A Hybrid Integer and Constraint Programming Approach to Solve Nurse Rostering Problems

Erfan Rahimian, Kerem Akartunalı
Dept. of Management Science, University of Strathclyde, Glasgow, G4 0GE, UK,
erfan.rahimian@strath.ac.uk, kerem.akartunali@strath.ac.uk

John Levine
Computer and Information Sciences, University of Strathclyde, Glasgow, G1 1XH, UK, john.levine@strath.ac.uk

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.

Key words: Timetabling; Nurse Rostering; Hybrid Algorithm; Integer Programming; Constraint Programming.

---

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 [30, 25]. 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 [13, 33].

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 [15, 6]. 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 the soft constraint violations. We refer the interested reader to [13] for further details on the NRP, and to [16] 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) [19, 28, 30] and Constraint Programming (CP) [18, 42], which are capable of finding the optimal solution, albeit often resulting in unacceptable computational times. In order to address the computational limitations of these methods, many heuristic methods have been proposed in the literature ranging from rather general Variable Neighbourhood Search [27, 37] and Genetic Algorithms [8, 2] to stochastic approaches [44] and tailor-made heuristics [5]. 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 [23] and Gurobi [20], have focused on using these methods in hybrid settings such as CP and heuristics [43], IP and heuristics [45], and the less well-investigated combination of IP and CP [35], 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 real-world 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 meta-heuristics [41]. Instead, we aim to develop a hybridisation of IP and CP which preserve the 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, the proposed algorithm 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 [10], 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 [10].

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 [14], 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. Section 3 and 4, we formulate this problem using IP and CP, respectively. For more details regarding the problem characteristics, we refer the reader to [10].

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 the 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 will 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 the weighted sum of all the associated auxiliary variables for soft constraints. Next, we present the parameters, variables, and constraints of the IP model:

Sets and Parameters:

E	Set of nurses.
D	Set of days.
A	Set of shift types.
A'	Set of all shift types except day off.
W	Set of weeks.
H_a	Set of shift types that cannot be assigned immediately after shift type $a \in A$.
Q_{ad}	Set of pre-assigned nurses to shift type $a \in A$ on day $d \in D$.
M_e^{min}, M_e^{max}	Minimum and maximum number of shifts that can be assigned to nurse $e \in E$ within the planning period.
W_w^{min}, W_w^{max}	Minimum and maximum number of shifts that can be assigned to a nurse within week $w \in W$.

V_d^{min}, V_d^{max} Minimum and maximum number of shifts that can be assigned to nurses on day $d \in D$.

A^{min}, A^{max} Minimum and maximum number of hours that can be assigned to each nurse during the planning period.

E_w^{min}, E_w^{max} Minimum and maximum number of hours that can be assigned to each nurse during week $w \in W$.

N^{min}, N^{max} Minimum and maximum number of consecutive working days over the planning period.

H_a^{min}, H_a^{max} Minimum and maximum number of consecutive shift type $a \in A$ over the planning period.

K^{min}, K^{max} Minimum and maximum number of worked weekends over the planning horizon.

C^{max} Maximum number of consecutive worked weekends over the planning period.

U_a Total workloads (hours) of shift type $a \in A$ within the planning period.

U_{aw} Total workloads (hours) of shift type $a \in A$ during week $w \in W$.

B_{ew}^{wa} The associated weight with the minimum and maximum number of shift assignments for nurse $e \in E$ within week $w \in W$.

B_{ew}^{cwx} The associated weight with the maximum number of consecutive weekends for nurse $e \in E$ within week $w \in W$.

B_{ead}^{rso} The associated weight with requested shift $a \in A$ within day $d \in D$ for nurse $e \in E$.

Decision variables:

x_{ead} $= 1$ if shift type $a \in A$ on day $d \in D$ is assigned to nurse $e \in E$, $= 0$ otherwise.

p_{ed} $= 1$ if nurse $e \in E$ works on day $d \in D$, $= 0$ otherwise.

k_{ew} $= 1$ if nurse $e \in E$ is assigned to weekend $w \in W$, $= 0$ otherwise.

y_{ea} Total number of times that shift type $a \in A$ is assigned to nurse $e \in E$ over the planning period.

z_{ewa} Total number of shift type $a \in A$ assigned to nurse $e \in E$ during week $w \in W$.

$v_{ew}^{wam}, v_{ew}^{wax}$ Total incurred penalty relevant to the minimum and maximum number of shift assignments for nurse $e \in E$ within week $w \in W$.

v_{ew}^{cwx} Total incurred penalty relevant to maximum number of consecutive weekends for nurse $e \in E$ within week $w \in W$.

v_{ead}^{rso} Total incurred penalty relevant to requested shift $a \in A$ within day $d \in D$ for nurse $e \in E$.

Constraints:

Next, we present the relevant IP constraints in the same order as the order of constraints presented in Section 2:

$$\sum_{a \in A} x_{ead} = 1, \ \ \forall e \in E, d \in D \tag{1}$$

$$\begin{cases} p_{ed} = \sum\limits_{a \in A'} x_{ead}, & \forall e \in E, d \in D \\ V_d^{min} \leq \sum\limits_{e \in E} p_{ed} \leq V_d^{max}, & \forall d \in D \end{cases} \tag{2}$$

$$\begin{cases} y_{ea} = \sum\limits_{d \in D} x_{ead}, & \forall e \in E, a \in A \\ M_e^{min} \leq \sum\limits_{a \in A} y_{ea} \leq M_e^{max}, & \forall e \in E \end{cases} \tag{3a}$$

$$\begin{cases} \sum\limits_{i=1}^{N^{min}-1} p_{e(d+i)} \leq p_{ed} + p_{e(d+N^{min})} & \forall e \in E, d \in \{1 \dots |D| - N^{min}\} \\ + N^{min} - 2, & \\ \sum\limits_{g=d}^{N^{max}+d} p_{eg} \leq N^{max}, & \forall e \in E, d \in \{1 \dots |D| - N^{max}\} \end{cases} \tag{3b}$$

$$\begin{cases} A^{min} \leq \sum\limits_{a \in A} y_{ea} U_a \leq A^{max}, & \forall e \in E \\ z_{ewa} = \sum\limits_{d=7(w-1)+1}^{7w} x_{ead}, & \forall e \in E, a \in A, w \in W \\ E_w^{min} \leq \sum\limits_{a \in A} z_{ewa} U_{aw} \leq E_w^{max}, & \forall e \in E, w \in W \end{cases} \tag{3c}$$

$$W_w^{min} - v_{ew}^{wam} \leq \sum_{a \in A} z_{ewa} \leq W_w^{max} + v_{ew}^{wax}, \ \ \forall e \in E, w \in W \tag{3d}$$

$$\begin{cases} k_{ew} \leq p_{e(7w-1)} + p_{e(7w)} \leq 2k_{ew}, & \forall e \in E, w \in W \\ K^{min} \leq \sum\limits_{w \in W} k_{ew} \leq K^{max}, & \forall e \in E \end{cases} \tag{3e}$$

$$\begin{cases} \sum\limits_{i=1}^{H_a^{min}-1} x_{ea(d+i)} \leq x_{ead} & \forall e \in E, a \in A, d \in \{1 \dots |D| - H_a^{min}\} \\ + x_{ea(d+H_a^{min})} + H_a^{min} - 2, & \\ \sum\limits_{g=d}^{H_a^{max}+d} x_{eag} \leq H_a^{max}, & \forall e \in E, a \in A, d \in \{1 \dots |D| - H_a^{max}\} \end{cases} \tag{3f}$$

$$\begin{cases} x_{end} \leq x_{en(d+1)} + 1 - p_{e(d+1)}, & \forall e \in E, d \in \{1 \dots |D| - 1\} \\ x_{end} - p_{e(d+1)} \leq 1 - p_{e(d+2)}, & \forall e \in E, d \in \{1 \dots |D| - 2\} \end{cases} \tag{4}$$

$$x_{er(7w-1)} = x_{er(7w)}, \ \ \forall e \in E, w \in W \tag{5}$$

$$x_{en(7w-2)} \leq p_{e(7w-1)} + p_{e(7w)}, \ \ \forall e \in E, w \in W \tag{6}$$

$$\sum_{i=0}^{C^{max}} k_{e(w+i)} \leq C^{max} + v_{ew}^{cwx}, \ \ \forall e \in E, w \in \{1 \dots |W| - C^{max}\} \tag{7}$$

$$x_{ead} = 1 - v_{ead}^{rso}, \ \ \forall e \in Q_{ad}, a \in A, d \in D \tag{8}$$

$$x_{ead} + x_{eh(d+1)} \leq 1, \ \ \forall e \in E, a \in A, h \in H_a, d \in \{1 \dots |D| - 1\} \tag{9}$$

$$x_{ead}, p_{ed}, k_{ew} \in \{0, 1\}, y_{ea}, z_{ewa} \in \mathbb{Z}, \ \ \forall e \in E, a \in A, d \in D, w \in W$$

