Les Cahiers du GERAD ISSN: 0711–2440

A dedicated pricing algorithm to solve a large family of nurse scheduling problems with branch-and-price

A. Legrain, J. Omer

G–2023–02

January 2023
Revised: November 2023

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2023
– Bibliothèque et Archives Canada, 2023

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2023
– Library and Archives Canada, 2023

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A dedicated pricing algorithm to solve a large family of nurse scheduling problems with branch-and-price

Antoine Legrain [a, b, c]

Jérémy Omer [d]

[a] *Polytechnique Montreal, Montréal (Qc), Canada, H3T 1J4*

[b] *CIRRELT, Montréal (Qc), Canada, H3T 1J4*

[c] *GERAD, Montréal (Qc), Canada, H3T 1J4*

[d] *IRMAR, INSA de Rennes, Rennes, France*

antoine.legrain@polymtl.ca
jeremy.omer@insa-rennes.fr

January 2023
Revised: November 2023
Les Cahiers du GERAD
G–2023–02

Abstract : In this paper, we describe a branch-and-price algorithm for the personalized nurse scheduling problem. The variants that appear in the literature involve a large number of constraints that can be hard or soft, meaning that they can be violated at the price of a penalty. We capture the diversity of the constraints on individual schedules by seven generic constraints characterized by lower and upper bounds on a given quantity. The core of the column generation procedure is in the identification of indivual schedules with minimum reduced cost. For this we solve a shortest path problem with resource constraints (SPPRC) where several generic constraints are modeled as resource constraints. We then describe dominance rules adapted to the presence of both upper and lower bounds on the resources and leverage soft constraints to improve the dominance. We also describe several acceleration techniques for the solution of the SPPRC, and branching rules that fit the specificities of the problem. Our numerical experiments are based on the instances of three benchmarks of the literature including those of the two international nurse rostering competitions (INRC-I and INRC–II). Their objective is three-fold: assess the dominance rules and the acceleration techniques, investigate the capacity of the algorithm to find provable optimal solutions of instances that are still open, and conduct a comparison to best published results. The most noticeable conclusion is that the improved solution of the SPPRC allows to solve optimally all the INRC–II instances where a 4-weeks planning horizon is considered and 40% of the 8-weeks instances.

Keywords : Nurse scheduling, column-generation, decomposition, branch-and-price, dynamic programming, soft constraints

1 Introduction

1.1 The nurse scheduling problem

Personalized nurse scheduling problem (NSP) is more important than ever in this post-pandemic context, where important nurses shortages still exist and have even increased during the pandemic in most western healthcare systems. Proposing high-quality schedules that fit the nurses preferences while ensuring the right coverage is a key-element to help managers retain nurses in their care units. Moreover, these schedules need to cover an horizon long enough, typically 4 to 8 weeks, to let the nurses plan their own personal time based on their work schedules.

The NSP aims to design a schedule that meets the needs of nurses and operational constraints while minimizing costs. Each day is divided into multiple *shifts*, and each nurse possesses a specific *skills* set that corresponds to assigned tasks. Nurses are assigned to a shift based on their skills, and days without any assignment is called a *day off*. The NSP involves creating a *roster* for each nurse that outlines their work schedule and days off for the planning horizon. Most rostering problems aim to balance workloads among employees while ensuring that individual schedules comply with employment regulations and contractual obligations. As a result, the cost of the entire schedule typically factors in an assessment of the overall quality of service it offers, along with penalties linked to imbalanced customized rosters and breaches of contracts and labor regulations. The quality of service is generally measured by assessing the demand for each shift on a daily basis, and a potentially significant number of constraints define the specifications for each customized roster.

The review article by (Cheang et al. 2003) lists 16 types of constraints that are commonly found in the literature. For instance, the workload of each nurse usually has to lie between a minimum and a maximum number of assignments on the planning horizon. It is also usual that the number of consecutive assignments be lower and upper bounded. Depending on the considered country and hospital, each constraint may be *hard*, meaning that a feasible solution necessarily satisfies it, or *soft*, in which case it can be violated at the the price of a penalty. As an example, the two international nurse rostering competitions (INRC), INRC-I (Haspeslagh et al. 2014) and INRC–II (Ceschia et al. 2019), specify two rostering problems where most constraints are soft. The hard constraints of the INRC instances specify that nurses can perform only one assignment per day, that they need the corresponding skill to perform a task and that a minimum number of nurses must be assigned to each shift. Up to 15 other soft constraints then characterize the required working conditions of each nurse. It should be noted that the INRC–II problem originally considered a stochastic NSP where the demands are dynamically revealed on a weekly basis, but the literature has often focused on its *static* version where all data is known in advance.

The personalized aspect of nurse schedules contributes greatly to the complexity of the NSP. The same schedule can have varying costs depending on the nurse receiving it due to differences in contracts or preferences for days off. In contrast, workforce scheduling problems are often solved using anonymous rosters.

1.2 Main contributions

The purpose of the research described in this article is to implement a branch-and-price algorithm that can handle most of the NSP instances described in the literature while finding provable optimal solutions for a large number of instances that are still open. Up to now, most static INRC–II instances studied in the literature were still open, and so were some INRC-I instances. Moreover, most studies have focused on specific instances and either the corresponding code has not been shared publicly or it is dedicated to the specific constraints considered in these instances.

Algorithmic contributions In the development of our algorithm, we have looked into opportunities of improvements in every components of the branch-and-price algorithm, but we have drawn a particular

focus on the generation of individual rosters. As a consequence, we have designed a new dominance rule that better takes into consideration both upper and lower bounded resources, in particular soft ones, in the solution of the shortest path problem with resource constraints (SPPRC) with dynamic programming. Moreover, the literature on generic SPPRC, on the NSP and on other related problems provides a large number of algorithmic features that sometimes produced a significant speed-up in the solution of the pricing problem. We have implemented the most promising among them and assessed their impact on a set of INRC instances. Finally, we have also implemented strong-branching, diving heuristic, and a very promising rotation MIP heuristic that produces high quality solutions in parallel of the branch-and-price resolution.

Software design contributions We designed a generic representation of the constraints so that each generic constraint can capture a large set of operational constraints including those described for the INRC competitions. These generic constraints should also be sufficient to take into consideration any type of constraints listed by Cheang et al. (2003). Moreover, considering that other needs may appear in the future, we opted for a code structure that should allow future users to integrate them in the solver. The already developed generic constraints provide many examples to make this task easier.

The proposed algorithms are implemented in C++ and depend only on free and open third party libraries. The resulting code is publicly shared (Legrain and Omer 2023) for reproduction of the results, future comparisons, improvements and extensions. To the best of our knowledge, it is the first open-source code that can handle all the hard and soft constraints of the INRC instances.

Computational results Our tests are based on the three benchmarks that were most commonly used in the literature: the Nurse Rostering Problem (NRP)[1] and INRC-I benchmarks and the static INRC–II instances that have already been considered in the literature. Those benchmarks capture a large diversity of the hard and soft constraints found in the literature. They involve schedules of 8 to 150 nurses over a planning horizon of 2 to 52 weeks, where the nurses can have up to four different skills and each day is divided into up to 32 shifts. Our algorithm is able to solve every instance of INRC-I except one, all static 4-weeks instances of INRC–II, and several 8-weeks instances of INRC–II to optimality. It is also able to find the best known solutions for 19 over 24 NRP instances and it improves two lower bounds with respect to the literature. For some of these results, we had to distribute several tasks on up to eight cores and set a the time limit to one day. Moreover, the numerical tests are an opportunity to report on the experimental comparison of roster and rotation-based formulations on the INRC–II instances.

1.3 Organization of the article

The generic constraints are first detailed in Section 3. We then give the outline of our branch-and-price algorithm in Section 4. In there, we also provide a generic extended formulation of the NSP and describe our branching strategies and the primal heuristics that we developed to improve the upper bounds found by branching. The solution of the pricing problem with a specific dynamic programming algorithm is then described in Section 5. Finally, in Section 6, we report the experimental results of our algorithm, we assess the impact of the acceleration techniques we implemented and carry a comparative study with the best published results.

2 Solution of the nurse scheduling problem

Most approaches that were used to tackle NSP instances are based on metaheuristics or mathematical programming techniques. Among them, several methods solely based on metaheuristics have shown their capacity to compute good rosters in a small computational time on a large variety of benchmarks (see *e.g.* (Burke et al. 2013, Ceschia et al. 2020, Abuhamdah et al. 2021, Abdelghany et al. 2021)).

[1] Instances and best results can be found here http://www.schedulingbenchmarks.org/nrp/index.html

However, the results of the two INRCs emphasized that these approaches were outperformed by methods based on integer programming formulations of the problem (Ceschia et al. 2020, Haspeslagh et al. 2014) except for instances with very long horizons (16 and 32 weeks) (Ceschia et al. 2020). Another limit of these approaches is that they do not provide lower bounds on the optimal value, so they cannot prove the optimality of the solutions by themselves.

The mathematical programming approaches can be categorized in two families. In the first one, the problem is formulated as a compact integer linear program (ILP) where the variables refer to working assignments and resting days. This has been investigated by Römer (2016) to successfully handle the stochastic problem described in INRC–II. Santos et al. (2016b) have also been able to find provable optimal solutions of many INRC-I instances and improve on the optimality gap of others. However, eleven instances remain open.

The method that we describe in this article belongs to the second family, where the problem is modeled as an extended formulation where variables refer to individual rosters or sequences of working assignments. This formulation generally relies on a *column generation* procedure embedded in a *branch-and-price* algorithm to get integer solutions. These methods, possibly coupled with matheuristics, have been able to produce the best results on most benchmarks. Below, we draw a detailed review of the previously developed branch-and-price approaches for the NSP and related problems, with a particular focus on the computation of individual rosters.

2.1 Branch-and-price approaches

In most extended formulations of the NSP, the variables of the linear problem correspond to rosters and the linear constraints include partitioning equalities or covering inequalities, each of which guarantees that a necessary task is performed by an adequate number of nurses. Due to the lengthy planning horizon and numerous tasks that each nurse may perform, it is unfeasible to enumerate all columns in advance and resolve the resulting ILP with branch-and-cut. To overcome this numerical challenge, the initial step is to limit the number of variables by considering a small set of rosters. At every node of the branch-and-bound tree, the linear relaxation is resolved through column generation. This leads to the development of the branch-and-price algorithm, where new rosters are typically generated based on SPPRC solutions. Refer to (Barnhart et al. 1998) for detailed information regarding the implementation of branch-and-price approaches.

For the SPPRC, one considers a network where vertices represent the tasks to complete and arcs the possible transitions. Each arc is weighted with a cost and its traversal consumes certain resources. The SPPRC then consists in finding the minimum cost path between two artificial origin and sink vertices such that the resources consumed along the path lie within given bounds. Either a new column of negative reduced cost is generated or it is proven that the optimal solution of the restricted linear relaxation is optimal for the non-restricted version. An overview of column-generation algorithms can be found in (Desaulniers et al. 2006), and for a comprehensive review of SPPRC, (Irnich et al. 2005) can be consulted.

The initial effort to address the NSP using branch-and-price techniques can be traced back to 1998 (Jaumard et al. 1998). In their study, the authors establish a universal framework that addresses a wide range of constraints. To generate new columns, they resolve an SPPRC that considers as many as seven distinct constraints. The personalized nature of the issue is later emphasized by Bard and Purnomo (2005). This approach prompts the generation of fresh schedules through a swap heuristic and subsequent feasibility checks a posteriori. In (Maenhout and Vanhoucke 2010), a heuristic is initially executed in an attempt to generate negative reduced cost rosters, and the SPPRC is only solved if the attempt is unsuccessful. Furthermore, they compare over fifteen different branching rules on one or more variables. In Burke and Curtois (2014), columns are generated by solving an SPPRC using heuristic dynamic programming methods. They report outstanding results on the INRC-I benchmark. The problem of pricing can also be tackled with constraint programming (He and Qu 2012). To handle

static INRC–II instances with a 4-weeks horizon, Gomes et al. (2017) improve their branch-and-price approach with a variable neighborhood search. The authors implemented a classical decomposition strategy, focusing on the roster of each nurse.

Another approach to addressing the problem is inspired by airline crew scheduling problems. Similarly, in (Kohl and Karisch 2004), each column represents a rotation or a sequence of flights that begin and end at the same base. This approach identifies the rostering problem as a task of selecting a sequence of rotations to create a roster for every crew member. Legrain et al. (2020) adapted this idea to the NSP by defining a rotation as a sequence of working assignments preceded and followed by a sequence of days off. The subproblems thus generate such rotations, while the master problem builds a complete roster for each nurse by selecting a sequence of rotations that are separated by at least one day off. They also enriched their branch-and-price algorithm with a large neighborhood search to find good feasible solutions in limited time. This approach produced the best results for a large number of INRC–II instances, as shown in the recent experimental comparison carried out by Ceschia et al. (2020). In his PhD thesis, Lensing (2020) reports that his implementation of this rotation decomposition is outperformed by the classical roster decomposition on the benchmark proposed by Curtois and Qu (2014), where most constraints are hard.

2.2 Generation of individual rosters

A key step of the branch-and-price algorithms for NSP is the generation of individual rosters with negative reduced costs. In our previous works on the subject, we have observed that this is the part of the algorithm where most of the computational effort is spent. In most works of the literature, this is done by solving an SPPRC in a graph where the nodes stand for possible assignments or resting days and where the arcs represent allowed transitions. The specificity of the NSP is that resource constraints may be soft and that they specify lower and upper bounds on the same quantity. While describing the first dynamic programming approach to the SPPRC, Joksch (1966) mentioned that in the presence of two inequalities imposing upper and lower limits on a resource consumption, a path can be identified as less efficient than another only if the resource consumption is the same along both paths. He then stressed that dynamic programming approaches would certainly not be computationally worthwhile unless many paths share the same resource values. Later, the dynamic programming recursion proposed by Beasley and Christofides (1989) also kept one path for each possible value of the resources, thus leading to the same computational issue.

The research on SPPRC has often been driven by the study of vehicle routing problems (VRPs). In this context, when both lower and upper bounds are considered, they are related to time windows $[a_i, b_i]$ where each node i must be visited. In this application, it is usually possible to wait at a node before serving it. As a consequence, the lower bound of a time window will not impact the validity of a path arriving at node i earlier than a_i. Instead, it impacts the departure time from node i: a vehicle reaching i before a_i will wait until a_i to serve it and then leave node i. Classical time-windows are enforced as hard constraints, but some works considered soft time windows. In the latter variant, waiting is still possible without extra cost at any node, but if a node is served earlier or later than its time window a linear penalty must be paid. A section is devoted to soft time windows in the review article by Costa et al. (2019). Sexton and Choi (1986) were the first to consider soft time windows in a VRP. Those were introduced to distinguish working time windows, outside of which a customer could not be served, and the preference time slots of the customers, which can be violated by paying a penalty. As emphasized by Liberatore et al. (2011), the greatest difficulty that they raise is in the price of earliness. A compromise has to be found between earliness penalty and tardiness in subsequent nodes, hence dominance happens a lot less when solving the subproblems with label-setting algorithms. To mitigate this issue, Liberatore et al. (2011) associate a non-increasing linear stepwise cost function with each label to represent an infinite number of states where the other resources' consumption and the list of visited nodes are the same. This prevents from having one label for each possible waiting

time at each node. A similar approach is followed by (Bettinelli et al. 2014) and (Abdallah and Jang 2014).

