The nurse rerostering problem: strategies for reconstructing disrupted schedules

Toni I. Wickert[a,b,*], Pieter Smet[a], Greet Vanden Berghe[a]

[a] *KU Leuven, Department of Computer Science, CODeS & imec, Gebroeders De Smetstraat 1, 9000 Gent, Belgium*
[b] *Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), 91501-970 Porto Alegre, Brazil*

Abstract

The nurse rostering problem is a well-known optimization problem within the field of operational research which seeks to assign nurses to shifts during a scheduling horizon subject to a set of hard and soft constraints. The nurse rerostering problem, meanwhile, occurs when one or more nurses already scheduled to work cannot be present due to unforeseen events such as, for example, illness. Such absences may render an existing solution infeasible and thus a fast method is required to recreate the roster. The present research explores several novel strategies for rerostering based on relaxations of different problem parameters, including soft constraints and the rescheduling horizon. A general integer programming formulation is developed considering multi-skilled nurses and various constraints commonly found in real-world problems. Secondly, the nurse rerostering problem is explored by rescheduling both the entire scheduling horizon and only a limited part. Additionally, the impact of considering both the soft constraints from the original nurse rostering problem and a relaxation of them is evaluated when solving the nurse rerostering problem. Finally, a variable neighborhood descent heuristic is developed to address the problem without the use of a solver. A computational study on instances adapted from the Second International Nurse Rostering Competition and on real-world instances from a Lisbon hospital demonstrates that the proposed

[*] toniismael.wickert@kuleuven.be
Email addresses: `toniismael.wickert@kuleuven.be` (Toni I. Wickert), `pieter.smet@cs.kuleuven.be` (Pieter Smet), `greet.vandenberghe@cs.kuleuven.be` (Greet Vanden Berghe)

Preprint submitted to Computers & Operations Research *October 24, 2018*

strategies solve realistic large-scale rerostering problems to (near-)optimality in limited computation time.

Keywords: Nurse Rerostering Problem, Nurse Rostering Problem, Integer Programming, Variable Neighborhood Descent

1. Introduction

Work absences occur due to a variety of reasons. According to Forbes (2013), common causes of employee absenteeism include heavy workloads, childcare obligations, disengagement with the coworkers or company, illness, medical appointments, injuries caused by accidents, interviews for other jobs, bullying or harassment by coworkers or bosses, among others. Depending on the working area, unscheduled absence rates typically range from 5% to 10%. Emergency services and healthcare in hospitals have the highest absenteeism rates (10.7%) when compared against other sectors such as, for example, utilities (8.7%), transportation (8.5%), customer services (7.7%) or manufacturing (6.4%) (Aguirre & Kerin, 2014). Forbes (2013) estimates the annual cost of lost productivity in the United States to be $3.6 billion for nurses and $0.25 billion for physicians.

Effectively managing disruptions caused by high absenteeism rates is thus clearly of vital importance but at the same time presents a difficult task to manual planners. An automated method based on computational techniques is essential in supporting the decision maker whenever a quick solution for handling disruptions is required. The Nurse Rerostering Problem (NRRP), as defined by Moz & Pato (2003), occurs when one or more nurses cannot work in the shifts they were previously assigned. If no pool of reserve nurses exists to replace those absent, the current roster must be rebuilt using a rerostering method.

This work revisits the NRRP by exploring several strategies based on relaxations of specific problem parameters which may be applied when addressing the disruptions. First, a general integer programming formulation is developed which includes both the constraints from the Nurse Rostering Problem (NRP) and the additional NRRP restrictions. Second, two types of relaxation strategies are proposed and evaluated. The first strategy determines which part of the scheduling horizon to consider when rerostering: either the complete period or only a restricted part. The second strategy concerns which constraints are included when solving the NRRP. An approach which relaxes some of the NRP constraints and only includes the NRRP constraints is evaluated. Finally, a Variable Neighborhood Descent (VND) heuristic is developed to address the problem without the use of a Mixed Integer Programming (MIP) solver. The primary objective of the VND heuristic is to provide hospitals with a solver free from third-party dependencies and which can be implemented without any additional cost. Computational experiments are conducted on adapted instances from the Second International Nurse Rostering Competition (INRC-II) and real-world instances from a Lisbon hospital.

The present paper addresses four primary research questions, namely:

- Is it necessary to include all original hard and soft constraints from the NRP when attempting to generate good quality solutions for the NRRP?
- What is the difference, in terms of computation time and solution quality, between solving the full NRRP model and a surrogate model which only considers disruption minimization objectives?
- What is the impact of the considered scheduling horizon when rerostering? Should only those days where nurses are absent be considered? The complete scheduling horizon? From the first absent day until the last absent day? Should this restricted period be extended with some days before and after?
- Is it possible to generate competitive results using a simple heuristic, compared against an integer programming formulation using a state of the art commercial solver?

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature related to NRP and NRRP. Section 3 specifies the NRRP and discusses how it differs from the NRP. Section 4 presents the general integer programming formulation, including the NRRP and NRP constraints. Following this, Section 5 details the Variable Neighborhood Descent (VND) algorithm. Section 6 describes the computational experiments, while Section 7 concludes the paper and indicates directions for future research.

2. Literature review

Due to its practical relevance, the nurse rostering problem has been the subject of many research papers throughout recent decades (Van den Bergh et al., 2013). Interested readers are referred to Ernst et al. (2004) for a general overview of staff scheduling, while Burke et al. (2004) provide a survey focused on nurse rostering problems. Several solving methods have been proposed throughout the academic literature for addressing the NRP. These include integer programming (Mischek & Musliu, 2017), decomposition combined with integer programming (Valouxis et al., 2012; Legrain et al., 2017) column generation combined with variable neighborhood search (Gomes et al., 2017), branch and price and variable depth search (Burke & Curtois, 2014), and adaptive variable neighborhood search (Tassopoulos et al., 2015). Despite the NRRP representing a common problem in hospitals, Clark et al. (2015) identified only eight relevant papers in the academic literature. The solution approaches proposed in these studies are typically based on heuristic search and integer programming.

Moz & Pato (2003) were the first authors to formally define the NRRP. In addition to a multi-commodity network flow formulation, they also introduced an aggregated integer programming model which decreased the model size enabling the problem to be solved faster (Moz & Pato, 2004). To evaluate their formulations, 16 real instances from a Lisbon hospital were utilized. Using CPLEX, 15 out of these 16 instances were solved to optimality within a time limit of two hours. More recently, Moz & Pato (2007) developed a Genetic Algorithm (GA) and performed tests on the same set of real-world instances. The GA outperformed the constructive heuristic of Moz & Pato (2003) in terms of solution quality within an acceptable time limit.

