European Journal of Operational Research 237 (2014) 71–81

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

New approaches to nurse rostering benchmark instances

Edmund K. Burke [a], Tim Curtois [b,*]

[a] Department of Computing and Mathematics, University of Stirling, Cottrell Building, Stirling FK9 4LA, UK
[b] School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK

ARTICLE INFO

Article history:
Received 12 April 2012
Accepted 19 January 2014
Available online 30 January 2014

Keywords:
Staff scheduling
Nurse rostering
Branch and price
Ejection chain

ABSTRACT

This paper presents the results of developing a branch and price algorithm and an ejection chain method for nurse rostering problems. The approach is general enough to be able to apply it to a wide range of benchmark nurse rostering instances. The majority of the instances are real world applications. They have been collected from a variety of sources including industrial collaborators, other researchers and various publications. The results of entering these algorithms in the 2010 International Nurse Rostering Competition are also presented and discussed. In addition, incorporated within both algorithms is a dynamic programming method which we present. The algorithm contains a number of heuristics and other features which make it very effective on the broad rostering model introduced.

1. Introduction

Rostering problems are found in a wide range of workplaces and industries including healthcare, manufacturing, transportation, emergency services, call centres and many more. Using a computational search algorithm to address these problems results in cost savings and better work schedules. As such, rostering problems in various forms have received a large amount of research attention over the years. This body of research grew steadily throughout the 1960s, 1970s and 1980s and then accelerated in growth as more powerful desktop personal computers became commonplace in workplaces during the 1990s. As the computational and processing power has grown so has the range and complexity of algorithms that can be applied and the size and complexity of the instances that can be solved. For an overview of rostering problems and solution methodologies see (Ernst, Jiang, Krishnamoorthy, & Sier, 2004). A very large annotated bibliography of publications relating to staff scheduling is also provided by Ernst, Jiang, Krishnamoorthy, Owens, and Sier (2004). For a literature review specifically aimed at the nurse rostering problem, see (Burke, De Causmaecker, Vanden Berghe & Van Landeghem, 2004).

As these review papers show, many different approaches have been used to solve nurse rostering problems. These include metaheuristics (Bellanti, Carello, Croce, & Tadei, 2004; Burke, Curtois, Post, Qu, & Veltman, 2008; Burke, Curtois, Qu, et al., 2010; Ikegami & Niwa, 2003; Moz & Pato, 2007), constraint programming (Darmoni et al., 1995; Meyer auf'm Hofe, 2000; Weil, Heus, Francois, & Poujade, 1995), mathematical programming (Azaiez & Al Sharif, 2005; Bard & Purnomo, 2005), other artificial intelligence techniques (such as case-based reasoning (Beddoe & Petrovic, 2007) and hybrid approaches (Burke, Li, & Qu, 2010; Qu & He, 2008). Each method has strengths and weaknesses. For example, as will be shown in this paper, a mathematical programming approach may be able to solve some instances to optimality extremely quickly but on other instances it may take infeasible amounts of time or use too much memory. A metaheuristic, on the other hand, may be able to find a good solution to difficult instances quite quickly but may not be able to find the optimal solution to another instance which an exact method can solve very quickly. An obvious solution to this well-known phenomenon is to combine and hybridise different techniques. This is one of the principles behind adaptive approaches such as hyperheuristics.

The aim of this paper, however, is to provide new results (upper bounds and lower bounds) for a large collection of diverse rostering benchmark instances. This is the first occasion that a branch and price method has been applied to these instances. We also introduce the dynamic programming algorithm which is at the core of the branch and price method and we present a general rostering model which was used for all the instances tested.

Branch and price is a branch and bound method in which each node of the branch and bound tree is a linear programming relaxation which is solved using column generation. The column generation consists of a restricted master problem and a pricing problem. Solving the pricing problem provides new negative reduced cost columns to add to the master problem. The pricing problem can be considered as the problem of finding the optimal

* Corresponding author. Tel.: +44 (0)1158466521.
E-mail addresses: e.k.burke@stir.ac.uk (E.K. Burke), tim.curtois@nottingham.ac.uk (T. Curtois).

http://dx.doi.org/10.1016/j.ejor.2014.01.039
0377-2217/© 2014 Elsevier B.V. All rights reserved.

work schedule for an individual employee but with the addition of dual costs, that is, additional (possibly negative) costs based on which shift assignments are made or not made. In non-root nodes of the branch and bound tree, there may also be additional branching constraints on certain assignments that must or must not be made.

