Journal of Scheduling

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

--Manuscript Draft--

Manuscript Number:	
Full Title:	A Hybrid Constraint and Integer Programming Approach to Solve Nurse Rostering Problems
Article Type:	S.I. : MISTA 2015
Keywords:	Timetabling; Nurse Rostering; Hybrid Algorithm; Integer Programming; Constraint Programming
Corresponding Author:	Erfan Rahimian, MSC University of Strathclyde Glasgow, UNITED KINGDOM
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	University of Strathclyde
Corresponding Author's Secondary Institution:	
First Author:	Erfan Rahimian, MSC
First Author Secondary Information:	
Order of Authors:	Erfan Rahimian, MSC
	Kerem Akartunali, PhD
	John Levine, PhD
Order of Authors Secondary Information:	
Funding Information:	

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

Latex file (tex) Click here to download Manuscript JOS paper-for submission-twocolumn 02.tex

Click here to view linked References

Journal of Scheduling manuscript No.
(will be inserted by the editor)

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

Erfan Rahimian · Kerem Akartunalı · John Levine

Received: date / Accepted: date

Abstract The Nurse Rostering Problem can be simply defined as assigning a series of shift sequences (schedules) to several nurses over a planning horizon according to some constraints and preferences. The inherent benefits of having higher-quality and more flexible schedules are a reduction in outsourcing costs and an increase of job satisfaction in health organizations. In this paper, we present a novel hybrid algorithm, which combines Integer Programming (IP) and Constraint Programming (CP) to efficiently solve highly-constrained Nurse Rostering Problems. We utilize 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 obtain high-quality solutions as well as strong lower bounds during a short time, we apply some innovative ways to extract useful information such as the computational difficulty of constraints from different steps of the search process. In fact, by employing the CP solver to facilitate the IP solver and improve its efficiency, we design some supplementary algorithmic components to further enhance the overall performance. We test our algorithm based on some real-world benchmark instances. Competitive results are reported compared to the state-of-the-art algorithms from the recent literature as well as pure IP and CP solvers, showing that the proposed algorithm is able to solve a wide variety of real-world instances with different complex structures.

Keywords Timetabling · Nurse Rostering · Hybrid Algorithm · Integer Programming · Constraint Programming

Erfan Rahimian, Kerem Akartunalı
Dept. of Management Science, University of Strathclyde, Glasgow, G4 0GE, UK
Tel.: +44-141-548-4361
Fax: +44-141-552-6686
E-mail: {erfan.rahimian, kerem.akartunali}@strath.ac.uk

John Levine
Computer And Information Sciences, University of Strathclyde, Glasgow, G1 1XH, UK
E-mail: john.levine@strath.ac.uk

1 Introduction

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

The Nurse Rostering Problem (NRP) aims to generate schedules for several nurses over a predetermined planning horizon. A schedule consists of a sequence of different types of shifts (e.g. early, late, vacations) spanning over the whole planning period. The pattern of shifts is 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 their complex and highly-constrained structure, most NRPs in real-world situations are computationally challenging and they can be also classified as *NP*-hard (Chuin Lau, 1996; Brucker et al, 2011). The inherent nature of the problem usually leads us to divide all constraints to two categories in practice: hard and soft constraints. Hard constraints must be satisfied to have a feasible roster, whereas soft constraints may be violated. To evaluate the quality of a roster, one can minimize the sum of all penalties incurred due to soft constraint violations. For more information regarding NRPs and generally staff scheduling problems, we refer interested readers to Burke et al (2004); Ernst et al (2004).

The focus of this paper, which is an extension to our previous work (Rahimian et al, 2015) is on integrating Integer Programming (IP) and Constraint Programming (CP) to solve NRPs, where we exploit the problem-specific information in order to improve both IP and CP performance. In the literature, there are two areas of general methods used to solve these problems: exact and heuristic methods. Exact methods include IP (M'Hallah and Alkhabbaz, 2013; Glass and Knight, 2010; Maenhout and Vanhoucke, 2010) and CP (Soto et al, 2013; Girbea et al, 2011), which are capable of finding the optimal solution, albeit often resulting in unacceptable computational times. However, recent research in Operations Research and Artificial Intelligence communities, combined with powerful solvers such as IBM CP OPTIMIZER (IBM, 2015) and Gurobi (Gurobi Optimization, 2015), focused on using these methods in hybrid settings (Stolevik et al, 2011; Valouxis et al, 2012; Fung et al, 2005). On the other hand, in order to address the computational limitations of exact methods, many heuristic methods have been proposed in the literature. However, these methods sacrifice the guarantee of an optimal solution (or even any information about the solution quality) in order to generate good solutions in acceptable computational times. We note Lu and Hao (2012); Burke et al (2012); Brucker et al (2010) as some recent examples of using heuristic methods in the NRP literature.

In recent years, some researchers experimented with hybridizations of different methods, e.g. CP and heuristics (Stolevik et al, 2011), IP and heuristics (Valouxis et al, 2012), and less well-investigated combination of IP and CP (Fung et al, 2005), in order to utilize the complementary strengths of all methods together. In this paper, we propose a new systematic hybrid algorithm using IP and CP approaches, which utilizes their capabilities, respectively in finding the optimal and a feasible solution very efficiently. Due to the exact nature of the proposed algorithm, it can generate a good solution as well as a good lower bound in contrast to heuristic methods. Furthermore, the hybrid algorithm exploits the problem-specific information to reduce the search space, to fine tune the search parameters, and to improve the efficiency of the search process in a novel way. In fact, using an IP approach as the main solution method, we employ a CP approach and some other algorithmic aids to improve the efficiency of the algorithm. During the search process, we try to identify the computationally expensive constraints, and predict the performance of the IP solver to make a decision regarding its inclusion for the remaining steps. Moreover, the proposed algorithm is designed to obtain the best result in a pre-defined limited computational time. We model the problem according to a general comprehensive model reported in the literature (Burke et al, 2008b), and evaluate it using some test instances published therein.

The rest of this paper is organized as follows: problem definition and assumptions are explained in Section 2. The IP and CP formulations are presented in Sections 3 and 4. In Section 5, we describe the proposed hybrid algorithm and the relevant components. Computational results are reported in Section 6, and some conclusions and potential future research directions are drawn in Section 7.

