A Constraint Programming based Column Generation Approach to Nurse Rostering Problems

Fang He and Rong Qu*

The Automated Scheduling, Optimisation and Planning (ASAP) Group
School of Computer Science, The University of Nottingham
Nottingham, NG8 1BB, UK

Abstract

This paper presents our investigations on a hybrid constraint programming based column generation (CP-CG) approach to nurse rostering problems. We present a complete model to formulate all the complex real-world constraints in several benchmark nurse rostering problems. The hybrid CP-CG approach is featured with not only the effective relaxation and optimality reasoning of linear programming but also the powerful expressiveness of constraint programming in modeling the complex logical constraints in nurse rostering problems. In solving the CP pricing subproblem, we propose two strategies to generate promising columns which contribute to the efficiency of the CG procedure. A Depth Bounded Discrepancy Search is employed to obtain diverse columns. A cost threshold is adaptively tightened based on the information collected during the search to generate columns of good quality. Computational experiments on a set of benchmark nurse rostering problems demonstrate a faster convergence by the two strategies and justify the effectiveness and efficiency of the hybrid CP-CG approach.

Key words: column generation; constraint programming; heuristics; nurse rostering problems; scheduling.

1 Introduction

1.1 Background

The nurse rostering problem (NRP) represents an important administration activity in modern hospitals. The problem consists of generating a configuration of individual schedules for all available nurses satisfying a set of constraints including working regulations and nurses' preferences. Solving the problem properly has a positive impact on nurses' working conditions and contributes to a higher quality of healthcare [1, 2]. NRPs belong to the personnel scheduling problem, which is one of the combinatorial optimization problems. Solving complex NRPs, which is NP-hard [3], presents a challenge to both practitioners in practice and researchers in Operations Research (OR) and Artificial Intelligence (AI) communities.

Due to their strengths in optimality reasoning and relaxation, OR techniques such as linear programming [4], integer programming [5] and mixed-integer programming [6] have been applied to NRPs. The core problem of NRPs can be modeled as the set partitioning or set covering problem. These well structured models, although are often rather simplified, can help to reveal important problem properties and define efficient (and often polynomial) algorithms [7]. In personnel scheduling, researchers have investigated column generation and branch-and-price approaches [8-11].

However, real-world large scale NRPs are usually highly constrained and difficult to solve efficiently by using pure OR techniques. Constraint Programming (CP) offers a rich modeling paradigm to handle complex and heterogeneous constraints in combinatorial problems. Its strength of feasibility reasoning has been widely recognized in the literature, where NRPs are modeled as constraint satisfaction problems and solved subject to a set of complex constraints [12, 13].

In solving complex NRPs efficiently, the real challenges come from both the feasibility and optimality reasoning. This motivated our research on integrating OR and AI techniques within an integrated scheme. Constraint Programming based Column Generation (CP-CG) is such a scheme to integrate these two

* Dr Rong Qu, School of Computer Science, The University of Nottingham, Nottingham, NG8 1BB, UK; Tel: ++44 115 8466503, Fax: ++44 115 8467877, rxq@cs.nott.ac.uk

techniques. The overall efficiency of CP-CG depends on both the linear program relaxation and the solving of the CP pricing subproblem. Problem specific features can also be considered in modeling the problem to further improve the efficiency of CP-CG. In this paper we aim to investigate the effective interactions between CG and CP within CP-CG to solve complex NRPs.

1.2 Related work in CP-CG

The CP-CG approach has been firstly introduced in [14] and [15] to model and solve the crew assignment problem. It is a decomposition approach where CP is used to solve the pricing subproblem and CG is used to solve the master problem. The results show that, compared to using CP, using dynamic programming to solve the pricing subproblem is very time consuming due to the large set of constraints [14]. The CP-CG approach has since been widely applied to airline crew scheduling [16], vehicle routing [17] and bin packing [18] problems. In this work, motivated by the above work on different problems, CP is used to effectively model the complex constraints in NRPs and solve the pricing subproblem in CP-CG.

All the problems addressed by CP-CG share the similar structure. They may be inherently decomposed into a pricing subproblem and master problem. CG in the master problem is used to select a subset of patterns from a huge pool of all possible weighted patterns to construct the best complete solution to the problem. CP in the subproblem is used to generate the large pool of patterns with desired features of the problem solution [19]. For example, in the airline crew scheduling problem, each schedule (individual pattern) for the crew should satisfy a specific set of working regulations. Integer variables in the master problem of CP-CG represent which columns (i.e. patterns) are chosen to construct the complete solution. The linear relaxation of the integer program in the master problem is used to iteratively derive the optimality and choose a subset of columns. Through this integration, the hybrid CP-CG approach benefits from both the feasibility reasoning of CP and the optimality reasoning of linear programming.

In personal scheduling, CP-CG has been applied to airline crew assignment problems [14-16] and a personnel scheduling problem in a bank [19]. Research mainly focuses on solving the pricing subproblem using CP. The efficiency of the CP-CG algorithms mainly relies on the development of efficient *cost filtering* (propagation) algorithms in CP. In [19], all the key features of the problem have been modelled as hard constraints in the CP subproblem. The approach has showed to be inefficient when the working regulations are complex and the scheduling period is long, which are often the case in real-world NRPs.

In solving a physician scheduling problem [20], instead of focusing on *cost filtering* in CP as in most of the previous work [14-16, 19], the researchers have proposed two search strategies, a *dual strategy* and a *master strategy*, to improve the efficiency of CP-CG. To speed up the convergence of the objective value, in the linear relaxation of the master problem, the *dual strategy* selects the shift with the *l*th largest dual value (where *l* is randomly chosen) to drive the search towards columns with negative reduced cost. To speed up the convergence of the integer solutions, the *master strategy* stores the information of shifts that have been assigned, and chooses the least used shifts first. This strategy can be seen as a heuristic which chooses diverse columns for constructing integer solutions.

1.3 Motivations

Theoretically, at each iteration of CG in the master problem, the pricing subproblem searches for those columns with the greatest reduced cost (negative for the minimisation problem concerned here). When no such columns with negative reduced cost can be found, we obtain the optimal solution to the original problem [21, 22]. In CG, a column with the greatest (negative) reduced costs may not lead to the largest decrease of the objective function. The decrease of the objective function also depends on how much the entering variables can be increased. Thus, choosing the optimal column (with the minimum negative cost) is not always the best choice. In practice, any feasible columns which have negative reduced cost can serve as candidates to enter the master problem.

Generating feasible columns rather than optimal ones presents an easier problem, but leads to a slower convergence to the optimum with respect to the number of required iterations in CG. In this paper we address the issue of slow convergence in CG approaches. Another issue we face in developing efficient CP-CG is that the CP pricing subproblem tends to generate similar columns [21, 22] due to the default Depth First Search in CP, thus it is difficult to quickly reach integer solutions for the master problem.

In this paper we aim to address the above research issues in applying the CP-CG approach to complex NRPs. To deal with the slow convergence of CP-CG, we propose a cost threshold to select *good quality* columns. That is, not only the generated columns have negative reduced costs, but also their cost coefficients in the objective function are below a threshold. This cost threshold is adaptively updated (reduced) according to the solutions collected during the CG procedure. With regard to similar columns generated, we apply the Depth Bounded Discrepancy Search to obtain diverse columns. This strategy, to some extent, quickly leads to integer solutions of the master problem. The main contributions of the paper are as follows:

- We propose a CP-CG solution procedure where a complete model formulates all, rather than a subset of, real-world constraints in several benchmark NRPs concerned.
- We apply the Depth Bounded Discrepancy Search in CP-CG to obtain diverse columns for the master problem in CG concerning integrality. An adaptive bound tightening strategy is devised to obtain high quality columns during the problem solving based on a repeatedly updated cost threshold.

The remainder of this paper is organized as follows. Section 2 presents the integer program model in the master problem and the CP model in the subproblem for NRPs. Section 3 presents the overall solution procedure with details of the key steps. We demonstrate the efficiency of the CP-CG approach in Section 4 by an in-depth analysis over a set of experiments. This leads to the conclusions and future work in Section 5.

2 Modelling the Nurse Rostering Problem

2.1 The nurse rostering problem

In this work, we denote a sequence of day-on and day-off duties for a nurse within the scheduling period as the *schedule* of the individual nurse. On each day-on, the nurse can be assigned to a particular shift (e.g., early, day, evening or night shift). The overall timetable for all nurses (all schedules) is denoted as the *roster*. The NRP is to assign a schedule to each available nurse of a specific skill category. The problem data such as the number of personnel in a ward, and in each skill category, the demand of each category of nurses and the definition of shift types is determined in staff planning, which is the first stage of overall nurse workforce management [2].

Nurse rostering in hospitals involves a large number of constraints including working regulations, legal requirements and nurses' preferences, etc. The NRPs concerned in this paper are derived from real-life scenarios in hospital wards, and are a set of mostly tested benchmark problems in the literature. The rules and regulations have been directly taken from real-world cases and preserved with essential characteristics. The constraints in these problems are categorized into hard constraints and soft constraints, explained as follows:

- *Hard constraints* must be satisfied to obtain *feasible* solutions for use in practice. In our benchmarks we treat hospital rules as hard constraints to ensure a continuous service, i.e. hospitals must provide some minimum level of care in terms of the number of nurses for each shift type. Each nurse cannot work more than one shift on the same day. Other hard constraints include the working time (e.g. maximum 38 hours per week for full time nurses) and shift patterns (e.g. a minimum of 42 hours rest is required after two consecutive night shifts).