Maenhout & Vanhoucke (2011) investigated whether it is necessary to re-optimize the complete scheduling horizon or if restricting rerostering possibilities represents a feasible strategy. The best results were obtained when only a very limited fraction of the roster, typically between 10% and 30%, was re-optimized. In a follow-up study, Maenhout & Vanhoucke (2013) further explored which parts of a disrupted roster to re-optimize when rerostering. An empirical study confirmed previous results insofar as they determined that it is unnecessary to consider the complete scheduling horizon to obtain good solutions. Instead, only a period before and after the disruptions should be considered, including the period of disruptions themselves. The total number of absent nurses had little impact on the length of the rerostering period that should be considered. However, the more clustered the disruptions, the more important the days before and after the disruptions become. Regarding which resources to consider when rerostering, they concluded that it is unnecessary to consider all nurses but instead only those whose roster is disrupted along with an additional subset of nurses, specially selected for any particular reason or just random. The fewer absent nurses, the smaller this additional number of required nurses.

Bäumelt et al. (2016) developed two parallel algorithms executed on a Graphics Processing Unit (GPU) to solve the NRRP, using the instances of Moz & Pato (2007). Two models of parallelization were compared: a homogeneous model in which the entire algorithm runs on a GPU, and a heterogeneous model where the algorithm is partially solved on a CPU and partially on a GPU. The homogeneous model resulted in solutions being generated between 12.6 and 17.7 times faster for instances with 19 and 32 nurses, respectively. By contrast, the heterogeneous model provided average speedups of 2.3 and 2.4 for the same datasets. The results demonstrated that the parallel algorithm achieves the same quality of results in significantly shorter computation time compared against the sequential algorithm.

Table 1 details the scope of this work compared to the existing literature. The second column classifies each variant of the NRRP according to the $\alpha|\beta|\gamma$ notation proposed by De Causmaecker & Vanden Berghe (2011). In general, the problem considered in the present paper strongly generalizes previously-published models by including more constraints from practice. Moreover, in contrast to the existing literature, the proposed model considers multi-skilled nurses. The third, fourth and fifth columns compare which rerostering strategies have been applied with respect to existing rerostering models. The comparison shows that most previous studies have not considered any specific strategy when rerostering. Only Maenhout & Vanhoucke (2011, 2013) have investigated different relaxations of the available nurses and the scheduling horizon, the latter of which is also explored in the present paper. Finally, the last two columns detail which techniques have been employed to solve the NRRP. This comparison reveals that in the last decade the focus has shifted towards heuristic methods.

Table 1: Existing approaches to the NRRP.

		Strategies			Solving technique	
Reference	Problem classification	Constraint relaxation	Scheduling horizon relaxation	Staffing size	Exact method	Heuristic approach
This paper	*ASN/VN/PLR*	✓	✓		✓	✓
Moz & Pato (2003)	*AS/RN/PR*				✓	✓
Moz & Pato (2004)	*AS/RN/PR*				✓	
Moz & Pato (2007)	*AS/RN/PR**					✓
Pato & Moz (2008)	*AS/RN/PRM**					✓
Maenhout & Vanhoucke (2011)	*ASB/RN/PLRM**		✓	✓		✓
Maenhout & Vanhoucke (2013)	*ASB/V3/PLR**		✓	✓		✓
Bäumelt et al. (2016)	*AS/RN/PR**					✓

(*) Papers without a mathematical model, but, the problems considered in their studies are classified as described.

3. The nurse rerostering problem

The NRP assigns nurses to shifts during a scheduling horizon. These assignments are subject to a set of hard and soft constraints. Hard constraints must be respected, while violations of the soft constraints are penalized in the objective function. Table 2 shows an example of a feasible NRP solution with five nurses and a scheduling horizon of seven days. The shifts are Early (E), Late (L) and Night (N), while dashes indicate a day off. At least one nurse is required to work during each shift on each day.

Table 2: Example of a roster with seven days and five nurses.

Nurse	Mon	Tue	Wed	Thu	Fri	Sat	Sun
N1	N	–	E	E	E	–	–
N2	L	L	–	–	L	L	L
N3	N	N	N	N	–	–	–
N4	–	–	–	–	N	N	N
N5	E	E	L	L	–	E	E

Table 3: Roster with two absent nurses in gray.

Nurse	M	T	W	T	F	S	S
N1	N	–	E	E	E	–	–
N2	L	L	–	–	L	L	L
N3	N	~~N~~	N	N	–	–	–
N4	–	–	–	–	~~N~~	N	N
N5	E	E	L	L	–	E	E

Table 4: New solution after rerostering.

Nurse	M	T	W	T	F	S	S
N1	N	–	E	E	E	–	–
N2	L	L	–	–	L	L	L
N3	N	~~N~~	N	N	N	–	–
N4	–	N	–	–	~~N~~	N	N
N5	E	E	L	L	–	E	E

The NRRP occurs when a nurse who is scheduled to work, cannot be present due to unforeseen events. This disruption may make the solution infeasible, thereby requiring another nurse to be reallocated to cover the absence. However, this new solution must still comply with labor rules and institutional constraints as per the original rostering problem. Moreover, it should be as similar as possible to the original roster given that volatile and unpredictable schedule changes can wreak havoc with workers' child-care arrangements, school classes and other personal responsibilities (Williams et al., 2017).

Table 3 shows a disruption in which two absent nurses are highlighted in gray. Considering the minimum of one nurse per day/shift, these two absences render the solution infeasible, as now nobody is working the Night shifts on Tuesday and Friday. Table 4 presents one possible new solution after rerostering, wherein Nurse 4, who before had a free day on Tuesday, is now working a Night shift. Nurse 3, who had a free day on Friday, is now also working the Night shift.

In this trivial example, the solution's feasibility may be restored by simply swapping two nurses. This operation has minimal impact on the existing solutions and maintains the same number of working shifts as in the original roster. Generally, however, the situation is more complex as various time-related constraints impose additional restrictions on the solution, such as, for example, minimum/maximum number of consecutive days worked or minimum rest time between two consecutive working days. In addition to minimizing the number of changes, these constraints must also be respected. Moreover, if they are modeled as soft constraints, their violations should be minimized.