2 Problem Definition

NRP is the process of assigning a number of nurses to a number of work shifts during a planning horizon according to a set of requirements and constraints. These constraints are usually categorized as hard and soft constraints. In the following, we define decision variables and constraints according to the conceptual model described in Burke et al (2008b), which will be used to construct an IP model.

We define our decision variables for each nurse, for each day, and for each shift type. This way of modeling allows us to better utilize the problem-specific structure in order to reduce the search space, although it is less flexible and contains more symmetry compared to the pattern-based modelling, e.g. Burke et al (2012), which generates all possible weekly shift sequences (patterns), and hence potentially considers all constraints except coverage constraints. We assume 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. We also consider a day-off as a shift type for modelling purposes. For the sake of simplicity, we assume all nurses belong to the same skill category. In addition, we assume all rosters start from Monday and are made from a complete week (includes seven days with a two-day weekend). The constraints of the model are:

1. Maximum one assignment per shift type per day,
2. Coverage constraints: 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. Complete weekends: 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, where there is no assignment at all,
7. Maximum number of consecutive worked weekends, where there is at least one assignment,
8. Requested shifts (days) on or off,
9. Forbidden shift type patterns (e.g. the “ND” pattern, where the shift type “D” is not allowed to be assigned right after the shift type “N”).

In the next two sections, i.e. Sections 3 and 4, we formulate this problem using Integer Programming (IP) and Constraint Programming (CP). We also note that the above constraints can be considered hard or soft according to different settings. For the sake of simplicity, we only provide here a formulation assuming that all constraints are hard. In case any soft constraints exist in the model, our objective function can be defined as the weighted sum of all associated slack variables in the IP model or the relevant reified variables in the CP model for each soft constraint.

3 IP Formulation

Here, we present our mathematical formulation using Integer Programming based on the definitions and assumptions provided in Section 2. The variables, parameters, and constraints of the IP model are defined as follows:

Decision variables:

x_{ead}	$= 1$ if shift type a on day d is assigned to nurse e, $= 0$ otherwise.
p_{ed}	$= 1$ if nurse e works on day d, $= 0$ otherwise.
k_{ew}	$= 1$ if nurse e is assigned to weekend w, $= 0$ otherwise.
y_{ea}	Total number of times that shift type a assigned to nurse e over the planning period.
z_{ewa}	Total number of shift type a assigned to nurse e during week w.

Parameters:

N	Set of nurses.
D	Set of days.
A	Set of shift types.
W	Set of weeks.
H_a	Set of shift types that cannot be assigned immediately after shift type a.
PR_{ad}	Set of pre-assigned nurses to shift type a on day d.
ML_e, MU_e	Minimum and maximum number of shifts that can be assigned to nurse e within the planning period.
WL_w, WU_w	Minimum and maximum number of shifts that can be assigned to a nurse within week w.
VL_d, VU_d	Minimum and maximum number of shifts that can be assigned to nurses on day d.
AL, AU	Minimum and maximum number of hours that can be assigned to each nurse during the planning period.
EL_w, EU_w	Minimum and maximum number of hours that can be assigned to each nurse during week w.
NL, NU	Minimum and maximum number of consecutive working days over the planning period.
HL_a, HU_a	Minimum and maximum number of consecutive shift type a over the planning period.
KL, KU	Minimum and maximum number of worked weekends over the planning horizon.
CU	Maximum number of consecutive worked weekends over the planning period.
UT_a	Total workloads (hours) of shift type a within the planning period.
UT_{aw}	Total workloads (hours) of shift type a during week w.

Constraints:

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

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

$$p_{ed} = \sum_{a\in A\setminus\{r\}} x_{ead}, \quad \forall e \in N, d \in D \tag{2}$$

$$VL_d \le \sum_{e\in N} p_{ed} \le VU_d, \quad \forall d \in D$$

$$y_{ea} = \sum_{d\in D} x_{ead}, \quad \forall e \in N, a \in A \tag{3a}$$

$$ML_e \le \sum_{a\in A} y_{ea} \le MU_e, \quad \forall e \in N$$

$$\sum_{g=d}^{NU+d} p_{eg} \le NU, \quad \forall e \in N, d \in \{1 \ldots |D| - NU\} \tag{3b}$$

$$\sum_{i=1}^{NL-1} p_{e(d+i)} \le p_{ed} + p_{e(d+NL)} + NL - 2, \quad \forall e \in N, d \in D$$

$$AL \leq \sum_{a\in A} y_{ea}UT_a \leq AU, \quad \forall e \in N \tag{3c}$$

$$z_{ewa} = \sum_{d=7(w-1)+1}^{7w} x_{ead}, \quad \forall e \in N, a \in A, w \in W$$

$$EL_w \leq \sum_{a\in A} z_{ewa}UT_{aw} \leq EU_w, \quad \forall e \in N, w \in W$$

$$WL_w \leq \sum_{a\in A} z_{ewa} \leq WU_w, \quad \forall e \in N, w \in W \tag{3d}$$

$$k_{ew} \leq p_{ed} + p_{e(d+1)} \leq 2k_{ew}, \quad d = 7w-1, \forall e \in N, w \in W \tag{3e}$$

$$KL \leq \sum_{w\in W} k_{ew} \leq KU, \quad \forall e \in N$$

$$\sum_{i=1}^{HL_a-1} x_{ea(d+i)} \leq x_{ead} + x_{ea(d+HL_a)} + HL_a - 2, \quad \forall e \in N, a \in A, d \in D \tag{3f}$$

$$\sum_{g=d}^{HU_a+d} x_{eag} \leq HU_a, \quad \forall e \in N, a \in A, d \in \{1 \ldots |D| - HU_a\}$$

$$x_{end} \leq x_{en(d+1)} + 1 - p_{e(d+1)}, \quad \forall e \in N, d \in \{1 \ldots |D|-1\} \tag{4}$$

$$x_{end} - p_{e(d+1)} \leq 1 - p_{e(d+2)}, \quad \forall e \in N, d \in \{1 \ldots |D|-2\}$$

$$x_{erd} = x_{er(d+1)}, \quad \forall e \in N, d \in \{6, 13, \ldots |D|-1\} \tag{5}$$

$$x_{end} \leq p_{e(d+1)} + p_{e(d+2)}, \quad \forall e \in N, d \in \{5, 12, \ldots |D|-2\} \tag{6}$$

$$\sum_{i=0}^{CU} k_{e(w+i)} \leq CU, \quad \forall e \in N, w \in \{1 \ldots |W| - CU\} \tag{7}$$

$$x_{ead} = 1, \quad \forall e \in PR_{ad}, a \in A, d \in D \tag{8}$$

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

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

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 shift types **n**ight and **r**est, respectively.

