Knowledge-Based Systems 98 (2016) 185–199

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A tensor based hyper-heuristic for nurse rostering

Shahriar Asta*, Ender Özcan, Tim Curtois

ASAP, School of Computer Science, University of Nottingham, NG8 1BB Nottingham, UK

ARTICLE INFO

Article history:
Received 10 September 2015
Revised 21 December 2015
Accepted 23 January 2016
Available online 3 February 2016

Keywords:
Nurse rostering
Personnel scheduling
Data science
Tensor factorization
Hyper-heuristics

ABSTRACT

Nurse rostering is a well-known highly constrained scheduling problem requiring assignment of shifts to nurses satisfying a variety of constraints. Exact algorithms may fail to produce high quality solutions, hence (meta)heuristics are commonly preferred as solution methods which are often designed and tuned for specific (group of) problem instances. Hyper-heuristics have emerged as general search methodologies that mix and manage a predefined set of low level heuristics while solving computationally hard problems. In this study, we describe an online learning hyper-heuristic employing a data science technique which is capable of self-improvement via tensor analysis for nurse rostering. The proposed approach is evaluated on a well-known nurse rostering benchmark consisting of a diverse collection of instances obtained from different hospitals across the world. The empirical results indicate the success of the tensor-based hyper-heuristic, improving upon the best-known solutions for four of the instances.

1. Introduction

Hyper-heuristics are high level improvement search methodologies exploring space of heuristics in problem solving [18]. According to Burke et al. [19], hyper-heuristics can be categorized in different ways. A hyper-heuristic either *selects* from a set of available low level heuristics or *generates* new low level heuristics from existing components to solve a problem, leading to a distinction between *selection* and *generation* hyper-heuristics, respectively. Also, depending on the availability of feedback from the search process, hyper-heuristics can be categorized as *learning* and *no-learning*. Learning hyper-heuristics can further be classified into online and offline methodologies depending on the nature of the feedback received as a part of the search process. Online hyper-heuristics learn *while* solving a given problem instance, whereas offline hyper-heuristics process collected data gathered from the training instances and learn prior to solving the unseen instances.

Nurse rostering is a highly constrained scheduling problem which was proven to be NP-hard [33] in its simplified form. Solving a nurse rostering problem requires assignment of shifts to a set of nurses so that (1) the minimum staff requirements are fulfilled and (2) the nurses' contracts are respected [17]. The general problem can be represented as a constraint optimisation problem using 5-tuples consisting of set of nurses, days (periods) including the relevant information from the previous and upcoming schedule, shift types, skill types and constraints.

In this study, a novel selection hyper-heuristic approach is employed to tackle the nurse rostering problem. The proposed framework (which is an extension to the previous work in [4,7]) is a single point based search algorithm which fits best in the online learning selection hyper-heuristic category, even if it is slightly different to the other online learning selection hyper-heuristics. A selection hyper-heuristic has two main components: heuristic selection and move acceptance method. While the task of the heuristic selection is to select low level heuristics based on a strategy, the acceptance method decides whether or not the solution produced by the selected heuristic shall be accepted. Over the years many heuristic selection and move acceptance methods have been proposed. Examples of heuristics selection strategies are Simple Random (SR) and Random Gradient (RG) [25], Choice Function (CF) [25], Reinforcement Learning (RL) [42] and Tabu Search (TS) [20]. Examples of early (primitive) acceptance mechanisms are Improvement Only (IO) [25], Improving and Equal (IE) [10], and Naive Acceptance (NA) [12]. The IO acceptance criteria only accepts improving solutions (compared to the current solution) and solutions equal or worsening to the quality of the current solution are rejected. The IE acceptance criterion accepts improving as well as equal solutions. While the improving solutions are always accepted by the NA acceptance method and non-improving solutions are accepted with a predetermined probability. More sophisticated acceptance algorithms such as Simulated Annealing(SA), Late Acceptance (LA) and Great Deluge (GD) can be found in the scientific literature [18].

* Corresponding author. Tel.: +44 115 9514251.
E-mail addresses: sba@cs.nott.ac.uk (S. Asta), exo@cs.nott.ac.uk (E. Özcan), tec@cs.nott.ac.uk (T. Curtois).

http://dx.doi.org/10.1016/j.knosys.2016.01.031
0950-7051/© 2016 Elsevier B.V. All rights reserved.

The proposed approach consists of running the simple random heuristic selection strategy (SR) in four stages. In the first two stages, the acceptance mechanism is NA with the fixed probability of 0.5, while in the second stage, we use IE as acceptance mechanism. The trace of the hyper-heuristic in each stage is represented as a 3rd order tensor. After each stage commences, the respective tensor is factorized which results in a score value associated to each heuristic. The space of heuristics is partitioned into two distinct sets, each representing a different acceptance mechanism (NA and IE, respectively) and lower level heuristics associated to it. Subsequently, a hyper-heuristic is created which uses different acceptance methods in an interleaving manner, switching between acceptance methods periodically. In the third stage, the parameter values for heuristics is extracted by running the hybrid hyper-heuristic and collecting tensorial data similar to the first two stages. Subsequently, the hybrid hyper-heuristic equipped with heuristic parameter values is run for a specific time. The above mentioned procedure continues until the maximum allowed time is reached.