Objective function:

$$\min \sum_{e \in E} \sum_{w \in W} \left(B_{ew}^{wa} \left(v_{ew}^{wam} + v_{ew}^{wax}\right) + B_{ew}^{cwx} v_{ew}^{cwx}\right) + \sum_{e \in Q_{ad}} \sum_{a \in A} \sum_{d \in D} \left(B_{ead}^{rso} v_{ead}^{rso}\right)$$

The formulated constraints are very straightforward according to the problem definition, hence only the non-trivial ones are further explained here. In constraint 3b, to take into account the number of consecutive working days below the minimum, we count all the sequences individually up to the minimum exclusively. However, for the maximum part, we only need to count all the sequences which have a length of one day more than the maximum. Thus, all the sequences having a length of more than the maximum are counted accordingly. Constraint 3f is similar to constraint 3b, but only sequences of a particular shift type are counted. In constraint 4, we assume that there

should be two days off after a night shift or a series of night shift types. Furthermore, in constraints 2, 4, 5, and 6, n and r indicate night and rest shift types, respectively. The objective function is the weighted sum of all the auxiliary variables associated with soft constraints 3d, 7, and 8, and their relevant weights.

The proposed IP formulation is similar to the most IP formulations reported in the literature [38, 9]. However, there are often some differences in the softness of constraints (i.e. which constraints are considered hard or soft), and therefore, the associated objective function. For example, [38] proposed an IP formulation for the problem instances introduced in the first International Nurse Rostering Competition (INRC-I) [22]. In their formulation, all constraints are considered soft, except Constraint 2. That said, the objective function is also different, which is the weighted sum of the associated penalties with the remaining soft constraints. In this paper, we are not rely on a particular problem setting (i.e. softness of constraints) due to the particular design of the proposed hybrid algorithm. In Section 6, we benchmark the algorithm by using two different datasets including the instances introduced in INRC-I competition, which have different problem settings. In fact, this is one of the strengths of the proposed hybrid algorithm which is not dependent on a particular problem setting such as the softness of constraints, making it a proper option for practical circumstances. For more information regarding the softness of constraints, we refer the interested readers to [36]. It should be also noted that the presented formulation is based on the single-skill requirement, which can be easily extended to a multi-skill formulation with some minor adjustments similar to [38].

4. CP Formulation

Here, we present our CP formulation based on the Constraint Satisfaction Problem (CSP) model using the definitions and assumptions provided in Section 2. To the best of our knowledge, we are the first researchers presenting the CP formulation for the described problem. It is worth noting that we only utilise CP for generating feasible solutions in the configuration of the proposed hybrid algorithm, and thus, there is no need to incorporate the softness of constraints nor to define a Constraint Optimisation Problem (COP). In this section, first, we concisely explain the two types of global constraints, which are used in the CP model: *Cardinality* and *Stretch* global constraints. For more information about global constraints in CP, we refer the interested readers to [26], [46], and [3].

Cardinality constraints (aka. *GCC* or *Generalised Cardinality*) bounds the number of times

that variables take a certain set of domain values. It is written as:

$$cardinality\,(x, v, l, u)$$

where x is a set of variables $(x_1, \ldots, x_n)$; v is an m-tuple of domain values of the variables x; l and u are m-tuples of non-negative integers defining the lower and upper bounds of the times value v being taken by variable x, respectively. The constraint defines that, for $j = 1, \ldots, m$, at least l_j and at most u_j of the variables x take value v_j.

Stretch constraints bound the sequence of consecutive variables that take the same value (stretch), i.e. $x_{j-1} \neq 1, x_j, \ldots, x_k = v, x_{k+1} \neq v$. It is expressed as:

$$stretch\,(x, v, l, u, P)$$

where x is a set of variables $(x_1, \ldots, x_n)$; v is an m-tuple of possible domain values of x; l and u are m-tuples of lower and upper bounds for x, respectively. P is a set of patterns, i.e. pairs of values (v_j, v_k), requiring that when a stretch of value v_j immediately precedes a stretch of value v_k, the pair (v_j, v_k) must be in P.

As we use the same parameters as defined in Section 3, we define any additional parameters as well as the variables and constraints of the CP model next.

Sets and Parameters:

U The vector of total workloads (hours) of all the shift types within the planning period.

U_w The vector of total workloads (hours) of all the shift types during week $w \in W$.

Decision variable:

s_{ed} Integer variable indicating the shift type assigned to nurse $e \in E$ on day $d \in D$.

Constraints:

Having defined the required global constraints, we present the relevant CP constraints, where the numbering of the constraints is the same as the constraints presented in Section 2:

$$cardinality\left(\bigcup_{e \in E} s_{ed}, A, V_d^{min}, V_d^{max}\right), \quad \forall d \in D \tag{2}$$

$$cardinality\left(\bigcup_{d \in D} s_{ed}, A, M_e^{min}, M_e^{max}\right), \quad \forall e \in E \tag{3a}$$

$$stretch\left(\bigcup_{d\in D} s_{ed}, A', N^{min}, N^{max}, P\right),\ \forall e \in E, P = \{\} \tag{3b}$$

$$\begin{cases} A^{min} \leq prod(\bigcup\limits_{d\in D} s_{ed}, U) \leq A^{max}, & \forall e \in E \\ E_w^{min} \leq prod(\bigcup\limits_{d=7(w-1)+1}^{7w} s_{ed}, U_w) \leq E_w^{max}, & \forall e \in E, w \in W \end{cases} \tag{3c}$$

$$cardinality\left(\bigcup_{d=7(w-1)+1}^{7w} s_{ed}, A, W_w^{min}, W_w^{max}\right),\ \forall e \in E, w \in W \tag{3d}$$

$$cardinality\left(\bigcup_{w\in W} s_{e(7w)} + s_{e(7w-1)}, r, |W| - K^{max}, |W| - K^{min}\right), \quad \forall e \in E \tag{3e}$$

$$stretch\left(\bigcup_{d\in D} s_{ed}, a, H_a^{min}, H_a^{max}, P\right),\ \forall e \in E, a \in A, P = \{\} \tag{3f}$$

$$if\ s_{ed} = n, s_{e(d+1)} = r \wedge s_{e(d+2)} = r,\ \forall e \in E, d \in \{1, \dots |D| - 2\} \tag{4}$$

$$s_{e(7w-1)} = s_{e(7w)},\ \forall e \in E, w \in W \tag{5}$$

$$if\ s_{e(7w-2)} = n, s_{e(7w-1)} \neq r \vee s_{e(7w)} \neq r,\ \forall e \in E, w \in W \tag{6}$$

$$stretch\left(\bigcup_{w\in W} s_{e(7w)} + s_{e(7w-1)}, r, |W| - C^{max}, |W|, P\right),\ \forall e \in N, P = \{\} \tag{7}$$

$$s_{ed} = a,\ \forall e \in Q_{ad}, a \in A, d \in D \tag{8}$$

$$if\ s_{ed} = a, s_{e(d+1)} \notin H_a,\ \forall e \in E, d \in \{1, \dots |D| - 1\} \tag{9}$$

$$s_{ed} \in A,\ \forall e \in E, d \in D$$

Constraint 2 and 3a restrict the number of assigned shifts using the cardinality constraint in vertical and horizontal dimensions, respectively. In constraint 3b, sequences of working shifts

($a \in A'$) using the stretch constraint are limited to N^{min} and N^{max}, where there is no need to any specific pattern. Constraint 3c is very similar to its relevant IP form, where function $prod(x, v)$ is defined as the product of all the values of set x, and the relevant possible domain values in vector v. Constraints 3d and 3e resemble to constraint 2. Constraint 3f is the same as constraint 3b, where sequences of a particular shift type taken into account. In constraint 4, we assume that there should be two days off after a night shift or a series of night shift types. Using the stretch global constraint, constraint 7 restricts the sequences of rest shifts (instead of working shifts) to $|W| - C^{max}$ and $|W|$, since it is easier to be counted in terms of modelling. The remaining constraints are straightforward, where we use some simple reified (if-then) relations.