- *Soft constraints* are not obligatory but desired to be satisfied as much as possible. Balanced workload and individual preferences are usually treated as soft constraints. A weight (a non-negative value) is assigned to each soft constraint to reflect its importance (especially in comparison with other soft constraints). The higher the weight, the more strongly the constraint is desired to be satisfied. These weights are set either by the head nurses or based on feedback from the nurses with regard to the quality of their schedules they desire.

We present a summary of these constraints in Table 1. Details of all constraints in different problems concerned in this paper are listed in Appendix A.

Table 1 Summary of constraint categories in the benchmark NRPs, more details are presented in Appendix A.

Hard constraints	Only one shift on a day for each nurse
	Exact coverage requirement (no over/under cover of shifts)
	Working time
	Shift patterns
Soft constraints	Workload balance
	Shift patterns
	Pattern preferences

The objective of NRPs can be defined as to find a feasible roster (i.e. which satisfies all hard constraints) with the lowest possible penalty caused by soft constraint violations, i.e. to minimize a weighted sum of the penalties from all violations of soft constraints.

When modeling the master problem within the CP-CG approach (see Section 2.3), we only need to consider the coverage constraint in Table 1. All the other detailed constraints are encapsulated as the features of columns, and formulated in CP within CP-CG (see Section 2.4 and Appendix B).

2.2 The basic column generation formulation

Column generation (CG) is an efficient algorithm for solving linear programs with a large number of variables [23,24]. The basic idea is to, by exploiting problem substructures, decompose a linear program into two complementary components: a *master problem* (MP) and a *pricing subproblem* (Psub). The master problem (MP) has a compact linear program formulation as follows:

$$\begin{aligned}&\text{(MP) min } CX\\&\text{subject to } AX \geq B\\&\qquad\qquad X \geq 0\end{aligned}$$

Where vector X represents decision variables x_j; C represents objective function coefficients c_j; B represents the right hand side coefficients b_i; and matrix A represents constraint coefficients a_{ij}; $i \in I = \{1, \ldots, n\}, j \in J = \{1, \ldots, m\}$.

It is often impossible to directly solve the large master problem. CG provides a way to obtain the solution indirectly [16] by solving a much smaller problem, termed as *restricted master problem* (RMP) as follows:

$$\begin{aligned}&\text{(RMP) min } C\tilde{X}\\&\text{subject to } \tilde{A}\tilde{X} \geq B\\&\qquad\qquad \tilde{X} \geq 0\end{aligned}$$

where $\tilde{A} \subset A$ is a subset of $m' < m$ columns and $\tilde{X}$ are the corresponding variables. An optimal solution to the restricted master problem provides *dual values* λ_i of each constraint i, i.e. $\sum_{j \in J} a_{ij} x_j \geq b_i$, of the original linear program.

We then need to add variables to $\tilde{X}$ and the corresponding columns to $\tilde{A}$ to yield a linear problem with the same solution as the original master problem. The linear programming duality theory proves that only columns with *negative reduced cost* ($\pi_i < 0$) can be candidates to $\tilde{A}$, where a *reduced cost* measures the improvement of the objective function coefficient for the value change of the corresponding variable. This is the same way as how the Simplex algorithm chooses columns internally. It is also used to generate external columns in the CG method [16].

Let vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$ represent a new column we generate for the corresponding variables X of the master problem, and c_α denote the objective function coefficient associated to column $\boldsymbol{\alpha}$. Let F denote the feasibility region of the combinatorial objects represented by the columns of A. With the optimal dual values λ_i associated with constraints $\sum_{j \in J} a_{ij} x_j \geq b_i$, we have the following *pricing subproblem* (PSub):

$$\text{(PSub)} \quad \pi_i = c_\alpha - \sum_{i}^{i=n} \lambda_i \alpha_i < 0$$
$$\boldsymbol{\alpha} \in F$$

where $\boldsymbol{\alpha}$ are the decision variables of the pricing subproblem which encode the combinatorial objects and solution characteristics, i.e. characterize columns of A. If $\boldsymbol{\alpha}$ is a valid solution for the subproblem, i.e. $\boldsymbol{\alpha} \in F$, $\boldsymbol{\alpha}$ is added to $\tilde{A}$ in the master restricted problem.

Based on the above general formulations, next we define our NRPs within the CP-CG scheme, where the pricing subproblem defined in CP provides columns to CG of the master problem.

2.3 Formulating the master problem as integer program

Problem size parameters:

--N: set of nurses (index i)
--D: set of days in the scheduling period (index j)
--S: set of shift types, i.e. *Late*, *Early*, *Night*, *Off*, etc (index k)

Nurse parameters:

--G: set of nurse categories (i.e. different working contracts, e.g. 20, 32 or 36 hours per week, respectively) (index m)
--F_m: set of feasible schedules for nurse of category m with respect to related constraints of contract stipulations (index l)
--a_{ilmjk}: is 1 if schedule l for nurse i in category m covers the required shift k on day j; 0 otherwise
--c_{il}: penalty of schedule l which violates the constraints of contract stipulations of nurse i

Demand coverage parameter:

--R_{jk}: coverage demand of shift type k on day j

Decision variables in the master problem:

-- y_{il}: takes value 1 if nurse i is assigned to schedule l; 0 otherwise.

The **Master Problem Formulation (NRPs MP)** is modeled as follows:

$$\text{(NRPs MP)} \quad \min \sum_{m \in G} \sum_{i \in N} \sum_{l \in F_m} c_{il} y_{il} \tag{1}$$

$$\text{s.t.} \quad \sum_{m \in G} \sum_{i \in N} \sum_{l \in F_m} a_{ilmjk} y_{il} = R_{jk}, \forall j \in D, k \in S \tag{2}$$

$$\sum_{l \in F_m} y_{il} = 1, \forall i \in N \tag{3}$$

In (NRPs MP), the objective function is linear over the schedules y_{il}. The penalty of the entire solution (roster) is defined as the sum of the penalties of the selected schedules, i.e. objective (1) of the master problem aims to minimize the sum of penalties associated with the individual schedules y_{il} each nurse i is assigned to. Constraint (2) defines the required number of nurses for each shift type on each day (exact coverage). Formulating the coverage constraint as such allows flexible substitutability between nurses, i.e. schedules are exchangeable among nurses of the same category m. Constraint (3) assigns exactly one feasible schedule to each nurse subject to his or her specific constraints. A similar formulation of the restricted master problem has also been applied in [11].

In the above seemingly simple model, a large amount of complexity is actually hidden in the definition of schedule l in y_{il}, i.e. all the other constraints in Appendix A are implicitly modeled by the definition of schedule l, and the generation of each column (feasible schedule l) by CP must be subject to these constraints (see Section 2.4).

2.4 Formulating the pricing subproblem in CP

The subproblem concerning all complex constraints (except the coverage constraint) is modeled in CP to provide columns of desired features in CP-CG. The pricing subproblem is in the general form as follows:

$$\text{(P)} \quad \pi_i = c_\alpha - \sum_{i}^{i=n} \lambda_i \alpha_i < 0 \tag{4}$$

$$\boldsymbol{\alpha} \in F \tag{5}$$

Two groups of constraints, namely the negative reduced cost (4) and the feasibility constraints (5), are concerned.

In the reduced cost constraint (4), we define the reduced cost π_{il} in our problem as follows:

$$\pi_{il} = c_{il} - \gamma_i - \sum_{m\in G} \sum_{i\in N} \sum_{l\in F_m} \lambda_{jk} a_{ilmjk} \tag{6}$$

--π_{il} is the reduced cost of column l for nurse i
--c_{il} is the cost coefficient of column l for nurse i
--γ_i is the dual value of constraint (3) for nurse i
--λ_{jk} is the dual value of constraint (2) for shift k on day j
--a_{ilmjk} corresponds to the coefficients matrix in (2) of the master problem **(**NRPs MP**)**

In (6), γ_i and λ_{jk} are the dual values obtained from the linear program solution of the master problem (NRPs MP). c_{il} and a_{ilmjk} are obtained from the solution of the pricing subproblem. Each schedule l for nurse i is a sequence of shifts that satisfies all the related constraints ($l \in F_m$). It introduces a new column in the master problem with cost coefficient c_{il} (we use the term “cost” in the CP pricing subproblem which corresponds to “penalty” in the master problem).

The modeling of feasibility constraint (5) in our CP-CG requires careful consideration of the complex constraints in NRPs. The issue of how to efficiently formulate the complex constraints in CP has been studied in our previous work [25]. In constraint satisfaction problems, a constraint is defined as a logical relation among several variables which restricts the possible values that variables can simultaneously take from their domains. In NRPs, we could define working regulations, etc. by using simple primitive constraints. For example, “if s_{ij} = late, $s_{ij+1} \neq$ early” defines that no early shift is allowed after a late shift, where s_{ij} is a decision variable with finite domain. However, this will lead to a massive number of primitive constraints in complex NRPs. What’s more, one of the essential ideas in CP is propagation, where inconsistent values to variables are detected as early as possible. Using only the primitive constraints significantly restricts the power of propagation in CP [26].