Unfortunately, given that waiting is not relevant in general for resource constraints, the above cannot be generalized to nurse scheduling. In contrast, Tagmouti et al. (2007) have studied a variant of the traveling salesman problem where a time-dependent service cost is given by a convex function of the arrival time at each node. They solve the problem with a column-generation algorithm where the subproblem is a shortest elementary path problem with one resource constraint. Their algorithm is a direct adaptation of the classical label-setting algorithm where the cost of a partial path is given by the sum of the arc traveling costs and the service costs. In (Qureshi et al. 2010), early arrivals are penalized linearly. The authors clearly emphasize that soft constraints result in a dramatic increase of the computational time that is required to find optimal paths, especially when early arrivals are penalized. But in every case, the soft constraints yield no more adaptation than in the computation of the cost of a partial path. When early arrivals are penalized, the shortest path problem is solved with a genetic algorithm heuristics (Qurashi et al. 2010).

In the context of NSP, only a few works have handled soft constraints in a branch-and-price approach where the pricing problem is solved by a label-setting algorithm. For instance, Strandmark et al. (2020) developed a heuristic column generation approach, but the algorithm does not produce lower bounds because their SPPRC does not capture the soft roster constraints. In contrast, Burke and Curtois (2014) straightforwardly adapted the classical dominance rule to handle lower bounds: a label can dominate another label only if the consumption of every upper bounded resource is smaller or equal and if the consumption of every lower-bounded resource is larger or equal. As a consequence, dominance happens only when the consumption is equal for the resources that are both upper and lower-bounded. In (Legrain et al. 2020), the network of the SPPRC is modified to reduce the pricing problem to a standard SPPRC with only one upper bounded resource. For this, short sequences of assignments are enumerated, and the nodes and arcs are duplicated into layers where the number of consecutive assignments to the same shift can be tracked. Dominance is much more efficient with this approach, but the network is much larger. Moreover, dominance is not cross-checked between the nodes that belong to different layers but correspond to the same assignment.

3 Generic constraints for a general NSP

In the remainder of the article, we respectively denote as $\mathcal{N}$, $\mathcal{D}$, $\mathcal{S}$ and Σ the sets of nurses, days, shifts and skills. The number of days in the planning horizon is denoted as $n_{\mathcal{D}}$, therefore we set $\mathcal{D} = \{1, \ldots, n_{\mathcal{D}}\}$. Given that some constraints apply specifically to weekends, we also introduce $n_{\mathcal{W}}$ the number of weekends over the horizon. An *assignment* is then given by any pair (d, s) where $d \in \mathcal{D}$ and $s \in \mathcal{S}$. For a more concise presentation of our models, we include in $\mathcal{S}$ a Rest shift that corresponds to no assignment on a given day. Stated otherwise, an assignment (d, Rest) corresponds to a day off whereas (d, s) is a working day, for any $s \neq \text{Rest}$. We will also refer to Rest as the *rest shift* and to the others as *work shifts*.

Browsing the literature on the NSP, it quickly appears that each benchmark is based on a specific set of constraints. Cheang et al. (2003) have listed the 16 different families of constraints they could find in the articles published before their review. One purpose of this work is to implement an algorithm that will be able to run on most (if not all) instances of the literature provided that a proper parser is implemented. For this, we defined a set of generic constraints that are able to capture most specific constraints that we encountered in the literature.

The first step is in the definition of generic structures for the shifts, days, weekends and patterns, which are most commonly targeted by the constraints.

- A *shift type* is any set of shifts, *i.e.*, any set $S \subseteq \mathcal{S}$. For instance, a shift type may either stand for an actual working shift such as a work period from 9 a.m. to 5 p.m., or it may refer to any working shift, the rest shift or even any afternoon shift.
- A *day type* is any set of days, *i.e.*, any set $D \subseteq \mathcal{D}$. A day type may thus refer to Monday, May 28, 2022, to any Monday, or to any day. In the remainder, we will thus use the days of the week Monday, Tuesday, etc., to refer to the corresponding day types.
- A *weekend type* is given by any sequence of consecutive days of the week, *e.g.*, Saturday-Sunday, Friday-Saturday-Sunday or Saturday-Sunday-Monday.
- Finally, a *pattern* Π is given by tuple of pairs formed of one day type and one shift type, *i.e.*, $\Pi = ((D_1, S_1), \ldots, (D_K, S_k))$ where K is the length of Π and $D_k \subseteq \mathcal{D}$ and $S_k \subseteq \mathcal{S}$ for $k = 1, \ldots, K$.

Given these structures, we divided the generic constraints into two groups: those that define a feasible individual nurse schedule and its cost, and those that impact the selection of one schedule per nurse. This decomposition is mostly justified by the column generation procedure we have chosen where the master problem includes the second group while the subproblems include the first one. Each generic constraint can then either be hard or soft. In case it is soft, a penalty is associated to its violation that is generally defined by either the problem description or the instance. When a quantity is both upper and lower bounded, the penalties may be different for the upper and lower bounds.

In short, the master problem constraints are given below.

1. *Coverage*: a minimum number of nurses must work on each day and shift with each skill;
2. *Assignment*: each nurse must be assigned one roster, and they can only be assigned a skill that they possess.

The subproblems generic constraints are more numerous.

3. *Total shift types*: the total number of assignments of each nurse to a given shift type over the planning horizon is bounded. For instance, the total number of working days is usually upper and lower bounded. The total number of night shifts over the horizon may also be bounded with such constraint;
4. *Total worked weekends*: the number of weekends a nurse can work is bounded;
5. *Consecutive shift types*: the number of consecutive assignments to the same shift type is bounded. This constraint may refer to consecutive working or resting days, to consecutive specific shifts, etc.;
6. *Consecutive worked weekends*: the number of consecutive worked weekends is bounded;
7. *Identical weekend*: assignments on a weekend must all be to identical shift types. This can capture that a weekend must be either completely worked or completely rested, and that every assignment on a weekend must be to the exact same shift;
8. *Forbidden pattern*: the schedule of each nurse must not include a given pattern. For instance, it is usually forbidden to work a morning shift after a night shift, and it may be forbidden to rest less than two days after a sequence of night shifts;
9. *Preferences*: each nurse may have several individual wishes for their schedule. These may refer to a day that they want off or to a specific shift that they want to be assigned to on a particular day.

The above constraints allow to model the problems described in the two INRC competitions (which differ in several ways). They also capture the constraints used in the benchmark described by Curtois and Qu (2014), which has often been used in the literature. What is more, new constraints can be added to our implementation by defining one corresponding subclass. The condition is that they comply with the framework of our solution algorithm: the master problem needs to be formulated as an ILP and the subproblems as the search for a resource constrained shortest path.

4 Outline of the roster-based branch-and-price

4.1 Roster-based column generation

For all $i \in \mathcal{N}$, we denote as Ω_i the set of all feasible rosters for i, where each roster $j \in \Omega_i$ is characterized by a binary vector a^j such that $a^j_{d,s} = 1$ if and only if roster j includes assignment (d, s) for all $(d, s) \in \mathcal{D} \times \mathcal{S}$.

We formulate the NSP as the search for the minimum cost assignment of nurses to rosters under demand constraints. For this, we define three sets of decision variables: for $i \in \mathcal{N}$ and $j \in \Omega_i$, $x_{ij} = 1$ if roster j is assigned to nurse i and 0 otherwise; for $i \in \mathcal{N}$, $p \in \mathcal{D} \times \mathcal{S}$ and $\sigma \in \Sigma$, $y_{i,p,\sigma} = 1$ if nurse i performs skill σ on assignment p; and for $p \in \mathcal{D} \times \mathcal{S}$ and $\sigma \in \Sigma$, $z_{p\sigma}$ is the demand that is not satisfied on assignment p for skill σ. The vector of roster costs is given by $\mathbf{c_x}$, while the penalties corresponding to unsatisfied demands and to the assignment to a skill that a nurse does not possess are respectively denoted as $\mathbf{c_z}$ and $\mathbf{c_y}$. The latter costs are only consistent with soft constraints, but it is more convenient to treat hard and soft constraints in a unified way by assuming a large penalty when a constraint is hard. Also, the skill assignment cost $\mathbf{c_y}$ is zero when it corresponds to a skill that a nurse can perform. As a result, we may formulate the NSP as the following extended integer program, where $b_{p,\sigma}$ is the minimum demand on assignment p for skill σ. We also indicate the dual variables associated to assignment and demand constraints between brackets.

$$\min \quad \mathbf{c_x}^T\mathbf{x} + \mathbf{c_y}^T\mathbf{y} + \mathbf{c_z}^T\mathbf{z} \tag{1a}$$

$$\text{s.t.} \quad \sum_{j \in \Omega_i} x_{ij} = 1, \qquad \forall i \in \mathcal{N}, \qquad [\alpha_i] \tag{1b}$$

$$\sum_{j \in \Omega_i} a^j_p x_{ij} = \sum_{\sigma \in \Sigma} y_{i,p,\sigma}, \qquad \forall i \in \mathcal{N}, \forall p \in \mathcal{D} \times \mathcal{S} \qquad [\beta_{i,p}] \tag{1c}$$

$$\sum_{i \in \mathcal{N}} y_{i,p,\sigma} + z_{p\sigma} \geq b_{p,\sigma}, \qquad \forall p \in \mathcal{D} \times \mathcal{S}, \forall \sigma \in \Sigma \tag{1d}$$

$$x_{ij}, y_{ip\sigma} \in \{0, 1\}, \qquad \forall i, j, p, \sigma \tag{1e}$$

$$\mathbf{z} \geq 0 \tag{1f}$$

Given that it is not possible to enumerate the feasible rosters of a nurse in reasonable time for standard instances, we solve the above formulation by column generation. The algorithm starts with an initial set of rosters $\mathcal{R}_i \subseteq \Omega_i$ for each nurse i. It then considers a *restricted master problem* (RMP) that is obtained from Formulation (1) by replacing Ω_i by $\mathcal{R}_i$ for all nurses $i \in \mathcal{N}$. Assuming a feasible solution can always be obtained by adding artificial rosters at a prohibitive cost, we can suppose that the linear relaxation of RMP, known as RMP^{LR}, is feasible. At each iteration of the column generation procedure, RMP^{LR} is solved until optimality. Next, a *pricing problem* is solved using an optimal dual solution of this relaxation to find rosters of $\Omega_i \setminus \mathcal{R}_i$ that have negative reduced costs. If such a roster is generated, it will be added to the restricted formulation, otherwise, the optimal solution of the RMP^{LR} is proven to be optimal for the linear relaxation of Formulation (1).

The description of how an integer solution can be obtained by branching or by running a primal heuristic algorithm can be found in Appendix B as these are known techniques. However, we have also implemented a modified version of the classical MIP heuristic for column generation. More specifically, this heuristic searches for an integer solution by solving the restricted master problem obtained with the columns generated after the solution of the relaxation of the master problem. This is usually done at the root node, but it can also be done at any other branching node. However, compatible rosters are difficult to find, and solving the classical MIP may lead to solutions of poor quality (and, therefore greater than the current upper-bound), and even infeasibility. To overcome this challenge, we come back to the rotation-based formulation that allows more flexibility in the combination of rotations to obtain a good feasible solution. From a current solution of the roster-based formulation, all the

roster columns are cut into rotation columns and the associated MIP is then solved. In a parallel environment, this MIP can run permanently: as soon as a better solution is obtained, the new upper-bound is passed to the branch-and-price algorithm and a new MIP based on the current solution is then solved. This rotation MIP heuristic leads to the best solutions, as shown in the experiments. In the remainder of the section, we describe the pricing problem and sketch its solution algorithm.

4.2 Pricing rosters by dynamic programming

Construction of individual roster graphs The pricing problem consists in searching for individual rosters with negative reduced costs. Given that different nurses are not linked at this stage, pricing can be done independently for each nurse. For this, we consider an acyclic directed *roster graph* $G_i = (V_i, A_i)$ for each nurse i, where

- V_i includes two artificial origin and sink vertices o and t, and one vertex for each assignment $(d, s) \in \mathcal{D} \times \mathcal{S}$. For each $v \in V_i \setminus \{o, t\}$, we denote as $(\text{day}(v), \text{shift}(v))$, the assignment corresponding to v;
- A_i includes one arc from the origin to each vertex of the first day, one arc from each vertex of the last day to the sink, and for all days but the last one, one arc from each corresponding vertex to each vertex of the following day. More formally,

$$A_i = \{(o, v) : \text{day}(v) = 1\} \cup \{(v, t) : \text{day}(v) = n_{\mathcal{D}}\} \cup \{(u, v) : \text{day}(v) = \text{day}(u) + 1\}.$$

A small example is given in Figure 1.

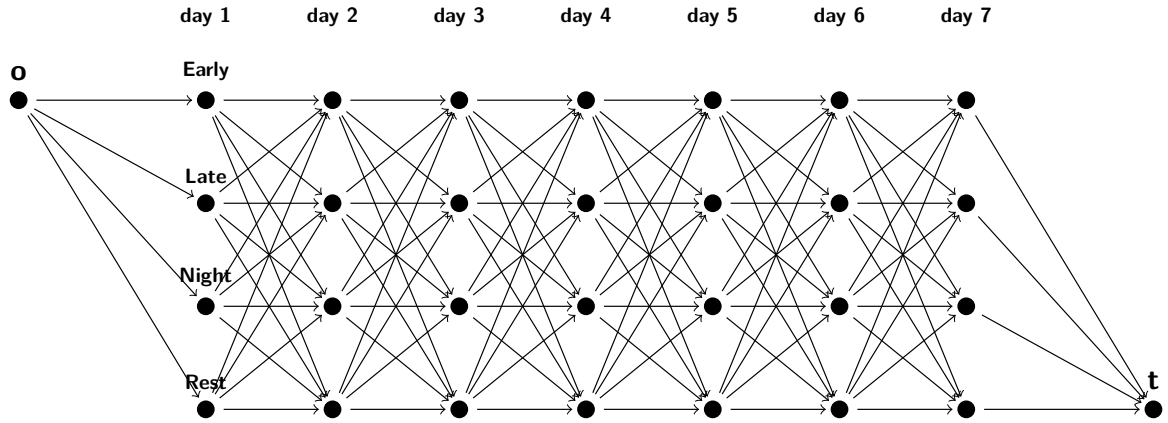

Figure 1: Example of a roster graph network. Here, the horizon includes 7 days and $\mathcal{S} = \{$Early, Late, Night, Rest$\}$.

A roster corresponds to a path from o to t in the roster graph. A feasible roster must also satisfy the individual roster constraints listed in Section 3. We take them into account by either modifying the roster graph or by considering resource constraints. In particular, the constraints on identical weekends, preferences and two-shifts long patterns may all be modeled by modifying the roster graph. For instance, a hard day-off preference is considered by removing all the arcs going to a working shift node on this day, while the same soft preference will lead to adding the corresponding penalty to the cost. Similar modifications can be done on the arcs joining vertices that correspond to different shift types on weekends for the identical weekends constraints.