It should be noted that the first constraint is implicitly satisfied due to the inherent structure of the CP model, which only a single shift type can be assigned to each nurse at a day. To resolve some symmetry issues and increase the performance of the CP solver over the CSP model, we also add some lexicographic ordering constraints [39, 17] to the main variable s_{ed} by taking into account the "requested shifts on/off" constraints for each group of nurses working on the same working contract in a similar way described in [40].

5. Integration of IP and CP

For small- to medium-size problems, IP solvers are often efficient to find the optimal solution and to generate strong lower-bounds. Similarly, CP solvers are capable of finding feasible solutions very efficiently. However, using these approaches on their own for solving large-scale problems, or even small-scale problems with a highly-constrained structure often leads to poor performance. For example, solving most of the benchmark instances using the model presented in Section 3 with a standard IP approach, we were not able to obtain an optimal solution (and in some cases even a good-quality solution) in a reasonable amount of time, where some instances took more than 24 hours to solve (e.g. see Table 3). Similarly, a standard CP approach results in poor performance, since it often takes a long time to achieve an optimal solution. Therefore, it is intuitive to hybridise them in order to utilise their strengths for efficiently solving the NRP. That said, we extend the reach of IP and CP through a novel hybridisation setting. In fact, we combine the strength of IP in obtaining lower-bounds and finding an optimal solution with the capability of CP in finding feasible solutions in a novel way. Applying the IP approach as our core solution component, we use a CP approach and other algorithmic modules to increase its efficiency. Moreover, IP and CP on their own are quite efficient in solving some specific problem structures such as network flow and bin packing

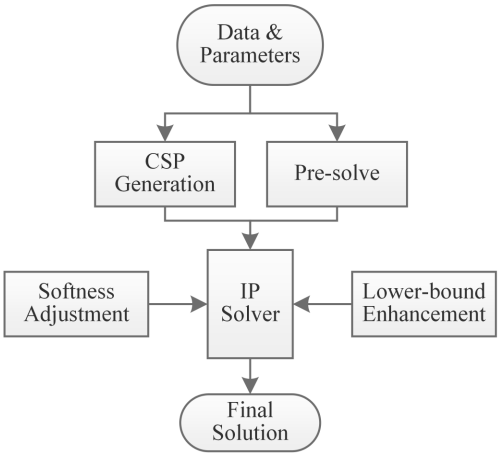

Figure 1: Schematic diagram of the proposed hybrid algorithm

problems [20, 23], which further motivates us to combine such strengths together in order to achieve better overall performance. To improve the efficiency of the hybrid algorithm, we also exploit the problem structure to provide some valuable information such as difficulty of constraints, thereby setting the parameters of the proposed algorithm adaptively. Applying a high-level hybridisation scheme of IP and CP, our contribution is a hybrid algorithm that can be implemented easily in practice, and that efficiently generates good-quality solutions for a wide range of problem instances, particularly within a relatively short computational time.

In the following, we provide a brief description of the performance of the hybrid algorithm, and then elaborate on each associated component individually. After a quick pre-processing in order to create appropriate data structures for the algorithm, at the first step, we employ an IP solver to pre-solve the problem in order to identify any valuable information, which can be used to adjust the parameters of other components accordingly. In the next step, we employ a CP solver to iteratively solve various CSP models in order to generate a good-quality solution, and to identify difficult constraints. Then, using the best-obtained solution provided in the CSP generation step, we further improve the attained solution by an IP solver in the remaining time and report the final solution to the user. We also add two more components to reinforce the search process using the exploited problem-specific information: *Softness adjustment* and *Lower-bound enhancement*. The former attempts to modify the softness of each constraint based on a pre-defined threshold in order to tighten the problem formulation, thereby aiming to help the IP solver to generate better bounds; and the latter aims to provide a stronger lower-bound for the IP solver by decomposing the problem.

It should be noted that the proposed hybrid algorithm runs in a pre-defined time limit to solve

the problem. Thus, the user is able to determine the running time of each component by setting the relevant computational time parameter. It is noteworthy to mention that all the components of the hybrid algorithm work with the IP model except the CSP generation step, for which the CP model is employed. The schematic diagram of the proposed algorithm is depicted in Figure 1. Next, we explain each component individually in more detail:

Pre-solve: In a nutshell, the duty of this step is to extract some information from the problem, and to regulate the parameters of the main IP solver adaptively. In fact, this component is the first step in most of the commercial solvers to analyse and simplify the problem structure, and also to identify any specific structures such as network flow or assignment problems [20]. If the IP solver can identify any particular structures, it is often led to better performance during the branch-and-bound algorithm. Here, we only call the pre-solve step of an IP solver from the hybrid algorithm as a black-box. We use the information obtained from this step including the obtained lower-bound and relaxed objective function to predict if there are any specific structures, thereby setting the parameters of the IP solver. According to our experiments, within the pre-solve step, if the IP solver provides a stronger (bigger in our problem settings) lower-bound compared with the relaxed objective function value (which is obtained by relaxing all the integer constraints), the employed IP solver might be able to solve the problem in better performance due to the identification of a specific data structure. That said, we switch on the relevant parameter for the pre-solve step of the main IP solver to the highest degree (aggressive mode) (e.g. setting the *presolve* parameter in Gurobi). Moreover, using the reported number of constraints and variables in this step, if they are more than the user-defined threshold $psThr$, we call the problem "difficult", thus, we set the search strategy of the main IP solver to spend more efforts on obtaining a feasible solution rather than on proving optimality. We do not change the default search strategy in case a problem is not difficult to solve. In most of the modern solvers, the user can modify the search strategy by setting a specific parameter defined therein. For example, in Gurobi IP solver, the user can tailor the search strategy by setting the *mipfocus* parameter.

CSP generation: After we set some parameters of the main IP solver, a CP solver is employed to generate a good-quality initial solution and to identify difficult constraints. This solver solves the problem according to the CSP model presented in Section 4. However, in our preliminary experiments on the benchmark instances, CP approach does not provide very good-quality or even feasible solutions within a limited computational time (e.g. see Table 3). To address this issue, we implement the following procedure: at each iteration i, we modify and then solve the CSP model p considering all those constraints that have a weight higher than the user-defined threshold

```
Solutions, p'_{i-1} = empty;
i = 1;
while true do
    p'_i = generateCSP(p, cspThr);
    if p'_i is feasible then
        for j = 1 to numSols do
            Solutions[p'_i].add(solve(p'_i));
        end
        if |1 − best(Solutions[p'_{i-1}]) / best(Solutions[p'_i])| ≤ q then
            break;
        end
    else
        cspThr = cspThr - 1;
        if p'_i == p then
            break;
        end
    end
    i++;
end
return [best(Solutions)]
```

Algorithm 1: The pseudo code of CSP generation in the hybrid algorithm

cspThr. If the modified problem p'_i is feasible, we generate a number of different solutions based on p'_i according to the user-defined parameter *numSols*. Otherwise, we decrease the threshold value by one unit, and go to the next iteration. Therefore, on each threshold level or iteration, there might be several feasible solutions, which are stored in the data structure *Solutions*. This process continues until the quality gap between the best-obtained solutions in two consecutive levels is less than q percent, or p'_i is equal to$p$. Finally, all the stored solutions are evaluated according to p and the best-quality solution is reported. Then, the reported solution is imported to the IP solver for further improvement. The pseudo code of this procedure is presented in Algorithm 1, where p, p'_i, *cspThr*, q, and *numSols* indicate the original problem, the new generated problem in each iteration i, the user-defined threshold level to cut off the constraints, the maximum gap between the quality of the best-obtained solutions in two consecutive threshold levels, and the user-defined number of solutions required to be generated for each threshold level, respectively. In this algorithm, *Solutions* is a data structure which stores a CSP and all the associated solutions, and function *best* evaluates the quality of the input solution according to the original problem settings.