2.4.1 Global constraints

In this work we use global constraints and soft versions of some global constraints to model some of the constraints in our NRPs. Global constraint is a substitute of a set of primitive constraints and is usually equipped with efficient propagation algorithms to remove inconsistent values of variables (see a list of global constraints with propagation algorithms in [27]).

Global cardinality constraint gcc(s, v, l, u), also named as *distribute* (see [27] pages 420-450), bounds the number of times, in range [l, u], certain values v can be taken by variables s. For example, $gcc(s_{ij}, Night, 0, 3)$, $j=1\ldots n$, defines that nurse i should work at most 3 night shifts in the whole period j.

Another global constraint *stretch*(s, v, l, u, P) (see [27] pages 420-450) bounds [l, u] the number of *consecutive* days (i.e. stretch) a nurse can work on a shift v, and restricts the shift v' that immediately follows the stretch, i.e. the pair of values (v, v') must be in pattern P. A *stretch* is a sequence of consecutive variables that take the same value. For example, $stretch(s_{ij}, Night, 2, 3, P)$, $P = \{(Night, Off)\}$, $j=1\ldots n$, restricts nurse i to have consecutive night shifts within length [2, 3], and the only shift type allowed following the night shifts is Off, as given in P.

When dealing with nurse preferences in our NRPs, *soft* constraints are used. The objective is to minimize violations of these constraints by using the associated violation measure. We use ~ hereinafter to denote the extended soft version of the above global constraints to handle soft constraint violations.

The violation measure $\mu^{\tilde{}gcc}$ of ~$gcc(s, v, l, u)$ calculates the deviations of shift assignments from the lower bound l and the upper bound u as in [28]. The penalty of such soft constraint violations is defined as the weighted sum $w\mu^{\tilde{}gcc}$ of the deviations. For example, if nurse i prefers to work on Day shifts within the range [4, 5] per week, represented by ~$gcc(s_{ij}, Day, 4, 5)$, $j = 1\ldots7$, and a schedule below or over this range leads to a penalty of weight 100, then for the schedule of a week $l = [Day, Day, Day, Off, Off, Off, Off]$, a penalty can be calculated as $w\mu^{\tilde{}gcc}(l) = 100 \times |4 - 3| = 100$, where $|s_{ij} = Day| = 3$, $l = 4$, $u = 5$. The violation measure $\mu^{\tilde{}gcc}$ for ~$gcc(s_{ij}, Day, 4, 5)$, $j = 1\ldots7$ is 1 for nurse i.

The deviations as calculated in ~*gcc* are also used to measure the violations of ~*stretch* constraint. To indicate the starting point of a given sequence auxiliary variables are introduced. Although this leads more variables, the resulting constraints are linear and easy to solve in CP. Our experiments have demonstrated a satisfactory performance of the search.

For the *gcc* and *stretch* constraints, we can achieve general arc consistence using the filtering algorithm (i.e. propagation) in CP. However, for ~*gcc* and ~*stretch*, the time spent in filtering is highly dependent on the specific propagation algorithms. In our work, instead of implementing the cost propagation algorithms for soft constraints, we focus on the search strategies, namely a cost threshold and Depth Bounded Discrepancy Search, to improve the efficiency of CP within CP-CG.

2.4.2. The CP model

Based on the primitive and (soft) global constraints, we model our CP pricing subproblem within CP-CG by defining the detailed feasibility constraint (5) $\alpha \in F$. The constraints we model here are from one of the most complex benchmark NRPs. Other problems can be modeled in a similar way.

Parameters:

--h_m: available working hours for nurses of category m in the scheduling period;
--n: number of days in the scheduling period;

Decision variables in the pricing subproblem:

--s_{ij}: decision variables with finite domain S. i.e. $D(s_{ij}) = S$.

The NRPs concerned are defined as follows:

$$\text{(NRPs P)} \quad c_{il} = E(l) = \sum_{Ce \in C} w_{Ce} \mu^{Ce}(l)$$

where $E(l)$ represents the evaluation of schedule l. C is the set of constraints. The evaluation of the soft constraint C_e is calculated by $w_{Ce}\mu^{Ce}(l)$, where w_{Ce} is the weight of the constraint given in Appendix A and $\mu^{Ce}(l)$ is the violation measure of the soft constraint. Based on the above defined (soft) global constraints, the hard and soft constraints are modeled as shown in Appendix B.

In our CP-CG approach, we concern two different types of decision variables, namely the binary variables in the integer program model and finite domain variables in the CP model. In the implementation of the approach, a set of communication variables has been introduced to reflect the interactions between the two models:

--x_{imjk}: binary variables as the communication variables between the master problem and the pricing subproblem. $x_{imjk} = 1$ if nurse i in category m is assigned to shift k on day j; $x_{imjk} = 0$ otherwise. For example, if $s_{ij} = k$, where nurse i is in category m, then $x_{imjk} = 1$.

After (NRPs P) is solved and the values of c_{il} and s_{ij} have been obtained, the communication variables x_{imjk} transform s_{ij} in the pricing subproblem to the coefficient a_{ilmjk} in the master problem. For example, a schedule l = [Day, Day, Night, Night, Off, Off, Day] can be transferred as a column $[110000100110000000110]^T$. Values of c_{il} and a_{ilmjk} are used to calculate the reduced cost π_{il} in (6).

3 The Solution Procedure

Fig. 1 illustrates the general CG procedure for a linear program. Theoretically, it is necessary to generate all possible columns with negative reduced cost before the procedure terminates, i.e. $\{\alpha^{(1)}, \ldots, \alpha^{(k)}\} = \varnothing$. However, in practice the procedure is usually terminated when some conditions are met, i.e. a total number of iterations or time limit. The master problem is then seen as being solved [21, 22].

```
Algorithm 1. Column Generation for linear program
$\tilde{A}$: the subset of feasible columns of $A$
$\lambda$: dual values
$\{\alpha^{(1)}, \ldots, \alpha^{(k)}\}$: columns with negative reduced cost
$\tilde{A}$ := obtain initial columns
Repeat
    $\lambda$ := solve the restricted master problem $\tilde{A}$ to obtain dual values $\lambda$
    $\{\alpha^{(1)}, \ldots, \alpha^{(k)}\}$ := solve the pricing subproblem based on $\lambda$ to obtain columns with
    negative reduced costs
    add columns $\{\alpha^{(1)}, \ldots, \alpha^{(k)}\}$ to matrix $\tilde{A}$
Until ($\{\alpha^{(1)}, \ldots, \alpha^{(k)}\} = \varnothing$) or the termination condition is met
```

Fig. 1. The column generation method for linear program [16]

The variables of the linear program in Fig. 1 are continuous. The solution obtained is the linear program relaxation solution, and quite often is not valid when solving an integer program. In practice, two general techniques are usually used to generate integer solutions for the master problem. Branch-and-bound (B&B) enumerates all candidate solutions in the search tree and prunes worse solutions using the upper and/or lower bounds obtained. It can be used to produce feasible integer solutions to the restricted master problem with the current columns, although the optimality is not guaranteed. Another technique, Branch-and-Price [21], generates columns at each node of the search tree after branching to find the optimal solution.

The overall CP-CG procedure is illustrated in Fig. 2. A feasible initial solution is firstly generated and fed into the restricted master problem (NRPs MP). Each column in the initial solution is associated with a cost $c_{il} = E(l) = \sum_{Ce \in C} w_{Ce} \mu^{Ce}$, and the highest cost is used as the threshold in the CP pricing subproblem. Candidate columns which preserve feasibility and have a cost below the cost threshold are generated by solveCPSubProblem($\tilde{c}$, *DDS*) using the Depth Bounded Discrepancy Search (DDS) strategy [29]. Columns with negative reduced costs are then priced out and added to the restricted master problem to solve the linear program problem again. The new lower bound of the linear program relaxation and the new dual value λ are derived in the linear program, as shown in the inner loop in Fig. 2.