4 CP Formulation

Here, we present our CP formulation based on Constraint Satisfaction Problem (CSP) model using the definitions and assumptions provided in Section 2. The presented model is detailed enough for the needs of this paper, however, we would add other redundant constraints or variables to increase the efficiency of the CP solver (Smith, 2006). In this section, first, we concisely explain the two types of global constraints which we use in the CP model: *Cardinality* and *Stretch* global constraints. For more information about global constraints in Constraint Programming, we refer the interested reader to Laburthe and Jussien (2011); van Hoeve and Katriel (2006); Beldiceanu et al (2005).

Cardinality constraints (aka. *GCC* or *Generalized 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 a 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 a 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.

Next, we define the variables, parameters, and constraints of the model. We use the same parameters as defined in the IP formulation (Section 3), and we define here only additional ones.

Decision variable:

s_{ed}	Integer variable indicating the shift type assigned to nurse e on day d.

Parameters:

$\tilde{H}_a$	Set of shift types that can be assigned immediately after shift type a.
UT	The vector of total workloads (hours) of the shift types within the planning period.
UT_w	The vector of total workloads (hours) of the shift types during week w.

Constraints:

Next, we present the relevant CP constraints based on the *cardinality* and *stretch* global constraints, where the order of the constraints is preserved the same as the order of the constraints presented in Section 2:

$$cardinality\left(\bigcup_{e\in N} s_{ed}, A, VL_d, VU_d\right), \quad \forall d \in D \tag{2}$$

$$cardinality\left(\bigcup_{d\in D} s_{ed}, A, ML_e, MU_e\right), \quad \forall e \in N \tag{3a}$$

$$stretch\left(s_{ed}, A, NL, NU, P\right), \quad \forall e \in N, d \in D, P = \{(a,r)|a \in A\} \tag{3b}$$

$$AL \leq prod(s_{ed}, UT) \leq AU, \quad \forall e \in N, d \in D \tag{3c}$$

$$EL_w \leq prod(s_{ed}, UT_w) \leq EU_w, \quad \forall e \in N, w \in W, d = 7(w-1)+1$$

$$cardinality\left(\bigcup_{d=7(w-1)+1}^{7w} s_{ed}, A, WL_w, WU_w\right), \quad \forall e \in N, w \in W \tag{3d}$$

$$cardinality\left(s_{ed}, r, KL, KU\right), \quad \forall e \in N, d \in \{7w-i|w \in W, i \in \{0,1\}\} \tag{3e}$$

$$stretch\left(s_{ed}, a, HL_a, HU_a, P\right), \quad \forall e \in N, d \in D, a \in A, P = \{\} \tag{3f}$$

$$stretch\left(s_{ed}, n, 2, 3, P\right), \quad \forall e \in N, d \in D, P = \{(n,r)\} \tag{4}$$

$$s_{ed} = s_{e(d+1)}, \quad \forall e \in N, d \in \{6, 13, \ldots |D|-1\} \tag{5}$$

$$stretch\left(s_{ed}, n, 2, 3, P\right), \quad \forall e \in N, d \in \{7w-i|w \in W, i \in \{0,1,2\}\}, P = \{(n,r)\} \tag{6}$$

$$stretch\left(s_{ed}, r, 2(|W|-CU), 2|W|, P\right), \quad \forall e \in N, d \in \{7w-i|w \in W, i \in \{0,1\}\}, P = \{(r,r)\} \tag{7}$$

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

$$stretch\left(s_{ed}, a, 0, 2, P\right), \quad \forall e \in N, d \in D, P = \{(a, \tilde{H}_a)|a \in A\} \tag{9}$$

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

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 the mentioned constraints, n and r indicate a night shift type and a day-off, respectively. It should be noted that constraint (1) is already satisfied due to the inherent structure of the CP model.

5 Integration of CP and IP

For small- to medium-sized 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 NRP instances using the model presented in Section 3 with a pure 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 the result for instance ORTEC01 in Table 3). Similarly, a pure CP approach results in poor performance as well, since it often takes a long time to achieve an optimal solution. Therefore, it is intuitive to hybridize them in order to utilize their strengths for efficiently solving NRPs. In this paper, we integrate IP and CP approaches in a co-operative fashion to solve the problem and utilize their strengths altogether. Indeed, 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. Moreover, according to our preliminary experiments, IP and CP on their own are quite efficient in solving some specific problem structures such as network flow and bin packing problems (Gurobi Optimization, 2015; IBM, 2015), which further motivates us to combine them together in order to achieve a better performance in overall. To improve the efficiency of the hybrid algorithm, we exploit the problem structure to provide valuable information about search space, hence improve the efficiency of the proposed algorithm. In fact, we use a CP approach and some other algorithmic modules to help the IP approach as our main solution method.

The algorithm presented in this paper is tested on nine different instances published in Burke et al (2008b). The diversity in the structure and complexity of these instances allows us to test our algorithm thoroughly. Table 1 provides more information about these instances, where the reported number of variables and constraints are based on the described IP model presented in Section 3. It is noteworthy to mention that although we try to solve a few instances based on real-world cases, we develop our solution method without any fine tuning. Therefore, we believe that our approach can be easily generalized to solve different instances based on the presented models.