Compared to the method proposed in [7], the framework here has few modifications. First, the framework in [7] has been extended to accommodate for an arbitrary number of acceptance criteria to be involved in the framework. That is, in contrast to the work in [7] where tensor data was collected for one acceptance criteria and the space of heuristics was partitioned into two disjoint sets, in this study, data collection and tensor analysis is performed for each hyper-heuristic separately. Moreover, low level heuristics are partitioned dynamically, rather than only once which was the case in [7] where ten (nominal) minute runs were considered. Mining search data periodically allows us to investigate whether the framework is capable of extracting new knowledge as the search makes progress. This could be useful in a variety of applications (i.e. life-long learning as in [30,47,48] or apprenticeship learning as in [5,8]. Finally, the framework here is different to the one proposed in [7] when parameter control for each low level heuristic is considered. While no parameter control was done in [7], in this study, parameters of each heuristic is tuned using tensor analysis. The good results achieved in this study shows that tensor analysis can also play a parameter control role.

The paper is organized as follows. Section 2 overviews the nurse rostering problem covering the problem definition, benchmarks in the area and related work. An introduction to tensor analysis is given in Section 3 where tensor representation of data, its advantages along with techniques widely employed to analyze tensorial data are explained. A detailed account of the proposed approach is provided in Section 4. The settings used in our experiments and the results of these experiments are laid out in Sections 5.1 and 5, respectively. Finally, concluding remarks and plans towards future work are discussed in Section 6.

2. Nurse rostering

In this section, we define the nurse rostering problem dealt with. Additionally, an overview of related work is provided.

2.1. Problem definition

The constraints in the nurse rostering problem can be grouped into two categories: (i) those that link two or more nurses and (ii) those that only apply to a single nurse. Constraints that fall into the first category include the cover (sometimes called demand) constraints. These are the constraints that ensure a minimum or maximum number of nurses are assigned to each shift on each day. They are also specified per skill/qualification levels in some instances. Another example of a constraint that would fall into this category would be constraints that ensure certain employees do or do not work together. Although these constraints do not appear in most benchmark instances (including those used here), they do occasionally appear in practise to model requirements such as training/supervision, carpooling, spreading expertize etc. The second group of constraints model the requirements on each nurse's individual schedule. For example, the minimum and maximum number of hours worked, permissible shifts, shift rotation, vacation requests, permissible sequences of shifts, minimum rest time and so on.

In this study, our aim is to see whether any improvement is possible via the use of machine learning, particularly tensor analysis. We preferred using the benchmark provided at [26] as discussed in the next section. These benchmark instances are collected from a variety of workplaces across the world and as such have different requirements and constraints, particularly the constraints on each nurse's individual schedule. This is because different organizations have different working regulations which have usually been defined by a combination of national laws, organizational and union requirements and worker preferences. To be able to model this variety, in [13] a regular expression constraint was used. Using this domain specific regular expression constraint allowed all the nurse specific constraints found in these benchmarks instances to be modeled. The model is given below.

Sets

E = Employees to be scheduled, $e \in E$
T = Shift types to be assigned, $t \in T$
D = Days in the planning horizon, $d \in \{1, \ldots |D|\}$
R_e = Regular expressions for employee e, $r \in R_e$
W_e = Workload limits for employee e, $w \in W_e$

Parameters

r^{max}_{er} = Maximum number of matches of regular expression r in the work schedule of employee e.
r^{min}_{er} = Minimum number of matches of regular expression r in the work schedule of employee e.
a_{er} = Weight associated with regular expression r for employee e.
v^{max}_{ew} = Maximum number of hours to be assigned to employee e within the time period defined by workload limit w.
v^{min}_{ew} = Minimum number of hours to be assigned to employee e within the time period defined by workload limit w.
b_{ew} = Weight associated with workload limit w for employee e.
s^{max}_{td} = Maximum number of shifts of type t required on day d.
s^{min}_{td} = Minimum number of shifts of type t required on day d.
c_{td} = Weight associated with the cover requirements of shift type t on day d.

Variables

x_{etd} = 1 if employee e is assigned shift type t on day d, 0 otherwise.
n_{er} = The number of matches of regular expression r in the work schedule of employee e.
p_{ew} = The number of hours assigned to employee e within the time period defined by workload limit w.
q_{td} = The number of shifts of type t assigned on day d.

Constraints

Employees can be assigned only one shift per day.

$$\sum_{t \in T} x_{etd} \leq 1, \quad \forall e \in E,\ d \in D \tag{1}$$

Objective function

$$Min f(s) = \sum_{e \in E} \sum_{i=1}^{4} f_{e,i}(x) + \sum_{t \in T} \sum_{d \in D} \sum_{i=5}^{6} f_{t,d,i}(x) \tag{2a}$$

where

$$f_{e,1}(x) = \sum_{e \in R_e} max\{0, (n_{er} - r^{max}_{er}) a_{er}\} \tag{2b}$$

$$f_{e,2}(x) = \sum_{e \in R_e} max\{0, (r^{min}_{er} - n_{er}) a_{er}\} \tag{2c}$$