Computers and Operations Research 82 (2017) 83–94

Contents lists available at ScienceDirect

Computers and Operations Research

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

A hybrid integer and constraint programming approach to solve nurse rostering problems

Erfan Rahimian[a,*], Kerem Akartunalı[a], John Levine[b]

[a] Department of Management Science, University of Strathclyde, Glasgow, G4 0GE, UK
[b] Department of Computer and Information Sciences, University of Strathclyde, Glasgow, G1 1XH, UK

ARTICLE INFO

Article history:
Received 1 August 2016
Revised 26 January 2017
Accepted 26 January 2017
Available online 27 January 2017

Keywords:
Timetabling
Nurse rostering
Hybrid algorithm
Integer programming
Constraint programming

ABSTRACT

The Nurse Rostering Problem can be defined as assigning a series of shift sequences (schedules) to several nurses over a planning horizon according to some limitations and preferences. The inherent benefits of generating higher-quality schedules are a reduction in outsourcing costs and an increase in job satisfaction of employees. In this paper, we present a hybrid algorithm, which combines Integer Programming and Constraint Programming to efficiently solve the highly-constrained Nurse Rostering Problem. We exploit the strength of IP in obtaining lower-bounds and finding an optimal solution with the capability of CP in finding feasible solutions in a co-operative manner. To improve the performance of the algorithm, and therefore, to obtain high-quality solutions as well as strong lower-bounds for a relatively short time, we apply some innovative ways to extract useful information such as the computational difficulty of instances and constraints to adaptively set the search parameters. We test our algorithm using two different datasets consisting of various problem instances, and report competitive results benchmarked with the state-of-the-art algorithms from the recent literature as well as standard IP and CP solvers, showing that the proposed algorithm is able to solve a wide variety of instances effectively.

1. Introduction

In order to ensure the right staff is on the right duty at the right time, *Nurse Rostering* has drawn significant attention during the last few decades, helping many health organisations to increase their efficiency and productivity. Creating a high-quality roster raises the recruitment and retention levels of nursing personnel, and maintains a reasonable overtime budget for nursing staff. From a financial perspective, it can reduce outsourcing and planning costs due to hiring fewer bank nurses to compensate gaps in rosters, and having flexible schedules (Kazahaya, 2005; M'Hallah and Alkhabbaz, 2013). In terms of human resource perspective, it can increase job satisfaction and diminish fatigue and stress, and hence results in improving caring services provided to patients (Burke et al., 2004; Ozcan, 2005).

The *Nurse Rostering Problem (NRP)* aims to generate rosters for several nurses over a predetermined planning horizon. A roster consists of a sequence of different types of shifts (e.g. early, late, vacations) spanning over the whole planning period. The patterns of shifts are generated according to a set of requirements such as hospital regulations, and a number of preferences such as fair distribution of shifts between nurses. Due to the complex and highly-constrained structure, the real-world NRP is often computationally challenging, and many variants of this problem are classified as *NP*-hard (Brucker et al., 2011; Chuin Lau, 1996). In practice, the inherent nature of the problem usually leads us to divide constraints into two categories: hard and soft constraints. Hard constraints must be satisfied to have a feasible roster, whereas soft constraints may be violated, albeit with a penalty. To evaluate the quality of a roster, one can minimise the sum of all penalties incurred due to soft constraint violations. We refer the interested reader to Burke et al. (2004) for further details on the NRP, and to Ernst et al. (2004) for a thorough review of staff scheduling problems.

In the literature, there are two areas of general methods used to solve these problems: exact and heuristic methods. Exact methods mostly include Integer Programming (IP) (Glass and Knight, 2010; Maenhout and Vanhoucke, 2010; M'Hallah and Alkhabbaz, 2013) and Constraint Programming (CP) (Girbea et al., 2011; Soto et al., 2013), which are capable of finding the optimal solution, albeit often resulting in unacceptable computational times. In order to address the computational limitations of these approaches, many heuristic methods have been proposed in the literature ranging from rather general Variable Neighbourhood Search

* Corresponding author.
E-mail addresses: erfan.rahimian@strath.ac.uk, er.rahimian@gmail.com (E. Rahimian), kerem.akartunali@strath.ac.uk (K. Akartunalı), john.levine@strath.ac.uk (J. Levine).

http://dx.doi.org/10.1016/j.cor.2017.01.016

(Lü and Hao, 2012; Rahimian et al., 2016a) and Genetic Algorithms (Ayob et al., 2013; Burke et al., 2001) to stochastic approaches (Tassopoulos et al., 2015) and tailor-made heuristics (Brucker et al., 2010). However, these methods sacrifice the guarantee of an optimal solution (or even any information regarding the solution quality) in order to generate good-quality solutions in acceptable computational times.