In the following, we provide a brief description of the performance of the hybrid algorithm, and later we will elaborate each associated component. After a quick pre-processing in order to create appropriate data structures for the algorithm, at first step, we employ an IP pre-solver in order to identify any valuable information. If any valuable information is identified, we continue to use the IP solver (rather than a CP solver) for the next steps, since it has more potential to be successful in solving the problem, as we have observed in our preliminary experiments. In the next step, we employ a CP solver to solve the problem considering only those constraints which will not make the problem difficult to solve. Identifying difficult constraints is achieved by solving a hierarchy of different CSPs iteratively. Next, using the information provided by the CP solver operated on a modified version of the problem and generated CSPs, we solve the problem by an IP solver (or the CP solver based on the obtained information from the IP pre-solver) in the remaining time. We also add three other components to reinforce the search process using the exploited problem-specific information: 1) Symmetry breaker, which tries to remove (or mitigate) the symmetric structures; 2) Weight balancer, which tries to modify each constraint's weight based on a pre-defined threshold in order to tighten the problem formulation; and 3) Decomposer, which aims to provide a strong lower bound for the IP solver.

It should be noted that the proposed hybrid algorithm runs in a pre-defined time to solve the problem. In fact, the user determines the running time of each component by setting the relevant computational time parameter.

The schematic diagram of the proposed algorithm is depicted in Figure 1. Next, we explain each component individually in more details:

IP Pre-solver: In fact, this component is the first step in most of the commercial solvers to analyze and simplify the problem structure, and also identify any specific structures such as network flow or assignment problems (Gurobi Optimization, 2015). If the IP solver can identify any particular structures, it often leads us to a better performance during the search process. 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 to predict if there are any specific structures, and therefore improving the performance of the IP solver. Particularly, we use the obtained lower bound and relaxed objective function value to predict the existence of any specific structures in this black-box indirectly. According to our experiments, if the IP pre-solver component provides a stronger (greater in our problem settings) lower bound compared to the relaxed objective function value (which is obtained by relaxing all integer constraints), the employed IP solver is a better choice to solve the problem most probably due to the identification of a specific data structure, otherwise we will use the CP solver instead. We also switch on the relevant parameter for the pre-solve step of the IP solver to the highest degree (aggressive mode) in this component (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 a user-defined threshold (*psThr*), we will change the search strategy of the IP Solver accordingly. We will explain this setting in the IP Solver component in more details.

CP Solver: During the search process, the hybrid algorithm may call the CP solver in two cases: first, as the main solver if the IP pre-solver does not provide strong enough lower bound for the problem due to its complex and highly-constrained structure; second, as an aid for the IP solver to provide a good-quality initial solution. This solver solves the problem based on the Constraint Satisfaction Problem (CSP) model presented in Section 4. In our experiments on the benchmark instances, CP approach did not provide very good-quality solutions in a limited computational time (see Table 3). To address this issue, we implement the following procedure: First, we generate a CSP model considering all constraints that have a weight higher than a user-defined threshold (*cspThr*). If the problem is infeasible, we will in-

Table 1 Benchmark instances and the relevant characteristics

Instance	Nurses	Shift Types	Days	Shift Permutations	Variables	Constraints
GPOST	8	3	28	3136	5680	5504
GPOSTB	8	3	28	3136	5680	5496
ORTEC01	16	5	33	7821	19096	19170
ORTEC02	16	5	33	7821	19101	19175
Valouxis-1	16	4	28	5824	9776	9968
SINTEF	24	6	21	6867	8118	6927
WHPP	30	4	14	5880	6000	5842
MILLAR-1	8	3	14	784	1956	1820
LLR	27	4	7	1323	1139	979

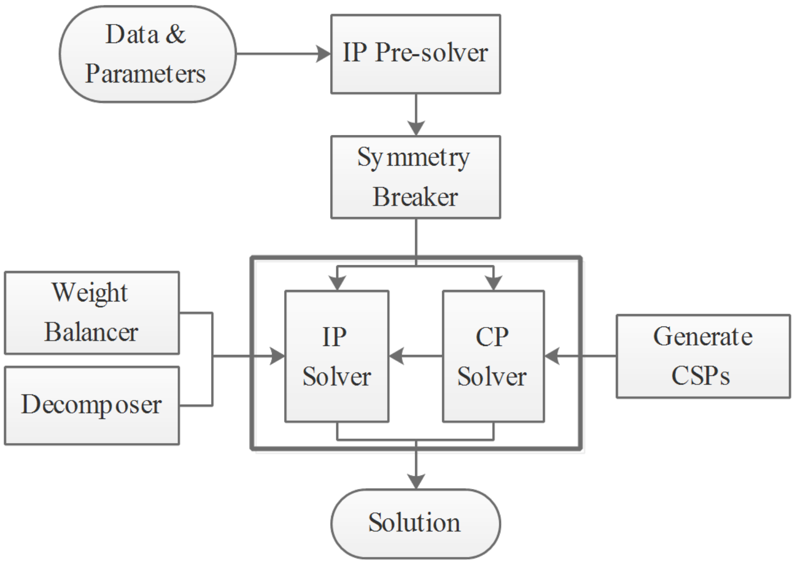


Fig. 1 Schematic diagram of the proposed hybrid algorithm

crease the threshold by one unit. Otherwise, we will generate a number of solutions based on the modified model according to a user-defined parameter (*numSols*). Therefore, on each threshold level, there might be several feasible solutions. This process continues until the quality of the best obtained solution in the current level is worse than by q percent in comparison to the best one in the last level. Finally, we report the best-quality solution in terms of the original model's objective function value. Next if the IP solver is a candidate for solving the problem, the reported solution will be imported to the IP solver. Otherwise, we continue solving the problem using the CP solver in the remaining time. We will explain this setting in the IP solver component in more details. The pseudo code of this procedure is presented in Algorithm 1, where p, p', *cspThr*, q, and *numSols* indicate the original problem, the new generated problem in each threshold level, the user-defined threshold level, the maximum gap between the quality of the best obtained solutions in two consecutive levels, and the user-defined number of solutions needs to be generated in each threshold level, respectively.

Using the information provided in this procedure by solving a variety of CSP problems, we can also find out an estimate for the computational difficulty of each constraint. If the solution time by removing a constraint from a problem in order to solve a new modified problem is significant, we will count it as a "difficult constraint". To our experiments, 10 seconds is sufficient for most of the benchmark instances. This simple inference helps us later in the Weight Balancer component to make the formulation of the problem tighter.