4. General integer programming formulation for the NRRP

This section presents a general integer programming formulation for the NRRP. Moz & Pato (2003, 2004) were, so far, the only authors to address the nurse rerostering problem using integer programming. Moz & Pato (2003) formulated the problem as an integer multi-commodity flow problem with side constraints in a multi-level acyclical network. This formulation was further improved in Moz & Pato (2004) by including node aggregation in the network. In contrast to these flow models, the formulation proposed in the present paper is based on an assignment problem, thereby enabling more general problem characteristics to be included such as multi-skilled nurses and various hard and soft constraints typically found in hospitals (Ceschia et al., 2014). Appendix A details the full integer programming formulation containing all original NRP constraints. In what follows, the presentation of the integer programming formulation is restricted to constraints and objectives relevant to the rerostering problem. Table 5 presents the problem's parameters in addition to the main and auxiliary decision variables employed in the NRRP formulation.

$$\textbf{Min} \quad \sum_{n\in N}\sum_{d\in D} v^{13}_{nd}\omega^{13} + \quad \sum_{n\in N}\sum_{i\in\{14,15\}} \hat{v}^{i}_{n}\omega^{i} + A.1 \tag{1}$$

Subject to

$$\hat{c}_{nd} + \sum_{s\in S}\sum_{k\in K} x_{ndsk} \leq 1 \qquad \forall n\in N, d\in D \tag{2}$$

$$\sum_{k\in K}(c_{ndsk} + x_{ndsk}) \leq 2y'_{nds} \qquad \forall n\in N\setminus\hat{N}, d\in D, s\in S \tag{3}$$

$$\sum_{k\in K}(c_{ndsk} + x_{ndsk}) + y''_{nds} \geq 2y'_{nds} \qquad \forall n\in N\setminus\hat{N}, d\in D, s\in S \tag{4}$$

$$\sum_{s\in S} y''_{nds} - 2v^{13}_{nd} \leq 0 \qquad \forall n\in N\setminus\hat{N}, d\in D \tag{5}$$

$$\sum_{d\in D}\sum_{s\in S}\sum_{k\in K} x_{ndsk} + \hat{v}^{14}_{n} \geq \delta_n \qquad \forall n\in N \tag{6}$$

$$\sum_{d\in D}\sum_{s\in S}\sum_{k\in K} x_{ndsk} - \hat{v}^{15}_{n} \leq \delta_n \qquad \forall n\in N \tag{7}$$

Objective function (1) minimizes a weighted sum of different terms related to the NRRP and NRP. The first term penalizes changes made in the rerostering solution with

Table 5: Sets and variables employed in the formulation.

Symbol	Definition
Parameters	
$n \in N$	index of the nurse, where N is the set of nurses;
$\hat{N} \subseteq N$	set of absent nurses;
$d \in D$	index of the day, where D is the set of days;
$s \in S$	s index of the shift, where S is the set of shifts
$k \in K$	k index of the skill, where K is the set of skills
ω^i	weight for violating the lower and/or upper limits of soft constraint i.
δ_n	original number of assignments associated with nurse n;
$c_{ndsk} \in \{0,1\}$	value which is 1 if nurse n is allocated to shift s and day d with skill k in the original roster, 0 otherwise;
$\hat{c}_{nd} \in \{0,1\}$	parameter modeling the disruptions which is 1 if nurse n is absent on day d, 0 otherwise;
Decision Variables	
$x_{ndsk} \in \{0,1\}$	1 if nurse n is allocated to shift s and day d with skill k, 0 otherwise;
Auxiliary Variables	
$y'_{nds} \in \{0,1\}$	1 if nurse n works in the original schedule or in the new roster on day d and shift s, 0 otherwise;
$y''_{nds} \in \{0,1\}$	auxiliary variable to calculate the number of changes compared to the original roster;
$v^{13}_{nd} \in \mathbb{N}_0$	auxiliary variable to calculate the violations of the number of changes compared to the original roster;
$\hat{v}^{14}_{n} \in \mathbb{N}_0$	auxiliary variable to calculate the violations of the number of working days less than the original roster;
$\hat{v}^{15}_{n} \in \mathbb{N}_0$	auxiliary variable to calculate the violations of the number of working days more than the original roster

respect to the original roster. The second term minimizes the difference in number of working days in the roster before and after rerostering. The last part of the objective function consists of the soft constraint violations of the NRP as defined in Equation (A.1) (Appendix A). Constraints (2) ensure that an absent nurse is not scheduled to work. Constraints (3), (4) and (5) determine the number of changes in the new roster compared to the original roster. Constraints (6) and (7) calculate the change in number of working days.

An example of how Constraints (3), (4) and (5) count the number of changes for a single nurse is provided in Tables 6 and 7. The formulation penalizes whenever a working day is changed to a day off (or vice versa) and whenever a shift which was assigned is modified. Changes regarding assigned skills are not penalized, as these cases are not considered to have a significant impact on the nurses. The example in

Table 6 considers three days and three shifts: Early (E), Late (L) and Night (N). The first row of integer values represents the current solution. In this case, the nurse works a Late shift on the first day, the second day is free, and a Night shift on the third day. The second row represents the solution after rerostering. The nurse now works a Night shift on the first day, an Early shift on the second day and another Night shift on the third day. The third row is the sum of the current solution and the newly rerostered solution. Finally, the fourth row shows the values variables y'_{nds} assume. In this example there are two changes, one is a shift change from Late to Night on the first day, and the second change concerns the second day where the nurse previously had a day off, but for which she is now scheduled to work an Early shift. There are no changes on the third day as the nurse continues working on the already-scheduled Night shift. Table 7 presents the penalization results stored in variable v^{13}_{nd} (last column of the right table). On the first and second day, the variable assumes a value of 1, meaning that there is one change (one violation), and on the third day the variable assumes a value of 0, denoting zero violations.

Table 6: Example of a reroster solution with two changes.

Day 1			Day 2			Day 3			
E	L	N	E	L	N	E	L	N	Description
0	1	0	0	0	0	0	0	1	$\sum_{k\in K} c_{ndsk}$ (current solution)
0	0	1	1	0	0	0	0	1	$\sum_{k\in K} x_{ndsk}$ (newly rerostered solution)
0	1	1	1	0	0	0	0	2	$\sum_{k\in K}(c_{ndsk}+x_{ndsk})$
0	1	1	1	0	0	0	0	1	$\sum_{k\in K}(c_{ndsk}+x_{ndsk}) \leq 2y'_{nds}$ (y'_{nds} values)

5. Variable neighborhood descent

This section presents a Variable Neighborhood Descent (VND) heuristic specifically designed to address the NRRP based on the general concepts first introduced by Mladenović & Hansen (1997). Legrain et al. (2014) developed simple procedures based on local search for the NRP. Their solving method addresses the lack of choice

Table 7: Example of how Constraints (4) and (5) are evaluated.

