have a single algorithm for any optimization problem [58] but it does not mean that we cannot aim for more general algorithms than are available today.

Early work on hyper-heuristics (for example [16,32,17,15,46,24]) focused on searching the heuristic space by deciding which heuristic to call at a given point in the search. That is, a set of *low level* heuristics are provided and the *high level* hyper-heuristic algorithm chooses which heuristic to apply at each iteration. One of the motivations for this type of hyper-heuristic is that the set of low level heuristics can be replaced and the hyper-heuristic is able to address a different problem, without changing the high level search algorithm. This type of hyper-heuristic is often referred to as *heuristics to choose heuristics* or a *selection hyper-heuristic* [49,11,30,8].

Hyper-heuristics have been applied to many domains, with timetabling receiving particular attention. Early timetabling/hyper-heuristic papers focused on selection hyper-heuristics. Burke et al. [6] investigated the hybridization of two graph coloring heuristics within a selection hyper-heuristic. Their results demonstrated the generality of the methodology. In [32] a new examination timetabling problem was introduced (the largest known problem in Malaysia). The authors used a tabu search based hyper-heuristic to produce better quality solutions than could be produced using a manually created solution. Rattadilok et al. [47] investigated parallel architectures within a hyper-heuristic framework, using a university timetabling problem as an experimental testbed.

Researchers have also investigated how their hyper-heuristic would cope with more than one domain. In [4], two health care timetabling problems (patient admission and nurse rostering) are studied. The paper demonstrates the effectiveness of the hyper-heuristic in addressing two different problems and was able to produce new best known results for patient