In contrast, modeling the constraints on total working time/weekends, consecutive shift types/weekends and long forbidden patterns cannot be done without adding a large number of vertices and arcs to the graph. This has been done for instance in (Legrain et al. 2020) in a rotation-based decomposition. But in their approach, they could enumerate the rotations whose lengths were smaller than the minimum number of worked shifts, because this only included rotations with length smaller

than three. For rosters, the lower bound may be close to the number of days of the horizon, which would lead to intractable enumeration. As a consequence, these constraints are modeled with a set of resource constraints denoted as $\mathbf{R}$. With classical resource constraints, an $o-t$ path is feasible if the total consumption along the path is at most equal to a given upper bound. Here, the constraints may be soft, lower bounds also need to be considered, and the consumption of resources associated to consecutive shifts and forbidden patterns cannot be represented by a single consumption value on each arc. We will give more details on this matter in Section 5. In the remainder of this section, we give an abstract description of the pricing solution algorithm.

With the above modifications of the roster graph and the resource constraints $\mathbf{R}$, a feasible path and its cost correspond to a feasible roster and its cost. To get the reduced cost of a roster, we also need to model the dual costs of the restricted master problem constraints. Consider nurse $i \in \mathcal{N}$ and a roster $j \in \Omega_i$ with cost c_j that covers assignments $(p_1, \ldots, p_l)$. Referring to Formulation (1a)–(1f), we compute the reduced cost of j as:

$$\overline{c}_j = c_j - \alpha_i - \sum_{k=1}^{l} \beta_{i,p_k}. \tag{2}$$

To make sure that the cost of any path in the roster graph is equal to the reduced cost of the corresponding roster, we thus subtract α_i from the costs of the arcs going out from the source and we subtract $\beta_{i,p}$ from the costs of the arcs going out from the vertex corresponding to each assignment $p \in \mathcal{D} \times \mathcal{S}$.

Search for a resource-constrained shortest path Following the above construction of the roster graph G, the search for a negative reduced cost roster is equivalent to the search for a negative cost resource-constrained path in G. Searching for the minimum cost feasible path, we thus get a negative reduced-cost roster for nurse i or prove that none exists. Most state-of-the-art approaches for the SPPRC are based on a dynamic programming approach. The algorithm starts with the partial path containing only vertex o before extending it along each arc outgoing from o. The algorithm then proceeds by extending the partial paths obtained at the last iteration along the arcs outgoing from their end vertices until t is reached. Instead of recording each partial path, the algorithm only stores its state as given by its costs, resources consumptions and the last arc it traversed. The efficiency of the algorithm is then determined by its ability to keep only a limited number of *efficient* partial paths at each vertex by removing those that are *dominated*. For a more explicit characterization of extension and dominance, we start with some formal definitions.

Definition 1. Let $v \in V_i$ and let $P = (o, v_1, \ldots, v_k)$ be a path from o to v_k with length $|P| = k$. We denote as $\gamma^0(P)$ the cost of P. For any resource $r \in \mathbf{R}$, $\gamma^r(P)$ is the consumption of r along P.

Extension Let $(v_k, v_{k+1}) \in A$, the extension of P along $(v_k, v_{k+1}) \in A_i$ is the path $[P, v_{k+1}] = (o, v_1, \ldots, v_k, v_{k+1})$.

Completion A completion of P, $[P, \overline{P}]$, is a path from o to t. The path $\overline{P}$ is a *suffix* of P.

Feasibility Path P is feasible if and only if every hard resource constraint is satisfied along P.

Dominance Let Q be a second path from o to v. Path P dominates Q if and only if any feasible completion of Q, $[Q, \overline{Q}]$, yields a feasible completion of P, $[P, \overline{Q}]$, such that $\gamma^0\left([P, \overline{Q}]\right) \leq \gamma^0\left([Q, \overline{Q}]\right)$, and if there is a feasible completion of Q, $[Q, \tilde{Q}]$ such that $\gamma^0\left([P, \tilde{Q}]\right) < \gamma^0\left([Q, \tilde{Q}]\right)$.

Equivalence Path Q is equivalent to P if and only if any feasible completion of Q, $[Q, \overline{Q}]$, yields a feasible completion of P, $[P, \overline{Q}]$, such that $\gamma^0\left([P, \overline{Q}]\right) = \gamma^0\left([Q, \overline{Q}]\right)$. The set of equivalent paths from o to v defines an equivalence class.

To illustrate the above definitions, we may consider the classical SPPRC where each resource constraint corresponds to an upper bound on the resource consumption. Denoting as U_r the upper bound of each resource r, P is infeasible if there is r such that $\gamma^r(P) > U_r$. If P is feasible, path P dominates another feasible path Q if and only if $\gamma^0(P) \leq \gamma^0(Q)$, $\gamma^r(P) \leq \gamma^r(Q), \forall r \in \mathbf{R}$ and

$\exists r \in \mathbf{R}, \gamma^r(P) < \gamma^r(Q)$. Paths P and Q are equivalent if and only if $\gamma^0(Q) = \gamma^0(P)$ and $\gamma^r(Q) = \gamma^r(P), \forall r \in \mathbf{R}$.

From these definitions, it is straightforward to verify that only non-dominated paths need to be considered for extension, and that at most one representative of each equivalence class needs to be extended at each node. This leads to the generic solution algorithm below (Algorithm 1). The algorithm starts with a partial path including only the origin o, and it extends this path along each arc outgoing from o to build the set of partial paths $\mathcal{P}(v)$ that will be extended from neighbor of o. Since the graph is acyclic and structured into layers corresponding to the days of the planning horizon, the algorithm may then iterate over the days of the horizon to extend the non-dominated partial paths of each corresponding vertex. It may be noted though that the identification of all dominated partial paths requires a necessary and sufficient condition that cannot be verified in a short computational time in general. It may thus be useful to consider only a sufficient condition with the risk that some dominated partial paths will be uselessly extended. Section 5 will be devoted to the specialization of the algorithm for the specific constraints of the NSP.

initialization: $\mathcal{P}(o) := \{(o)\}$, $\mathcal{P}(v) := \emptyset, \forall v \in V_i \setminus \{o\}$.
for $(o, v) \in A_i$ **do**
  if path (o, v) is feasible **then** $\mathcal{P}(v) \leftarrow \{(o, v)\}$
for $d \in \mathcal{D}$ **do**
  for $u \in V_i(d)$ **do**
    Filter $\mathcal{P}(u)$ to keep only one path per equivalence class
    Filter $\mathcal{P}(u)$ by removing dominated paths
    for $P \in \mathcal{P}(u)$ **do**
      for $(u, v) \in A_i$ **do**
        Extend P along (u, v) to form $[P, v]$
        if $[P, v]$ is feasible **then** $\mathcal{P}(v) \leftarrow \mathcal{P}(v) \cup \{[P, v]\}$
if $\mathcal{P}(t) = \emptyset$ **then return** $\emptyset$
else return $\arg\min_{P \in \mathcal{P}(t)} \{\gamma^0(P)\}$

Algorithm 1: Dynamic programming search for a constrained shortest path in G_i

5 Solution of the shortest path with soft constraints

In this section, we provide a detailed description of the solution of the pricing problem. Given that each nurse can be considered independently in the pricing problem, we focus on a particular nurse i with rotation graph $G_i = (V_i, A_i)$. We denote as $\mathbf{H}$ and $\mathbf{S}$ the set of hard and soft individual constraints of nurse i. For a more concise computation of constraints' violations, we also denote as $[x]^+ = \max\{x, 0\}$ the positive part of any $x \in \mathbb{R}$. Finally, for every constraint $r \in \mathbf{R}$, we denote as U^r the corresponding upper bound and as L^r the lower bound (we can set $L^r = 0$ when there is none). If $r \in \mathbf{S}$, we also denote as c_r^U and c_r^L the linear penalties for violations of the upper and lower limits (respectively).

5.1 Representation of the NSP constraints as resources

As already stated in Section 4.2, the constraints on identical weekends, preferences and two-shifts long patterns may all be modeled by modifying the roster graph. We gave some examples of such modification, and the others are quite straightforward to derive. In the remainder, we show how the remaining constraints can be modeled as resource constraints.

In its most general definition, the SPPRC is a variant of the shortest path problem where a set of resources accumulate along paths and whose value is constrained at vertices and arcs of the path. As a consequence, the description of a resource must specify how it accumulates and how its value is constrained. For each resource $r \in \mathbf{R}$, the accumulation of resource is characterized by a set of resource extension functions (REF) $f_{uv}^r : \mathbb{R} \to \mathbb{R}, \forall (u, v) \in A_i$. To model soft constraints, we similarly define cost extension functions (CEF) $g_{uv}^r : \mathbb{R} \to \mathbb{R}, \forall (u, v) \in A_i$. For a given path P with end node u, the extension of resources consumption and cost along some arc $(u, v) \in A_i$ can then be computed as:

- $\gamma^r([P,v]) = f^r_{uv}(\gamma^r(P)), \forall r \in \mathbf{R}$ and
- $\gamma^0([P,v]) = \gamma^0(P) + c_{uv} + \sum_{r\in\mathbf{S}} g^r_{uv}(\gamma^r(P))$.

In the remainder of the section, we provide the specific REFs and CEFs for each constraint of the NSP that needs to be modeled as a resource constraint. For the hard constraints, we also mention the constraints that resource consumptions must satisfy.

Total shift types Let r be a constraint on the total number of assignments to a given shift type S. This constraint can be modeled by a classical resource constraint where the consumption of resource along each arc $(u,v) \in A_i$ is given by a constant value that is equal to 1 if shift$(v) \in S$ and 0 otherwise. At any arc $(u,v) \in A_i$, the resource/cost extensions functions and the hard constraints on a partial path P can then be defined as follows.

- REF: $f^r_{uv}(\gamma) = \begin{cases} \min\{\gamma+1, U^r\} & \text{if shift}(v) \in S \\ \gamma & \text{otherwise} \end{cases}$
- CEF: if $r \in \mathbf{S}$, $g^r_{uv}(\gamma) = \begin{cases} c^U_r & \text{if shift}(v) \in S \text{ and } \gamma = U^r \\ c^L_r [L^r - \gamma]^+ & \text{if } v = t \\ 0 & \text{otherwise} \end{cases}$
- Hard constraints: if $r \in \mathbf{H}$, $\begin{cases} \text{if shift}(v) \in S : \gamma^r(P) \le U^r - 1 \\ \text{if } v = t : \gamma^r(P) \ge L^r \end{cases}$

Observe that penalties due to the violation of the upper limit are taken into consideration when extending a partial path along any arc with the above CEFs. This is done for identifying more dominated paths. In contrast, the penalties due to a violation of the lower bound are added only when reaching the sink, because this violation is non-increasing (with the extension of the path).

Consecutive shift types Let r be a constraint on the consecutive number of assignments to shift type S. These constraints are similar to those found in the variant of the SPPRC studied by Bolívar et al. (2014) where the resource consumption is reset when so-called *replenishment arcs* are traversed. Here, any arc (u,v) such that shift$(u) \in S$ and shift$(v) \notin S$ is similar to a replenishment arc. When traversing such arc, any started sequence of assignments to the shift type S is ended and the resource value is reset to 0. As a consequence, the lower limit must be satisfied every time a replenishment arc is traversed. For soft constraints, violations of the lower bound must also be paid when traversing such arc. We may thus define the resource/cost extensions and the constraints at any arc $(u,v) \in A_i$ as follows.

- REF: $\forall (u,v) \in A_i, f^r_{uv}(\gamma) = \begin{cases} \min\{\gamma+1, U^r\} & \text{if shift}(u) \in S \text{ and shift}(v) \in S \\ 0 & \text{otherwise} \end{cases}$
- CEF: if $r \in \mathbf{S}$, $g^r_{uv}(\gamma) = \begin{cases} c^U_r & \text{if shift}(v) \in S \text{ and } \gamma = U^r \\ c^L_r [L^r - \gamma]^+ & \text{if shift}(u) \in S \text{ and shift}(v) \notin S \\ 0 & \text{otherwise} \end{cases}$
- Hard constraints: if $r \in \mathbf{H}$, $\begin{cases} \text{if shift}(v) \in S : \gamma^r(P) \le U^r - 1 \\ \text{if shift}(u) \in S \text{ and shift}(v) \notin S : \gamma^r(P) \ge L^r \end{cases}$

Constraints on weekends The only impacting difference between constraints on shift types and constraints on weekends is that the completion of two different partial paths with the same suffix may lead to (slightly) different consumptions over the suffix. To illustrate this, we must assume the weekends of nurse i is at least three days long, *e.g.*, from Friday to Sunday. We then consider two partial paths P and Q from o to the same vertex u such that shift(u) = Rest and day(u) is the first Saturday of the planning horizon. Assume also that P goes through a rest shift on the preceding Friday whereas Q

goes through a work shift. In such case, if r is a constraint on the total number of working weekends, $\gamma^r(P) = 0$ and $\gamma^r(Q) = 1$. Now, if we extend both P and Q along the same arc going to some node v such that shift$(v) \neq$ Rest, the consumption along Q will be unchanged whereas that along P will increase by one unit.

As a consequence, the resource value $\gamma^r(P)$ is not sufficient to represent the state of a partial path P with respect to a constraint on worked weekends. For this, we also need a binary resource value $\delta_r \in \{0, 1\}$ indicating whether the current weekend has already been counted as worked in a partial path P (i.e., $\delta_r(P) = 1$) or not (i.e., $\delta_r(P) = 0$). This resource value has common features with a resource with replenishment since it must be reset at each end of a weekend. To characterize the extension of this resource value, we define one new set of REFs, $h^r_{uv} : \mathbb{R} \to \mathbb{R}, \forall (u, v) \in A_i$, where

$$h^r_{uv}(\delta) = \begin{cases} 1 & \text{if shift}(v) \neq \text{Rest and day}(v) \text{ is on a weekend} \\ \delta & \text{if shift}(v) = \text{Rest and day}(v) \text{ is on a weekend} \\ 0 & \text{otherwise} \end{cases}$$

When extended along some arc, the value of $\gamma^r(P)$ can increase only if $\delta_r(P) = 0$. The adaptation of the REFs and CEFs of shift type constraints is then straighforward.

Forbidden patterns They are closely related to the forbidden paths considered by Villeneuve and Desaulniers (2005), but the structure of the roster graph makes it possible to model it with a single upper-bounded resource.

Let $\Pi = ((D_1, S_1), \ldots, (D_K, S_K))$ be a forbidden pattern with length $K \geq 3$, where $D_1, \ldots, D_K$ are day types and $S_1, \ldots, S_K$ shift types. Any sub-sequence $((d_1, s_1), \ldots, (d_k, s_k))$, $k \leq K$, such that $d_1 \in D_1, \ldots, d_k \in D_k, s_1 \in S_1, \ldots, s_k \in S_k$ is called a *subpattern* of Π with length k. Denoting r the index of the constraint corresponding to Π, we model it with one resource value with upper bound $U^r = K$. For any partial path P from o to $u \in V_i$, $\gamma^r(P)$ indicates the length of the longest subpattern of Π ending at u in P.