IP Solver: In this component, we use a state-of-the-art IP solver to solve the problem during the remaining time. The only difference between this component and running a pure IP solver is the initial solution and parameter settings provided to the solver supplied from other relevant components. We use the solution obtained from the CP solver as a warm start for the IP solver. Moreover, we change some parameters of the IP solver based on the information provided by the IP Pre-solver. Indeed, if the IP Pre-solver provides a strong enough lower bound (elaborated in the IP Pre-solver component), we switch off the pre-solve step in this component (e.g. setting the *Presolve* parameter in Gurobi). We also change the search strategy based on the number of constraints and variables provided by the IP Pre-solver, and a user-specified threshold, i.e. *psThr*. If the number of constraints and vari-

Algorithm 1: The procedure to generate CSPs in CP Solver component of the algorithm

```
Solutions = empty;
i = 0;
$p_i^{'}$ = p;
while true do
    $p_i^{'}$ = generateCSP(p, cspThr);
    if $p_i^{'}$ is feasible then
        for j = 1 to numSols do
            Solutions.add(solve($p_i^{'}$));
        end
    else
        cspThr++;
    end
    if (solutionObj($p_i^{'}$) - solutionObj($p_{i-1}^{'}$)) / solutionObj($p_i^{'}$)
    >= q then
        break;
    end
    i++;
end
return [bestObj(Solutions)]
```

ables of the problem are more than *psThr*, we set the search strategy to spend more efforts on obtaining a feasible solution rather than 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 change the search strategy by setting a specific parameter defined therein. For example, in Gurobi solver, the user can tailor the search strategy by setting the *MIPFocus* parameter. Furthermore, using the lower bound provided by the Decomposer component, we enforce it on the IP solver by setting the relevant parameter accordingly (e.g. setting the *Start* parameter in Gurobi).

Symmetry Breaker: As we mentioned in Section 2, modeling the problem using indexed variables can create symmetry issues. To resolve these issues, we add lexicographic ordering constraints (Beldiceanu et al, 2005; Smith, 2006) to both CP and IP models applied to the main variables (i.e. x_{ead} and s_{ed}, respectively), by taking into account "request on or off" constraints (constraint (9) described in Section 2). We then use the new model for both the IP and CP Solver components. In Section 6, we will mention that breaking a symmetric structure in the model is often beneficial for the solver.

Weight Balancer: In order to improve the efficiency of the IP solver during the search process, we modify the weights in the objective function due to the difficulty degree of constraints, which we obtained from the CP Solver component. Based on this degree, if a constraint is not difficult, we impose it to the IP solver as a hard constraint. Theoretically, this process may lead to an infeasible problem. In this case, we undo the relevant change 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, which often results in a better efficiency during the search process.

Decomposer: One of the design aspect of the proposed hybrid algorithm is to generate a good lower bound for most of the benchmark instances. In this component, we decompose the problem to weekly rosters, and then we evaluate all possible shift patterns according to "forbidden shift pattern" and "request on or off" constraints (constraints (8) and (9) described in Section 2). Indeed, we try to find out whether there is an inevitable conflict in the model, which can be discovered before solving the problem. When there is an inherent conflict in the model according to the current data, we can calculate the associated penalty based on the objective function and consider it as a new lower bound. We do this particular evaluations for all decomposed weekly rosters in a problem. This process is very similar to the one elaborated in Glass and Knight (2010), where the authors try to infer a lower bound for two specific instances. However, here we use the same technique but for all decomposed weekly rosters, and not only for particular instances. Apart from this process, we also solve all decomposed weekly rosters by an IP solver to discover any further potential lower bounds. Finally, the best lower bound calculated in this component will be imported to the IP solver by setting the relevant parameter (e.g. setting the *Start* parameter in Gurobi).

6 Computational Results

To evaluate the proposed hybrid algorithm, we implemented our algorithm in Java 1.7, and used the IBM ILOG CP solver 1.7 and Gurobi IP solver 5.0 for solving all CSPs and IPs, respectively. The reason to use the aforementioned solvers is that we found them easier to implement in terms of modeling, and also they suit our hybrid framework better than other software packages. In addition, we note that the benchmarks reported in Mittelmann (2008) show that Gurobi produces very similar results for most of the instances comparing to other state-of-the-art IP solvers such as IBM ILOG Cplex. We run our experiments on a PC with Intel 3.4 GHz processor and 4 GB of RAM, and we used the benchmark instances introduced in Section 5. The variety in benchmark instances helps us to test and analyze our algorithm in different circumstances. To the best of our knowledge, we are one of the few researchers experimenting with all these instances altogether. Furthermore, we run all our experiments on one CPU core to have a fairer and more accurate comparison.

For evaluation purposes, we run the hybrid algorithm for 10 minutes, and distribute 10%, 30%, and 50% of the computational time to IP Pre-solver, CP Solver, and IP Solver components, respectively. The remaining time is distributed equally to other components as they require very short times in comparison. The reasons for benchmarking the proposed

Table 2 The hybrid algorithm results for different settings

Instance	Default Setting			No Symmetry Breaker			No Weight Balancer		
	Obj.	LB	G(%)	Obj.	LB	G(%)	Obj.	LB	G(%)
GPOST	5	5	0	8	5	37.5	5	5	0
GPOSTB	5	0	100	3	0	100	5	0	100
ORTEC01	380	158	58.42	530	140	73.58	680	140	79.41
ORTEC02	370	150	59.46	570	140	75.44	340	140	58.82
Valouxis-1	20	10	50	20	10	50	20	0	100
WHPP	5	0	100	5	0	100	5	0	100

algorithm in 10 minutes are two-fold: 1) we primarily designed the hybrid algorithm to perform in a short time; 2) 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 the first International Nurse Rostering Competition (INRC-I) (Haspeslagh et al, 2014), and hence provides a platform for a fair comparison. Furthermore, we set the threshold parameters for the IP Pre-solver and CP Solver components, i.e. *psThr* and *cspThr*, to 10000 and 10 respectively. We also set the *q* and *numSols* parameters for the CP Solver component to 5 and 500, respectively. The design of the algorithm is primarily deterministic, however to address the minor random behavior due to the intrinsic nature of the employed solvers, we run it three times per instance for each experiment and report best values.