		$\sum_{k\in K}(c_{ndsk}+x_{ndsk})+y''_{nds} \geq 2y'_{nds}$		$\sum_{s\in S} y''_{nds} - 2v^{13}_{nd} \leq 0$	
Day 1	E	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$	$2 - 2v^{13}_{nd} \leq 0$	$v^{13}_{nd}=1$
	L	$1+0+y''_{nds} \geq 2$	$y''_{nds}=1$		
	N	$0+1+y''_{nds} \geq 2$	$y''_{nds}=1$		
Day 2	E	$0+1+y''_{nds} \geq 2$	$y''_{nds}=1$	$1 - 2v^{13}_{nd} \leq 0$	$v^{13}_{nd}=1$
	L	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$		
	N	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$		
Day 3	E	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$	$0 - 2v^{13}_{nd} \leq 0$	$v^{13}_{nd}=0$
	L	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$		
	N	$1+1+y''_{nds} \geq 2$	$y''_{nds}=0$		

aside from either a manual solution method or a high-cost commercial package. The present research addresses this same dilemma by introducing a VND heuristic for the NRRP. The VND heuristic is selected primarily due to its simplicity, its ability to integrate several neighborhood structures, and the successful application of this algorithm and its variants for the NRP (Burke et al., 2008; Zheng et al., 2017; Gomes et al., 2017). The proposed method seeks to generate results quickly and without the dependency of expensive third-party software packages when funding is limited.

5.1. Main method - VND

Algorithm 1 outlines the main method which takes as input parameters the current solution, the maximum number of top-level loop iterations and the number of iterations after which an intensification/diversification procedure is called. Section 6.1.4 provides additional details regarding the algorithm's parameters. Function *OFV(cs)* returns the objective function value (OFV) of the solution *cs*. Note that it is only when the objective value is calculated for the first time that all nurses, days, shifts and skills are considered. When neighboring solutions are evaluated, a relative recalculation of the objective value is employed in order to speed up computation time.

In each iteration of the top-level loop (lines 5-17), a sequence of procedures which explore different neighborhoods is executed until either no improvement is found or a feasible solution is reached. All neighborhoods, except for *changeShift*, *assignMissingShiftDeleteNext*, *intensDiverLS* , consider only the days on which absences occur.

However, it is not always possible to remedy infeasibilities by exploring only these restricted neighborhoods. For this reason, *changeShift*, *assignMissingShiftDeleteNext* and *intensDiverLS* are also included in the VND heuristic as they explore a larger proportion of the search space, thereby increasing the likelihood that infeasibilities will be solved should the deterministic neighborhoods fail in doing so, albeit at the expense of longer computational runtimes. The algorithm terminates by returning the best solution found. Input parameters are passed to each subroutine, however, in the main pseudo-code they are omitted for presentation reasons.

Algorithm 1: Variable Neighborhood Descent (VND).

```
Input    : cs current solution, maxTrials maximum number of iterations, maxTrialsIntDiv number of
           iterations after which the intensification/diversification procedure is called
Output   : solution
cs ← assignMissingShift;
cs ← changeAssignMissingShift;
bestOFV ← ∞;
iterations ← 0;
while bestOFV > OFV(cs) or (hasHardViolation(cs) and iterations < maxTrials) do
    bestOFV ← OFV(cs) ;                  // returns the OFV from the current solution cs
    cs ← assignDeleteShift;
    cs ← changeShift;
    cs ← swapShift;
    cs ← assignMissingShift;
    cs ← changeAssignMissingShift;
    cs ← assignMissingShiftDeleteNext;
    if hasHardViolation(cs) and iterations > maxTrialsIntDiv then
        cs ← intensDiverLS;
    end
    iterations ← iterations + 1;
end
return cs ;                                    // returns the best solution found
```

5.2. *Assign and delete shift neighborhood*

Algorithm 2 moves an assigned shift from one nurse to another. Lines 4-18 iterate over each day with insufficient coverage. Line 7 iterates over each working nurse w and each idle nurse f on day d. Line 8 generates a neighboring solution in which the shift and skill assignments of nurse w are reassigned to nurse f. If the neighboring solution does not violate any hard constraints and is the best neighbor found (line 9), the variables are updated (lines 10-11). The neighborhood's size is $O(|D_v||W_d||F_d|)$. The procedure terminates by returning the best solution found.

Algorithm 2: Assign and delete shift.

```
Input  : D_v set of days with insufficient coverage, W_d set nurses working on day d, F_d set of idle nurses on
         day d, cs current solution
Output : Solution
1  improved ← true;
2  while improved do
3      improved ← false;
4      foreach d ∈ D_v do
5          bestNeighbor ← null;
6          bestNeighborOFV ← OFV(cs);
7          foreach w ∈ W_d, f ∈ F_d do
8              cs′ ← assignDelete(cs, d, w, f) ;            // returns null if infeasible
9              if cs′ ≠ null and OFV(cs′) < bestNeighborOFV then
10                 bestNeighbor ← cs′;
11                 bestNeighborOFV ← OFV(cs′);
12             end
13         end
14         if bestNeighbor ≠ null then                      // if an improved neighbor is found
15             cs ← bestNeighbor;
16             improved ← true;
17         end
18     end
19 end
20 return cs;
```

5.3. Change shift neighborhood

Algorithm 3 changes nurses' assignments on consecutive days. The method *assignShift* (line 8) generates a neighboring solution in which shift s is assigned to nurse n on day $d+d'$. If there is a feasible solution and the objective value of the neighboring solution is lower than that of the current solution, the variables are updated accordingly (lines 9-12). If shift s is a working shift (line 14), the nurses' skills are iterated over (loop 15-22). Method *assignSkill* (line 16) changes the assigned skill of nurse n on day $d+d'$ in shift s to k. If there is a feasible solution and the new objective value is lower than the current objective value (line 17), the current solution and variables are updated (lines 18-20). The neighborhood's size is $O(|D||S||w||K_n|)$. The goal of the variable d' is to change the assignments in a sequence of days, thereby aiming to reduce the number of violations of constraints concerning the minimum/maximum number of consecutive working days and similar constraints. Preliminary experiments demonstrate that the most suitable value of the parameter w is four. The procedure terminates by returning the best solution found.

Algorithm 3: Change shift.

```
Input  : D set of all days, N set of nurses, S set of shifts, S' set of working shifts,
         K_n set of skills of nurse n, cs current solution, w maximum consecutive days window size
Output : Solution
1  improvedLS ← true;
2  while improvedLS do
3      improvedLS ← false;
4      foreach d ∈ D, n ∈ N, s ∈ S, d' ← 1 to w do
5          improved ← false;
6          csBackup ← cs;
7          if ((d + d') < |D|) then
8              cs' ← assignShift(cs, n, (d + d'), s) ;        // returns null if infeasible
9              if cs' ≠ null and OFV(cs') < OFV(cs) then
10                 improved ← true;
11                 improvedLS ← true;
12             end
13             cs ← cs';
14             if s ∈ S' then
15                 foreach k ∈ K_n do
16                     cs' ← assignSkill(cs, n, (d + d'), s, k);
17                     if cs' ≠ null and OFV(cs') < OFV(cs) then
18                         cs ← cs';
19                         improved ← true;
20                         improvedLS ← true;
21                     end
22                 end
23             end
24         end
25         if not improved then
26             cs ← csBackup;
27         end
28     end
29 end
30 return cs;
```

5.4. *Swap shift neighborhood*

Algorithm 4 swaps the shift and skill assignments of two nurses. The loops (lines 7-8) iterate over each pair of nurses *n1* and *n2*. Line 9 generates a neighboring solution by swapping nurse *n1*'s shift and skill with nurse *n2*'s, and vice versa. If the new solution does not violate any hard constraints and is the best neighbor found (line 10), the variables are updated accordingly (lines 11-12). The neighborhood's size is $O(|D_v||N|^2)$. The procedure terminates by returning the best solution found.