If $D_1, \ldots, D_K$ refer to specific days or to some specific week days (*e.g.*, Monday, Tuesday and Wednesday), the resource extension along an arc (u, v) is quite straightforward. For $k = \gamma^r(P)$, one must simply compare day(v) with D_{k+1} and shift(v) with S_{k+1}. However, if $D_1, \ldots, D_K$ include every day of the horizon, some sub-sequences of Π may appear several times in Π. For instance, assume that Π forbids any sequence of shifts (Late, Late, Early) on any sequence of days. If $\gamma^r(P) = 2$, then P ends with a sequence of assignments to shifts (Late, Late). Obviously, if we wish to extend P along some arc (u, v) such that shift(v) = Early, we get $\gamma^r([P, v]) = 3$. The specificity is that if shift(v) = Late, the consumption is not reset to 0: the longest subpattern of Π ending at v in $[P, v]$ is (Late, Late) so $\gamma^r([P, v]) = 2$.

To describe the REF, we denote as $\Pi_{\leq l}$ the subpattern of Π with length l, for any $l \leq K$. We then build $\Gamma_\Pi(l)$ as the ordered list of the lengths of the subpatterns of Π that match the end of $\Pi_{\leq l}$. For the simplicity of the algorithm computing the extension, the list $\Gamma_\Pi(l)$ ends with 0 for all l and starts with l for all $l < K$, but it does not include K if $l = K$. In the example above where Π =(Late, Late, Early), we would get $\Gamma_\Pi(1) = [1, 0]$, $\Gamma_\Pi(2) = [2, 1, 0]$ and $\Gamma_\Pi(3) = [0]$. We finally characterize the resource extension over any arc $(u, v) \in A_i$ as follows.

- REF: $f^r_{uv}(\gamma)$ as described in Algorithm 2 which checks, by decreasing size, if any subpattern could be extended.
- CEF: if $r \in \mathbf{S}$, $g^r_{uv}(\gamma) = \begin{cases} c^U_r & \text{if } f^r_{uv}(\gamma) = K \\ 0 & \text{otherwise} \end{cases}$
- Hard constraints: if $r \in \mathbf{H}$, $f^r_{uv}(\gamma^r(P)) \leq U^r$.

```
... function f^r_uv(γ):
    for k ∈ Γ_Π(γ) do
        if day(v) ∈ D_{k+1} and shift(v) ∈ S_{k+1} then
            return k + 1 ;
    return 0;
```

Algorithm 2: Resource extension of forbidden patterns

5.2 Dominance rules

The extension functions and the constraints given above guarantee that the dynamic programming (Algorithm 1) will return a minimum cost feasible *o*-*t* path. The computational time of the algorithm will depend on its capacity to filter dominated paths though. Our aim is to design rules that are fast to verify and that provide sufficient conditions of dominance. For each considered constraint, one specific test needs to be carried out, and the satisfaction of the sufficient condition depends on the result of every test.

In the remainder of the section, we focus on the comparison of two partial paths P and Q from o to some vertex $v \in V_i$. The size of path P is denoted as $|P|\,(=|Q|)$. When considering a suffix $\overline{Q}$ of Q, the sequence of vertices of $\overline{Q}$ is denoted as $\overline{Q} = \left(v_1, \ldots, v_{|\overline{Q}|+1}\right)$ and we set $v_0 = v$. Observe that the first filtering task executed in Algorithm 1 is to remove equivalent labels until only one is left per equivalence class. It is straightforward to verify that if $\gamma^0(P) = \gamma^0(Q)$ and $\gamma^r(P) = \gamma^r(Q), \forall r \in \mathbf{R}$, then P and Q are equivalent. As a consequence, we assume that either $\gamma^0(P) \neq \gamma^0(Q)$ or there is $r \in \mathbf{R}$ such that $\gamma^r(P) \neq \gamma^r(Q)$. The proofs of the main results are detailed in Appendix A.

5.2.1 Dominance rules for hard constraints

An *o*-*t* path is feasible only if it satisfies every hard constraint. As a consequence, P dominates Q only if for all $r \in \mathbf{H}$ and for every completion $[Q, \overline{Q}]$:

$$[Q, \overline{Q}] \text{ satisfies } r \implies [P, \overline{Q}] \text{ satisfies } r. \tag{3}$$

If (3) is true, we say that P *dominates* Q *with respect to (w.r.t.)* r. The rule that has often been applied for a resource with both upper and lower limits is that the resource consumption should be the same in both partial paths. We can actually do better in some cases by leveraging that for some paths P, any completion of P is guaranteed to satisfy the upper and/or lower limits of some resources. For this, we denote as $\overline{D}$ the number of days after day(v) in the planning horizon ($\overline{D} = n_{\mathcal{D}} - \text{day}(v)$) and $\overline{W}$ the number of weekends on the planning horizon after day(v) (excluding the ongoing weekend if day(v) is on a weekend). The resulting dominance rule chosen for each type of resource constraint is detailed below.

Proposition 1. *Let $r \in \boldsymbol{H}$ be any hard resource constraint.*

1. *If r is a constraint on total shift types or on consecutive shift types, P dominates Q w.r.t. r if*

$$\left[\gamma^r(P) + \overline{D} - U^r\right]^+ \leq \left[\gamma^r(Q) + \overline{D} - U^r\right]^+ \text{ and } \left[L^r - \gamma^r(P)\right]^+ \leq \left[L^r - \gamma^r(Q)\right]^+. \tag{4}$$

2. *If r is a constraint on total or consecutive worked weekends, P dominates Q w.r.t. r if*

$$\begin{cases} \left[\gamma^r(P) + [\delta_r(Q) - \delta_r(P)]^+ + \overline{W} - U^r\right]^+ \leq \left[\gamma^r(Q) + \overline{W} - U^r\right]^+ \\ \left[L^r - \gamma^r(P)\right]^+ \leq \left[L^r - \gamma^r(Q)\right]^+. \end{cases} \tag{5}$$

3. *If r is a constraint on a forbidden pattern Π with length three or more, P dominates Q w.r.t. r if*

$$\gamma^r(P) = \gamma^r(Q) \text{ or } \gamma^r(P) \in \Gamma_\Pi(\gamma^r(Q)). \tag{6}$$

In our implementation, we have used the sufficient conditions given in Proposition 1 as dominance tests for hard constraints. In the remainder, for any hard constraint $r \in \mathbf{H}$, the satisfaction of the corresponding test is denoted as $P \succ_r Q$.

5.2.2 Dominance rules for soft constraints

Although soft constraints have usually been identified as difficulties for the solution of the SPPRC, they may actually be leveraged to identify more dominated paths and accelerate the dynamic programming. As an illustrative example, we may consider the simple case with only one soft upper bounded resource and two partial paths P and Q, from o to some vertex v, such that $\gamma^0(P) = 10$ and $\gamma^0(Q) = 20$. Assume that the linear penalty associated with the violation of the resource is equal to 5 and that the consumption along each path is equal to $\gamma(P) = 2$ and $\gamma(Q) = 1$. With the classical dominance rule recalled in Section 4.2, it is impossible to decide whether P dominates Q (because $\gamma(P) > \gamma(Q)$) or Q dominates P (because $\gamma^0(Q) > \gamma^0(P)$). Yet, we see that for any completion $[Q, \overline{Q}]$, $\gamma\left([P, \overline{Q}]\right) = \gamma\left([Q, \overline{Q}]\right) + 1$ which immediately yields $\gamma\left([P, \overline{P}]\right) \leq \gamma\left([Q, \overline{P}]\right) + 5$. Given that there is no hard constraint, $[P, \overline{Q}]$ and $[Q, \overline{Q}]$ are both feasible, so the above means that P dominates Q. In the remainder of this section, we formalize this idea and generalize it to all the constraints met in the NSP. One may observe that if the penalization is not linear, but convex, we could design dominance conditions in the same spirit.

For any $r \in \mathbf{S}$, let $G^r(P, \overline{Q})$ and $G^r(Q, \overline{Q})$ be the penalty paid for violations of constraint r along $\overline{Q}$ in $[P, \overline{Q}]$ and $[Q, \overline{Q}]$, respectively. More formally, with $\overline{Q} = (v_1, \ldots, v_{|\overline{Q}|+1})$,

$$G^r(P, \overline{Q}) = \sum_{k=0}^{|\overline{Q}|} g^r_{v_k v_{k+1}} \left(\gamma^r\left([P, v_1, \ldots, v_k]\right)\right).$$

We then denote $\Delta^r(P, Q, \overline{Q}) = G^r(P, \overline{Q}) - G^r(Q, \overline{Q})$. Given that the costs of the arcs of $\overline{Q}$ will be paid in both $[P, \overline{Q}]$ and $[Q, \overline{Q}]$, it is straightforward to verify that the following condition is necessary and sufficient for P to dominate Q.

Property 1. Partial path P dominates Q if and only if for every feasible completion $[Q, \overline{Q}]$, $[P, \overline{Q}]$ is feasible and

$$\gamma^0(Q) > \gamma^0(P) + \sum_{r \in \mathbf{S}} \Delta^r(P, Q, \overline{Q}). \tag{7}$$

In general, though, the above condition can be difficult to verify for arbitrary partial paths. It may for instance be necessary to extend the two partial paths, which we want to avoid. An alternative is to settle with a sufficient condition that will be based only on the state of the two partial paths. For each soft constraint $r \in \mathbf{S}$, we will thus compute an upper bound $\overline{\Delta}^r(P, Q)$ such that for every completion $[Q, \overline{Q}]$, $\Delta^r(P, Q, \overline{Q}) \leq \overline{\Delta}^r(P, Q)$. Assuming that these quantities are available, we obtain the following sufficient condition of dominance.

Property 2. For all $r \in \mathbf{S}$, let $\overline{\Delta}^r(P, Q) \geq \Delta^r(P, Q, \overline{Q})$ for all completion $[Q, \overline{Q}]$. Then, P dominates Q if

$$\begin{cases} P \succ_r Q, \forall r \in \mathbf{H}, \\ \gamma^0(Q) > \gamma^0(P) + \sum_{r \in \mathbf{S}} \overline{\Delta}^r(P, Q) \end{cases} \tag{8}$$

In our implementation, we used the sufficient condition given in (8) as dominance rule. If $P \succ_r Q, \forall r \in \mathbf{H}$ and $\gamma^0(Q) = \gamma^0(P) + \sum_{r \in \mathbf{S}} \overline{\Delta}^r(P, Q)$, either P dominates Q or the two paths are equivalent. In such case, we kept only P for extension. The following proposition provides the actual computation of an upper bound $\overline{\Delta}^r(P, Q)$ for each type of constraint met in the NSP.

Proposition 2. *Let $r \in \boldsymbol{S}$ be a soft resource constraint, and let $\overline{Q}$ be a suffix of Q such that $[P, \overline{Q}]$ and $[Q, \overline{Q}]$ are feasible.*

1. *If r is a constraint on the total number of shift types or on consecutive shift types,*

$$\Delta^r(P,Q,\overline{Q}) \leq \begin{cases} c_r^L \left([L^r - \gamma^r(P)]^+ - [L^r - \gamma^r(Q)]^+ \right) & \text{if } \gamma^r(P) \leq \gamma^r(Q) \\ c_r^U \left(\left[\gamma^r(P) + \overline{D} - U^r\right]^+ - \left[\gamma^r(Q) + \overline{D} - U^r\right]^+ \right) & \text{if } \gamma^r(P) > \gamma^r(Q) \end{cases} \tag{9}$$

2. *If r is a constraint on total or consecutive worked weekends,*

$$\Delta^r(P,Q,\overline{Q}) \leq \begin{cases} c_r^L \left([L^r - \gamma^r(P)]^+ - [L^r - \gamma^r(Q)]^+ \right), \\ \qquad \text{if } \gamma^r(P) \leq \gamma^r(Q) \\ c_r^U \left(\left[\gamma^r(P) + 1 - \delta_r(P) + \overline{W} - U^r\right]^+ - \left[\gamma^r(Q) + 1 - \delta_r(Q) + \overline{W} - U^r\right]^+ \right), \\ \qquad \text{if } \gamma^r(P) > \gamma^r(Q) \end{cases} \tag{10}$$

3. *If r is a constraint on a forbidden pattern Π with length three or more,*

$$\Delta^r(P,Q,\overline{Q}) \leq c_r^U \left| \Gamma_\Pi(\gamma^r(Q)) \setminus \Gamma_\Pi(\gamma^r(P)) \right|. \tag{11}$$

5.3 Implementation of acceleration techniques

Several acceleration techniques have been used in the past to speed-up the solution of the SP-PRC (Bolívar et al. 2014). Good results have also been reported for staff scheduling by Dohn and Mason (2013) and Gérard et al. (2016) or for the VRP by Costa et al. (2019). Here we describe the techniques that we implemented.

Given the huge number of acceleration techniques that appear in the literature, we had to restrict our tests to those that we felt were the most likely to provide significant improvements. For instance, we did not implement preprocessing methods and Lagrangian-based filtering techniques similar to those described in (Dumitrescu and Boland 2003) and (Carlyle et al. 2008), because they cannot be adapted to capture the specificity of soft constraints.

5.3.1 Heuristic pricing methods

Decremental state-space relaxation (DSSR). One may observe that in a context of high demand, the total number of assignments of each nurse will tend to be closer to its upper bound than to its lower bound, and the same goes for the number of assignments in each rotation. As a consequence, we propose to relax the constraints on the lower bounds of these constraints when checking for dominance. If the optimal solution has a negative cost, we add the corresponding column to the master problem. Otherwise, we solve the subproblem once more, but with the lower bound constraints. For a more efficient re-optimization, we store the partial paths that would not have been dominated if lower bounds had been considered.

Restrict the number of propagated labels (RNPL). In (Burke and Curtois 2014), the number of paths that can be extended is restricted to a given number and the extended paths are chosen heuristically. Given that the graph is acyclic, we instead decided to extend the partial paths with minimum costs at each vertex. If no negative cost roster was found with this heuristic, a second phase is run where every partial path is extended.

5.3.2 Computation of lower bounds

Shortest reverse paths from the sink (SRP). For any path P from o to v, a lower bound on the cost and on the consumption of each resource in any completion of P can quickly be obtained by searching for reverse shortest paths from the sink t to v. Path P can then be deleted if these lower bounds

indicate that every completion will have a non-negative reduced cost or that hard resource constraints will not be satisfied. This technique has often been used for the general SPPRC (Bolívar et al. 2014, Zhu and Wilhelm 2012), and Burke and Curtois (2014) also mentioned having implemented it to solve INRC-I instances but they did not assess its impact.

In the NSP, a lower bound on a resource's consumption is of no real use, because it is usually easy to build a completion that does not consume any unit of resource. As a consequence, we only tested this technique to get lower bounds on the cost of a completion. The reverse shortest paths from t can be computed with only one execution of a $\mathcal{O}(|A_i|)$ shortest path algorithm, because the roster graph is acyclic. However, it has to be executed at every solution of the pricing problem, because dual costs change after every solution of the master problem.