We conduct two experiments to test the proposed algorithm: first, we investigate the benefit and efficiency of the Symmetry Breaker and Weight Balancer components, and how they affect the performance of the algorithm. Then, we compare the hybrid algorithm against pure IP and CP solvers as well as the five most recent best algorithms in the relevant literature.

The first experiment is designed to investigate the effects of breaking symmetry and modifying weights on overall performance of the hybrid algorithm. For each test, the best objective function value (*Obj.*), lower bound (*LB*), and duality gap which will be described later (*G(%)*) were recorded. The results are shown in Table 2. It should be noted that the algorithm solved instances SINTEF, MILLAR-1, and LLR in less than 10 seconds, therefore, we only report the results for the rest of the instances (six instances) in this experiment.

The results of running the hybrid algorithm using all the components are indicated as *default setting* in the first part of Table 2. For the next two parts, we remove the Symmetry Breaker and Weight Balancer components, respectively. As it can be seen, having symmetry structures in the problem worsens the duality gap for three of instances, i.e. GPOST, ORTEC01, ORTEC02, whereas it does not change the duality gaps for instances GPOSTB, Valouxis-1, and WHPP. The reason to obtain the same results is because of the limited complexity in the structure of these instances. As a result, the hybrid algorithm solved them easily compared with the other instances, although they have symmetry issues. Therefore, Symmetry Breaker component seems to improve the efficiency of the hybrid algorithm in general, in particular for problems with a very complex structure.

In the third part, we remove only the Weight Balancer component. The results show similar duality gaps for all the instances except two increases for instances ORTEC01 and Valouxis-1, and a reduction for instance ORTEC02, which are not significant. Similar to the second part of Table 2, we obtained the same results for some instances due to the limited complexity in the structure of those instances. Consequently, we decided to include this component in default settings of the algorithm.

In order to compare the performance of the hybrid algorithm against the pure IP and CP solvers, we run the algorithm for 10 min, and report the best obtained solution (*Obj.*), lower bound (*LB*), duality gap (*G(%)*), and the solution (clock) time in seconds (T(s)) in Table 3. The duality 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 dual problem). When the duality gap is zero, the current feasible solution is an optimal solution (Wolsey, 1998). Column *BKS* shows the best known solution for the benchmark instances according to Burke et al (2008b), which were obtained using column generation and relaxation techniques with an IP solver or other heuristic methods for a long runtime. In the remaining parts of Table 3, the results for running the IP solver for 10 minutes (short runtime) and maximum 60 minutes (long runtime) as well as the CP solver for maximum 60 minutes are shown, respectively. It should be noted that for both IP and CP solvers, we run them in default settings without any tuning, though for the CP solver, we have reinforced the CP formulation described in Section 4 by adding some symmetry and redundant constraints, which have shown promise in our preliminary experiments (Smith, 2006).

Running the IP solver for 10 minutes, one can see that our algorithm found better solutions in all instances except instances MILLAR-1 and LLR, which both solvers obtain the same results due to the lower complexity in these instances. Furthermore, the algorithm improved the lower bounds for three instances, which is an evidence for the effectiveness

Table 3 Benchmark results for the hybrid algorithm versus pure IP and CP solvers

Instance	BKS	Hybrid Algorithm				IP (10 min)				IP (60 min)				CP (60 min)	
		Obj.	LB	G(%)	T(s)	Obj.	LB	G(%)	T(s)	Obj.	LB	G(%)	T(s)	Obj.	T(s)
GPOST	5	5	5	0.00	323	9	5	44.44	600	5	5	0.00	701	11	3600
GPOSTB	3	5	0	100.00	600	7	0	100.00	600	5	0	100.00	3600	8	3600
ORTEC01	270	380	158	58.42	600	1970	140	92.89	600	720	210	70.83	3600	–	3600
ORTEC02	270	370	150	59.46	600	15415	140	99.09	600	700	210	70.00	3600	830	3600
Valouxis-1	20	20	10	50.00	600	1090	0	100.00	600	90	0	100.00	3600	180	3600
SINTEF	0	0	0	0.00	6	8	0	100.00	600	0	0	0.00	694	0	1931
MILLAR-1	0	0	0	0.00	1	0	0	0.00	112	0	0	0.00	112	0	877
WHPP	5	5	0	100.00	600	24061	0	100.00	600	1004	0	100.00	3600	67050	3600
LLR	301	301	301	0.00	8	301	301	0.00	241	301	301	0.00	241	301	574

of Decomposer component. For instances MILLAR-1 and LLR, the hybrid algorithm found the optimal solution in a significantly shorter time in comparison to the IP solver.

To have a better and fairer comparison, we run the IP solver for a longer time, i.e. 60 minutes. It can be seen that the IP solver improves the duality gap only for three instances, i.e. GPOST, GPOSTB, SINTEF, in comparison to when it is run in a shorter time. Moreover, it improves the lower bounds for instances ORTEC01 and ORTEC02 considerably, but without making any changes for instance Valouxis-1. However, in overall, the hybrid algorithm achieves better duality gaps in five and three instances for short and long runtime, respectively, in comparison to the IP solver. It should be noted that for some instances such as ORTEC01, we do not obtain any optimal solution even after running the pure IP solver for more than 24 hours.

Similarly, running the CP solver with a long time limit of 60 minutes, we obtained the optimal solution for three instances and could not even reach to any feasible solution for instance ORTEC01. Therefore, the hybrid algorithm with 10 minutes time limit outperformed the CP solver in six instances. We note that we do not report the results for running the CP solver in 10 minutes since in most of the instances, it obtains very poor solutions.

To compare the performance of the current algorithm against the stat-of-the-art algorithms reported in the literature, Table 4 shows the best-published results from: a hybrid Variable Neighborhood Search (Burke et al, 2008a), denoted as VNS-1, a Memetic Algorithm (Burke et al, 2001), denoted as MA, a Variable Depth Search (Burke et al, 2013), denoted as VDS, a Harmony Search Algorithm (Hadwan et al, 2013), denoted as HSA, a Scatter Search (Burke et al, 2010), denoted as SS, and another hybrid Variable Neighborhood Search (Metivier et al, 2009), denoted as VNS-2. Unfortunately, to the best of our knowledge, we are not aware of any exact approaches tested on the studied instances and hence we did not include any in our benchmarking. We report the best results and their computational times (in seconds, except for VNS-1 which is in hour) in columns *Best* and *T*, respectively. Although we run all experiments only for 10 minutes, we report the computational times because for some instances the hybrid algorithm could find an optimal solution sooner.

