Knowledge-Based Systems 24 (2011) 322–330

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Integrating neural networks and logistic regression to underpin hyper-heuristic search

Jingpeng Li [*], Edmund K. Burke, Rong Qu

School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, United Kingdom

ARTICLE INFO

Article history:
Received 23 April 2010
Received in revised form 21 October 2010
Accepted 21 October 2010
Available online 26 October 2010

Keywords:
Hyper-heuristic
Data mining
Educational timetabling
Neural network
Logistic regression

ABSTRACT

A hyper-heuristic often represents a heuristic search method that operates over a space of heuristic rules. It can be thought of as a high level search methodology to choose lower level heuristics. Nearly 200 papers on hyper-heuristics have recently appeared in the literature. A common theme in this body of literature is an attempt to solve the problems in hand in the following way: at each decision point, first employ the chosen heuristic(s) to generate a solution, then calculate the objective value of the solution by taking into account all the constraints involved. However, empirical studies from our previous research have revealed that, under many circumstances, there is no need to carry out this costly 2-stage determination and evaluation at all times. This is because many problems in the real world are highly constrained with the characteristic that the regions of feasible solutions are rather scattered and small. Motivated by this observation and with the aim of making the hyper-heuristic search more efficient and more effective, this paper investigates two fundamentally different data mining techniques, namely artificial neural networks and binary logistic regression. By learning from examples, these techniques are able to find global patterns hidden in large data sets and achieve the goal of appropriately classifying the data. With the trained classification rules or estimated parameters, the performance (i.e. the worth of acceptance or rejection) of a resulting solution during the hyper-heuristic search can be predicted without the need to undertake the computationally expensive 2-stage of determination and calculation. We evaluate our approaches on the solutions (i.e. the sequences of heuristic rules) generated by a graph-based hyper-heuristic proposed for exam timetabling problems. Time complexity analysis demonstrates that the neural network and the logistic regression method can speed up the search significantly. We believe that our work sheds light on the development of more advanced knowledge-based decision support systems.

1. Introduction

The hyper-heuristic approach is an emerging search technology motivated by the goal of raising the level of generality at which search methodologies can be effective [8,7]. The aim can be thought of as being to formulate a high level search methodology to choose lower level heuristics. Hence, the most distinct feature of a hyper-heuristic lies in its exploration of a search space of heuristics, rather than of directly searching a problem solution space. As the neighbourhood structures of these two spaces are described very differently, this feature enables a hyper-heuristic to be capable of jumping within the solution space of the problem through moves among the neighbourhoods defined by the search space of heuristics.

It is clear that most of the metaheuristic implementations in the literature are very problem specific, especially when considering real-world problems. In this paper, we are seeking to build hyper-heuristic approaches (with the high level search algorithms often being meta-heuristics) that have several features. They should be faster to implement and more flexible to use than current state-of-the-art metaheuristic implementations. Also, they should produce good quality solutions without requiring too much domain knowledge. In hyper-heuristic development there is an obvious trade-off between "generality" and "solution quality".

For the purposes of illustration, consider a very simple timetabling problem of assigning m events to n time slots. A hyper-heuristic uses some method at a higher level to choose from k heuristics at the lower level. According to the way that a solution is built, the hyper-heuristics can be grouped into two classes: constructive hyper-heuristics and improvement hyper-heuristics. The former class builds solutions by applying a sequence of heuristic rules from scratch, while the latter improves solutions by applying one of the heuristics which are often related to the shifting or swapping of the solution components.

Clearly, the solution space of the original problem is n^m. For a constructive hyper-heuristic, its solution space is k^m. For an

* Corresponding author.
E-mail addresses: jpl@cs.nott.ac.uk (J. Li), ekb@cs.nott.ac.uk (E.K. Burke), rxq@cs.nott.ac.uk (R. Qu).

0950-7051/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2010.10.004

improvement hyper-heuristic, its solution space is $\sum_{j=1}^{t}\sum_{i=1}^{k} f(h_i) = O(tkn^r)$, where $f(h_i)$ is the number of possible moves that the ith heuristic can take (e.g. $f(h_i) = n(n-1)/2$ if the ith heuristic is to randomly swap the time slots of two events), $r \in [1, n-1]$ is the maximum number of concurrent shifts allowed in all k heuristics (e.g. $r = 2$ if the ith heuristic is a swap one), and $t \in [1, n^m - 1]$ is the number of improvements that the hyper-heuristic could make.

Among the three solution spaces described above, the improvement hyper-heuristic space is the smallest because it normally consists of an r value (being smaller than 3) and a t value being fixed, and thus in practice it does not increase exponentially with the problem size. Hence, the improvement hyper-heuristic is the most flexible and it works particularly well if a good initial solution is provided. A constructive hyper-heuristic also operates on a much reduced search space since k is mostly smaller than n. The problem induced by an improvement hyper-heuristic is NP-hard in the worst case where $r = n - 1$ and $t \to n^m - 1$, while the problem induced by a constructive-heuristic is NP-hard under any circumstance. Hence, the optimal solutions for both problems are normally impossible to find. Even if we find them, they might only correspond to one of the many possible solutions of the original problem and thus we cannot claim that the original problem has been well solved.