Indeed, the aim of this step is to construct a weak CSP and iteratively strengthen it until some stopping criteria are met. Note that solving a CSP and generating a number of solutions accordingly is done quite fast, often within a few seconds. Nevertheless, experimentally speaking, the time limit of the CP solver is set to 10 seconds. Furthermore, it should be noted that although a

stronger CSP (i.e. a CSP with more number of constraints) is generated while progressing through lower threshold levels, and therefore, it is always expected to obtain better solutions afterwards, the quality of generated solutions is varied within each level due to the internal mechanisms of the applied CP solver.

Using the information obtained from a variety of threshold levels within the CSP generation step, we can also find out an estimate for the computational difficulty of each constraint: if the difference in solution times attained by removing a constraint from a problem in order to solve a new modified problem is significant, we count it as a "difficult" constraint. In our preliminary experiments, we concluded that 10 seconds is sufficient for most of the benchmark instances. Identifying difficult constraints helps us in the Softness adjustment step to tighten the formulation of the problem.

IP Solver: In this component, we use an IP solver to solve the problem within the remaining time, to which an initial solution generated from the CSP generation step is imported. Indeed, we use the solution obtained from the CP solver as a warm start for the IP solver. To improve the performance of the IP solver, we also apply two more techniques performed within the Softness adjustment and Lower-bound enhancement steps as described next. According to our preliminary experiments, these two steps are effective when either the problem instance is highly-constrained or huge in size.

Softness adjustment: In order to improve the efficiency of the IP solver during the search process, we modify the weights (e.g. see B^{wa}_{ew} and B^{rso}_{ead} in Section 3) in the objective function due to the difficulty degree of constraints obtained within the CSP generation step. According to this degree, if a constraint is not difficult, we impose it to the IP solver as a hard constraint. Obviously, we only consider soft constraints in this step, which have a contribution to the objective function. In fact, only difficult constraints are kept in the objective function, and the remaining ones are moved to the set of hard constraints. This is quite similar to modifying the associated weights with hard constraints in the objective function to a significantly large value. Theoretically, this process may lead to an infeasible problem. In this case, we undo the relevant change (i.e. declare the constraint as soft again) and continue the process for the rest of the constraints. Finally, we solve the new modified problem using the IP solver. This technique helps to reduce the search space by tightening the problem formulation, which often results in better efficiency during the search process, and higher-quality feasible solutions.

Lower-bound enhancement: In this step, we try to find out a better lower-bound by decomposing the problem to different sub-problems, and then, to solve each by using an IP solver. For this purpose, we decompose the problem in three ways: first, we break down the problem to weekly

rosters. For each weekly roster, all the constraints are taken into account except those associated with the whole planning horizon. These constraints are not examined unless their imposed restriction is valid for a week. For example, if the total number of assignments for each nurse within the planning period is 5, it is added to the involved constraints. Second, we decompose the problem to personal schedules for each nurse. In this case, all the constraints are considered excluding Constraint 2. Finally, we decompose the problem to groups of similar nurses. To identify similar nurses, we iterate through the set of nurses and constraints except Constraint 2 to find out nurses who are included in similar set of constraints. In order to facilitate this process, we define data structure $LBE = \{(e, c_e)|e \in E, c_e = \{c \leftrightarrows e|c \in C\}\}$, where E and C show the set of nurses and constraints, c_e denotes the set of constraints affects nurse e, and $\leftrightarrows$ indicates involvement. Then, we group all nurses who have the same set of c_e. Thus, each group shows a sub-problem. It should be noted that the identified groups do not necessarily partition the set of nurses or complement each other. Experimentally speaking, these groups are often associated with different working contracts in the problem data. Indeed, we try to find out whether there is an inevitable conflict in the model, which can be discovered before solving the problem by decomposing the problem into smaller sub-problems. Finally, the best lower-bound calculated in this step by solving different sub-problems is imposed to the IP solver by setting the relevant parameter (e.g. setting the *start* parameter in Gurobi).

6. Computational Results

The algorithm presented in this paper is tested on 9 real-world instances published in [10], and 36 instances introduced in INRC-I competition [22]. The first set of instances is called *real-world dataset*, and the second set of instances is called *competition dataset-I* and *competition dataset-II*. The difference between the two competition datasets is that the former is according to the single-skill assumption, on which the proposed hybrid algorithm is defined, and the latter contains multi-skill nurses to allow further analyses. Moreover, competition dataset-II contains very challenging large-scale instances, which are mostly computationally intractable for exact methods. Nevertheless, we benchmark the hybrid algorithm by using these instances to investigate how it performs in comparison with other meta-heuristic algorithms. The diversity in the structure and complexity of these instances allows us to test the algorithm thoroughly. In addition, the softness of constraints is varied for each instance. To the best of our knowledge, we are one of the few researchers experimenting with all these instances altogether. Table 1 provides more information regarding these instances. In this table, competition datasets are denoted as *Medium* and *Long* with the

Table 1: Benchmark instances and the relevant characteristics

Instance	Nurses	Shift types	Days	Instance	Nurses	Shift types	Days
Real-world Dataset:				*Competition Dataset-II:*			
GPOST	8	3	28	MediumH01	30	5	28
GPOSTB	8	3	28	MediumH02	30	5	28
ORTEC01	16	5	33	MediumH03	30	5	28
ORTEC02	16	5	33	MediumH04	30	5	28
Valouxis-1	16	4	28	MediumL05	30	5	28
SINTEF	24	6	21	LongE01	49	5	28
WHPP	30	4	14	LongE02	49	5	28
MILLAR-1	8	3	14	LongE03	49	5	28
LLR	27	4	7	LongE04	49	5	28
Competition Dataset-I:				LongE05	49	5	28
MediumE01	31	5	28	LongH01	50	5	28
MediumE02	31	5	28	LongH02	50	5	28
MediumE03	31	5	28	LongH03	50	5	28
MediumE04	31	5	28	LongH04	50	5	28
MediumE05	31	5	28	LongH05	50	5	28
MediumH05	30	6	28	LongL01	50	5	28
MediumL01	30	5	28	LongL02	50	5	28
MediumL02	30	5	28	LongL03	50	5	28
MediumL03	30	5	28	LongL04	50	5	28
MediumL04	30	5	28	LongL05	50	5	28
MediumT01	30	5	28	LongT01	50	5	28
MediumT02	30	5	28	LongT02	50	5	28
MediumT03	30	5	28	LongT03	50	5	28

suffix of *E*, *H*, *L*, *T*, indicating the *Early*, *Hidden*, *Late*, and *Hint* instances, respectively. We do not include *Sprint* instances in our benchmarks, since even a sole IP solver is able to find the optimal solution in a few minutes for each of them.

To evaluate the proposed hybrid algorithm, we implement our algorithm in Java 1.7, and use the IBM ILOG CP solver 1.7 [23] and Gurobi IP solver 5.6 [20] for solving all CP and IP models, respectively. The reason to use the aforementioned solvers is that they are easier to implement in terms of modelling, and also they suit our hybrid framework better than other software packages. In addition, we note that the benchmarks reported in [31] show that Gurobi produces very similar results for most of the instances comparing with other state-of-the-art IP solvers such as IBM ILOG Cplex [24]. We run our experiments on a PC with Intel 3.4 GHz processor and 4 GB of RAM, and only on one CPU core, in order to have a fairer and more accurate comparison.