As we can see, our proposed hybrid algorithm is able to outperform other algorithms for six instances, and obtained competitive results for instances ORTEC01 and ORTEC02. For instance WHPP, we could not find out any reported result in the literature other than the best known solution mentioned in Burke et al (2008b). Furthermore, for instances GPOST, SINTEF, MILLAR-1, and LLR, the hybrid algorithm obtained the optimal solution in a very short time compared to other algorithms. Comparing the results of our algorithm with the Scatter Search, for instances GPOSTB, MILAAR-1, and LLR, we obtained the same results, but in a shorter computational time. For instances GPOST, Valouxis-1, and SINTEF, the hybrid algorithm found the best solutions, which are significantly better than the others.

It is worth noting that the proposed algorithm found the solutions reported in Table 4, while our aim of designing the hybrid algorithm was not only to find a good feasible solution, but also to achieve a better duality gap for ensuring solution quality. In fact, due to the heuristic nature of the benchmarked algorithms, none of them are able to identify a lower bound for the obtained solution.

7 Summary and Conclusion

This paper proposed a new hybrid algorithm combining IP and CP to solve real-world Nurse Rostering Problems. The algorithm utilized 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 constraints for 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 to

Table 4 Benchmark results for the hybrid algorithm versus other state-of-the-art algorithms reported in the literature: VNS-1 (Burke et al, 2008a), MA (Burke et al, 2001), VDS (Burke et al, 2013), HSA (Hadwan et al, 2013), SS (Burke et al, 2010), and VNS-2 (Metivier et al, 2009)

Instance	BKS	Hybrid Algorithm		VNS-1		MA		VDS		HSA		SS		VNS-2	
		Best	T(s)	Best	T(h)	Best	T(s)	Best	T(s)	Best	T(s)	Best	T(s)	Best	T(s)
GPOST	5	5	323			915	121					9	861	8	234
GPOSTB	3	5	600			789	95					5	791		
ORTEC01	270	380	600	541	12	535	1516	360	300	310	412	365	680		
ORTEC02	270	370	600							330	446				
Valouxis-1	20	20	600			560	593					100	800	160	3780
SINTEF	0	0	6			8	175					4	821		
MILLAR-1	0	0	1			100	8					0	182	0	1
WHPP	5	5	600												
LLR	301	301	8			305	38					301	423	314	79

generate a high-quality solution as well as a strong lower bound in order to guarantee the solution quality. Moreover, we provided both CP and IP models of the problem.

We tested our algorithm on a diverse test bed of nine real-world instances from the literature. We conducted two experiments to evaluate the effectiveness of different components of the proposed algorithm, and its performance compared to some state-of-the-art algorithms as well as pure IP and CP solvers. The results show that proposed algorithm is capable of obtaining competitive results.

Our future work will investigate different formulations for the Nurse Rostering Problems compared with the classical model consisting of indexed variables. We will also try to add a heuristic component to the proposed hybrid algorithm to improve its performance. Exploiting the problem-specific information, we will attempt to design a more sophisticated framework involving various heuristics and novel automation approaches, to accommodate different characteristic of the problem. Finally, we are going to extend our algorithm to other scheduling problems, and implement our techniques in a real hospital environment.

References

Beldiceanu N, Carlsson M, Rampon J (2005) Global constraint catalog. URL `http://eprints.sics.se/2366`

Brucker P, Burke EK, Curtois T, Qu R, Berghe GV (2010) A shift sequence based approach for nurse scheduling and a new benchmark dataset. Journal of Heuristics 16(4):559–573

Brucker P, Qu R, Burke E (2011) Personnel scheduling: Models and complexity. European Journal of Operational Research 210(3):467–473

Burke E, Cowling P, De Causmaecker P, Berghe GV (2001) A memetic approach to the nurse rostering problem. Applied Intelligence 15(3):199–214

Burke EK, De Causmaecker P, Berghe GV, Van Landeghem H (2004) The state of the art of nurse rostering. Journal of Scheduling 7(6):441–449

Burke EK, Curtois T, Post G, Qu R, Veltman B (2008a) A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. European Journal of Operational Research 188(2):330–341

Burke EK, Curtois T, Qu R, Berghe GV (2008b) Problem model for nurse rostering benchmark instances. Tech. rep., http://www.cs.nott.ac.uk/˜tec/NRP/papers/ANROM.pdf

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

Burke EK, Li J, Qu R (2012) A Pareto-based search methodology for multi-objective nurse scheduling. Annals of Operations Research 196(1):91–109

Burke EK, Curtois T, Qu R, Vanden Berghe G (2013) A time predefined variable depth search for nurse rostering. Tech. Rep. 3, School of Computer Science and Information Technology, University of Nottingham

Chuin Lau H (1996) On the complexity of manpower shift scheduling. Computers & Operations Research 23(1):93–102

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

Fung SKL, Leung HF, Lee JHM (2005) Guided complete search for nurse rostering problems. In: Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI, vol 2005, pp 706–707

Girbea a, Suciu C, Sisak F (2011) 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 pp 477–482

Glass Ca, Knight Ra (2010) The nurse rostering problem: A critical appraisal of the problem structure. European Journal of Operational Research 202(2):379–389

Gurobi Optimization I (2015) Gurobi. URL `http://www.gurobi.com`

Hadwan M, Ayob M, Sabar NR, Qu R (2013) A harmony search algorithm for nurse rostering problems. Information Sciences 233(0):126–140

Haspeslagh S, De Causmaecker P, Schaerf A, Stolevik M (2014) The first international nurse rostering competition 2010

van Hoeve W, Katriel I (2006) Global constraints. Handbook of constraint programming

IBM (2015) IBM ILOG CPLEX CP Optimizer. URL `http://www.ibm.com/software/integration/optimization/cplex-cp-optimizer/`