Enumeration of subpaths (ENUM). To strengthen these bounds, we also built a graph where the soft costs of some constraints could be integrated in the arc costs. More specifically, for the constraints on consecutive days off that are found in most benchmarks and for those on consecutive shifts that appear in INRC–II, Legrain et al. (2020) describe how the roster graph can be modified to model the soft constraints with additional arc costs. We adapted their approach by adding arcs that enumerate possible sequences of days off and of assignments to the same shift, and then the shortest reverse path from the sink (**ENUM SRP**).

5.3.3 Bidirectional extension (BIDI)

Bi-directional dynamic programming has originally been used to speed-up Dijstra's algorithm for the computation of shortest paths, and it has been reported to bring significant improvements to the solution of VRPs with branch-and-price (Righini and Salani 2006). Instead of extending paths only from the origin to the destination, the algorithm also executes a backward extension from the destination and merges these forward and backward partial paths into complete $o-t$ paths when they join. The vertices where these two paths join are chosen with a view to minimizing the total number of paths generated. This technique can be very efficient for VRPs because the pricing problem consists in the search of elementary paths where the number of paths generated can be exponential in the number of arcs in the paths.

We adapted this technique to NSP, because the number of paths generated may also grow quickly with the number of days in a roster due to the presence of lower bounds. This involves the implementation of two methods for each resource constraint: one for backward extension and the other for merging paths. The vertices corresponding to the middle day of the scheduling horizon were chosen to merge forward and backward paths. This appeared to be the most natural choice for the generation of individual schedules.

6 Computational experiments

Our experiments illustrate the capacity of our approach to handle a large diversity of hard and soft constraints. First, we wish to study the impact of all the algorithmic developments that we made in our search for numerical improvements, and which are described in the previous sections. Second, we assess the performance of our algorithm by solving all the Nurse Rostering Problem (NRP) and INRC-I instances and the INRC–II instances that have previously been used in studies of the static NSP. We also report a comparison with the other methods of the literature that were used to solve the same instances. Only the most important results are displayed in the main body, detailed computational results can be found in Appendix C.

In most tests, we aim at proving optimality for as many instances as possible, so we implemented parallel computing where relevant: pricing problems, strong branching and the rotation MIP heuristic. All experiments have been run on processors Intel Gold 6148 Skylake @ 2.4 GHz with 32 GB of

memory over 8 threads using our developed open-source C++ code, which depends on the latest Coin-OR libraries (BCP 1.4, CLP 1.17.7 and CBC 2.10.8[2]) and Gurobi 9.5.0.

6.1 Description of the test benchmarks

In our experiments, we have used all the INRC-I instances: 30 sprint, 15 medium and 15 long instances, which consider respectively 10, 30 and 50 nurses. They all consider a four weeks horizon, three to four different shifts, and one or two different skills. This benchmark is split into three subsets (*early*, *late*, *hidden*) depending on the stage of the competition when they were released. The hidden instances were only published once the competitors had all submitted the final versions of their codes.

Given that the INRC–II benchmark was meant for a stochastic version of the problem, we had to restrict our study to the subset of the instances used by the organizers of the competition when evaluating the competitors' algorithms. Those are the same that have already been used in (Legrain et al. 2020, Ceschia et al. 2020). Their names all start with the same pattern nXXXwY where XXX stands for the number of nurses and Y is the number of weeks of the planning horizon. The pattern is then followed by a long sequence identifying files where the demands are written. For a more compact presentation of the results, we only use shorter names to identify them. The correspondence is given in Appendix C. The instances include 30 to 120 nurses over a four or an eight weeks horizon. They consider four different shifts and four different skills. Among other important differences between the two benchmarks: the weekend is always on Saturday and Sunday in INRC–II whereas it can include the Friday in INRC-I; INRC-I instances include soft constraints on forbidden patterns with length more than two, whereas they are hard and their length is equal to two in INRC–II; possessing a required skill may be a soft constraint in INRC-I while it is always hard in INRC–II; there is a soft and a hard lower bound on demand in INRC–II whereas there is only one hard equality constraint in INRC-I. For the reader who plans on working on the INRC-I benchmark, we should also mention that the correct understanding of the original description of the constraints can be challenging. For this, we had to refer to the validator provided by the organizers of the competition and to the results and solutions already shared by other researchers and competitors.

In comparison to the two INRC benchmarks, the NRP instances can be extremely large, certainly larger than what could be needed for detailed and personalized schedules. The benchmark includes 24 instances whose main characteristics are reported in Table 6. The largest ones consider a one year horizon with up to 32 different shifts per day. Another specificity of the benchmark is that most constraints are hard. In particular, all the constraints on total and consecutive shift types are hard. A detailed comparison of the constraints can be found on the website of the benchmark.[3]

6.2 Assessment of pricing solution

The algorithmic features described in Section 5 are all meant to speed-up the solution of the pricing problems. As a consequence, their impact on computational time can be studied by solving only the linear relaxation of the roster-based formulation (1a)–(1f). Still with a view to spending a reasonable amount of computational resources in this assessment, we focused on six INRC–II instances – three with a 4 weeks horizon and three with a 8 weeks horizon – and three INRC-I instances. We have also run the experiments over only one thread. Given that the number of nurses will not impact our conclusions, we chose the following instances: n30w4-1, n35w4-1, n040w4-1, n030w8-1, n035w8-1, n040w8-1, long3, long_late3 and medium_hidden1. While we also assess the speed-up strategies with both dominance rules, the main goal of these experiments is to show that the improved dominance is by far the most impacting feature when solving NSP with soft constraints.

We first assessed the impact of the dominance rules described in Section 5.2 by comparing them to the basic dominance rule where a path P may dominate a path Q only if $\gamma^r(P) = \gamma^r(Q)$ for each upper

[2] https://github.com/wssuite/coin
[3] http://www.schedulingbenchmarks.org/nrp/instances1_24.html

and lower bounded resource r. The results are reported in Table 1, where the acronyms used in column titles have been defined in Section 5.3. Those results show that nearly every strategy implemented is able to decrease significantly the computational time except the enumeration of subpaths which requires a heavy preprocessing. This confirms that the implemented strategies work well with a basic dominance rule, as already reported in the literature.

Table 1: Computational time improvements for the basic dominance rule.

Strategy	Default	ENUM	SRP	ENUM SRP	BIDI	DSSR	RNPL=10	DSSR RNPL=5	DSSR RNPL=10
Instance	Time (s)	Improvement							
n030w4-1	465	4%	58%	2%	55%	37%	36%	33%	33%
n035w4-1	385	18%	54%	0%	50%	51%	32%	36%	29%
n040w4-1	482	30%	58%	-1%	55%	28%	11%	43%	22%
n030w8-1	13772	-1%	65%	1%	61%	34%	14%	20%	28%
n035w8-1	20516	-18%	73%	1%	70%	38%	3%	10%	39%
n040w8-1	16027	13%	67%	-1%	57%	43%	7%	32%	33%
long3	217	0%	24%	-1%	10%	61%	3%	51%	14%
long_late3	3385	-1%	13%	-1%	61%	66%	70%	73%	73%
medium_hidden1	4315	1%	18%	1%	61%	49%	76%	72%	83%
Average		**5%**	**48%**	0%	**53%**	**45%**	**28%**	**41%**	**39%**

We then assess heuristic pricing algorithms, lower bound computations and bidirectional extension by measuring their impact when considered individually with the improved dominance rule. For heuristic pricing and lower bounds, it was possible to code the accelerating techniques for both INRC-I and INRC–II without extra effort due to the similarity of the two benchmarks. However, each new resource constraint requires the implementation of methods for the backward extension of partial paths and for merging forward and backward partial paths, which requires a decent amount of time to be efficiently implemented. The results are reported in Table 2. They confirm that the improved dominance rule allows to eliminate a lot of dominated partial paths, and thus to speed-up the whole solution process by factors up to 28. Unfortunately, the improved dominance also makes most other strategies inefficient. Only the bidirectional approach is still worth to be implemented, especially for bigger instances, and will remain activated for the remaining of the tests.

Table 2: Computational time improvements for the improved dominance rule.

Strategy	Default	ENUM	SRP	ENUM SRP	BIDI	DSSR	RNPL=10	DSSR RNPL=5	DSSR RNPL=10
Instance	Time (s)	Improvement							
n030w4-1	39	-26%	-8%	-13%	8%	-15%	-5%	-5%	-18%
n035w4-1	40	-23%	-20%	-10%	-15%	-5%	-33%	-23%	-33%
n040w4-1	27	-22%	-7%	-11%	-26%	-19%	-52%	-15%	-59%
n030w8-1	544	-21%	-1%	-4%	29%	-2%	-18%	-13%	-5%
n035w8-1	725	-64%	2%	-8%	30%	-6%	-15%	-26%	-22%
n040w8-1	654	-10%	2%	-2%	19%	-20%	-17%	-18%	-36%
long3	83	-1%	-1%	-6%	45%	4%	-118%	-35%	-112%
long_late3	302	0%	-2%	-4%	33%	-10%	-81%	-71%	-86%
medium_hidden1	303	-1%	-7%	-2%	31%	-1%	-61%	-33%	-61%
Average		**-27%**	**-5%**	**-8%**	**7%**	**-11%**	**-23%**	**-16%**	-29%

6.3 Assessment of the branching and primal heuristics

The assessment of the branching and primal heuristics is performed over more difficult instances: n080w4-2, n110w4-1, n030w8-1, n035w8-2, n040w8-1, long3, and medium_hidden1. Our tests focus on the assessment of a strong branching strategy and of two primal heuristics: a diving heuristic (see Appendix B) and the MIP heuristic based the rotation formulation. In the evaluation of the rotation

MIP, the MIP solver runs permanently in parallel. However, the selection parameters (μ, ρ) that we used in the default branching strategy (see Appendix B) have been tuned in preliminary tests and never modified afterwards. Their values in all our tests are $(\mu, \rho) = (0.2, 0.1)$.

Table 3: Assessment of the strong branching strategy and of primal heuristics: improvement of the lower (LB) and upper (UB) bounds compared to the default branch-and-price algorithm; time limit was set to two hours.

Strategy	diving	rotation MIP	strong branching	diving	rotation MIP	strong branching
Instances		LB			UB	
n080w4-2	0.00%	0.01%	0.05%	1.24%	1.39%	0.31%
n110w4-1	0.21%	-0.01%	0.06%	59.48%	59.48%	0.00%
n030w8-1	-0.15%	0.00%	-0.12%	-0.99%	0.74%	-1.23%
n035w8-2	-0.25%	-0.01%	0.03%	54.53%	56.45%	0.77%
n040w8-1	-0.11%	0.03%	0.17%	-1.54%	2.70%	-2.70%
long3	0.00%	0.00%	0.00%	0.41%	0.41%	0.41%
medium_hidden1	0.00%	0.00%	4.70%	13.87%	10.95%	-21.90%
Average	**-0.04%**	**0.00%**	**0.70%**	**18.14%**	**18.88%**	**-3.48%**

The results of these tests, reported in Table 3, clearly indicate that the primal heuristics decrease the upper bounds while strong branching increases the lower bound. At the same time, one can note that the diving heuristic also decreases the lower bound as more time is spent diving, while strong branching increases the upper bound as more time is spent evaluating the branching decisions. However, the rotation MIP heuristic does to not impact the quality of the lower bound, because computations are done in parallel of the tree exploration. All three strategies are used and combined for the following experiments.

Among the algorithmic features that are classically implemented in branch-and-price algorithms, we do not report any result related to stabilization techniques. Actually, we have implemented a dual stabilization based on penalties as detailed in Du Merle et al. (1999), but we have been discouraged to go further in our analysis by disappointing preliminary results. Nevertheless, Pessoa et al. (2018) state that those techniques are particularly difficult to tune, so there might be some space for improvement.

6.4 Comparison to the literature

The assessment of the default algorithm is done on all the INRC-I instances and on the INRC–II instances used by Ceschia et al. (2020). This amounts to a total of 100 instances where the number of nurses ranges from 10 to 120 and the schedules must be computed on 4 to 8 weeks. Finally, the default algorithm is also assessed on the NRP instances that are very different as most constraints are hard, the horizon ranges from 2 weeks to a whole year, and there are up to 32 different shifts per day. The purpose of this comparison is to illustrate the capacity of our code to handle several different benchmarks and assess its capacity to solve the instances to optimality or improve on the literature. In our tests, we set the default time limit to one hour per week in the planning horizon, not exceeding a one-day limit.

6.4.1 INRC-I

A review of the methods applied to the INRC-I and their results are summarized in (Abuhamdah et al. 2021). The best known upper bounds are listed on the competition website. Many solution algorithms are based on metaheuristics and they do not compute lower bounds. The only reports on lower bounds that we found in the literature are made by (Burke and Curtois 2014) and Santos et al. (2016a), but the former did not give any result on the 20 hidden instances.

We report the results of our algorithm in Table 4. As will be the case in the other tables below, the lower and upper bounds displayed with **bold** characters are optimal, the underlined values indicate an improvement with respect to the literature and a dash in the Time column indicates that time limit

Table 4: Results for the INRC-I ; time limit was set to 4 hours.

Type	**early**				**late**				**hidden**			
Instance	Time (s)	root LB	LB	UB	Time (s)	root LB	LB	UB	Time (s)	root LB	LB	UB
sprint1	107	56	**56**	**56**	144	37	**37**	**37**	29	31.5	**32**	**32**
sprint2	62	58	**58**	**58**	15	41.4	**42**	**42**	11	32	**32**	**32**
sprint3	145	51	**51**	**51**	36	47.8	**48**	**48**	21	62	**62**	**62**
sprint4	30	58.5	**59**	**59**	107	72.5	**73**	**73**	26	66	**66**	**66**
sprint5	134	57	**58**	**58**	174	43.7	**44**	**44**	90	59	**59**	**59**
sprint6	22	54	**54**	**54**	61	41.5	**42**	**42**	31	129.7	**130**	**130**
sprint7	26	56	**56**	**56**	41	42	**42**	**42**	11	153	**153**	**153**
sprint8	49	56	**56**	**56**	77	17	**17**	**17**	36	204	**204**	**204**
sprint9	103	55	**55**	**55**	87	17	**17**	**17**	267	337.3	**338**	**338**
sprint10	49	52	**52**	**52**	85	42.9	**43**	**43**	29	306	**306**	**306**
medium1	94	240	**240**	**240**	128	156	**157**	**157**	–	95.7	103	120
medium2	26	239.2	**240**	**240**	94	18	**18**	**18**	–	212.4	216	219
medium3	136	235.5	**236**	**236**	155	28.2	**29**	**29**	341	33.4	**34**	**34**
medium4	363	236.2	**237**	**237**	58	34.4	**35**	**35**	4817	75.5	**78**	**78**
medium5	155	302.1	**303**	**303**	450	106.7	**107**	**107**	329	117.2	**118**	**118**
long1	68	197	**197**	**197**	360	235	**235**	**235**	830	345	**346**	**346**
long2	104	218.5	**219**	**219**	336	229	**229**	**229**	170	88.5	**89**	**89**
long3	80	240	**240**	**240**	–	218.5	219	220	3408	37.7	**38**	**38**
long4	61	303	**303**	**303**	138	220.7	**221**	**221**	684	21.8	**22**	**22**
long5	183	284	**284**	**284**	163	82.5	**83**	**83**	1012	41	**41**	**41**
Improvements			**0**	**0**			**1(1)**	**0**			**9(7)**	**0**

was reached before optimality could be proven. The last row summarizes the number of improved bounds, and, in parentheses, the number of new proofs of optimality obtained thanks to these new bounds. Overall, these results show that most instances have been proven optimal in a very short computational time (over 8 threads) and that time limit was reached on only three instances. For the two instances medium hidden 1 and 2, optimality was reached during other tests though. More precisely our algorithm was able to prove optimality with other values of μ and ρ (0 and 0.5). However, our algorithm was never able to converge when executed on instance long late 3. Whatever computational time available, the lower bound remains at the root lower bound. One explication could come from the fact that this particular instance has no personalized aspect: there is no request from any nurse for a day/shift on/off. A non-personalized rostering method should be used for this special instance. Finally, the upper-bounds reported in Table 4 were already known, but optimality proofs had not been reported for all of them. More precisely, we see that our algorithm was able to improve the lower bounds of ten instances. Among those, eight allowed to prove the optimality of the best known upper bound.