For evaluation purposes, we run our hybrid algorithm for 10 minutes for real-world dataset. The reason is two-fold: first, the hybrid algorithm is primarily designed to perform well in a short time, and second, the selected time is in line with the testing times used by most of the algorithms reported in the literature, including the time used in INRC-I competition, so that a platform for a fair comparison is established. For competition datasets, we run the hybrid algorithm for 10 minutes for Medium instances and 10 hours for Long instances according to the competition time rules.

After extensive preliminary testing of the algorithm using different settings, the following parameters are chosen: We dedicate 5%, 25%, and 50% of the computational time to the Pre-solve, CSP generation, and IP Solver steps, respectively. The remaining time is distributed equally to Softness adjustment and Lower-bound enhancement steps as they require significantly shorter times in comparison. Furthermore, we set the threshold parameters $psThr$ and $cspThr$ for the Pre-solve and CSP generation steps to 10,000 and 1000, and parameters q and $numSols$ for the CSP generation step to 5 and 1000, respectively. Since the design of the algorithm is deterministic, we run it once per instance for each experiment and report the values accordingly.

We conduct three experiments to test the proposed algorithm: first, we analyse the performance of different components of the hybrid algorithm, and how they affect the efficiency of the algorithm. Second, we compare the hybrid algorithm with standard IP and CP solvers to investigate whether the hybrid algorithm is able to extend the reach of each of these methods solely. Finally, we compare the performance of the hybrid algorithm with other state-of-the-art methods in the relevant literature for all the benchmark datasets.

The first experiment is designed to investigate the effectiveness of Softness adjustment and

Table 2: Results of the hybrid algorithm using different settings

Instance	CG solution	LE bound		WA + LE (default)			No WA			No LE		
		RR	LB	Obj.	LB	G(%)	Obj.	LB	G(%)	Obj.	LB	G(%)
Real-world Dataset:												
GPOST	21	1	1	5	5	**0.0**	5	5	**0.0**	5	5	**0.0**
GPOSTB	20	0	0	5	0	**100.0**	5	0	**100.0**	5	0	**100.0**
ORTEC01	1031	40	**60**	380	158	**58.4**	392	158	59.7	380	158	**58.4**
ORTEC02	984	40	**60**	370	150	**59.5**	380	148	61.1	390	148	62.1
Valouxis-1	1651	0	0	20	10	**50.0**	20	10	**50.0**	20	10	**50.0**
WHPP	728	0	0	5	0	**100.0**	5	0	**100.0**	5	0	**100.0**
Competition Dataset-I:												
MediumE02	593	235	235	240	240	**0.0**	240	240	**0.0**	240	240	**0.0**
MediumE03	451	232	232	236	236	**0.0**	236	236	**0.0**	236	236	**0.0**
MediumE04	670	233	233	237	237	**0.0**	237	237	**0.0**	237	237	**0.0**
MediumE05	554	275	275	303	303	**0.0**	303	303	**0.0**	303	303	**0.0**
MediumH05	3263	56	56	174	56	**67.8**	192	56	70.9	174	56	**67.8**
MediumL01	2910	61	**92**	159	147	**7.5**	170	144	15.3	161	144	10.6
MediumL02	422	7	7	18	18	**0.0**	18	18	**0.0**	18	18	**0.0**
MediumL03	1529	9	9	29	25	**13.8**	29	25	**13.8**	29	25	**13.8**
MediumL04	527	6	6	35	32	**8.6**	35	32	**8.6**	35	32	**8.6**
MediumT01	1220	10	**14**	39	27	**30.8**	41	24	41.5	39	25	35.9
MediumT02	1544	25	25	80	52	**35.0**	80	52	**35.0**	80	52	**35.0**
MediumT03	9632	35	**41**	121	80	**33.9**	131	77	41.2	131	78	40.5

Lower-bound enhancement steps on overall performance of the hybrid algorithm, where we run the hybrid algorithm for 10 minutes and record the detailed results in Table 2. We do not report the results for instances SINTEF, MILLAR-1, LLR, and MediumE01, since they have very short solution time, hence giving us no insight into the impact of each component. In this table, the results of running the hybrid algorithm with default settings (i.e. keeping both Softness adjustment (WA) and Lower-bound enhancement (LE) steps), and without WA or LE are reported. For each setting, the obtained objective function value (*Obj.*), lower-bound (*LB*), and optimality gap (*G(%)*) are recorded. The optimality gap is defined as the discrepancy between the value of the current feasible solution (for the primal problem) and the value of the lower-bound (feasible for the dual problem). When the optimality gap is zero, the current feasible solution is optimal. Furthermore, we report the initial solution generated from CSP generation (CG) step, and the lower-bound generated from LE step in comparison with the Root Relaxation (RR).

As it can be seen, removing WA and LE from the main algorithm setting results in poorer performance for 6 and 4 instances, respectively (shown as bold style). As we have explained in Section 5, the expected impact of WA and LE steps is mostly realised when the problem instance is highly-constrained, and therefore, it is computationally difficult to solve. Thus, the impact of removing these two steps is only seen for some difficult instances such as ORTEC02 and MediumT01. Indeed,

Table 3: Benchmark results of the hybrid algorithm and standard IP and CP solvers within 10 and 60 minutes using the real-world dataset and competition dataset-I

Instance	Hybrid Algorithm						IP						CP
	10 min			60 min			10 min			60 min			60 min
	Obj.	LB	G(%)	Obj.	LB	G(%)	Obj.	LB	G(%)	Obj.	LB	G(%)	Obj.
Real-world Dataset:													
GPOST	5	5	**0.0**	5	5	**0.0**	9	5	44.4	5	5	**0.0**	11
GPOSTB	5	0	**100.0**	3	0	**100.0**	7	0	**100.0**	5	0	**100.0**	8
ORTEC01	380	158	**58.4**	270	210	**22.2**	1970	140	92.9	720	210	70.8	-
ORTEC02	370	150	**59.5**	270	210	**22.2**	15 415	140	99.1	700	210	70.0	830
Valouxis-1	20	10	**50.0**	20	20	**0.0**	1090	0	100.0	90	0	100.0	180
SINTEF	0	0	**0.0**	0	0	**0.0**	8	0	100.0	0	0	**0.0**	0
MILLAR-1	0	0	**0.0**	0	0	**0.0**	0	0	**0.0**	0	0	**0.0**	0
WHPP	5	0	**100.0**	5	5	**0.0**	24 061	0	100.0	1004	0	100.0	67 050
LLR	301	301	**0.0**	301	301	**0.0**	301	301	**0.0**	301	301	**0.0**	301
Competition Dataset-I:													
MediumE01	240	240	**0.0**	240	240	**0.0**	240	240	**0.0**	240	240	**0.0**	255
MediumE02	240	240	**0.0**	240	240	**0.0**	240	240	**0.0**	240	240	**0.0**	602
MediumE03	236	236	**0.0**	236	236	**0.0**	236	236	**0.0**	236	236	**0.0**	576
MediumE04	237	237	**0.0**	237	237	**0.0**	237	237	**0.0**	237	237	**0.0**	-
MediumE05	303	303	**0.0**	303	303	**0.0**	303	303	**0.0**	303	303	**0.0**	-
MediumH05	174	56	**67.8**	129	56	**56.6**	188	56	70.2	141	56	60.3	-
MediumL01	159	147	**7.5**	157	151	**3.8**	175	137	21.8	159	144	9.43	-
MediumL02	18	18	**0.0**	18	18	**0.0**	18	16	7.75	18	18	**0.0**	-
MediumL03	29	25	**13.8**	29	29	**0.0**	30	21	30.3	29	22	22.4	-
MediumL04	35	32	**8.6**	35	35	**0.0**	36	31	14.4	35	32	8.6	-
MediumT01	39	27	**30.8**	34	31	**8.8**	44	27	39.2	34	29	14.7	-
MediumT02	80	52	**35.0**	72	72	**0.0**	95	50	46.9	73	54	26.0	-
MediumT03	121	80	**33.9**	117	89	**23.9**	149	80	46.6	117	87	25.64	-

when a problem instance is computationally easy to solve, even removing the extra components does not affect the whole performance of the search process. In general, according to the obtained results, including these components within the search process is preferable. It is worth mentioning that running the algorithm using default settings generates also better lower-bounds for 5 instances in total, which shows the effectiveness of LE step in our algorithm.