Although this is the first time that branch and price has been applied to these instances, it has previously been used on the nurse rostering problem (Eveborn & Rönnqvist, 2004; Jaumard, Semet, & Vovor, 1998; Maenhout & Vanhoucke, 2010; Mason & Smith, 1998). All these earlier applications have similar structure and the same structure is adopted here. The master problem is modelled as a set covering problem and solved using a linear programming method such as the simplex method. The pricing problem is formulated as a resource constrained shortest path problem and solved using a dynamic programming approach. The branch and bound tree is generally too large for a complete search and so heuristic, constraint branching schemes are adopted in which branching is performed on shift assignments in the roster. Although the dynamic programming algorithms all use the same principles (dominance pruning and bound pruning), the actual implementations are dependent on the constraints and objectives present in the pricing problem. For a recent overview of column generation see (Lubbecke & Desrosiers, 2005) and for further reading on resource constrained shortest path problems see (Irnich & Desaulniers, 2005).

In the next section, we discuss the challenge of modelling such a wide variety of instances and how it was solved. In Section 3, we introduce the benchmark instances and Section 4 presents the branch and price algorithm. Section 5 contains the results of applying the algorithms to the benchmark instances. In Section 6, we discuss the International Nurse Rostering Competition and finish with conclusions in Section 7.

2. Modelling the problem

One of the most significant challenges in addressing a large diverse collection of instances is developing a model which can be used for all the instances with their varying types of constraints and objectives. In all the instances, there are common types of constraints/objectives which are relatively straightforward to model. These include the cover constraints (ensuring that there is a correct or a preferable number of employees assigned to each shift). However, the types of constraints that can be present in each employee's work schedule can vary significantly from instance to instance. This is due to the reality of each workplace having its own set of rules and requirements defined by different employers, employees, unions and national legislation. Furthermore, each employee often has a different contract to reflect such features as full-time employment, part-time employment and night shift working. To provide a system which can incorporate these variations, we developed a general constraint based on pattern/string matching or more specifically regular expressions. Regular expressions are a powerful yet compact way of specifying patterns to be found or matched. They are commonly used in Computer Science and so we will not expand upon the subject here. Instead, we refer readers to one of the many textbooks on the subject such as (Friedl, 2006). Using a regular expression constraint in staff scheduling problems appears to be a natural fit and this is not the first example of its application to these type of problems (Côté, Gendron, Quimper, & Rousseau, 2011; Demassey, Pesant, & Rousseau, 2006; Pesant, 2004). However, in order to fully include all the variations in the instances we used, our approach is broader than some of this earlier work. First though, we will illustrate by example how this constraint can be applied in staff rostering problems. The basic idea behind the constraint is to consider the employee's work schedule as the 'search text' containing the regular expressions to be matched and the regular expressions to be matched are sequences of shifts. After presenting the examples below, we also provide a figure to illustrate how the constraint works in practice (Figs. 1–3). The figures show a short section of a single employee's schedule. The coloured squares labelled *E*, *D* and *N* represent early, day and night shifts respectively. The highlighted days show where the regular expression in question has been matched.

Example 1. If a night shift (*N*) can only be followed by another night shift or a day off then it could be modelled by the constraint "maximum zero matches of the pattern '*N* followed by any shift other than *N*'". Note that we use the expression "maximum zero" here as another way of saying this pattern must not appear at all. We use this expression instead though because all the matches are expressed as either a maximum or minimum number of matches in order to provide more modelling power (Fig. 1).

Example 2. If an employee must not work more than five consecutive shifts then it could be modelled by the constraint "maximum zero matches of the pattern '*Any*, *Any*, *Any*, *Any*, *Any*, *Any*'" where *Any* is any shift (that is, not a day off) (Fig. 2).

Example 3. If an employee must have a minimum of two consecutive night shifts then the constraint would be "maximum zero matches of the pattern 'anything but *N*, followed by *N*, followed by anything but *N*'" (Fig. 3).

As can be seen, the constraint is based on the idea of string/pattern matching. However, it is more like a regular expression and extends some of the previous work because we also allow:

- *Grouping*: Matching one of a group of shifts at a point in the sequence.
- *Negation*: Matching anything but a specific shift or group of shifts at a point in the sequence.
- *Alternation*: Matching multiple patterns.
- *Quantifiers*: The pattern(s) must appear a minimum or maximum number of times.
- Restricting the search text to a specific region of the work schedule.
- Only matching a pattern if it starts on a particular day in the work schedule.

This enables us to model some of the more complicated constraints such as those relating to weekend work or constraints that only apply between certain dates in the planning period. Using this general regular expression constraint, we can model many of the constraints found in staff scheduling problems. An example list is provided below.

- Minimum/maximum consecutive work days.
- Minimum/maximum consecutive non-work days.
- Day on/off requests.
- Shift on/off requests.
- Minimum/maximum number of shifts (optionally within a specific time frame).
- Minimum/maximum number of shifts of a specific type (optionally within a specific time frame).
- Minimum/maximum number of consecutive shifts of a specific type (optionally within a specific time frame).
- Days off after a series of shifts of a specific type.
- Shift rotations (which shifts can follow which shifts).
- Minimum/maximum shift rotations.