Algorithm 4: Swap shift.

```
   Input    : Dv set of days with insufficient coverage, N set of nurses, cs current solution
   Output   : Solution
1  improved ← true;
2  while improved do
3      improved ← false;
4      foreach d ∈ Dv do
5          bestNeighbor ← null;
6          bestNeighborOFV ← OFV(cs);
7          foreach n1 ← 1 to |N|-1 do
8              foreach n2 ← n1+1 to |N| do
9                  cs′ ← swapAssignments(cs, d, n1, n2) ;        // returns null if infeasible
10                 if cs′ ≠ null and OFV(cs′) < bestNeighborOFV then
11                     bestNeighbor ← cs′;
12                     bestNeighborOFV ← OFV(cs′);
13                 end
14             end
15         end
16         if bestNeighbor ≠ null then
17             cs ← bestNeighbor;
18             improved ← true;
19         end
20     end
21 end
22 return cs;
```

5.5. *Assign missing shift neighborhood*

Algorithm 5 assigns the shifts and skills associated with absences which have occurred to idle nurses in a greedy manner such that the largest decrease in objective value is obtained. Line 4 iterates over the days and shifts for which the number of nurses is below the minimum. For each idle nurse n (lines 7-13), a neighboring solution is generated by assigning the missing shift s and skill k on day d (line 8). If the neighboring solution does not violate any hard constraint and is the best neighbor found (line 9), the variables are updated (lines 10-11). The neighborhood's size is $O(|D_v||S_d||K_d||F_d|)$. The procedure terminates by returning the best solution found.

Algorithm 5: Assign missing shift.

Input : D_v set of days with insufficient coverage ordered by most violated days, S_d set of shifts with insufficient coverage on day d, K_d set of skills with insufficient coverage on day d, F_d set of idle nurses on day d, *cs* current solution

Output : Solution

1 improved ← true;
2 **while** *improved* **do**
3   improved ← false;
4   **foreach** $d \in D_v, s \in S_d, k \in K_d$ **do**
5   bestNeighbor ← null;
6   bestNeighborOFV ← OFV(cs);
7   **foreach** $n \in F_d$ **do**
8   cs' ← assignShiftSkill(cs, n, d, s, k) ; `// returns null if infeasible`
9   **if** $cs' \neq null$ **and** $OFV(cs') < bestNeighborOFV$ **then**
10   bestNeighbor ← cs';
11   bestNeighborOFV ← OFV(cs');
12   **end**
13   **end**
14   **if** $bestNeighbor \neq null$ **then** `// if an improved neighbor is found`
15   cs ← bestNeighbor;
16   improved ← true;
17   **end**
18   **end**
19 **end**
20 **return** cs;

5.6. Change and assign missing shift neighborhood

Algorithm 6 first moves a working shift to an idle nurse and then assigns the missing shift and skill. Line 4 iterates over the days where the number of nurses is below the minimum. For each working nurse w and each idle nurse f (lines 7-18), the currently assigned shift and skill are saved (lines 8-9) and the missing shift and skill are assigned to nurse w (line 10). If the resulting solution does not violate any hard constraints, the algorithm assigns an idle nurse f the shift and skill previously assigned to nurse w (line 12). If the resulting solution does not violate any hard constraints and is the best neighbor found (line 14), the variables are updated (lines 15-16). The neighborhood's size is $O(|D_v||S_d||K_d||W_d||F_d|)$. The procedure terminates by returning the best solution found.

Algorithm 6: Change and assign missing shift.

```
Input  : Dv set of days with insufficient coverage ordered by most violated days,
         Sd set of shifts with insufficient coverage on day d, Kd set of skills with insufficient coverage on
day d, Fd set of idle nurses on day d, Wd set of working nurses on day d, cs current solution
Output : Solution
1  improved ← true;
2  while improved do
3      improved ← false;
4      foreach d ∈ Dv, s ∈ Sd, k ∈ Kd do
5          bestNeighbor ← null;
6          bestNeighborOFV ← OFV(cs);
7          foreach w ∈ Wd, f ∈ Fd do
8              backupShift ← getShift(cs, w, d) ;              // backup current shift
9              backupSkill ← getSkill(cs, w, d, s) ;           // backup current skill
10             cs′ ← assignShiftSkill(cs, w, d, s, k) ;        // returns null if infeasible
11             if cs′ ≠ null then
12                 cs″ ← assignShiftSkill(cs′, f, d, backupShift, backupSkill);
13             end
14             if cs″ ≠ null and OFV(cs″) < bestNeighborOFV then
15                 bestNeighbor ← cs″;
16                 bestNeighborOFV ← OFV(cs″);
17             end
18         end
19         if bestNeighbor ≠ null then                   // if an improved neighbor is found
20             cs ← bestNeighbor;
21             improved ← true;
22         end
23     end
24 end
25 return cs;
```

5.7. Assign missing shift and delete next shift neighborhood

Algorithm 7 attempts to fix disruptions by inserting shifts on the days with an insufficient number of nurses. However, when inserting shifts results in an infeasible solution, the assignment on the following day is deleted. The function $\mathscr{U}(LB,\ldots,UB)$ returns a value between *LB* and *UB* sampled from a uniform distribution. Line 1 iterates over the days with insufficient coverage. Lines 2 and 3 randomly select an idle nurse on day d and assign the missing shift and skill. If the resulting solution cs' is infeasible and d is not the last day of the scheduling horizon (line 4), the algorithm removes the assignment on the next day. If the resulting solution is feasible, it replaces the current solution (lines 7-9). The neighborhood's size is $O(|D_v||S_d||K_d|)$. The procedure terminates by returning the best solution found.

Algorithm 7: Assign missing shift delete next.