Within the outer loop in Fig. 2, B&B is run based on the generated columns to obtain an integer solution y_{il}. We apply the 0/1 branching strategy on variable y_{il} of the master problem to assign a particular schedule l (y_{il} = 1) or prohibit the specific assignment (y_{il} = 0) to a nurse i. In CPLEX, this default branching rule automatically chooses which variable to branch first. The solution serves as the upper bound of the master problem. The cost threshold is then updated as the highest cost of all columns in the current solution. The column pool is then emptied and DDS restarts to generate columns with a tightened bound. This bound tightening strategy aims to avoid generating columns of high costs. If the cost threshold remains the same, parameters of the DDS search will be adjusted to direct the search to find columns in different parts of the search tree, see Section 3.2. The whole procedure stops when no improvement of $c_{il}y_{il}$ can be obtained in a certain number of iterations. An integer solution with a certain gap to the linear program relaxation is obtained.

```
**Algorithm 2. CP based Column Generation Approach**
$\tilde{A}$ : the subset of feasible columns of $A$
$\lambda$ : dual values
{$\alpha^{(1)}$, …, $\alpha^{(k)}$}: columns generated by CP
$\tilde{c}$ : cost threshold of columns

$\tilde{A}$ := get initial columns // see Section 3.1
$\tilde{c}$ := initial cost threshold
Repeat
    Repeat
        $A_p$ := empty the column pool
        $\lambda$ := solve the RMP $\tilde{A}$
        $A_p$ := {$\alpha^{(1)}$, …, $\alpha^{(k)}$} = solveCPSubProblem($\tilde{c}$ , DDS) // see Section 3.2
        price columns with negative reduced costs from $A_p$ and add them to $\tilde{A}$
    Until stop condition is met (i.e. a pre-defined number of iterations)
    $y_{il}$ = solve the integer solution of MP $\tilde{A}$ using B&B
    update the cost threshold $\tilde{c}$ // the bound tightening strategy, see Section 3.3
Until termination condition is met (without improvement of $c_{il}y_{il}$)
```

Fig. 2. The CP based column generation solution procedure

In the literature, integer solutions to the master problem are obtained by either running B&B on the linear program relaxation or running Branch-and-Price. In our CP-CG, instead of running Branch-and-Price at each node to derive optimal solutions, we run B&B on the generated diverse columns which are of *good quality* by using search strategies in CP to derive integer solutions to the master problem.

To illustrate what are columns of good structure, i.e. diverse columns, assume that we already have a set of columns as follows:

l_1 = [Day, Day, Night, Night, Off, Off, Day]
l_2 = [Day, Day, Day, Day, Off, Off, Off]
l_3 = [Day, Day, Day, Off, Off, Off, Off]
l_4 = [Day, Day, Day, Day, Day, Off, Off]

The master problem (NRPs MP) chooses exactly one column for each nurse to construct a whole roster which satisfies the coverage constraint, for example, 3 Day shifts and 1 Night shift should be assigned on the first day. All the current columns have a Day shift on the first day, so they are not diverse columns. To satisfy the coverage constraint, new diverse columns with Night shift on the first day (of course also with negative reduced cost) are expected.

3.1 Initial solutions

In our previous work [25], where CP is used to solve complex NRPs, it has been observed that finding initial feasible solutions with respect to all hard and soft constraints is very time consuming. This is due to that propagation upon soft constraints, compared with that on hard constraints, is very inefficient [30]. Specific propagation algorithms have thus been designed in the literature.

In our CP-CG approach, instead of specific propagation algorithms, an indirect and simple heuristic relaxation is used to quickly construct initial solutions to enter the restricted master problem, see Fig. 3. Firstly all soft constraints are treated crisply as hard constraints, which will obviously lead to no feasible solutions for the highly constrained problem concerned in our work. Then the soft constraint with the least weight is relaxed one by one until a feasible solution can be found. This method has shown to be very efficient due to the powerful propagation in CP to find feasible solutions.

Algorithm 3. Initial solution generation

```
1. Add all soft constraints and all the hard constraints into the constraint set $C$;
2. Solve the problem $P(X, D, C)$ as a constraint satisfaction problem by CP;
   If no feasible solution for $P(X, D, C)$
       Then remove the soft constraint with the least weight from $C$; go to step 2;
   Else return the feasible solution.
```

Fig. 3. Initial solution generation

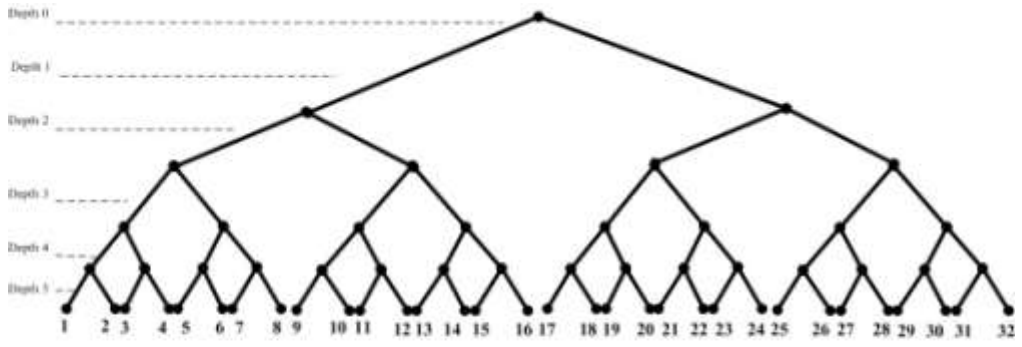
3.2 Depth Bounded Discrepancy Search to obtain diverse columns

Depth First Search (DFS) [31] is a standard search strategy in CP. It searches down to the leaf of one branch before starting another branch in the search tree. Whenever a dead-end of a branch with no solution is reached, the search goes back to an upper depth of the search tree, i.e. backtracking, and continues to another branch. The main drawback of DFS is that, given a limited computational time, even for problems of moderate size, it can only explore a very small part of the search tree, returning very similar solutions (with only the last several variables taking different values) [31, 32]. We apply Depth Bounded Discrepancy Search in our CP-CG to obtain diverse columns (solutions in different parts of the tree).

Depth Bounded Discrepancy Search (DDS) [29] is an alternative search strategy that iteratively explores the search tree based on the innovative idea of *discrepancy*. A discrepancy is "any decision point in a search tree where we go against the heuristic" [33]. Assume a heuristic orders the branches in a left-to-right manner by estimating which branch is more likely to contain solutions, taking the left branch is to follow the heuristic, and taking the right branch is a discrepancy. DDS explores the search tree in a series of DFS, where discrepancies (i.e. search goes to the right branches) happen at early stage of the search. As we know heuristics tend to be less informative and make more mistakes near the root of the search tree [29]. Discrepancies at early stage provide chances of exploring diverse and promising branches.

DDS explores the search tree iteratively in d iterations of DFS, d is the depth of the tree. In iteration i, $i = 0,\ldots,(d–1)$, all discrepancies must happen at and above depth i. That is, at depth i of the tree, the search must take the right branch (discrepancy) of the node. At depth above i, the search explores both the right and left branches. Below depth i, the search must take the left branch at all nodes (i.e. follow the heuristic, no discrepancy is allowed in the later stage). The latest discrepancy in DDS happens at depth i in the tree, i.e. bounded at depth *i* in iteration *i*. This controls (forces) the search to traverse to different parts of the tree, resulting into more diverse solutions compared to the standard DFS [32-34].

Fig. 4 illustrates how DDS traverses the tree. Without loss of generality, the binary tree represents assignments of the simplified variables s_j, $j = 1,\ldots,5$, the domain of s_j is (Day, Off), and the left and right branches take the value Day and Off, respectively. In the first iteration of DDS, depth $i = 0$, so no discrepancy happens. The search takes the left branch at all nodes, leading to path (1) DDDDD. In the second iteration, depth $i = 1$, DDS obtains path (17) ODDDD by taking the discrepancy at depth 1 (i.e. takes the value Off at the right branch). In the third iteration, the discrepancy must happen at depth $i = 2$, thus leading to paths (9) DODDD and (25) OODDD. It can be seen that the paths explored by DDS (illustrated at the bottom of Fig. 4) lead to diverse assignments, i.e. first variables in the assignments also take different values.



depth	path
0	1
1	17
2	9 25
3	5 13 21 29
4	3 7 11 15 19 23 27 31
5	2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32

Fig. 4. The Depth Bounded Discrepancy Search

In the default settings in ILOG Solver, based on the standard Depth Bounded Discrepancy Search (DDS) [29], an extended DDS search is defined by introducing three additional parameters (*depth, width, Max Discrepancy*) as follows. The first parameter *depth* restricts the depths at which the search explores to be between *depth*×*i* and *depth*×(*i*+1) in iteration *i*. That is, in the first iteration, $i = 0$, the search explores the nodes at depths above *depth*. In the second iteration, $i = 1$, it explores the nodes between *depth* and *depth*×2, and so on. The second parameter *width* is used to restrict the number of paths explored by limiting the number of discrepancies occurring between *depth*×*i* and *depth*×(*i*+1). The third parameter, *Max Discrepancy*, restricts the total number of discrepancies by defining the total number of times that the search is forced (diversified) to different parts of the tree.

3.3 Pricing subproblem with threshold

In our CP-CG approach, it is easy to generate feasible candidate columns due to the efficient constraint handling in CP. However, most of the huge number of columns are of poor quality with high cost, and are not helpful to reduce the objective function value in CG. The issue of selecting "good" candidate columns to reduce the computational time in CG has been first discussed in [35]. In [36], for the maximization problem concerned, columns with reduced cost of zero have been used with fast improvement algorithms to construct high quality solutions.

To eliminate poor columns in CP-CG, we introduce an additional cost bounding constraint, threshold $\tilde{c}$, to the pricing subproblem. The enhanced model for the subproblem based on (NRPs P) is presented as follows:

$$\text{(En NRPs P)}: \begin{array}{l} l \in F \\ \pi_i < 0 \\ c_i < \tilde{c} \end{array}$$

where l represents a feasible schedule and the reduced cost π_i is calculated by equation (6).