To address this weakness, we can consider more appropriate low-level heuristics, in addition to more iterations, during the hyper-heuristic search to facilitate more opportunity to undertake effective search. However, this would significantly increase the search space and consequently increase the computational time since the search and evaluation by each heuristic are costly. This is particularly the case for constructive heuristics.

Hence, so far almost all the research has been concentrated on the development of various innovative high level systems, such as refined heuristics [27,21,30], meta-heuristics [9,31,19] and Artificial Intelligence-based approaches [12,22,3,39]. Those systems act as a supervisor at a higher level to manage a small number of heuristics at a low level, with the aim of achieving a balance between the solution quality and the execution time. In this sense, this kind of research on hyper-heuristics can be thought of as not being too different from ordinary optimisation search algorithms, especially if we are simply considering each low-level heuristic as one of the solution components (sometimes with a fixed fitness value but usually with a variable fitness value).

More specifically, since the year 2000 when the term *hyper-heuristics* was first used [18], nearly 200 papers have appeared in the literature (see http://www.cs.nott.ac.uk/~gxo/hhbibliography.html). The tendency here is to use the following overall solution approach: at each decision point within the search space of heuristics, first use the chosen heuristic (such as a move, a swap or a sequence of heuristic rules) to generate a solution, and then calculate the objective value of the resulting solution to decide if the solution is worth keeping or not.

Intuitively, we feel that there is no need to carry out these costly 2-stage evaluations for every resulting solution. For example, an experienced human scheduler will not consider a move, a swap or a rule sequence that potentially results in an inferior solution, and will not take time to evaluate a solution that seems infeasible. That is, he/she would first obtain a reference of the solution, and then evaluate the exact solution if necessary. In addition, many real-world problems represent search spaces with feasible regions being rather scattered and small. For example, during our earlier work on two types of nurse rostering problems [2,10], we observed that the close neighbours of an infeasible solution are mostly infeasible, while the close neighbours of a feasible solution are often feasible as well. In that sense, in order to run more iterations on a larger heuristic search space (i.e. containing more heuristics) within the same computational time, the following two questions (which have attracted little attention before) naturally arise: Would it be possible for a hyper-heuristic to suspend the 2-stage evaluations if it predicts that the incoming heuristic (or sequence of heuristic rules) will not perform well (e.g. cannot generate a feasible solution)? If yes, what is the cost for carrying out such a prediction?

Several knowledge discovery systems have been recently proposed to deal with intractably large search space and to generate high-quality patterns [36,39,24,40], and we observe that the two papers by Santos et al. [36] and Thabtah and Cowling [39] have some connections with our work. The first paper is concerned with data mining in meta-heuristics, although it could be easily extended to the hyper-heuristic framework if its proposed DM-GRASP method operates on a set of heuristics rather than on just a single heuristic. The second paper is concerned with data mining in hyper-heuristics by using a standard associative classification. In essence, the methodologies proposed in these two papers use different associative classification methods to aid the decision of which lower level heuristic should be applied next. Once the single heuristic is determined, it will be used to construct a solution in a standard way (i.e. by implementing the 2-stage evaluations). In this context, these two methods are still similar to the other papers in the literature.

This paper will explore neural networks and logistic regression as a basis for underpinning hyper-heuristic research within the context of the above questions, with a focus on runtime comparisons. In the following sections, we will first introduce an example hyper-heuristic which is proposed for exam timetabling problems. We will then present two data mining techniques (neural networks and logistic regression) to classify the solutions generated by the hyper-heuristic. Next, we will highlight some classification results based on a set of benchmark test instances. Finally, we will present a concluding discussion and outline some future research directions.

2. Case study on exam timetabling problems

2.1. A graph-based hyper-heuristic

Timetabling has attracted a significant level of research interest since the 1960's. The general timetabling problem comes in many different guises such as nurse rostering [16,6], sports timetabling [20], transportation timetabling [25] and educational timetabling [13,14,37,29,32]. Educational timetabling problems are probably the most widely studied.

Exam timetabling can be considered to be the process of assigning a set of events (i.e. exams) into a limited number of timeslots subject to a set of constraints. Constraints are usually divided into two types: hard and soft. A hard constraint cannot be violated under any circumstances. A typical example is two exams with common students involved cannot be scheduled into the same timeslot. A soft constraint is one that should be satisfied if possible but its satisfaction is not essential. A typical example is represented by exams taken by common students having to be spread out over the available timeslots so that students do not have to sit two exams that are too close to each other.

Exam timetabling problems can be modeled as graphs, with each vertex representing an event and each edge representing a conflict (i.e. the situation where two vertices involve the same students) [6]. Naturally, graph heuristics can be employed to order the unscheduled events according to the difficulties of scheduling them into a timeslot without violating any hard constraints. The difficulty of an event is represented by its degree in the graph, or the number of conflicts it has with the others. The rationale behind this is to make sure the most difficult exams are scheduled first. The objective of graph colouring based timetabling heuristics is