6.4.2 Static INRC–II

A review of the methods that were applied to this static INRC–II benchmark and a detailed comparison of their results can be found in (Legrain et al. 2020) and (Ceschia et al. 2020). The results of our roster-based approach on all 4-weeks and 8-weeks instances of the INRC–II benchmark are respectively given in Table 5. The bounds that appear in these tables all improve on those already reported in the literature. To avoid redundancy, we thus omit to underline these values. For a more detailed analysis of the results, the superscripts B, D, and H refer to the moment when the best upper bound has been found: respectively, during the branch-and-price (either at a node or during strong-branching), when diving or when solving the MIP Heuristic.

First and foremost, Table 5 shows that all 4-weeks instances have been solved to optimality, while the algorithms described in the literature had all failed to do so up to now. Among these 20 instances, 18 were solved in less than one hour and all of them where solved in less than three hours.

Table 5: Results for the static INRC–II; time limit was set to one hour per week.

Instance	Time (s)	root LB	LB	UB	Instance	Time (s)	root LB	LB	UB
n030w4-1	67	1659.5	**1670**	**1670^{H}**	n030w8-1	425	1993.7	**2010**	**2010^{H}**
n030w4-2	377	1809.2	**1815**	**1815^{B}**	n030w8-2	471	1709.4	**1720**	**1720^{D}**
n035w4-1	66	1337.1	**1360**	**1360^{B}**	n035w8-1	-	2407.8	2425	2505^{H}
n035w4-2	196	1075.6	**1080**	**1080^{H}**	n035w8-2	-	2152.2	2205	2270^{H}
n040w4-1	161	1535.2	**1565**	**1565^{D}**	n040w8-1	-	2463.9	2490	2515^{H}
n040w4-2	33	1741.7	**1750**	**1750^{D}**	n040w8-2	-	2284	2305	2330^{H}
n050w4-1	483	1295.5	**1315**	**1315^{D}**	n050w8-1	-	4777.5	4805	4810^{B}
n050w4-2	175	1302.1	**1315**	**1315^{H}**	n050w8-2	7731	4743.3	**4765**	**4765^{B}**
n060w4-1	46	2434.9	**2450**	**2450^{D}**	n060w8-1	-	2099	2120	2150^{D}
n060w4-2	31	2664.3	**2675**	**2675^{D}**	n060w8-2	-	2393.4	2410	2435^{H}
n070w4-1	37	2370.3	**2380**	**2380^{D}**	n070w8-1	-	4474.7	4505	4530^{H}
n070w4-2	69	2105	**2115**	**2115^{H}**	n070w8-2	-	4636.3	4660	4690^{H}
n080w4-1	2811	3292	**3300**	**3300^{D}**	n080w8-1	-	3941.7	3955	4030^{H}
n080w4-2	7514	3177.1	**3185**	**3185^{H}**	n080w8-2	-	4286.9	4295	4335^{H}
n100w4-1	187	1167.2	**1170**	**1170^{B}**	n100w8-1	-	2013.5	2025	2030^{H}
n100w4-2	39	1777.7	**1780**	**1780^{D}**	n100w8-2	-	2128.6	2140	2145^{H}
n110w4-1	816	2321.3	**2330**	**2330^{D}**	n110w8-1	2456	3990	**3990**	**3990^{B}**
n110w4-2	133	2455	**2455**	**2455^{D}**	n110w8-2	-	3440	3440	3460^{H}
n120w4-1	11351	2011.5	**2020**	**2020^{H}**	n120w8-1	-	2440	2440	2450^{D}
n120w4-2	70	2045.7	**2050**	**2050^{D}**	n120w8-2	14162	2871.7	**2875**	**2875^{B}**
Improvements			**20**	**20(20)**				**5**	**5(5)**

Although, the 4-weeks instances have all been solved in a reasonable amount of time, a lot more time was generally necessary to prove optimality for the 8-weeks instances (when it could be done). Table 5 shows that for the 8-weeks horizon, 5 instances over 20 could be solved to optimality in less than eight hours. It also allowed to improve all the best known upper and lower bounds of the literature.

Out of curiosity, we also investigated the impact of the time limit on the results by running these tests with a 1-day time limit (see Appendix C). This has allowed to close three more instances and improve many upper and lower bounds. For all instances but one, the remaining gap is of the same order as one unit violation of any soft constraint. This may be the sign that the algorithm is not far from solving most other instances to optimality. Most of these upper bounds have been obtained by the MIP heuristic. For further analysis of the primal heuristics, the large neighborhood search (LNS) developed by Legrain et al. (2020) has also been tested with the roster formulation. These results are presented in the column **LNS UB**. They show that the LNS was almost never able to get better performances than the MIP heuristic approach: the LNS obtained a better upper bound only for instance n040w8-2. One may also observe in Appendix C that n110w8-2 was solved optimally in the second run with a 1-day time limit although less than 3 hours were necessary to converge. The reason is that, even though we set the random seed to the same value for every run, the code remains stochastic due to the parallel execution of the MIP heuristic.

We have then conducted the same experiments with CBC instead of Gurobi for the rotation MIP heuristic and those results can be seen in Appendix C. For the 4-weeks instances, all the instances but one have been solved to optimality, but in general with larger computational times. For the 8-weeks instances, four instances (instead of five) have been solved to optimality under 8 hours and all computational times are significantly higher. CBC can thus be a good option for a free and open-source solver as the reported differences may not be impacting in practice.

Finally, we have compared these results to the best ones obtained with the rotation approach (Legrain et al. 2020), and those results, which are presented in Appendix C, show the improvements in lower and upper bounds that could be achieved by using a roster instead of a rotation formulation. To be fair, we should mention that the rotations results were obtained in one day of computation over only one thread, while the roster formulation was solved on 8 threads with a 4 and 8 hours time limit for 4-weeks and 8-weeks instances, respectively. The results show that the roster approach outperforms

the rotation approach on all instances with improvements of the lower bounds by more than 4% on average and of the upper bounds by more than 3%. When using CBC though, it can be noticed that the rotation-based formulation finds a better upper bound for five 8-weeks instances. The observation of the increase in the root and final lower bounds also confirm that the most impacting property of the roster-based formulation is that it has a better linear relaxation.

6.4.3 Nurse rostering problem benchmark

The last comparison have been made on the 24 instances of the NRP benchmark. Only the largest instances, which cover a whole year, are still open. For several difficult instances though, there exists no publication describing the method that produced the best reported results. For a meaningful comparison, we will thus restrict our analysis to the best published results. The twelve first instances could be solved to optimality within a reasonable computational time (i.e., less than 5 hours) with a commercial mathematical programming solver, running on 16 threads and using up to 128GB RAM (see (Smet 2018)). However, the larger ones are much more difficult, which causes most approaches to encounter issues during their solution. The main difficulty comes from the fact there are many hard constraints on total shift types. The easier instances do not include any constraint on the maximum number of assignments to specific shifts whereas the most difficult ones may include 6 to 32 of them. If specifically interested by the NRP benchmark, the reader may refer (Strandmark et al. 2020) for more details on published results.

The description of the instance, the results of our roster-based approach and the best published lower and upper bound are reported in Table 6. In the best published columns, the superscript indicates the reference where each bound was published. To obtain our results, most accelerating strategies have been activated (RNPL=10, DSSR, SRP, BIDI). They show that our algorithm performs as well as other approaches on this benchmark: it finds the best known upper bounds of 19 instances an proves the optimality of 14 among them. Moreover, our approach was able to solve one open instance and improve one lower and two upper bounds with respect to the literature. Despite the large time limit, our approach was not able to solve the root linear relaxation of six instances nor compute any feasible solution for the largest four instances. We observed that the roster-based formulation gets stuck in the solution of the subproblems due to the large number of hard constraints. Although the subproblems are solved only with RNPL and DSSR strategies to overcome part of the difficulty, the impossibility to perform soft dominance (as described in Section 5.2.2) on the total shift type constraints yield an explosion of the number of nondominated labels. Regarding instances 15 and 19, this did not preclude the MIP heuristic from finding fairly good upper bounds though. Actually, the upper bound found for instance 19 is even far better than the best published one.

7 Conclusion

Extensive experiments (weeks of computation) have shown that the proposed improved dominance approach combined with many acceleration techniques has enabled the resolution of large instances to optimality. In particular, all 4-weeks instances of the INRC–II have now optimal solutions while they were open up to now, and optimality was also proven for eight 8-weeks instances while only small gaps remain for the others. Moreover, every INRC-I instance but one has also been closed, and our approach solves to optimality or finds the best upper bounds for most NRP instances. Finally, the proposed solver, when used with CBC, is available as a free and open source software, and still produces very good solutions.

Several improvements could be made to help with the instances that remain open. In particular, we think that instance long_late3 of the INRC-I can be solved to optimality by aggregating similar nurses. Regarding the largest NRP instances, one may raise the purpose of optimizing detailed and personalized schedules over six months or even one year, but if this is of importance, one step towards getting feasible (and hopefully good) solutions could be implement a receding horizon where smaller

Table 6: Results for the NRP instances: indices 1 and 2 refer to solutions published respectively in (Smet 2018) (a flow-based MIP) and (Strandmark et al. 2020) (a column generation-based heuristic); time limit was set to one hour per week.

				best published		our roster-based approach			
instance	weeks	nurses	shifts	LB	UB	time (s)	root LB	LB	UB
Instance1	2	8	1	**607**[1]	**607**[1]	2	558	**607**	**607**
Instance2	2	14	2	**828**[1]	**828**[1]	0	828	**828**	**828**
Instance3	2	20	3	**1001**[1]	**1001**[1]	1	1001	**1001**	**1001**
Instance4	4	10	2	**1716**[1]	**1716**[1]	2	1716	**1716**	**1716**
Instance5	4	16	2	**1143**[1]	**1143**[1]	9	1140.6	**1143**	**1143**
Instance6	4	18	3	**1950**[1]	**1950**[1]	9	1949	**1950**	**1950**
Instance7	4	20	3	**1056**[1]	**1056**[1]	112	1054.1	**1056**	**1056**
Instance8	4	30	4	**1300**[1]	**1300**[1]	-	1296.6	1298	**1300**
Instance9	4	36	4	**439**[1]	**439**[1]	-	405.7	406	**439**
Instance10	4	40	5	**4631**[1]	**4631**[1]	100	4631	**4631**	**4631**
Instance11	4	50	6	**3443**[1]	**3443**[1]	21	3443	**3443**	**3443**
Instance12	4	60	10	**4040**[1]	**4040**[1]	4131	4040	**4040**	**4040**
Instance13	4	120	18	1348[1]	1356[2]	-	1301	1301	1351
Instance14	6	32	4	**1278**[1]	**1278**[1]	4830	1278	**1278**	**1278**
Instance15	6	45	6	3820[1]	3853[1]	-	-	-	3879
Instance16	8	20	3	**3225**[1]	**3225**[1]	63	3223.5	**3225**	**3225**
Instance17	8	32	4	**5746**[1]	**5746**[1]	669	5746	**5746**	**5746**
Instance18	12	22	3	4404[1]	4459[1]	-	4417.3	4421	4459
Instance19	12	40	5	3144[1]	3204[1]	-	-	-	3174
Instance20	26	50	6	4765[1]	4913[1]	36089	4769	**4769**	**4769**
Instance21	26	100	8	21122[1]	21402[2]	-	-	-	-
Instance22	52	50	10	-	32126[2]	-	-	-	-
Instance23	52	100	16	-	19704[2]	-	-	-	-
Instance24	52	150	32	-	58480[2]	-	-	-	-
Improvements								**2(1)**	**3(1)**

schedules are iteratively computed. Finally, some other promising avenues could be explored with stochastic counterparts of the nurse rostering problem. As one may observe that the lower bounds computed with the root linear relaxation of the roster-based approach is within a few percents of the optimal value for all instances solved to optimality, the linear relaxation of the roster formulation can serve as an excellent approximation of the recourse in a two-stage stochastic counterpart where schedules need to be computed as a potential recourse.

A Proofs

Proof of Proposition 1.

We assume that there is no violation of r in P nor Q, and we consider a suffix $\overline{Q}$ such that the completion $[Q, \overline{Q}]$ satisfies r. As a consequence, we are interested only in the violations of constraints that can happen in $[P, \overline{Q}]$ and not in $[Q, \overline{Q}]$. In particular, for constraints on consecutive shift types or worked weekends, we are interested only in the ongoing sequence of shift types or worked weekends. To avoid heavy technicalities, we will refer to the length of this complete sequence as $\gamma^r\left([P, \overline{Q}]\right)$. This allows for a homogeneous presentation with constraints on total shift and weekend types. We proceed likewise for constraints on forbidden patterns, since we are interested only in patterns that are already started in P or Q.