This (En NRPs P) model is solved by CP as a constraint satisfaction problem to seek feasible solutions as candidate columns. In the research of constraint optimization problems (or constraint satisfaction problems with soft constraints), cost filtering algorithms associated with (global) constraints have been used for each soft constraint in the model [19, 30, 38] to reduce the domain of both *cost variables* and *decision variables* (we refer to Focacci's work [37] for further discussions). However, there is a tradeoff between the time needed and the achieved efficiency of the algorithm.

In our work, instead of using cost filtering algorithms on each soft constraint to filter the domain of the cost variable c_{il}, we use the cost (i.e. the violation measure of the soft constraint) in the bound tightening strategy to filter the domain of decision variables s_{ij} (see Fig. 2). The filtering rule is: if the cost of a sequence of assignments at node q is greater than the upper bound (i.e. c_{il} >= the cost threshold $\tilde{c}$), we remove this value from the domain of the assignments, i.e. node q is pruned. We should note that although this cost threshold encourages the generation of columns with good reduced cost, it may lead to bad (large) cost coefficient in the objective function. The bound tightening strategy adaptively tightens the bound. At the beginning of the procedure, the cost threshold is set at a relatively high value thus columns with large costs also have the chance to enter the master problem. By adaptively tightening the cost threshold, the search gradually accepts better columns with smaller cost.

The bound tightening strategy, working with feasibility pruning in CP, can help to accelerate the search. The search in the tree (i.e. the generation of columns) is thus controlled by both the search strategy DDS (with its parameters) and the adaptive upper bound of the columns (the cost threshold) to obtain good quality columns.

4 Computational Experiments

4.1 Problems and algorithm settings

We demonstrate the effectiveness of our CP-CG approach on several benchmark NRPs at http://www.cs.nott.ac.uk/~tec/NRP. The site presents a range of problems collected from practice and scientific publications. The rules, regulations and objectives have been directly taken from real-world hospital wards and preserved with essential characteristics, see Table 2. In the existing literature, to our knowledge, there are no general classifications on nurse rostering problems. Problems at the site are modeled in a unified XML template to offer a flexible format so that various constraints and rules in a range of real world scenarios can be easily handled and formulated. In the literature, most of the published academic papers have tested a single problem, and no published work has tested all or more than three problems from this site. Evaluating our approach over all the problems is a tedious task, where the same process of modeling different constraints has to be repeatedly carried out.

These three problems are the most complex and have been mostly tested, thus have become benchmark problems during the years. Considering them also makes it possible to compare our approach to a number of existing approaches in the literature on the same problems (see Table 9).

Table 2 Characteristics of the chosen benchmark NRPs

Problems	**No. of Nurses**	**No. of Shift types**	**Scheduling Period (days)**
Gpost	8	3	28
Valouxis	16	4	28
ORTEC	16	5	28,30,31

It is important to note that the difficulty of problems depends not only on the number of shift types, the number of nurses and the length of the scheduling period, but also on the complex constraints involved (see all the constraints in Appendix A). In Table 2, the problem ORTEC is the largest and also the most difficult, where 12 instances (of 12 months) have been widely tested by a number of approaches in the literature. This serves a perfect testbed to evaluate the effects of the search strategies in CP and the column management in our CP-CG approach. The other two simpler problems, although of relatively smaller size, are highly constrained, and are used to tune our CP-CG approach and provide insight of the effects of different components in the CP-CG approach.

All experiments have been carried out on an Intel(R) Core(TM) 2CPU 1.86GHz machine with 1.97GB memory. The hybrid CP-CG approach is implemented in C++, linking ILOG CPLEX 10.0 to solve the linear program and the integer program, and ILOG Solver 6.2 to solve the CP pricing subproblem as well as to provide initial solutions. Default parameters are used in all the CPLEX software packages unless otherwise stated. The parameter settings in the CP-CG approach for all problems are given in Table 3. The total computational time is set as one hour, the same as that in other existing methods in the literature (see Table 9). Other parameters are set based on observations of the algorithm performance on the two smaller problems Gpost and Valouxis in a number of initial tests. To control the size of the master problem solved by B&B in CG, the maximum number of columns evaluated is set as 10,000.

Table 3 Parameter settings for the CP-CG approach

Parameters	**Values**
Total CPU time limit	1 hour
Maximum CPU time for CP solver per iteration	60(sec)
Maximum number of iterations	50
Maximum number of columns in CG	10,000

4.2 Performance of initialisation methods and variable/value ordering

We first evaluate our initialization method in terms of solution quality and computational time, results shown in Table 4. All the problems tested are over-constrained, and no solution can be found if all constraints are imposed crisply. By relaxing the least important soft constraints one by one (see Section 3.1), a feasible solution can be obtained quickly. Due to the strong propagation in CP to detect infeasibilities, this initialisation method is very efficient (computational time is close to zero seconds).

In CP, the efficiency of search also depends on variable and value ordering. Table 5 presents the evaluation of six basic variable ordering heuristics for the problem Gpost. The number of choice points and fails encountered during the search indicate that the MinSizeInt and MinMaxInt heuristics perform the best, with no statistically significant differences between them. The MinSizeInt heuristic is randomly picked and used in the following DFS and DDS search procedures.

Table 4 Results from the initialization method. For the problem Gpost, "-" indicates feasible solutions have been found so there is no need to relax more soft constraints.

Problem	**All constraints**		**Relax $w(SoftCon) \leq 10$**		**Relax $w(SoftCon) \leq 40$**	
	obj	CPU(sec)	obj	CPU(sec)	obj	CPU(sec)
Gpost	infeasible	0	18	50	-	-
Valouxis	infeasible	0	infeasible	0	1120	65
ORTEC	infeasible	0	infeasible	0	1686	112

The value ordering heuristic we applied is *night shift first*, which has been tested in our previous work [25]. The night shift is the most important shift in the problems, due to the fact that it is involved in a number of hard constraints (H5, H7, and H9) and soft constraints (S2, S3) with high costs of 1000. It is also observed from experienced administrative nurses in the hospitals that the night shift is the most complicated shift and is more likely to cause conflicts. Therefore in value ordering, night shifts are assigned first before the other shifts are randomly selected and assigned to the nurses.

Table 5 Evaluation of six variable ordering heuristics for the problem Gpost in Table 2.

Heuristics	**No. of choice points**	**No. of fails**	**CPU (sec)**	**Variable ordering strategies**
MinSizeInt	8966	7995	1.3	the smallest domain first
MaxSizeInt	10706	9723	1.5	the largest domain first
MinMinInt	10703	9720	1.5	the least minimal bound first
MaxMinInt	11978	10995	1.8	the greatest minimal bound first
MinMaxInt	9290	8319	1.2	the least maximal bound first
MaxMaxInt	126003	11620	1.8	the greatest maximal bound first

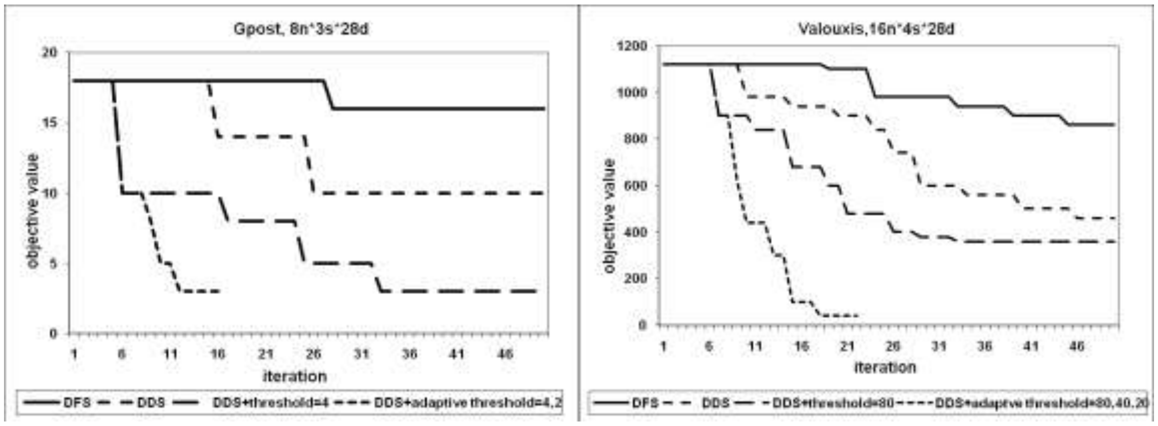
4.3 Performance of search strategies in CP-CG

First, we comment on the number of columns processed by DDS with and without the cost threshold in CP-CG, see Table 6. The settings of these threshold values for different problems are based on the weights of the soft constraints shown in Appendix A. Without the cost threshold, a large number of columns with very large cost can be generated. However, a majority of these columns makes no contribution to the restricted master problem. The last column in Table 6 demonstrates that, a large number of columns with costs above the threshold can be discarded by using the cost threshold to provide good columns at each iteration of CG.

Fig. 5 presents the decrease of objective function value over the iterations of CG. It can also be clearly seen from the faster convergence of DDS with the cost threshold, that "good" columns make the real contribution to the search procedure. Since the maximum CPU time per iteration in the CP solver is set as 60 seconds, shown in Table 3, Fig.5 can also indicate the relationship between the convergence of the objective value and the CPU time.

Table 6 Number of columns processed by DDS with and without cost thresholds in CP-CG. "up to limit" indicates that the search stopped after reaching the maximum number of 10000 columns, as shown in Table 3.