In order to compare the performance of the hybrid algorithm against standard IP and CP solvers, i.e. running the solvers in default settings without any tuning or extra reinforcing techniques, we run the algorithm for 10 and 60 minutes (short and long runtimes), and report the results in Table 3 and 4. We also run the IP solver for 10 and 60 minutes, and the CP solver for 60 minutes. In this table, *Obj.*, *LB*, *G(%)*, and *T(s)* show the obtained solution, lower-bound, optimality gap, and the solution (clock) time in seconds, respectively. The better results in each runtime are shown as bold style. It should be noted that for running the CP solver over the benchmark instances, and therefore incorporating the soft constraints into the CP model explained in Section 4, we apply a COP model in a similar way to [34], where the authors apply CP by using two different sub-

Table 4: Benchmark results of the hybrid algorithm and standard IP and CP solvers within 10 and 60 minutes using the competition dataset-II

Instance	Hybrid Algorithm						IP						CP
	10 min			60 min			10 min			60 min			60 min
	Obj.	LB	G(%)	Obj.	LB	G(%)	Obj.	LB	G(%)	Obj.	LB	G(%)	Obj.
MediumH01	192	84	**56.3**	117	89	**23.9**	631	84	86.7	215	89	58.6	-
MediumH02	312	186	**40.4**	248	191	**23.0**	M	183	100.0	287	190	33.8	-
MediumH03	73	22	**69.9**	36	25	**30.6**	363	22	93.9	101	23	77.2	-
MediumH04	124	68	**45.2**	81	68	**16.0**	M	68	100.0	81	68	**16.0**	-
MediumL05	107	107	**0.0**	107	107	**0.0**	114	104	8.8	107	107	**0.0**	-
LongE01	197	197	**0.0**	197	197	**0.0**	197	197	**0.0**	197	197	**0.0**	-
LongE02	219	219	**0.0**	219	219	**0.0**	219	219	**0.0**	219	219	**0.0**	-
LongE03	240	240	**0.0**	240	240	**0.0**	240	240	**0.0**	240	240	**0.0**	-
LongE04	303	303	**0.0**	303	303	**0.0**	303	303	**0.0**	303	303	**0.0**	-
LongE05	284	284	**0.0**	284	284	**0.0**	284	284	**0.0**	284	284	**0.0**	-
LongH01	488	324	**33.6**	368	337	**8.4**	M	321	100.0	498	337	32.3	-
LongH02	101	81	**19.8**	92	86	**6.5**	M	79	100.0	144	81	43.8	-
LongH03	59	21	**64.4**	47	26	**44.7**	M	17	100.0	M	17	100.0	-
LongH04	38	14	**63.2**	29	19	**34.5**	M	14	100.0	M	14	100.0	-
LongH05	114	36	**68.4**	41	41	**0.0**	M	36	100.0	M	40	100.0	-
LongL01	332	212	**36.1**	235	235	**0.0**	M	212	100.0	M	231	100.0	-
LongL02	289	216	**25.3**	229	229	**0.0**	M	214	100.0	279	229	17.9	-
LongL03	320	213	**33.4**	221	218	**1.4**	M	213	100.0	254	218	14.2	-
LongL04	310	201	**35.2**	230	213	**7.4**	M	196	100.0	M	213	100.0	-
LongL05	156	79	**49.4**	94	80	**14.9**	631	79	87.5	374	79	78.9	-
LongT01	164	14	**91.5**	89	14	**84.3**	M	14	100.0	M	14	100.0	-
LongT02	96	9	**90.6**	65	10	**84.6**	M	4	100.0	M	9	100.0	-
LongT03	272	36	**86.8**	67	39	**41.8**	M	36	100.0	M	39	100.0	-

models. For running the algorithm using the competition dataset-II, we extend the IP and CP models introduced in Section 3 and 4 in a similar way to [38] by making a few minor adjustments.

Running the IP solver for 10 minutes, one can see that the hybrid algorithm found better solutions for all real-world instances except instances MILLAR-1 and LLR, for which both methods obtain the same results. Furthermore, the algorithm improves the lower-bounds for three instances. Observing the results for the competition dataset-I, the hybrid algorithm generates better results for 8 and 5 out of 13 instances, respectively. For competition dataset-II, the hybrid algorithm generates better results for 18 out of 23 instances, where it is able to improve the lower-bounds for 7 instances. In overall, from 22 benchmark instances, the hybrid algorithm obtains better solutions for 14 instances, and stronger lower-bounds for 8 instances in comparison with the IP solver running within 10 minutes.

To have a fairer comparison, we run the IP solver for the longer time of 60 minutes using the real-world dataset. It can be seen that the hybrid algorithm is able to find better solutions for 4 instances in comparison with the IP solver, where it proves the optimality for instances Valouxis-1 and WHPP. Regarding the competition instances, the proposed algorithm generates better results for 7 and 16 instances included in the dataset I and II, respectively.

Given the extra time to run the hybrid algorithm using the long runtime, it can be seen that the obtained results using a short runtime are improved for 4 and 24 instances from the total of 9 and 36 instances included in the real-world and competition datasets, respectively.

Similarly, running the CP solver within the long runtime using all the benchmark instances, it obtains the optimal solution for 3 instances, while not even finding any feasible solutions for 34 instances. Apart from the generated optimal solutions, the performance of the CP solver is inferior due to the complexity and highly-constrained nature of the benchmark instances. Therefore, it can be seen that hybridising IP and CP leads to better performance for both the real-world and competition instances.

To compare the performance of our algorithm (HA) with the state-of-the-art algorithms reported in the literature by using real-world dataset, we present in Table 5 the best-published results for the real-world instances by a hybrid Variable Neighbourhood Search [11], denoted as VNS-1, a Memetic Algorithm [8], denoted as MA, a Variable Depth Search [12], denoted as VDS, a Harmony Search Algorithm [21], denoted as HSA, a Scatter Search [7], denoted as SS, another variant of hybrid Variable Neighbourhood Search [29], denoted as VNS-2, a hybrid Branch-and-price algorithm [9], denoted as BAP, and a stochastic VNS [41], denoted as SVN. In this table, *BKS* shows the best-known solutions from the literature for the benchmark instances, which are obtained using column

Table 5: Benchmark results of the hybrid algorithm and other state-of-the-art algorithms consisting of VNS-1 [11], MA [8], VDS [12], HSA [21], SS [7], VNS-2 [29], BAP [9], and SVN [41] for real-world dataset

Instance	BKS	HA	VNS-1	MA	VDS	HSA	SS	VNS-2	BAP	SVN
GPOST	**5**	**5**	-	915	-	-	9	8	**5**	-
GPOSTB	**3**	5	-	789	-	-	5	-	**3**	-
ORTEC01	**270**	380	541	535	360	310	365	-	**270**	-
ORTEC02	**270**	370	-	-	-	330	-	-	**270**	-
Valouxis-1	**20**	**20**	-	560	-	-	100	160	80	73
SINTEF	**0**	**0**	-	8	-	-	4	-	**0**	-
MILLAR-1	**0**	**0**	-	100	-	-	**0**	**0**	**0**	-
WHPP	**5**	**5**	-		-	-	-	-	**5**	**5**
LLR	**301**	**301**	-	305	-	-	**301**	314	**301**	**301**

generation and relaxation techniques (mostly instance specific) with an IP solver or other heuristic methods for a long runtime. In this column, the optimal results are underlined. Furthermore, "-" shows that there is no published result for the relevant instances using the benchmarking algorithm. All the algorithms are run for 10 minutes except VNS-1, which is executed for one hour.