Kazahaya G (2005) Harnessing technology to redesign labor cost management reports. Healthcare financial management : journal of the Healthcare Financial Management Association 59(4):94–100

Laburthe F, Jussien N (2011) CHOCO solver - Documentation. URL `http://choco.mines-nantes.fr/`

Lu Z, Hao JK (2012) Adaptive neighborhood search for nurse rostering. European Journal of Operational Research 218(3):865–876

Maenhout B, Vanhoucke M (2010) Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. Journal of Scheduling 13(1):77–93

Metivier JP, Boizumault P, Loudni S (2009) Solving nurse rostering problems using soft global constraints. In: Gent I (ed) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 5732 LNCS, Springer Berlin Heidelberg, chap 9, pp 73–87

M'Hallah R, Alkhabbaz A (2013) Scheduling of nurses: A case study of a Kuwaiti health care unit. Operations Research for Health Care 2(1-2):1–19

Mittelmann HD (2008) Decison Tree for Optimization Software. Tech. rep.

Ozcan Y (2005) Quantitative methods in health care management: techniques and applications, vol 4. John Wiley & Sons

Rahimian E, Akartunali K, Levine J (2015) 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

Smith B (2006) Modelling for Constraint Programming. In: Rossi F, Van Beek P, Walsh T (eds) Handbook of constraint programming, 2006th edn, Elsevier, chap 11, pp 377–406

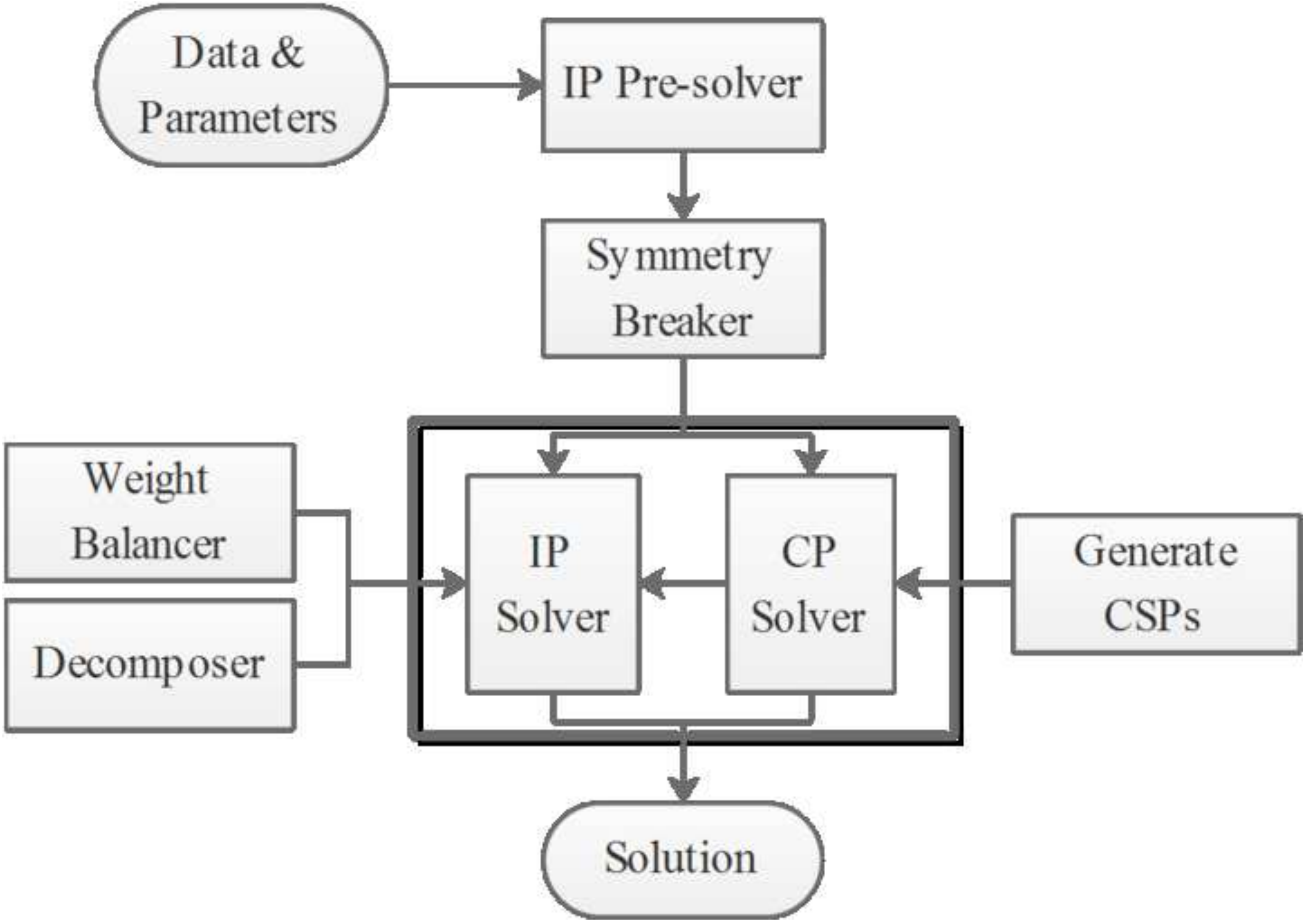
Soto R, Crawford B, Bertrand R, Monfroy E (2013) Modeling NRPs with Soft and Reified Constraints. AASRI Procedia 4(0):202–205

Stolevik M, Nordlander TE, Riise A, Froyseth H (2011) A hybrid approach for solving real-world nurse rostering problems. In: Lee J (ed) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 6876 LNCS, Springer Berlin Heidelberg, chap 9, pp 85–99

Valouxis C, Gogos C, Goulas G, Alefragis P, Housos E (2012) A systematic two phase approach for the nurse rostering problem. European Journal of Operational Research 219(2):425–433

Wolsey LA (1998) Integer programming. Wiley

Figure (png) Click here to download Figure Flowchart2.png


Data & Parameters
IP Pre-solver
Symmetry Breaker
Weight Balancer
Decomposer
IP Solver
CP Solver
Generate CSPs
Solution