1. Observe first that by definition of $\overline{D}$, $\gamma^r(P) \leq \gamma^r\left([P, \overline{Q}]\right) \leq \gamma^r(P) + \overline{D}$. By definition of the constraints on shift types, we also have

$$\gamma^r(P) \leq \gamma^r(Q) \Leftrightarrow \gamma^r\left([P, \overline{Q}]\right) \leq \gamma^r\left([Q, \overline{Q}]\right).$$

Looking at the first inequality of (4), we then see that if $\gamma^r(P) \leq \gamma^r(Q)$ then $\left[\gamma^r(P) + \overline{D} - U^r\right]^+ \leq \left[\gamma^r(Q) + \overline{D} - U^r\right]^+$. If on the contrary, $\gamma^r(P) > \gamma^r(Q)$,

$$\left[\gamma^r(P) + \overline{D} - U^r\right]^+ \leq \left[\gamma^r(Q) + \overline{D} - U^r\right]^+ \implies \gamma^r(P) + \overline{D} \leq U^r.$$

As a consequence, if $\left[\gamma^r(P) + \overline{D} - U^r\right]^+ \leq \left[\gamma^r(Q) + \overline{D} - U^r\right]^+$, then $\gamma^r\left([P, \overline{Q}]\right) \leq U^r$ or $\gamma^r(P) \leq \gamma^r(Q)$. Regarding the second inequality, we have similarly that

$$\gamma^r(P) \geq \gamma^r(Q) \implies [L^r - \gamma^r(P)]^+ \leq [L^r - \gamma^r(Q)]^+.$$

And if $\gamma^r(P) < \gamma^r(Q)$,

$$[L^r - \gamma^r(P)]^+ \leq [L^r - \gamma^r(Q)]^+ \implies \gamma^r(P) \geq L^r.$$

As a consequence, if $[L^r - \gamma^r(P)]^+ \leq [L^r - \gamma^r(Q)]^+$, $\gamma^r\left([P, \overline{Q}]\right) \geq L^r$ or $\gamma^r(P) \geq \gamma^r(Q)$.
We finally get that if both inequalities are satisfied and $L^r \leq \gamma^r\left([Q, \overline{Q}]\right) \leq U^r$ then $L^r \leq \gamma^r\left([P, \overline{Q}]\right) \leq U^r$.

2. For constraints on weekends, it is not true that $\gamma^r(P) \leq \gamma^r(Q) \Leftrightarrow \gamma^r\left([P, \overline{Q}]\right) \leq \gamma^r\left([Q, \overline{Q}]\right)$. Indeed, an ongoing weekend may have been worked in Q but not in P. If that is the case, we may have $\gamma^r\left([P, \overline{Q}]\right) - \gamma^r(P) = \gamma^r\left([Q, \overline{Q}]\right) - \gamma^r(Q) + 1$. Observe then that $[\delta_r(Q) - \delta_r(P)]^+ = 1$ if $\delta_r(P) = 0$ and $\delta_r(Q) = 1$, and 0 otherwise. So, the arguments developed for constraints on shift types remain valid if we consider $\gamma^r(P) + [\delta_r(Q) - \delta_r(P)]^+$ instead of $\gamma^r(P)$ when checking the upper bound.
3. We denote as Π_P and Π_Q the subpatterns of Π at the end of P and Q. Recall that $\gamma^r(P)$ and $\gamma^r(Q)$ are the lengths of Π_P and Π_Q. If $\gamma^r(P) = \gamma^r(Q)$, we then know that if Π appears in $[P, \overline{Q}]$, it must appear in $[Q, \overline{Q}]$. If $\gamma^r(P) \in \Gamma_\Pi(\gamma^r(Q))$, by definition of Γ_Π, it exists a sequence of assignments $\widetilde{\Pi}$ such that $\Pi_Q = \left[\widetilde{\Pi}, \Pi_P\right]$. As a consequence, if Π appears in $[P, \overline{Q}]$, it must also appear in $[Q, \overline{Q}]$. □

Proof of Proposition 2.

We consider a suffix $\overline{Q}$ of Q. Similarly to the proof of Proposition 1, we are interested in counting the penalties that will be paid while completing P with $\overline{Q}$ but not while completing Q. As a consequence, we can focus on the patterns and on the sequences of shift types/weekends that are already started at the end of P and Q, and we will refer to the length of these completed sequences as $\gamma^r\left([P, \overline{Q}]\right)$ and $\gamma^r\left([Q, \overline{Q}]\right)$.

1. Let $0 \leq x \leq \overline{D}$ be the consumption of r in $\overline{Q}$ (this quantity is equal in $[Q, \overline{Q}]$ and in $[P, \overline{Q}]$). Note that x is only the consumption of the initial sequence in $\overline{Q}$ for a constraint on consecutive shift types. We have

$$\gamma^r\left([P, \overline{Q}]\right) = \gamma^r(P) + x \text{ and } \gamma^r\left([Q, \overline{Q}]\right) = \gamma^r(Q) + x.$$

Referring to the arguments developed in the proof of Proposition 1, we can also deduce that

$$\begin{cases} \Delta^r(P, Q, \overline{Q}) \leq c_r^L \left([L^r - \gamma^r(P) - x]^+ - [L^r - \gamma^r(Q) - x]^+\right) & \text{if } \gamma^r(P) \leq \gamma^r(Q) \\ \Delta^r(P, Q, \overline{Q}) \leq c_r^U \left([\gamma^r(P) + x - U^r]^+ - [\gamma^r(Q) + x - U^r]^+\right) & \text{if } \gamma^r(P) > \gamma^r(Q) \end{cases} \tag{12}$$

Functions $x \mapsto [L^r - \gamma^r(P) - x]^+ - [L^r - \gamma^r(Q) - x]^+$ and $x \mapsto [\gamma^r(P) + x - U^r]^+ - [\gamma^r(Q) + x - U^r]^+$ are both stepwise-linear; the former reaches its maximum at 0 while the latter reaches its maximum at $\overline{D}$. Replacing x with these values yields (9).

2. As in Proposition 1, we must compensate for the difference of structure between constraints on shift types and constraints on weekends by replacing $\overline{D}$ with $\overline{W}$ and by counting possible consumption of resource on the ongoing weekend. This directly yields (10).
3. For each element $k > 0$ of $\Gamma_\Pi(\gamma^r(P))$, there is a subpattern of Π with length k at the end of P. Each subpattern of length $k \in \Gamma_\Pi(\gamma^r(P)) \cap \Gamma_\Pi(\gamma^r(Q))$ that will be completed in $[P, \overline{Q}]$ will also be completed in$[P, \overline{Q}]$; only the subpatterns with lengths in $\Gamma_\Pi(\gamma^r(Q)) \setminus \Gamma_\Pi(\gamma^r(P))$ may induce a penalty. □

B Implementation details

B.1 Branching

Classical branching decisions on columns do not work within a branch-and-price algorithm: forbidden columns at a given node in the tree can be regenerated by the subproblems, which makes this branching decision useless. Instead, branching decisions are performed on the variables of the original problem before the Dantzig-Wolfe decomposition, e.g., the flow on the edges of the roster graph network (see Figure 1). Therefore, a two-phase branching heuristic has been developed. First, we branch on the flow of the edges corresponding to a rest shift for a given nurse and day: respectively, rest and work edges are removed from the graph of each child node. Second, we branch on a subset of shift types for a given nurse and day: in the same spirit, some edges are removed from each graph.

Furthermore, the branching heuristic must select for each phase on which nurse and day (and subset of shifts for the second phase) the algorithm will branch. As the main goal is to increase the lower-bound when searching to prove optimality, the heuristic aims at maximizing the expected increase of the worst lower-bound of the children. Therefore, the variables closer to 0.5 are selected. When strong-branching is enabled, the linear relaxation of a subset of potential (10, i.e., 5 branching decisions) children is evaluated without column-generation, and the branching decision leading to the best expected increase of the worst lower-bound is selected.

Finally, some knowledge on the nature of the problem is introduced in the variable selection process. First, the assignments on weekends are in general more expensive to cover. Second, some nurses tend to have more fractional and/or expensive rosters. We aim at favoring the variables corresponding to these assignments and nurses in the heuristic. The following score is computed as follows for each variable:

$$\min(f_{ids}, 1 - f_{ids}) + \mu\ \mathbf{W}(d) + \rho\ \frac{1 + C_i}{1 + \max_{i'}(C_{i'})} * \frac{N_i}{\max_{i'}(N_{i'})},$$

where f_{ids} is the sum of the flows on the edges corresponding to shift types included in subset s for nurse i on day d, $\mathbf{W}(d)$ indicates whether day d is a weekend, $C_i = \sum_{j \in \Omega_i} c_j x_{ij}$ is the cost of the current fractional roster of nurse i, $N_i = Card\left(\{j \in \Omega_i : x_{ij} > 0\}\right)$ is the number of variables of nurse i with a positive value, and (μ, ρ) is a pair of parameters to control the weight of each term. To summarize, the score is composed of three parts: the first one favors fractional flows, the second advantages weekend days and the third one favors nurses with many fractional rosters and a high associated cost. We then branch on the edge with maximum score or test those with maximum scores if strong-branching.

B.2 Primal heuristics

In order to get good feasible solutions in the process and increase our chance to prune branching nodes with large lower bounds, we implemented three primal heuristic methods: the rotation MIP heuristic and the following two classical approaches which are simple adaptations of the techniques described in (Legrain et al. 2020) for rotation-based decomposition. In short, we used a diving method where we branch on columns and fix to one the values of those that are integer or close to integrality. Such dives are done frequently at the beginning of the branch-and-price and more scarcely as the algorithm proceeds. We also adapted their large neighborhood search (LNS). When this method is selected and

optimality has not been proved after half the time limit, the metaheuristic algorithm is executed to improve the upper bound. Starting with an initial feasible solution, the LNS proceeds by fixing most schedules of the solution and *destroying* the remainder. In a *repair* phase, it then solves a smaller problem where only the destroyed schedules need to be built. Several destruction operators can be considered and the choice of the operator is made by a random *roulette procedure* that gives priority to the operators that produced improvements in past iterations. In the roster-based decomposition, we mostly destroy complete individual rosters, but we also destroy four consecutive weeks of rosters when considering 8-weeks horizons. The number of nurses whose rosters are destroyed is chosen so that the total number of impacted weeks is equal to 48. Given that our roster-based branch-and-price has been able to solve small instances to optimality within a short computational time, this choice guarantees that the LNS will be able to execute a large number of iterations.

C Detailed experimental results

This section provides the detailed experimental results discussed in the main body as well as the full names of the INRC–II instances (Table 7). Among those, Tables 10 and 11 show the improvements of the roster approach compared to the rotation approach. The column "Increase" reports the increase of the lower bound obtained by the roster approach when compared to the rotation approach while the column "Decrease" reports the decrease of the upper bound.

Table 7: Complete names of the INRC–II instances.

$\|\mathcal{N}\|$	$\|\mathcal{W}\| = 4$		$\|\mathcal{W}\| = 8$	
	instance	short name	instance	short name
30	n030w4_1_6-2-9-1	n30w4-1	n030w8_1_2-7-0-9-3-6-0-6	n30w8-1
	n030w4_1_6-7-5-3	n30w4-2	n030w8_1_6-7-5-3-5-6-2-9	n30w8-2
35	n035w4_0_1-7-1-8	n35w4-1	n035w8_0_6-2-9-8-7-7-9-8	n35w8-1
	n035w4_2_8-8-7-5	n35w4-2	n035w8_1_0-8-1-6-1-7-2-0	n35w8-2
40	n040w4_0_2-0-6-1	n40w4-1	n040w8_0_0-6-8-9-2-6-6-4	n40w8-1
	n040w4_2_6-1-0-6	n40w4-2	n040w8_2_5-0-4-8-7-1-7-2	n40w8-2
50	n050w4_0_0-4-8-7	n50w4-1	n050w8_1_1-7-8-5-7-4-1-8	n50w8-1
	n050w4_0_7-2-7-2	n50w4-2	n050w8_1_9-7-5-3-8-8-3-1	n50w8-2
60	n060w4_1_6-1-1-5	n60w4-1	n060w8_0_6-2-9-9-0-8-1-3	n60w8-1
	n060w4_1_9-6-3-8	n60w4-2	n060w8_2_1-0-3-4-0-3-9-1	n60w8-2
70	n070w4_0_3-6-5-1	n70w4-1	n070w8_0_3-3-9-2-3-7-5-2	n70w8-1
	n070w4_0_4-9-6-7	n70w4-2	n070w8_0_9-3-0-7-2-1-1-0	n70w8-2
80	n080w4_2_4-3-3-3	n80w4-1	n080w8_1_4-4-9-9-3-6-0-5	n80w8-1
	n080w4_2_6-0-4-8	n80w4-2	n080w8_2_0-4-0-9-1-9-6-2	n80w8-2
100	n100w4_0_1-1-0-8	n100w4-1	n100w8_0_0-1-7-8-9-1-5-4	n100w8-1
	n100w4_2_0-6-4-6	n100w4-2	n100w8_1_2-4-7-9-3-9-2-8	n100w8-2
110	n110w4_0_1-4-2-8	n110w4-1	n110w8_0_2-1-1-7-2-6-4-7	n110w8-1
	n110w4_0_1-9-3-5	n110w4-2	n110w8_0_3-2-4-9-4-1-3-7	n110w8-2
120	n120w4_1_4-6-2-6	n120w4-1	n120w8_0_0-9-9-4-5-1-0-3	n120w8-1
	n120w4_1_5-6-9-8	n120w4-2	n120w8_1_7-2-6-4-5-2-0-2	n120w8-2

Table 8: Results for the static INRC–II over 8 weeks with time limit set to 8 and 24 hours.

	Time limit: 8 hours				Time limit: 1 day			
Instance	Time (s)	root LB	LB	UB	Time (s)	LB	UB	LNS UB
n030w8-1	425	1993.7	**2010**	**2010$^{\mathbf{H}}$**				2025
n030w8-2	471	1709.4	**1720**	**1720$^{\mathbf{D}}$**				1740
n035w8-1	-	2407.8	2425	2505^{H}	-	2430	2460^{H}	2475
n035w8-2	-	2152.2	2205	2270^{H}	-	2215	2265^{H}	2265
n040w8-1	-	2463.9	2490	2515^{H}	-	2495	2515^{B}	2605
n040w8-2	-	2284	2305	2330^{H}	-	2310	2325^{D}	2320
n050w8-1	-	4777.5	4805	4810^{B}	-	4805	4810^{H}	4830
n050w8-2	7731	4743.3	**4765**	**4765$^{\mathbf{B}}$**				4775
n060w8-1	-	2099	2120	2150^{D}	-	2125	2145^{H}	2200
n060w8-2	-	2393.4	2410	2435^{H}	-	2415	2430^{H}	2475
n070w8-1	-	4474.7	4505	4530^{H}	69115	**4520**	**4520$^{\mathbf{B}}$**	4555
n070w8-2	-	4636.3	4660	4690^{H}	20312	**4675**	**4675$^{\mathbf{H}}$**	4700
n080w8-1	-	3941.7	3955	4030^{H}	-	3955	4055^{H}	4110
n080w8-2	-	4286.9	4295	4335^{H}	-	4295	4325^{H}	4335
n100w8-1	-	2013.5	2025	2030^{H}	-	2025	2030^{H}	2065
n100w8-2	-	2128.6	2140	2145^{H}	-	2140	2145^{H}	2190
n110w8-1	2456	3990	**3990**	**1315$^{\mathbf{B}}$**				4000
n110w8-2	-	3440	3440	3460^{H}	11430	**3440**	**3440$^{\mathbf{B}}$**	3440
n120w8-1	-	2440	2440	2450^{D}	-	2440	2450^{B}	2450
n120w8-2	14162	2871.7	**2875**	**2875$^{\mathbf{B}}$**				2875

Table 9: Results for the INRC–II using COIN-OR CBC solver; time limit was set to one hour per week.