Input : D_v set of days with insufficient coverage ordered by most violated days,
D_a set of all days, S_d set of shifts with insufficient coverage on day d, K_d set of skills with insufficient coverage on day d, F_d set of idle nurses on day d

Output : Solution

```
1  foreach d ∈ D_v, s ∈ S_d, k ∈ K_d do
2      n ← U(1,...,|F_d|) ;                 // returns a random nurse not working on day d
3      cs' ← assignShiftSkill(cs, n, d, s, k) ;        // returns null if infeasible
4      if cs' is null and d+1 ≤ |D_a| then
5          cs' ← assignDayOff(cs', n, (d+1));
6      end
7      if cs' ≠ null then                        // if an improved neighbor is found
8          cs ← cs';
9      end
10 end
11 return cs;
```

5.8. Intensification and diversification neighborhood

Algorithm 8 details an intensification and diversification procedure for the NRRP. The objective here is to explore larger neighborhoods, thereby increasing the probability of finding a feasible solution when the other neighborhoods fail. Preliminary experiments on a subset of the instances revealed that the most suitable values for the input parameters *maxNoImprov*, *maxChanges*, *maxNoImprovDiv*, *maxChangesDiv* are 100, 3, 50 and 2, respectively. The algorithm is terminated after the maximum number of iterations without improvement is reached or if a feasible solution is found (line 3). The loop spanning lines 6-22 determines how many neighbors are generated in each iteration of the procedure's intensification phase. In this case, this value equals the number of nurses $|N|$. If the number of iterations without improvement is greater than a specific threshold, the diversification phase begins (line 30). In this phase, for each day, two random modifications are generated in the current solution (lines 32-38). After this diversification procedure, the local search operators *assignMissingShift, changeAssignMissingShift, assignDeleteShift, changeShift, swapShift* are called to further improve the solution (lines 39-40). The procedure terminates by returning the best solution found.

6. Computational results

This section analyzes a series of computational experiments to investigate whether the proposed IP formulation can be solved using a MIP solver for both small and large instances with multi-skilled nurses. The impact of relaxing soft constraints with respect to the original NRP in terms of solution quality and computational time is analyzed. Moreover, whether the degradation of solution quality is significant when rerostering a limited scheduling horizon and whether or not the proposed VND heuristic can generate competitive results compared to a MIP solver is also investigated.

6.1. Data sets and experimental setup

This section presents two sets of instances employed for the experiments. They cover both academic and realistic scenarios. This paper contributes a rerostering version of the INRC-II instances. The Lisbon instances were, until this work, the only public set of instances available in the literature related to the NRRP. They were developed by Moz & Pato (2007) and based on real data provided by a Lisbon hospital. The INRC-II instances incorporate a large number of soft constraints related to the NRP and are therefore less restrictive in terms of hard constraints compared to the Lisbon instances.

6.1.1. INRC-II instances

Table 8 describes the constraints in the INRC-II instances for which there are two main sets. The first concerns those related to the NRP that have less weight in the ob-

Algorithm 8: Intensification and diversification procedure.

Input : D_v set of days with absent nurses, N set of nurses, S set of shifts, *bestSolution* current solution, *maxNoImprov* maximum number of iterations without improvement, *maxChanges* maximum number of changes, *maxNoImprovDiv* maximum number of iterations to start the diversification phase, *maxChangesDiv* maximum number of changes in the diversification phase

Output : Solution

```
countNoImprov ← 0;
cs ← bestSolution;
while countNoImprov < maxNoImprov and hasHardViolation(cs) do
    bestNeighbor ← ∞;
    cs'' ← null;
    for x ← 1 to |N| do
        randNumChange ← 𝒰(1,...,maxChanges) ;      // returns a random number of changes
        for z ← 1 to randNumChanges do
            randNurse ← 𝒰(1,...,|N|) ;                // returns a random nurse
            randDay ← 𝒰(1,...,|D|) ;                  // returns a random day
            randShift ← 𝒰(1,...,|S|) ;                // returns a random shift
            cs' ← assignShift(cs, randNurse, randDay, randShift) ;  // returns null if infeasible
            if cs' ≠ null and isNotTabu(randNurse, randDay, randShift) then
                addTabu(randNurse, randDay, randShift) ;  // add nurse,day,shift to tabu list
                if OFV(cs') < bestNeighbor then
                    bestNeighbor ← OFV(cs');
                    cs'' ← cs';
                    break;
                end
            end
        end
    end
    if cs'' ≠ null and OFV(cs'') < OFV(cs) then
        countNoImprov ← 0;
        cs ← cs'';
        bestSolution ← cs;
    end
    else
        countNoImprov ← countNoImprov + 1;
        if countNoImprov > maxNoImprovDiv then
            cs ← bestSolution;
            foreach d ∈ D do
                foreach y ← 1 to maxChangesDiv do
                    randNurse ← 𝒰(1,...,|N|) ;        // returns a random nurse
                    randShift ← 𝒰(1,...,|S|) ;        // returns a random day
                    cs ← assignShift(cs, randNurse, d, randShift)
                end
            end
            cs ← assignMissingShift(cs);    cs ← changeAssignMissingShift(cs);
            cs ← assignDeleteShift(cs);    cs ← changeShift(cs);    cs ← swapShift(cs);
        end
    end
end
return bestSolution;
```

jective function when rerostering and, consequently, less importance associated with avoiding their violations. The second set of constraints concerns the specific objectives related to the NRRP. The number of changes, which is considered the most important objective to minimize by the objective function is assigned a weight of 100. Meanwhile, the change in number of assigned shifts is regarded as less important and receives a weight of 50. An in-depth discussion of each constraint is provided by Ceschia et al. (2014).

Table 8: Hard and soft constraints in the INRC-II instances.

Index	Constraint description	Weight	Eq.
Nurse Rostering Constraints			
-	A nurse can be assigned to at most one shift per day	HC	A.2
-	Minimum number of nurses per day/shift/skill	HC	A.3
-	A shift type succession must belong to a valid succession (for example, a Night shift cannot be followed by an Early shift)	HC	A.4
-	A shift requiring nurses with a given skill must necessarily be fulfilled by a nurse having that skill	HC	A.5
1	Preferred coverage	30	A.7
2	Minimum consecutive assignments (working days)	30	A.8, A.9
3	Maximum consecutive assignments (working days)	30	A.15
4	Minimum number of consecutive days off	30	A.11, A.12
5	Maximum number of consecutive days off	30	A.13
6	Minimum consecutive assignments to the same shift	15	A.14, A.15
7	Maximum consecutive assignments to the same shift	15	A.16
8	Individual nurse's undesired working day/shift	10	A.17
9	Complete weekend	30	A.18, A.19
10	Minimum number of assignments over the scheduling period	20	A.20
11	Maximum number of assignments over the scheduling period	20	A.21
12	Total working weekends	30	A.18, A.22
Nurse Rerostering Constraints			
-	Absent nurses cannot be assigned to any shift	HC	2
13	Each change in the new roster is penalized	100	3, 4, 5
14, 15	The original number of assigned shifts should be maintained	50	6, 7

Ingels & Maenhout (2015) simulate employee availability using a Bernoulli distribution. They performed simulations for short-term sick leave with a probability of 2.44% based on a study conducted by SD Worx (2013) in Belgium . Moreover, they reported that simulations using an absenteeism probability of 5% and 10% resulted in similar results, regarding sick leave probabilities.