More recently, research in Operations Research and Artificial Intelligence communities, combined with robust solvers such as IBM CP Optimiser (IBM, 2015a) and Gurobi (Gurobi Optimization, 2015), have focused on using these methods in hybrid settings such as CP and heuristics (Stølevik et al., 2011), IP and heuristics (Valouxis et al., 2012), and the less well-investigated combination of IP and CP (Rahimian et al., 2015), in order to utilise the complementary strengths of all methods together. In this paper, we propose a new hybrid algorithm integrating IP and CP to solve the NRP, utilising the strengths of IP in finding optimal solutions and of CP in finding feasible solutions efficiently while exploiting problem-specific information. Due to the exact nature of the proposed algorithm, it can also generate strong lower-bounds in contrast to heuristic methods. Furthermore, the hybrid algorithm exploits problem-specific information to reduce the search space, to fine tune the search parameters, and to improve the efficiency of the whole search process in a novel fashion. For instance, during the search process, we identify the potential constraints which are computationally expensive to predict the performance of the IP solver, and thus, setting the search parameters adaptively.

Our main purpose in this paper is to extend the reach of exact method through hybridisation of exact methods. Indeed, using an IP approach as the core solution method, we employ a CP approach and other algorithmic aids to improve the efficiency of the overall algorithm. We do not intend to design a hybrid algorithm capable of generating the best result for challenging problem instances, particularly in comparison with advanced hybrid metaheuristics (Solos et al., 2013). Instead, we aim to develop a hybridisation of IP and CP which preserves benefits of exact methods, and is able to outperform each of them alone. Moreover, aiming to ease the implementation process and to increase the applicability of the hybrid algorithm, we do not apply any low-level or convoluted hybridisation settings. The proposed algorithm is designed to obtain the best result in a pre-defined, relatively short computational time. In addition, it does not depend on any specific settings regarding the importance of constraints, hence each constraint can be defined as hard or soft during the search process. We formulate the problem according to a general model reported in the literature (Burke et al., 2008b), and evaluate the proposed algorithm using two different datasets exist in the literature.

The rest of this paper is organised as follows: problem definition and assumptions are presented in Section 2. The IP and CP formulations are presented in Sections 3 and 4, respectively. In Section 5, we elaborate the proposed hybrid algorithm and the associated components. Computational results are reported in Section 6, and conclusions and potential future research directions are briefly discussed in Section 7.

2. Problem definition

The NRP is defined as the process of assigning a number of nurses to some work shifts during a planning horizon according to a set of requirements and constraints. The output of this process is a roster consisting of the allocated shifts (e.g. early and late) to all the nurses within the planning period. The constraints of the problem are often categorised as hard and soft constraints. In the following, we define decision variables and constraints similar to the conceptual model described in Burke et al. (2008b).

We define our decision variables for each nurse, day, and shift type. Although this approach to model the problem is less flexible and contains more symmetry compared with the pattern-based modelling, where often all possible weekly shift sequences (patterns) are generated (Burke et al., 2012), it allows us to better utilise the problem-specific structure in order to reduce the search space. We assume that the current roster is modelled over a specified planning horizon in an isolated way, i.e. no information (history) from the previous roster is used to construct the current one. In addition, a day off is considered as a shift type for modelling purposes. We have observed in various health settings that nurse rostering is performed in each hospital ward separately, and therefore, a single skill set is more realistic in practice than multi skill set. Therefore, we make the assumption that all nurses belong to the same skill category. For the sake of simplicity, we assume that all rosters start from Monday and are made from a whole week (i.e. seven days with a two-day weekend). The constraints of the model are:

1. Maximum one assignment per shift type per day for each nurse,
2. The number of shift types for each day must be fulfilled,
3. The minimum and maximum number of:
   (a) shift assignments within the scheduling period,
   (b) consecutive working days over the planning horizon,
   (c) working hours within the scheduling period (and/or during a week),
   (d) shift assignments within a week,
   (e) shift assignments at the weekend,
   (f) consecutive shift types over the planning period,
4. Minimum number of days off after a night shift or a series of night shifts,
5. Over the weekends, there should be either an assignment to all days of weekends or no assignments at all,
6. No night shift before free weekends (i.e. no assignment at the weekend),
7. Maximum number of consecutive worked weekends, when there is at least one weekend assignment,
8. Requested shifts (days) on/off, where some user-defined shifts (days) must (not) be allocated for a particular nurse within the planning horizon,
9. Forbidden shift type patterns (e.g. the *ND* pattern, where the shift type *D* is not allowed to be assigned right after the shift type *N*).

In the next two sections, i.e. Sections 3 and 4, we formulate this problem using IP and CP, respectively. For more details regarding the problem characteristics, we refer the reader to Burke et al. (2008b).

3. IP formulation

Here, we present our mathematical formulation using IP based on the definitions and assumptions provided in Section 2. For the sake of consistency, we also use the same numbering of constraints as in Section 2. We also note that the defined constraints can be considered hard or soft in different settings and problem instances, reflecting inherently different natures of wards, hospitals or health systems. However, the design of the hybrid algorithm is not dependent on a particular setting of constraints, hence each constraint can be defined as hard or soft depending on the user preferences. For demonstration purposes, we provide here a formulation assuming that some constraints are soft, namely (3d), (7), and (8). In case one needs to consider any of our hard constraints as soft in their IP model, they need to introduce auxiliary (slack) variables for each family of constraints in order to store the associated penalty, and also update the objective function, which is defined as