As we can see, our proposed hybrid algorithm is able to generate the best solutions for 6 instances, and competitive results for the remaining instances. Comparing the results with the BAP algorithm, it is able to find better results only for instance Valouxis-1. Indeed, BAP is a reinforced branch-and-price algorithm with some low-level heuristics and embedded dynamic programming, which makes it an efficient candidate to solve the benchmark instances.

Table 6 and 7 show the results obtained by BAP and an upgraded variant of VDS [9], denoted as VDS-1, an adaptive VNS [44], denoted as VNS-3, a hyper-heuristic [4], denoted as HYP, a systematic two-phase algorithm [45], denoted as SYS, an adaptive neighbourhood search [27], denoted as ANS, a general constraint optimisation solver [32], denoted as COS, a mathematical programming heuristic, denoted as RIP [38], and a Harmony Search Algorithm [1], denoted as HS for the competition datasets. Observing the results reported in Table 6 for competition dataset-I, the hybrid algorithm generates the best solutions for 8 instances. Comparing with other algorithms, the results of our algorithm are promising. Therefore, it can be seen that for rather challenging instances, the proposed algorithm is able to generate competitive results, even though it is an exact approach.

Table 7 shows the result for the challenging competition dataset-II. As it can be observed, the hybrid algorithm obtains the best solutions for only 6 instances. Comparing with other algorithms,

Table 6: Benchmark results of the hybrid algorithm and other state-of-the-art algorithms consisting of VDS-1 and BAP [9], VNS-3 [44], HYP [4], SYS [45], ANS [27], COS [32], RIP [38], and HS [1] for competition dataset-I

Instance	BKS	HA	VDS-1	BAP	VNS-3	HYP	SYS	ANS	COS	RIP	HS
MediumE01	**240**	**240**	244	**240**	**240**	242	**240**	**240**	241	**240**	243
MediumE02	**240**	**240**	241	**240**	**240**	241	**240**	**240**	**240**	**240**	242
MediumE03	**236**	**236**	238	**236**	**236**	238	**236**	**236**	**236**	**236**	238
MediumE04	**237**	**237**	240	**237**	**237**	238	**237**	**237**	238	**237**	240
MediumE05	**303**	**303**	308	**303**	**303**	304	**303**	**303**	304	**303**	305
MediumH05	**119**	174	-	-	124	-	122	**119**	-	**119**	169
MediumL01	**157**	159	187	**157**	161	163	158	164	176	**157**	169
MediumL02	**18**	**18**	22	**18**	19	21	**18**	20	19	**18**	26
MediumL03	**29**	**29**	46	**29**	30	32	**29**	30	30	**29**	34
MediumL04	**35**	**35**	49	**35**	**35**	38	**35**	36	37	**35**	42
MediumT01	**34**	39	-	-	-	40	-	-	-	-	42
MediumT02	**72**	80	-	-	-	91	-	-	-	-	83
MediumT03	**117**	121	-	-	-	134	-	-	-	-	130

our obtained results are inferior. These results are not beyond our expectations, since competition dataset-II is computationally intractable for exact methods such as the proposed hybrid algorithm. In fact, hybridising exact methods results in extending the reach of each sole method, but not to the extent that it can outperform hybrid meta-heuristics.

It is worth noting that our proposed algorithm finds the solutions reported in Table 5 to 7, while our aim of designing the hybrid algorithm is not only to find a good-quality solution, but also to achieve a better optimality gap for ensuring solution quality.

7. Conclusion

In this paper, we proposed a new hybrid algorithm combining IP and CP to solve the real-world Nurse Rostering Problem. The algorithm utilised the strengths of CP to aid the IP solver to achieve better solutions. Concentrated on the problem structure, we developed some components to provide valuable problem-specific information such as the computational difficulty of instances and constraints to both IP and CP solvers so that better performance can be achieved in solving highly-constrained problem instances. In contrast to heuristic methods reported in the literature, we attempted to design a hybrid method for generating a high-quality solution as well as a strong lower-bound in order to guarantee the solution quality. Due to the high-level hybridisation of IP and CP, an important aspect of the proposed hybrid algorithm is its straightforward adaptability to practical circumstances in terms of implementation, which makes it advantageous compared with

Table 7: Benchmark results of the hybrid algorithm and other state-of-the-art algorithms consisting of VDS-1 and BAP [9], VNS-3 [44], HYP [4], SYS [45], ANS [27], COS [32], RIP [38], and HS [1] for competition dataset-II

Instance	BKS	HA	VDS-1	BAP	VNS-3	HYP	SYS	ANS	COS	RIP	HS
MediumH01	**111**	192	-	-	122	-	130	117	-	**111**	143
MediumH02	**221**	312	-	-	**221**	-	**221**	220	-	**221**	248
MediumH03	**34**	73	-	-	**34**	-	36	35	-	**34**	49
MediumH04	**78**	124	-	-	79	-	81	79	-	**78**	87
MediumL05	**107**	**107**	161	**107**	112	122	**107**	117	125	**107**	131
LongE01	**197**	**197**	198	**197**	**197**	**197**	**197**	**197**	**197**	**197**	**197**
LongE02	**219**	**219**	223	**219**	**219**	220	**219**	222	**219**	**219**	226
LongE03	**240**	**240**	242	**240**	**240**	**240**	**240**	**240**	**240**	**240**	**240**
LongE04	**303**	**303**	305	**303**	**303**	**303**	**303**	**303**	**303**	**303**	**303**
LongE05	**284**	**284**	286	**284**	**284**	**284**	**284**	**284**	**284**	**284**	**284**
LongH01	**346**	488	-	-	349	-	363	**346**	-	**346**	380
LongH02	**89**	101	-	-	**89**	-	90	**89**	-	**89**	110
LongH03	**38**	59	-	-	**38**	-	**38**	**38**	-	**38**	44
LongH04	**22**	38	-	-	**22**	-	**22**	**22**	-	**22**	27
LongH05	**41**	114	-	-	**41**	-	**41**	45	-	**41**	53
LongL01	**235**	332	286	**235**	239	241	**235**	237	**235**	**235**	253
LongL02	**229**	289	290	**229**	224	245	**229**	**229**	**229**	**229**	256
LongL03	**220**	320	290	**220**	227	233	**220**	222	**220**	**220**	256
LongL04	**221**	310	280	**221**	232	246	**221**	227	**221**	222	263
LongL05	**83**	156	110	**83**	**83**	87	**83**	**83**	**83**	**83**	98
LongT01	**31**	164	-	-	-	33	-	-	-	-	40
LongT02	**17**	96	-	-	-	**17**	-	-	-	-	28
LongT03	**53**	272	-	-	-	55	-	-	-	-	81

sophisticated customised methodologies from the literature that are not easy to implement and use in practice. We benchmarked our hybrid algorithm with two different datasets consisting of real-world and competition instances. In comparison with standard IP and CP solvers, the obtained results showed better performance, indicating the effectiveness of the hybridisation for extending the reach of exact methods. We also obtained competitive results against state-of-the-art algorithms for a diverse range of instances.

In our future research, we will investigate different formulations such as pattern-based modelling for the Nurse Rostering Problem, which have the potential to provide better performance for large-size problem instances when the modelling is designed in a customised fashion. Moreover, we plan to explore further opportunities to exploit the problem-specific information. From these further studies, we plan to go beyond the aim of designing an easy-to-implement and effective framework as accomplished in this paper. Therefore, we will attempt to design a more sophisticated and customised framework involving various heuristics and novel automation approaches, so that different characteristic of the problem can be accommodated, and the overall performance can be improved. We also plan to implement our developed tools in a real hospital environment, which we are currently discussing with various local health providers. Finally, we also plan to investigate possibilities to extend our algorithm to other scheduling problems that have similar problem structure, such as the course timetabling problem, so that we can exploit practical benefits in more general settings.