In this study, absences were randomly generated based on statistics observed by Aguirre & Kerin (2014) in the U.S. They report absenteeism rates ranging from 5% to 10% among all employees, meaning that at any given time 5% to 10% of the workforce is missing from work. While this rate varies by sector, the emergency services and healthcare, both known for their stressful working conditions, high rates of overtime, and are therefore unsurprisingly associated with the highest rates of absenteeism.

The first group of NRRP instances is named *Single-day Nurse Absence*, where absences are generated for randomly selected nurses and days based on an absenteeism rate of 5%. The instances have 35, 70 or 110 nurses and a scheduling horizon of either four or eight weeks.

The second group of instances is named *Consecutive-days Nurse Absence* and represents situations which simulate nurse illness. In these instances, a randomly selected nurse is absent for a sequence of days beginning from a first random day i until a later random day j which are chosen based on a uniform distribution. Instances with 35, 70 or 110 nurses and scheduling horizons of four or eight weeks have 5% of their associated nurses absent for a random number of consecutive days.

6.1.2. Lisbon instances

The Lisbon instances differ from the INRC-II in terms of constraints, shifts and nurses' skills. Rather than four shifts as per the INRC-II instances, the Lisbon instances have three shifts, namely: Early (08:00-16:00), Late (16:00-24:00) and Night (00:00-08:00). The nurses are single-skilled, while in the INRC-II instances they are multi-skilled. All constraints are hard and the objective simply concerns minimizing the number of changes compared to the original roster. The general model proposed in Section 4 and Appendix A details this subset of constraints. Table 9 presents the set of constraints and the related equations.

Inconsistencies regarding these Lisbon instances required adaptations as the current roster, provided by the hospital's head nurse, violated several hard constraints. The following changes were made to render the instances feasible:

Table 9: Hard and soft constraints considered for the Lisbon instances.

Index	Constraint Description	Weight	Eq.
Nurse Rostering Constraints			
-	A nurse can be assigned to at most one shift per day	HC	A.2
-	Minimum number of nurses per day/shift/skill	HC	A.3
-	A shift type succession must belong to a valid succession	HC	A.4
-	Every seven days sequence, nurses must have 1 day off when the contract is 42 hours per week, and 2 days off when the contract is 35 hours per week	HC	A.6
Nurse Rerostering Constraints			
-	Absent nurses cannot be assigned any shift	HC	2
13	Each change in the new roster is penalized	1	3, 4, 5

- Nurses with 30-hour contracts are now considered as having 35-hour contracts. In doing so, these nurses must have two days off per working week;

- The current solution for 32 nurses is infeasible due to some violations where the nurses' contracts permit them to work a maximum of 5 days every 7 days. However, some nurses work 6 days in the provided rosters. In this case, the nurses' contracts were adjusted to 42 hours, thereby permitting them to work 6 days every 7 days;

- The pattern file, provided in PDF format, was ignored during the conversion process since the patterns were not always respected in the roster provided by the hospital's head nurse.

6.1.3. Computational environment

All models and algorithms were implemented in Java and compiled with OpenJDK 1.8. The experiments were conducted on an AMD FX(tm)-8150 eight-Core Processor with 32GB of RAM memory running Linux Ubuntu 16.04.3 64-bit. The commercial MIP solver employed was CPLEX version 12.7.1. with default parameters and configured to use eight threads. For the experiments with an open-source solver, Coin-OR CBC 2.9.9 was employed with eight threads. Relative gaps in solution quality were calculated as $gap = 100 \times \frac{UB-OPT}{OPT}$, where UB (upper bound) corresponds to the objective function value of the VND heuristic, and OPT corresponds to the optimum solution value obtained by CPLEX. For each experiment, the VND heuristic was executed ten times with different seed values for the random number generator.

6.1.4. VND neighborhoods and parameters tunning

The primary objective of the experiments in this section is to analyze to which degree that each VND neighborhood impacts upon the heuristic's performance. Table 10 compares the results obtained by the MIP solver and different configurations of the VND heuristic. The complete scheduling horizon and all NRP and NRRP constraints were considered for these experiments. The following five VND heuristic configurations were investigated:

- *VND 1:* employs only the *assignMissingShift* neighborhood;
- *VND 2:* employs the *assignMissingShift* and *assignDeleteShift* neighborhoods;
- *VND 3:* employs the *assignMissingShift*, *assignDeleteShift* and *changeShift* neighborhoods;
- *VND 4:* employs all neighborhoods, except *intensDiverLS*;
- *VND:* employs all neighborhoods.

The second through fourth columns in Table 10 provide the objective values obtained by the MIP solver, the respective time to the optimum solution, and the time to prove optimality. The fifth through fourteenth columns show, for each configuration of the VND heuristic, the gap to the optimum objective value and the required computation time.

The average relative optimality gaps for *VND1*, *VND2*, *VND3*, *VND4* and *VND* are 1.10%, 1.07%, 0.62%, 0.61% and 0.61%, respectively. Comparing computation times, the MIP solver requires, on average, significantly more time than the VND heuristic to reach optimality. The VND heuristic generated near-optimum solutions within only a few seconds. It is worth noting that due to the numerous dynamic situations which may occur in the real world, fast management decisions and therefore short computational times are critical to ensure the best decision is made as quickly as possible.

Table 10: VND neighborhoods evaluation - Complete scheduling horizon, NRP and NRRP constraints

	Single-day absences												
	MIP solver			VND 1		VND 2		VND 3		VND 4		VND	
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)
n035w4	3206.0	3.7	5.7	0.8	0.7	0.8	0.9	0.7	1.0	0.6	1.0	0.6	1.0
n035w8	6490.5	8.5	14.0	1.1	0.9	1.1	1.3	1.0	1.8	1.0	1.8	1.0	1.8
n070w4	5850.0	5.8	16.6	0.7	1.0	0.7	1.4	0.0	2.4	0.0	2.8	0.0	2.8
n070w8	12379.0	51.5	186.3	1.5	1.2	1.4	2.2	0.4	4.4	0.4	6.6	0.4	6.6
n110w4	7495.0	12.1	42.1	0.9	1.1	0.8	2.1	0.7	3.0	0.7	3.9	0.7	3.9
n110w8	14366.0	78.7	237.5	1.5	1.5	1.5	2.9	1.0	5.8	1.0	10.2	1.0	10.2
average				1.10		1.07		0.62		0.61		0.61	
	Consecutive-days absences												
	MIP solver			VND 1		VND 2		VND 3		VND 4		VND	
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)
n035w4	3167.0	5.3	9.6	2.4	0.8	1.7	1.0	1.4	1.1	1.3	1.2	1.3	1.2
n035w8	6383.5	7.5	28.6	2.7	0.9	2.2	1.3	2.1	1.8	1.9	1.9	1.9	1.9
n070w4	5564.5	13.8	41.9	0.8	1.0	0.7	1.5	0.4	2.4	0.3	2.5	0.3	2.5
n070w8	11595.0	109.2	208.0	1.3	1.2	1.3	2.0	1.0	4.3	0.9	6.2	0.9	6.2
n110w4	7039.0	28.3	84.5	1.2	1.2	1.2	2.1	0.5	4.1	0.5	5.0	0.5	5.0
n110w8	13670.5	229.6	598.9	2.5	1.6	2.5	2.9	1.8	6.7	1.9	9.7	1.9	9.7
average				1.82		1.60		1.20		1.11		1.11	