Instance	Time (s)	LB	UB	Instance	Time (s)	LB	UB
n030w4-1	86	**1670**	**1670**	n030w8-1	918	**2010**	**2010**
n030w4-2	1425	**1815**	**1815**	n030w8-2	4774	**1720**	**1720**
n035w4-1	71	**1360**	**1360**	n035w8-1	28802	2425	2585
n035w4-2	81	**1080**	**1080**	n035w8-2	28824	2195	2365
n040w4-1	170	**1565**	**1565**	n040w8-1	28800	2490	2565
n040w4-2	36	**1750**	**1750**	n040w8-2	28847	2305	2435
n050w4-1	1076	**1315**	**1315**	n050w8-1	28804	4800	4835
n050w4-2	206	**1315**	**1315**	n050w8-2	10627	**4765**	**4765**
n060w4-1	42	**2450**	**2450**	n060w8-1	28825	2115	2275
n060w4-2	31	**2675**	**2675**	n060w8-2	28817	2410	2595
n070w4-1	37	**2380**	**2380**	n070w8-1	28802	4505	4540
n070w4-2	75	**2115**	**2115**	n070w8-2	28803	4660	4730
n080w4-1	7806	**3300**	**3300**	n080w8-1	28861	3950	4105
n080w4-2	14400	3185	3190	n080w8-2	28844	4295	4440
n100w4-1	138	**1170**	**1170**	n100w8-1	28866	2025	2080
n100w4-2	83	**1780**	**1780**	n100w8-2	28840	2140	2215
n110w4-1	637	**2330**	**2330**	n110w8-1	10108	**3990**	**3990**
n110w4-2	59	**2455**	**2455**	n110w8-2	28800	3440	3490
n120w4-1	14288	**2020**	**2020**	n120w8-1	28820	2440	2450
n120w4-2	67	**2050**	**2050**	n120w8-2	28819	2875	2895

Table 10: Comparison of the rotation and roster formulations over INRC–II 4-weeks instances; time limit was set to 4 hours.

Instance	Rotations		Rosters							
	LB	UB	Root LB	Increase	LB	Increase	UB	Decrease	CBC UB	Decrease
n030w4-1	1615	1685	1659.5	2.76%	1670	3.41%	1670	0.89%	1670	0.89%
n030w4-2	1740	1840	1809.2	3.98%	1815	4.31%	1815	1.36%	1815	1.36%
n035w4-1	1250	1415	1337.1	6.97%	1360	8.80%	1360	3.89%	1360	3.89%
n035w4-2	1045	1145	1075.6	2.93%	1080	3.35%	1080	5.68%	1080	5.68%
n040w4-1	1335	1640	1535.2	15.00%	1565	17.23%	1565	4.57%	1565	4.57%
n040w4-2	1570	1865	1741.7	10.94%	1750	11.46%	1750	6.17%	1750	6.17%
n050w4-1	1195	1445	1295.5	8.41%	1315	10.04%	1315	9.00%	1315	9.00%
n050w4-2	1200	1405	1302.1	8.51%	1315	9.58%	1315	6.41%	1315	6.41%
n060w4-1	2380	2465	2434.9	2.31%	2450	2.94%	2450	0.61%	2450	0.61%
n060w4-2	2615	2730	2664.3	1.89%	2675	2.29%	2675	2.01%	2675	2.01%
n070w4-1	2280	2430	2370.3	3.96%	2380	4.39%	2380	2.06%	2380	2.06%
n070w4-2	1990	2125	2105	5.78%	2115	6.28%	2115	0.47%	2115	0.47%
n080w4-1	3140	3320	3292	4.84%	3300	5.10%	3300	0.60%	3300	0.60%
n080w4-2	3045	3240	3177.1	4.34%	3185	4.60%	3185	1.70%	3190	1.54%
n100w4-1	1055	1230	1167.2	10.64%	1170	10.90%	1170	4.88%	1170	4.88%
n100w4-2	1470	1855	1777.7	20.93%	1780	21.09%	1780	4.04%	1780	4.04%
n110w4-1	2210	2390	2321.3	5.04%	2330	5.43%	2330	2.51%	2330	2.51%
n110w4-2	2255	2525	2455	8.87%	2455	8.87%	2455	2.77%	2455	2.77%
n120w4-1	1790	2165	2011.5	12.37%	2020	12.85%	2020	6.70%	2020	6.70%
n120w4-2	1820	2220	2045.7	12.40%	2050	12.64%	2050	7.66%	2050	7.66%
	Average			**7.64%**		**8.28%**		**3.70%**		**3.69%**

Table 11: Comparison of the rotation and roster formulations over INRC–II 8-weeks instances; time limit was set to 8 hours.

Instance	Rotations		Rosters (over 8 hours)							
	LB	UB	Root LB	Increase	LB	Increase	UB	Decrease	CBC UB	Decrease
n030w8-1	1920	2070	1993.7	3.84%	2010	4.69%	2010	2.90%	2010	2.90%
n030w8-2	1620	1735	1709.4	5.52%	1720	6.17%	1720	0.86%	1720	0.86%
n035w8-1	2330	2555	2407.8	3.34%	2425	4.08%	2505	1.96%	2585	-1.17%
n035w8-2	2180	2305	2152.2	-1.28%	2205	1.15%	2270	1.52%	2365	-2.60%
n040w8-1	2340	2620	2463.9	5.29%	2490	6.41%	2515	4.01%	2565	2.10%
n040w8-2	2205	2420	2284	3.58%	2305	4.54%	2330	3.72%	2435	-0.62%
n050w8-1	4625	4900	4777.5	3.30%	4805	3.89%	4810	1.84%	4835	1.33%
n050w8-2	4530	4925	4743.3	4.71%	4765	5.19%	4765	3.25%	4765	3.25%
n060w8-1	1970	2345	2099	6.55%	2120	7.61%	2150	8.32%	2275	2.99%
n060w8-2	2260	2590	2393.4	5.90%	2410	6.64%	2435	5.98%	2595	-0.19%
n070w8-1	4400	4595	4474.7	1.70%	4505	2.39%	4530	1.41%	4540	1.20%
n070w8-2	4540	4760	4636.3	2.12%	4660	2.64%	4690	1.47%	4730	0.63%
n080w8-1	3775	4180	3941.7	4.42%	3955	4.77%	4030	3.59%	4105	1.79%
n080w8-2	4125	4450	4286.9	3.92%	4295	4.12%	4335	2.58%	4440	0.22%
n100w8-1	2005	2125	2013.5	0.42%	2025	1.00%	2030	4.47%	2080	2.12%
n100w8-2	2125	2210	2128.6	0.17%	2140	0.71%	2145	2.94%	2215	-0.23%
n110w8-1	3870	4010	3990	3.10%	3990	3.10%	3990	0.50%	3990	0.50%
n110w8-2	3375	3560	3440	1.93%	3440	1.93%	3460	2.81%	3490	1.97%
n120w8-1	2295	2600	2440	6.32%	2440	6.32%	2450	5.77%	2450	5.77%
n120w8-2	2535	3095	2871.7	13.28%	2875	13.41%	2875	7.11%	2895	6.46%
	Average			**3.91%**		**4.54%**		**3.35%**		**1.46%**

References

Khaled S. Abdallah and Jaejin Jang. An exact solution for vehicle routing problems with semi-hard resource constraints. Computers & Industrial Engineering, 76:366–377, October 2014. doi: 10.1016/j.cie.2014.08.011.

Mohammed Abdelghany, Amr B. Eltawil, Zakaria Yahia, and Kazuhide Nakata. A hybrid variable neighbourhood search and dynamic programming approach for the nurse rostering problem. Journal of Industrial & Management Optimization, 17(4):2051, 2021. doi: 10.3934/jimo.2020058.

Anmar Abuhamdah, Wadii Boulila, Ghaith M. Jaradat, Anas M. Quteishat, Mutasem K. Alsmadi, and Ibrahim A. Almarashdeh. A novel population-based local search for nurse rostering problem. International Journal of Electrical and Computer Engineering, 11(1):471, 2021. doi: 10.11591/ijece.v11i1.pp471-480.

Jonathan F. Bard and Hadi W. Purnomo. Preference scheduling for nurses using column generation. European Journal of Operational Research, 164(2):510–534, 2005. doi: 10.1016/j.ejor.2003.06.046.

Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. Operations Research, 46(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

J. E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. Networks, 19(4):379–394, 1989. doi: 10.1002/net.3230190402.

Andrea Bettinelli, Alberto Ceselli, and Giovanni Righini. A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. Mathematical Programming Computation, 6(2):171–197, June 2014. doi: 10.1007/s12532-014-0064-0.

Manuel A. Bolívar, Leonardo Lozano, and Andrés L. Medaglia. Acceleration strategies for the weight constrained shortest path problem with replenishment. Optimization Letters, 8(8):2155–2172, 2014. doi: 10.1007/s11590-014-0742-x.

Edmund K. Burke and Tim Curtois. New approaches to nurse rostering benchmark instances. European Journal of Operational Research, 237(1):71–81, 2014. doi: 10.1016/j.ejor.2014.01.039.

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. doi: 10.1287/ijoc.1120.0510.

W Matthew Carlyle, Johannes O Royset, and R Kevin Wood. Lagrangian relaxation and enumeration for solving constrained shortest-path problems. Networks: an international journal, 52(4):256–270, 2008.

Sara Ceschia, Nguyen Dang, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf. The Second International Nurse Rostering Competition. Annals of Operations Research, 274(1-2):171–186, 2019. doi: 10.1007/s10479-018-2816-0.

Sara Ceschia, Rosita Guido, and Andrea Schaerf. Solving the static INRC-II nurse rostering problem by simulated annealing based on large neighborhoods. Annals of Operations Research, 288(1):95–113, 2020. doi: 10.1007/s10479-020-03527-6.

Brenda Cheang, Haibing Li, Andrew Lim, and Brian Rodrigues. Nurse rostering problems – a bibliographic survey. European Journal of Operational Research, 151(3):447–460, 2003.

Luciano Costa, Claudio Contardo, and Guy Desaulniers. Exact Branch-Price-and-Cut Algorithms for Vehicle Routing. Transportation Science, 53(4):946–985, July 2019. doi: 10.1287/trsc.2018.0878.

Tim Curtois and Rong Qu. Computational results on new staff scheduling benchmark instances. ASAP Research Group, School of Computer Science, University of Nottingham, page 5, 2014.

Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. Column generation, volume 5. Springer Science & Business Media, 2006.

Anders Dohn and Andrew Mason. Branch-and-price for staff rostering: An efficient implementation using generic programming and nested column generation. European Journal of Operational Research, 230(1):157–169, 2013. doi: 10.1016/j.ejor.2013.03.018.

Olivier Du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. Discrete Mathematics, 194(1–3):229–237, 1999.

Irina Dumitrescu and Natashia Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. Networks: An International Journal, 42(3):135–153, 2003.

Rafael A.M. Gomes, Túlio A.M. Toffolo, and Haroldo Gambini Santos. Variable neighborhood search accelerated column generation for the nurse rostering problem. 4th International Conference on

Variable Neighborhood Search. Electronic Notes in Discrete Mathematics, 58:31–38, 2017. doi: 10.1016/j.endm.2017.03.005.

Matthieu Gérard, François Clautiaux, and Ruslan Sadykov. Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce. European Journal of Operational Research, 252(3):1019–1030, 2016. doi: 10.1016/j.ejor.2016.01.036.

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. doi: 10.1007/s10479-012-1062-0.

Fang He and Rong Qu. A constraint programming based column generation approach to nurse rostering problems. Computers & Operations Research, 39(12):3331–3343, 2012. doi: 10.1016/j.cor.2012.04.018.

Stefan Irnich, Guy Desaulniers, et al. Shortest path problems with resource constraints. In Column generation, chapter 2, pages 33–65. Springer US, 2005.

Brigitte Jaumard, Frédéric Semet, and Tsevi Vovor. A generalized linear programming model for nurse scheduling. European Journal of Operational Research, 107(1):1–18, 1998. doi: 10.1016/S0377-2217(97)00330-5.

H.C Joksch. The shortest route problem with constraints. Journal of Mathematical Analysis and Applications, 14(2):191–197, 1966. doi: 10.1016/0022-247X(66)90020-5.

Niklas Kohl and Stefan E Karisch. Airline crew rostering: Problem types, modeling, and optimization. Annals of Operations Research, 127(1–4):223–257, 2004.

Antoine Legrain and Jérémy Omer. wssuite/nursescheduler: Soft domination, October 2023.

Antoine Legrain, Jérémy Omer, and Samuel Rosat. A rotation-based branch-and-price approach for the nurse scheduling problem. Mathematical Programming Computation, (12):417–450, 2020. doi: 10.1007/s12532-019-00172-4.

Warner Lensing. Heuristic Branch-and-Price algorithms for the nurse rostering problem. Unpublished PhD dissertation, University of Groningen, Faculty of Economics and Business, 2020.

Federico Liberatore, Giovanni Righini, and Matteo Salani. A column generation algorithm for the vehicle routing problem with soft time windows. 4OR, 9(1):49–82, 2011. doi: 10.1007/s10288-010-0136-6.

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. doi: 10.1007/s10951-009-0108-x.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS Journal on Computing, 30(2):339–360, 2018.

Ali Gul Qurashi, Eiichi Taniguchi, and Tadashi Yamada. Column generation-based heuristics for vehicle routing problem with soft time windows. Journal of the Eastern Asia Society for Transportation Studies, 8, 2010.

Ali Gul Qureshi, Eiichi Taniguchi, and Tadashi Yamada. Column Generation-based Heuristics for Vehicle Routing Problem with Soft Time Windows. Journal of the Eastern Asia Society of Transportation Studies, 8:15, 2010.

Giovanni Righini and Matteo Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optimization, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

Michael Römer. Future demand uncertainty in personnel scheduling: investigating deterministic lookahead policies using optimization and simulation. In Michael Manitz and Oliver Rose, editors, Proceedings of 30th European Conference on Modelling and Simulation, 2016.

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

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, 2016b. doi: 10.1007/s10479-014-1594-6.

Thomas R. Sexton and Young-Myung Choi. Pickup and delivery of partial loads with "soft" time windows. American Journal of Mathematical and Management Sciences, 6(3-4):369–398, 1986. doi: 10.1080/01966324.1986.10737200.

Pieter Smet. Constraint reformulation for nurse rostering problems. In Proceedings of the 12th international conference on the practice and theory of automated timetabling, pages 69–80. PATAT, 2018.

Petter Strandmark, Yi Qu, and Timothy Curtois. First-order linear programming in a column generation-based heuristic approach to the nurse rostering problem. Computers & Operations Research, 120:104945, 2020. doi: 10.1016/j.cor.2020.104945.

Mariam Tagmouti, Michel Gendreau, and Jean-Yves Potvin. Arc routing problems with time-dependent service costs. European Journal of Operational Research, 181:30–39, 2007. doi: 10.1016/j.ejor.2006.06.028.

Daniel Villeneuve and Guy Desaulniers. The shortest path problem with forbidden paths. European Journal of Operational Research, 165(1):97–107, 2005. doi: 10.1016/j.ejor.2004.01.032.

Xiaoyan Zhu and Wilbert E. Wilhelm. A three-stage approach for the resource-constrained shortest path as a sub-problem in column generation. Computers & Operations Research, 39(2):164–178, 2012. doi: 10.1016/j.cor.2011.03.008.