References

[1] Mohammed A. Awadallah, Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, Asaju La'aro Bolaji, and Mahmud Alkoffash. Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem. *Neural Computing and Applications*, pages 1–20, 2015.

[2] Masri Ayob, Mohammed Hadwan, Mohd Zakree Ahmad Nazri, and Zulkifli Ahmad. Enhanced harmony search algorithm for nurse rostering problems. *Journal of Applied Sciences*, 13(6):846–853, 2013.

[3] Nicolas Beldiceanu, M Carlsson, and Jx Rampon. Global constraint catalog. Technical report, SICS, 2005.

[4] Burak Bilgin, Peter Demeester, Mustafa Misir, Wim Vancroonenburg, and Greet Vanden Berghe. One hyper-heuristic approach to two timetabling problems in health care. *Journal of Heuristics*, 18(3):401–434, 2012.

[5] Peter Brucker, Edmund K. Burke, Tim Curtois, Rong Qu, and Greet Vanden Berghe. A shift sequence based approach for nurse scheduling and a new benchmark dataset. *Journal of Heuristics*, 16(4):559–573, 2010.

[6] Peter Brucker, Rong Qu, and Edmund Burke. Personnel scheduling: Models and complexity. *European Journal of Operational Research*, 210(3):467–473, 2011.

[7] E. K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe. A scatter search methodology for the nurse rostering problem. *Journal of the Operational Research Society*, 61(11):1667–1679, 2010.

[8] Edmund Burke, Peter Cowling, Patrick De Causmaecker, and Greet Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied Intelligence*, 15(3):199–214, 2001.

[9] Edmund K. Burke and Tim Curtois. New approaches to nurse rostering benchmark instances. *European Journal of Operational Research*, 237(1):71–81, 2014.

[10] Edmund K Burke, Tim Curtois, Rong Qu, and Greet Vanden Berghe. Problem model for nurse rostering benchmark instances. Technical report, ASAP, School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, UK, 2008.

[11] Edmund K. Burke, Timothy Curtois, Gerhard Post, Rong Qu, and Bart Veltman. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2):330–341, 2008.

[12] Edmund K. Burke, Timothy Curtois, Rong Qu, and Greet Vanden Berghe. A time predefined variable depth search for nurse rostering. *INFORMS Journal on Computing*, 25(3):411–419, 2013.

[13] Edmund K. Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–449, 2004.

[14] Edmund K. Burke, Jingpeng Li, and Rong Qu. A Pareto-based search methodology for multi-objective nurse scheduling. *Annals of Operations Research*, 196(1):91–109, 2012.

[15] Hoong Chuin Lau. On the complexity of manpower shift scheduling. *Computers and Operations Research*, 23(1):93–102, 1996.

[16] A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[17] P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. *Lecture Notes in Computer Science*, 2470:462–476, 2002.

[18] A. Girbea, C. Suciu, and F. Sisak. Design and implementation of a fully automated planner-scheduler constraint satisfaction problem. *SACI 2011 - 6th IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings*, pages 477–482, 2011.

[19] Celia A. Glass and Roger A. Knight. The nurse rostering problem: A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2):379–389, 2010.

[20] Inc. Gurobi Optimization. Gurobi IP Solver, 2015.

[21] Mohammed Hadwan, Masri Ayob, Nasser R Sabar, and Rong Qu. A harmony search algorithm for nurse rostering problems. *Information Sciences*, 233(0):126–140, 2013.

[22] Stefaan Haspeslagh, Patrick De Causmaecker, Andrea Schaerf, and Martin Stølevik. The first international nurse rostering competition 2010. *Annals of Operations Research*, 218(1):221–236, 2014.

[23] IBM. IBM ILOG CPLEX CP Optimizer, 2015.

[24] IBM. IBM ILOG CPLEX Optimizer, 2015.

[25] Glen Kazahaya. Harnessing technology to redesign labor cost management reports. *Healthcare financial management : journal of the Healthcare Financial Management Association*, 59(4):94–100, 2005.

[26] F. Laburthe and N. Jussien. CHOCO solver - Documentation, 2011.

[27] Zhipeng Lü and Jin Kao Hao. Adaptive neighborhood search for nurse rostering. *European Journal of Operational Research*, 218(3):865–876, 2012.

[28] Broos Maenhout and Mario Vanhoucke. Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. *Journal of Scheduling*, 13(1):77–93, 2010.

[29] Jean Philippe Metivier, Patrice Boizumault, and Samir Loudni. Solving nurse rostering problems using soft global constraints. In IanP Gent, editor, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5732 LNCS, chapter 9, pages 73–87. Springer Berlin Heidelberg, 2009.

[30] Rym M'Hallah and Amina Alkhabbaz. Scheduling of nurses: A case study of a Kuwaiti health care unit. *Operations Research for Health Care*, 2(1-2):1–19, 2013.

[31] Hans D. Mittelmann. Decison Tree for Optimization Software, 2008.

[32] K. Nonobe. INRC2010: An approach using a general constraint optimization solver. Technical report, Hosei University, 2010.

[33] Ya Ozcan. *Quantitative methods in health care management: techniques and applications*, volume 4. John Wiley & Sons, 2005.

[34] Rong Qu and Fang He. A hybrid constraint programming approach for nurse rostering problems. In Tony Allen, Richard Ellis, and Miltos Petridis, editors, *Applications and Innovations in Intelligent Systems XVI - Proceedings of AI 2008, the 28th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, chapter 16, pages 211–224. Springer London, 2009.

[35] Erfan Rahimian, Kerem Akartunalı, and John Levine. A Hybrid Constraint Integer Programming Approach to Solve Nurse Scheduling Problems. In *Proceedings of the Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA 2015*, pages 429–442, Prague, Czech Republic, 2015.

[36] Erfan Rahimian, Kerem Akartunalı, and John Levine. Integer Programming for Nurse Rostering: Modelling and Implementation Issues. In *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling, PATAT 2016*, pages 545–547, Udine, Italy, 2016.

[37] Erfan Rahimian, Kerem Akartunalı, and John Levine. A Hybrid Integer Programming and Variable Neighbourhood Search Algorithm to Solve Nurse Rostering Problems. *European Journal of Operational Research*, 258(2):411–423, 2017.

[38] Haroldo G. Santos, Túlio A. M. Toffolo, Rafael A. M. Gomes, and Sabir Ribas. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, 239(1):225–251, 2016.

[39] Ilya Shlyakhter. Generating effective symmetry-breaking predicates for search problems. *Discrete Applied Mathematics*, 155(12):1539–1548, 2007.

[40] Barbara Smith. Modelling for Constraint Programming. In Francesca Rossi, Peter Van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, chapter 11, pages 377–406. Elsevier, 2006.

[41] Ioannis Solos, Ioannis Tassopoulos, and Grigorios Beligiannis. A Generic Two-Phase Stochastic Variable Neighborhood Approach for Effectively Solving the Nurse Rostering Problem. *Algorithms*, 6(2):278–308, 2013.

[42] Ricardo Soto, Broderick Crawford, Rodrigo Bertrand, and Eric Monfroy. Modeling NRPs with Soft and Reified Constraints. *AASRI Procedia*, 4(0):202–205, 2013.

[43] Martin Stølevik, Tomas Eric Nordlander, Atle Riise, and Helle Frøyseth. A hybrid approach for solving real-world nurse rostering problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 85–99. Springer Berlin Heidelberg, 2011.

[44] Ioannis X. Tassopoulos, Ioannis P. Solos, and Grigorios N. Beligiannis. A two-phase adaptive variable neighborhood approach for nurse rostering. *Computers and Operations Research*, 60:150–169, 2015.

[45] Christos Valouxis, Christos Gogos, George Goulas, Panayiotis Alefragis, and Efthymios Housos. A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, 219(2):425–433, 2012.

[46] WJ Van Hoeve and I. Katriel. Global Constraints. In Francesca Rossi, P Van Beek, and T Walsh, editors, *Handbook of Constraint Programming*, chapter 6, pages 169–208. Elsevier B.V., first edition, 2006.