Problem	**Without threshold**	**With adaptive threshold**		
	no. of columns added into RMP	threshold values	no. of columns added into RMP	no. of columns discarded
Gpost	5862	2	2567	7433
Valouxis	8562	40	3860	6140
ORTEC	up to limit	100	4586	5414



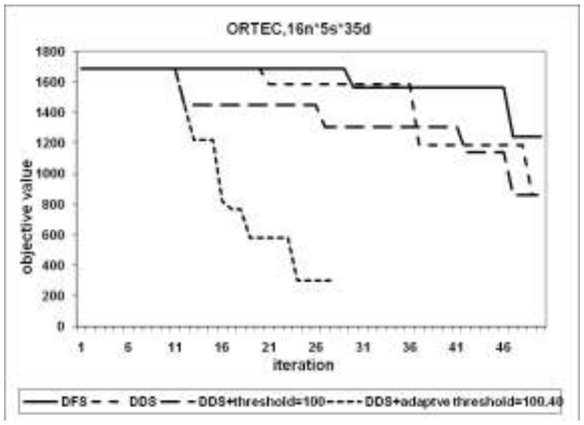


Fig. 5. The decrease of objective function value over iterations of CP-CG with different search strategies for the three problems. “16n*5s*35d” denotes a problem with 16 nurses, 5 shift types and 35 days.

Next we comment on the decrease of objective value in CP-CG with four search strategies, namely DFS, DDS, DDS with static threshold and DDS with adaptive threshold, shown in Fig. 5. Compared with DFS and pure DDS, the convergence of the objective value from DDS with static cost threshold is faster, i.e. continuously decreased. DDS is further improved by the bound tightening strategy which adaptively updates the threshold according to the information collected during the search. With the adaptive cost threshold, fewer columns are processed, so less iterations of CG are executed compared with DFS and pure DDS.

To provide an in-depth analysis on the search strategies in CP-CG concerning the lower bound of linear program and the integer solution obtained after B&B, Table 7 compares numerical results of DFS, DDS and DDS with the adaptive threshold. For each strategy, objective values of solutions after the initialization, linear program relaxation and B&B are presented. In terms of the linear program lower bound, DDS with the adaptive cost threshold makes the most improvement to initial solutions, although both DFS and DDS strategies can also improve the initial solutions to a certain scope. After B&B, for the small problem Gpost, integer solutions can be found by all three strategies, and the optimal solution can only be found by using DDS with adaptive cost threshold. For larger problems, integer solutions can only be obtained by DDS and DDS with adaptive cost threshold within the time limit, of which the latter obtained much better results for both problems.

Table 7 Numerical results of CP-CG with DFS, DDS and DDS + adaptive cost threshold. Optimal results are shown in bold. Avg CPU(sec): the average time of a single iteration of CG by CP; Z_{IN}: the objective value of the initial solution; Z_{LP}: the objective value of the best solution of the LP relaxation of the master problem obtained at the end of the CG procedure; Z_{IP}: the objective value of the best integer solution obtained after applying B&B on generated columns.

Problem	**Strategy**	Z_{IN}	Z_{LP}	Z_{IP}	**Avg CPU(sec)**
Gpost	DFS	18	16	16	5.62
	DDS	18	10	14	3.58
	DDS + adaptive cost threshold	18	3	**3**	3.68
Valouxis	DFS	1120	860	--	8.21
	DDS	1120	460	540	4.20
	DDS + adaptive cost threshold	1120	40	60	4.28
ORTEC	DFS	1686	1240	--	18.75
	DDS	1686	860	--	9.24
	DDS + adaptive cost threshold	1686	300	401	8.78

As we mentioned above, to obtain integer solutions to the master problem, we only run B&B on the generated columns at the root node, thus the obtained integer solution may not be optimal. To provide a measure of such loss of optimality, in Table 8, we compare Z_{LP} and Z_{IP} in Table 7 with the best lower bound Z_{LB} obtained in the literature.

Table 8 The gap between Z_{LP} and Z_{IP} in Table 7 to the best lower bound Z_{LB} in the literature, respectively. "--" denotes that the values are not available.

	Z_{LP}	Z_{LB}	gap	Z_{IP}	Z_{LB}	gap
Gpost	3	3	0%	3	3	0%
Valouxis	40	--	--	60	--	--
ORTEC	300	270	11%	401	270	33%

In Table 8, for both problem instances Gpost and ORTEC, the best lower bound Z_{LB} is obtained by solving the linear relaxation of the instances in the linear program solver CPLEX (see more details at http://www.cs.nott.ac.uk/~tec/NRP). The integrality constraint is relaxed. Hence Z_{LB} is not the result of an integer solution but the solution to the continuous relaxation. Our lower bound Z_{LP} is computed by the algorithm presented in Fig. 2, where a subset of negative reduced cost columns is generated. Our Z_{IP} is obtained by applying B&B on the generated columns. Z_{LP}, Z_{IP} and Z_{LB} are all obtained by considering the same model of the problem instances. Therefore, the gaps between their values are comparable.

The gaps in Table 8 give an idea of the loss of the optimality by applying our CP-CG approach. They are relatively small, indicating the good quality of our lower bounds. We should note that Z_{LB} is not obtained by running a Branch-and-Price algorithm which guarantees the optimality of the solution. It is obtained by running B&B on the relaxed problem instances, thus may not represent the real optimal objective value for the problem. In the literature Branch-and-Price has not been investigated for these datasets yet.

4.4 CP-CG compared with existing approaches in the literature

We finally evaluate our CP-CG approach on the most constrained problem, ORTEC in Table 2. We compare CP-CG with those existing approaches in the literature on 12 instances of the problem in Table 9. In all the other approaches in Table 9, meta-heuristic algorithms (e.g. genetic algorithms and variable neighbourhood search) have either been delicately designed using domain knowledge to solve the problem [39, 40], or played an important role in the procedure to improve the solutions obtained by integer programming or CP [25, 41]. It is interesting to see that both the hybrid integer programming [41] and hybrid CP [25] approaches have employed variable neighbourhood search to the solutions obtained from exact mathematical methods in a sequential manner. Our CP-CG approach does not employ any advanced meta-heuristic algorithm to solutions, but embeds heuristics in DDS more closely within the CG procedure.

Table 9 Existing approaches to the ORTEC benchmark in the literature, best results in bold. "p.d.": the percentage deviation from the best result. The threshold $\tilde{c} = 40$ in CP-CG. Hybrid GA: a genetic algorithm hybridised with local search, developed in the commercial software Harmony^{TM} at ORTEC, and compared with the hybrid variable neighbourhood search (Hybrid VNS) in [40]. Hybrid IP: IP solutions improved by a VNS. Hybrid CP: a CP approach followed by a VNS. Original results of Hybrid CP in [25] were obtained within 0.5 hours. We present the results of Hybrid CP from one hour for a fair comparison.

ORTEC Problem instances	**Hybrid GA [39] (1 hour)**		**Hybrid VNS [40] (1 hour)**		**Hybrid IP [41] (1 hour)**		**Hybrid CP [25] (1 hour)**		**CP-CG (1 hour)**	
	obj	**p.d.**	**obj**	**p.d.**	**obj**	**p.d.**	**obj**	**p.d**	**obj**	**p.d.**
Jan	775	157%	735	144%	460	53%	595	98%	**301**	**0%**
Feb	1791	42%	1866	48%	1526	21%	1721	36%	**1261**	**0%**
Mar	2030	19%	2010	17%	**1713**	**0%**	2706	58%	1975	15%
Apr	612	57%	457	17%	**391**	**0%**	915	134%	621	59%
May	2296	29%	2161	21%	2090	17%	**1786**	**0%**	1941	8%
Jun	9466	52%	9291	49%	8826	42%	8700	40%	**6231**	**0%**

Jul	781	84%	481	13%	**425**	**0%**	586	38%	751	77%
Aug	4850	123%	4880	125%	3488	61%	**2171**	**0%**	**2171**	**0%**
Sep	615	86%	647	96%	**330**	**0%**	1215	268%	401	22%
Oct	736	145%	665	121%	445	48%	616	105%	**301**	**0%**
Nov	2126	34%	2030	28%	1613	1%	1605	1%	**1590**	**0%**
Dec	625	54%	520	28%	**405**	**0%**	**405**	0%	450	11%

Within the same computational effort, results obtained by CP-CG are highly competitive (obtained the best solutions in the literature for 6 out of 12 instances), especially concerning that CP-CG is built upon the complete model formulating all constraints, and thus the quality of solutions is warranted, i.e. not relying on meta-heuristics which may return different results from different runs. In our CP-CG approach, no meta-heuristic improvement algorithms have been applied afterwards. The idea is to provide a clear indication of the guaranteed effectiveness of the pure CP-CG approach. Hybridizations of our CP-CG approach with meta-heuristics concerning effectiveness and computational efforts will be investigated in our future work, and is out of the scope of this paper.

5 Conclusions

In this paper, we investigate a hybrid column generation approach CP-CG, where constraint programming (CP) is integrated to solve the highly constrained nurse rostering problems. The complex problems have been modeled in a column generation (CG) scheme, where the master problem is formulated as an integer program and the pricing subproblem is modeled as a weighted constraint optimization problem in the CP paradigm.

In the standard CG procedure, the quality of columns is only measured by the reduced cost. Those columns which satisfy constraints of the pricing subproblem enter the restricted master problem. In our CP-CG approach, we propose two strategies to generate high quality diverse columns. A cost threshold has been introduced, and is adaptively updated by a bound tightening strategy during the search to choose those columns of good quality, i.e. only columns with negative reduced cost below the threshold enter the master problem. The Depth Bounded Discrepancy Search (DDS) has been used in CP to produce diverse columns and make real contributions to the convergence of the solution procedure. Computational results have demonstrated that a much smaller number of columns enter the master problem by using DDS compared with standard Depth First Search. Even with these fewer columns from DDS in the pricing subproblem, the objective value in linear relaxation converges faster than applying Depth First Search. Furthermore, the cost threshold also contributes a faster convergence in linear relaxation.

The effectiveness and efficiency of our CP-CG approach have been demonstrated on three benchmark nurse rostering problems with different profiles. The adopted strategies in CP-CG have been justified by comparison results against several existing approaches and the analysis of different approaches on the benchmark nurse rostering problems tested.

The main focus of this work is to design efficient search strategies which speed up the convergence of linear program relaxation while also satisfying the integrality request of the master problem. The Branch-and-Bound technique has been applied at the root node in CP-CG to produce integer solutions with a certain gap to the optimal. Given more computational time, our CP-CG approach may be plugged at each node of the tree to derive optimal integer solutions to the problem, i.e. the Branch-and-Price algorithm. Other future research directions include hybridizations of CP-CG with more advanced search algorithms such as meta-heuristics with neighbourhoods and move strategies designed based on problem specific information, as well as more efficient cost propagations in solving the CP pricing subproblem.

Acknowledgement

This research has been supported by the School of Computer Science at The University of Nottingham,

UK.

References

1. Zurn, P., Dolea, C., and Stilwell, B., *Nurse retention and recruitment: developing a motivated workforce.* Global Nursing Review Initiative, 2005. 4: p. 1-31.
2. Burke, E. K., De Causmaecker, P., Vanden Berghe, G., and Van Landeghem, H., *The State of the Art of Nurse Rostering.* Journal of Scheduling, 2004. 7(6): p. 441- 499.
3. Brucker, P., Qu, R., and Burke, E.K. *Personnel Scheduling: Models and Complexity*. European Journal of Operational Research, 2011. 210(3): 467-473.
4. Morris, J. G. and Showalter, M. J., *Simple approaches to shift, days-off, and tour scheduling programs.* Management Science, 1983. 29: p. 942-950.
5. Billionnet, A., *Integer programming to schedule a hierarchical workforce with variable demands.* European Journal of Operational Research, 1999. 114(1): p. 105 - 114.
6. Beaumont, N., *Scheduling staff using mixed integer programming.* European Journal of Operational Research, 1997. 98(3): p. 473 - 484.
7. Puchinger, J. and Raidl, G. R., *Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification*, in *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*. 2005. p. 41-53.
8. Jaumard, B., Semet, F., and Vovor, T., *A generalized linear programming model for nurse scheduling.* European Journal of Operational Research, 1998. 107: p. 1-18.
9. Bard, J. F. and Purnomo, H. W., *Preference scheduling for nurses using column generation.* European Journal of Operational Research, 2005. 164(2): p. 510 - 534.
10. Beliën, J. and Demeulemeester, E., *A branch-and-price approach for integrating nurse and surgery scheduling.* European Journal of Operational Research, 2008. 189: p. 652-668.
11. Maenhout, B. and Vanhoucke, M., *Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem.* Journal of Scheduling, 2010. 13(1): p. 77-93.
12. Cheng, B. M. W., Lee, J. H. M., and Wu, J. A. C. K., *A nurse rostering system using constraint programming and redundant modelling.* IEEE Transactions on information technology in biomedicine, 1997. 1(1): p. 44- 54.
13. Wonga, G. Y. C. and Chun, A. H. W., *Constraint-based rostering using meta-level reasoning and probability-based ordering.* Engineering Applications of Artificial Intelligence, 2004. 17: p. 599- 610.
14. Junker, U., E. Karisch, S., Kohl, N., Vaaben, B., Fahle, T., and Sellmann, M. *A Framework for Constraint Programming Based Column Generation*. in *Principles and Practice of Constraint Programming*. 1999.261- 275
15. Yunes, T. H., Moura, A. V., and de Souza, C. C. *Solving very large crew scheduling problems to optimality*. in *ACM symposium on Applied computing*. 2000.446- 451
16. Fahle, T., Junker, U., Karisch, S., Kohl, N., Sellmann, M., and Vaaben, B., *Constraint programming based column generation for crew assignment.* Journal of Heuristics, 2002. 8: p. 59–81.
17. Rousseau, L. M., Gendreau, M., Pesant, G., and Focacci, F., *Solving VRPTWs with constraint programming based column generation.* Annals of Operations Research, 2004. 13(1): p. 199–216.
18. Pisinger, D. and Sigurd, M., *Using Decomposition Techniques and Constraint Programming for Solving the Two-Dimensional Bin-Packing Problem.* INFORMS J. on Computing, 2007. 19(1): p. 36- 51.
19. Demassey, S., Pesant, G., and Rousseau, L.-M., *A Cost-Regular Based Hybrid Column Generation Approach.* Constraints, 2006. 11(4): p. 315- 333.
20. Gendron, B., Lebbah, H., and Pesant, G. *Improving the Cooperation Between the Master Problem and the Subproblem in Constraint Programming Based Column Generation*. in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. 2005.217- 227
21. Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H., *Branch-and-price: Column generation for solving huge integer programs.* Operations Research, 1998. 46: p. 316- 329.
22. Lubbecke, M. E. and Desrosiers, J., *Selected topics in column generation.* Operations Research, 2002. 53: p. 1007- 1023.
23. Dantzig, G. B. and Wolfe, P., *Decomposition Principle for Linear Programs.* Operations Research, 1960. 8: p. 101-111.
24. Gilmore, P. C. and Gomory, R. E., *A linear programming approach to the cutting-stock problem.* Operations Research, 1961. 9: p. 849- 859.
25. Qu, R. and He, F. *A Hybrid Constraint Programming Approach for Nurse Rostering Problems*. in *Applications and Innovations in Intelligent Systems XVI*. 2008.211-224
26. van Hoeve, W. J. and Katriel, I., *Global Constraints*, in *Handbook of Constraint Programming*, Rossi, F., Beek, P. van, and T.Walsh, Editors. 2006, Elsevier B.V. p. 169-208.
27. Hooker, J. N., *Integrated Methods for Optimization*. 2007: Springer.
28. van Hoeve, W. J., Pesant, G., and Rousseau, L. M., *On global warming: Flow-based soft global constraints.* Journal of Heuristics, 2006. 12: p. 347- 373.
29. Walsh, T. *Depth-bounded discrepancy search*. in *Proceedings of IJCAI-97* 1997.1388- 1393

30. Focacci, F., Lodi, A., and Milano, M., *Cost-Based Domain Filtering*, in *Principles and Practice of Constraint Programming – CP'99*. 1999. p. 189-203.
31. Apt, K. R., *Principles of Constraint Programming*. 2003: Cambridge University Press.
32. Sellmann, M., Zervoudakis, K., Stamatopoulos, P., and Fahle, T., *Crew Assignment via Constraint Programming: Integrating Column Generation and Heuristic Tree Search.* Annals of Operations Research, 2002. 115(1): p. 207- 225.
33. Harvey, W. D. and Ginsberg, M. L., *Limited Discrepancy Search*, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. 1995. p. 607- 613.
34. Rousseau, L.-M., *Stabilization Issues for Constraint Programming Based Column Generation*, in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. 2004. p. 402- 408.
35. Vanderbeck, F. and Wolsey, L. A., *An exact algorithm for IP column generation.* Operations Research Letters, 1996. 19(4): p. 151-159.
36. Sol, M. and Savelsbergh, M. W. P., 1994. *Column generation techniques for pickup and delivery problems*.
37. Focacci, F., Lodi, A., and Milano, M., *Optimization-Oriented Global Constraints.* Constraints, 2002. 7(3): p. 351-365.
38. Petit, T., Régin, J.-C., and Bessière, C., *Specific Filtering Algorithms for Over-Constrained Problems*, in *Principles and Practice of Constraint Programming — CP 2001*. 2001. p. 451-463.
39. Fijn van Draat, L., *Harmonious personnel scheduling.* Medium Econometrische Toepassingen, 2006. 14: p. 4-7.
40. Burke, E. K., Curtois, T., Post, G., Qu, R., and Veltman, B., *A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem.* European Journal of Operational Research, 2008: p. 330- 341.
41. Burke, E. K., Li, J., and Qu, R., *A Hybrid Model of Integer Programming and Variable Neighbourhood Search for Highly-constrained Rostering Problems.* European Journal of Operational Research, 2009. 203(2): p. 484-493.

Appendix A. Hard and Soft Constraints in the Nurse Rostering Benchmarks

Hard constraints	**Category**	**Details**
Gpost	One shift per day	One shift per day (D, N, O)*
	Coverage (no over/under cover)	Weekday: 3D 1N; Weekend: 3D 1N
	Working time	Full time: 18 shifts; Part time: 10 shifts
	Shift patterns	Maximum consecutive working days: 6
		Maximum consecutive N shifts: 3
		Maximum consecutive working weekends: 3
		After a series of work, at least 2 days off
		Complete weekends, i.e. free or work on both days
		After N shifts, at least 2 days off
Valouxis	One shift on one day	One shift one day (D, E, N, O)*
	Coverage (no over/under cover)	Weekday: 4D 4E 2N; Weekend: 3D 3E 2N
	Working time	18 shifts
	Shift patterns	Maximum consecutive working days: 5
		Maximum consecutive N shifts: 3
		Maximum consecutive working weekends: 3
		After a series of work, at least 2 days off
		Complete weekends, i.e. free or work on both days
		After N shifts, at least 2 days off
ORTEC	One shift one day	One shift on one day (E, L, D, N, O)*
	Coverage (no over/under cover)	Weekday: 3E 3D 3L 1N; Weekend: 2E 2D 2L 1N
	Working time	Group 1: 36 hours/week; Group 2: 32 hours/week; Group 3: 20 hours/week
	Shift pattern	Maximum consecutive working days: 6
		Maximum consecutive N shifts: 3
		Maximum consecutive working weekends: 3
		After a series of work, at least 2 days off
		Complete weekends, i.e. free or work on both days
		After N shift, at least 2 days off

*D: day shift; E: evening shift; L: late shift; N: night shift; O: day off.

Soft constraints	**Category**	**Details**	**Weights**	**Violation measure**
Gpost	Balanced workload	Full time: [4,5] shift/week Part time: [2,3] shift/week	1	*Difference between the no. of shifts received and the acceptable no. of shifts per week
		Full time: shifts series length [4,6]	1	*Difference between the no. of shifts

		Part time: shifts series length [2,3]		received and the acceptable series length
	Pattern preference	No stand alone shift, i.e. single day on	100	Number of isolated shifts
		No one shift over a weekend	100	Number of incomplete weekends
		No one day off between shift series	10	Number of one day off
Valouxis	Balanced workload	No. of D shifts: [5, 8] in the schedule	100	Difference between the no. of shifts received and the acceptable no. of shifts
		No. of E shifts: [5, 8] in the schedule	100	Difference between the no. of shifts received and the acceptable no. of shifts
		No. of N shifts: [2, 5] in the schedule	100	Difference between the no. of shifts received and the acceptable no. of shifts
	Pattern preference	No stand alone shift, i.e. single day on	1000	Number of isolated shifts
		No one shift over a weekend	1000	Number of incomplete weekends
		A D after E should be avoided	1000	Number of D shifts after E shift
		A E after N should be avoided	1000	Number of E shifts after N shift
		A D after N should be avoided	1000	Number of D shifts after N shift
		At least 2 days off between shift series	100	Number of one day off
		Series of D/E/N shift length:3	40	Difference between the series length and the acceptable length
		Series of D/E/N shift length: 3	20	Difference between the series length and the acceptable length
ORTEC	Balanced workload	Group 1: [4,5] shifts/week Group 2: [4,5] shifts/week Group 3: [2,3] shifts/week	10	*Difference between the no. of shifts received and the acceptable no. of shifts per week
		Group1: shift series length [4,6] Group2: shift series length [4,6] Group3: shift series length [2,3]	10	*Difference between the no. of shifts received and the acceptable series length
	Pattern preference	No stand alone shift, i.e. a single day on	1000	Number of isolated shifts
		No one shift at a weekend	1000	Number of incomplete weekends
		Length of a series of N shifts: [2,3]	1000	Difference between the series length and the acceptable length
		At least 2 days off between shift series	100	Number of one day off
		Length of a series of E shifts: [2,3]	10	Difference between the series length and the acceptable length
		Length of a series of L shifts: [2,3]	10	Difference between the series length and the acceptable length
		A E after D should be avoided	5	Number of E shifts after D shift
		A N after E should be avoided	1	Number of N shifts after E shift

* In order to have the same evaluation functions as those of other approaches in the literature, the constraints denoted by * is measured by the quadratic function. That is, the measure of violations is squared and multiplied by the corresponding weight.

Appendix B. The CP Model of Hard and Soft Constraints in the Nurse Rostering Benchmarks

Hard Constraints	
H1	Coverage constraint. A number of different shifts must be covered throughout the scheduling period in order to guarantee the coverage of service. In the CP-CG approach, H1 is presented as (2) in the master problem model (NRPs MP).
H2	For each day, one nurse can only start one shift. H2 is implicitly satisfied by assigning exactly one value to each decision variable.
H3	Within a scheduling period, a nurse is allowed to work at most 4 hours more than his/her available working time. Each shift has 8 hours working time. H3 is thus modeled as $sum(8\times f_{ij}) \le h_m+4$, $j = 1 \ldots n$, where $f_{ij} = \begin{cases} 1, if\ s_{ij} \ne off \\ 0, otherwise \end{cases}$. h_m is the available working hours for a nurse of category m in the scheduling period.
H4	Maximum 36 hours working time per week.

	Each shift corresponds to 8 working hours. H4 is modeled as *sum*($8\times f_{ij}$) ≤ 36, $j = 1....7, j = 8....14, \ldots$ for the corresponding week.
H5	Maximum 3 night shifts in the scheduling period. H5 is modeled as *gcc*(s_{ij}, Night, 0, 3), $j = 1\ldots n$.
H6	At least 2 weekends off in the scheduling period. H6 is modeled as *gcc*(s_{ij}, Off, 2, 5), in conjunction with an If-Then constraint: if s_{ij} = Off, then s_{ij+1} = Off, $j = 6,13,20,27,34$
H7, H8	Following a series of at least 2 consecutive night shifts, a minimum of 42 hours rest is required. At most 3 consecutive night shifts in the scheduling period. H7 and H8 are modeled as a single constraint *stretch*(s_{ij}, Night, 2, 3, *P*), *P* = {(Night,Off)}, $j = 1\ldots n$.
H9	At most 6 consecutive working days. H9 is modeled as *stretch*(s_{ij}, ~Off, 1, 6), $j = 1\ldots n$. Here ~Off represents a day-on shift, and *P* is omitted, representing no restriction on the pattern.
Soft Constraints	
S1	Complete weekend. From Friday 23:00 to Monday 0:00, a nurse should have either no shift or 2 shifts. Violation measure: $\mu_{ij}^{S1} = \begin{cases} 1, if\ s_{ij} \neq s_{ij+1} \\ 0, otherwise \end{cases}$ and $\mu_i^{S1} = \sum_{j=6,13,20,27,34} u_{ij}^{S1}$
S2	Avoid a sequence of shifts of length 1 for all nurses. Violation measure: $\mu_{ij}^{S2} = \begin{cases} 1, if\ s_{ij} = off, s_{ij+1} \neq off, s_{ij+2} = off \\ 0, otherwise \end{cases}$ and $j = 1\ldots n\text{-}3$.
S3	For all nurses, a series of night shifts should be within [2, 3]. It could be part of, but not before, another sequence of shifts. S3 is implicitly satisfied by constraints H7 and H8.
S4	At least 2 days off after a series of day, early or late shifts. S4 is modeled as ~*stretch* (s_{ij}, Off, 2, 5), $j = 1\ldots n$, violation measure is introduced in Section 2.4.1.
S5	For full time nurses, the number of day-on shifts should be within [4, 5] per week. S5 is modeled as ~*gcc*(s_{ij}, ~Off, 4, 5), $j = 1\ldots 7, j = 8....14\ldots$ for the corresponding week. ~Off represents a day-on shift. For part time nurses, the number of shifts should be within [2, 3] per week, modeled as ~*gcc*(s_{ij}, ~Off, 2, 3), $j = 1\ldots 7, j=8....14\ldots$ for the corresponding week.
S6	For full time nurses, the length of a series of shifts should be within [4, 6]. S6 is modeled as ~*stretch*(s_{ij}, ~Off, 4, 6), $j = 1\ldots n$, violation measure is introduced in Section 2.4.1. For part time nurses, the length of a series of shifts should be within [2, 3], modeled as ~*stretch*(s_{ij}, ~Off, 2, 3), $j = 1\ldots n$
S7	For all nurses, the length of a series of early shifts should be within [2, 3]. S7 is modeled as ~*stretch*(s_{ij}, Early, 2, 3), $j = 1\ldots n$
S8	For all nurses, the length of a series of late shifts should be within [2, 3]. S8 is modeled as ~*stretch*(s_{ij}, Late, 2, 3), $j = 1\ldots n$
S9	An Early shift after a Day shift should be avoided. Violation measure: $\mu_{ij}^{S9} = \begin{cases} 1, if\ s_{ij} = Day, s_{ij+1} = Early \\ 0, otherwise \end{cases}$ and $j = 1\ldots n$. Similar constraints include an Early shift after a Late shift should be avoided; a Day shift after a Late shift should be avoided, all can be modeled in a similar way.
S10	A Night shift after an Early shift should be avoided. S10 can be modeled in a similar way as S9.