The second part of Table 10 reports the results for the instances with consecutive-days absences. The average relative optimality gaps are 1.82%, 1.60%, 1.20%, 1.11% and 1.11%, respectively for the five VND heuristic variants. Again, the computation times were much lower compared to those required by the MIP solver for reaching its near-optimum solutions. For example, on the largest instances, the MIP solver spent 229.6 seconds to reach the optimum value, while the VND heuristic required only 9.7 seconds to reach a solution within 1.11% of the optimum solution. Experiments with different versions of the VND heuristic demonstrate that all the implemented components are important in contributing to obtaining near-optimum solutions or to solve infeasibilities. Moreover, when all neighborhoods are used, the algorithm generated the best results. In all remaining experiments, the VND heuristic is the one employed with all neighborhoods.

The VND heuristic has two main parameters. The first parameter determines after how many iterations the intensification and diversification procedure is called and was set to 30 in order to limit the algorithm's runtime while still providing sufficient possibilities to solve infeasibilities in the Lisbon instances. For the INRC-II instances, the intensification and diversification procedure was not required to solve infeasibilities. However, for the Lisbon instances this neighborhood was essential and solved infeasibilities in 10 out of 64 instances. The second main parameter is the maximum number of top-level loop iterations which was set to 100 to avoid infinite loops in the algorithm whenever an instance did not have a feasible solution. Table B.21 details the number of top-level loop iterations for different instances and strategies. Considering the complete scheduling horizon and all constraints, the average number of trials was less than two for the INRC-II instances, 12.9 for the Lisbon instances with 19 nurses and 4.8 for the Lisbon instances with 32 nurses.

Figure 1 presents the evolution of the objective function value throughout the VND heuristic's execution on an INRC-II instance with 110 nurses and a scheduling horizon of eight weeks. The algorithm begins with an initial objective value of 161755 and after 4.98 seconds ends with an objective function value of 12425. After some initial small improvements, the algorithm quickly finds several significant improvements. The algorithm then ends like it began: with a series of minimal improvements. This experiment demonstrates how the algorithm fulfills its two primary objectives: to find high-quality solutions within short computational runtimes.

6.2. Computational results for the INRC-II instances

This section presents the experiments employing the INRC-II instances. All tables present average results for each group of instances. The first column details the Instance Id, where *n035, n070, n110* represent the number of nurses and where *w4* and *w8* correspond to the number of weeks. Each group contains 10 instances for a total of 60. The column *std. dev.* provides the standard deviation on the average value which is reported in the previous column. Detailed computational results are publicly available online[1].

[1] `http://www.inf.ufrgs.br/~tiwickert/download/2017/reroster`

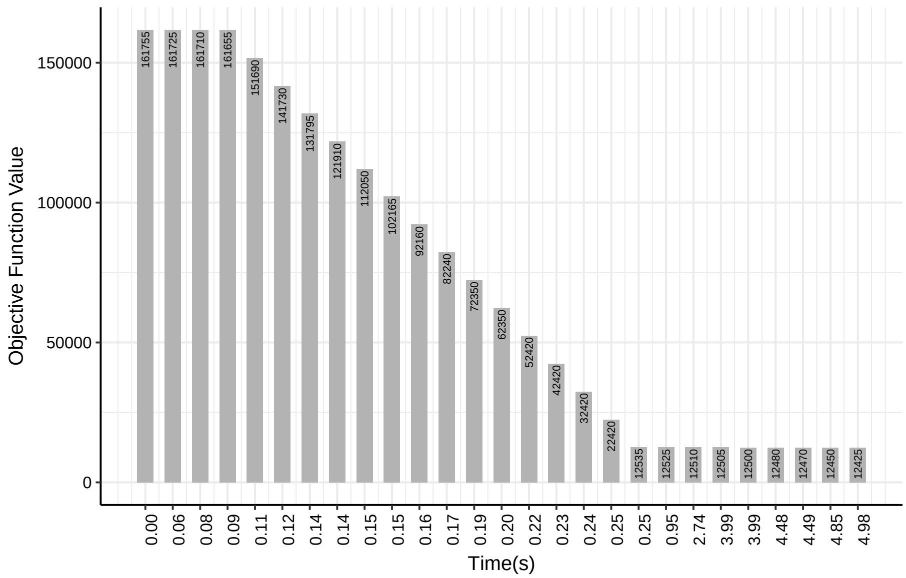

Figure 1: OFV evaluation throughout VND execution.

6.2.1. Complete scheduling horizon, complete set of NRP and NRRP constraints

In these experiments, the complete scheduling horizon and all the NRP and NRRP constraints are considered when rerostering. Table 11 presents the average results for both single-day absences and the consecutive-days absences instances. The first block (second and third columns) provides data concerning the initial infeasible solution, the second provides the NRP objective value, while the third provides the NRP+NRRP objective value. Note that to estimate the initial objective value, each unit below the minimum coverage violation was penalized with a weight of 10000.

The second block (fourth to ninth columns) shows the results obtained by the MIP solver. The fourth and fifth columns detail the NRP and the NRP+NRRP objective values, respectively. The sixth and eighth columns provide the time to reach the optimum solution and the time to prove it, while the seventh and ninth columns provide the respective standard deviations. The last block (tenth to twelfth columns) details the results regarding the VND heuristic. The tenth column provides the relative gap to the optimum value, while the eleventh and twelfth columns provide the time in seconds to reach the value and its respective standard deviation.

Table 11: Complete scheduling horizon NRP+NRRP constraints.

	Single-day absences										
	Initial infeasible solution		MIP						VND		
Instance Id	NRP OFV	NRP+NRRP OFV	NRP OFV	NRP+NRRP OFV	Opt Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.	Gap(%)	Time(s)	Std. Dev.
n035w4	2370.0	39770.0	2546.0	3206.0	3.7	1.1	5.7	1.1	0.6	1.0	0.2
n035w8	4912.5	84712.5	5165.5	6490.5	8.5	5.5	14.0	4.5	1.0	1.8	0.5
n070w4	4704.5	42504.5	4665.0	5850.0	5.8	3.7	16.6	7.5	0.0	2.8	0.6
n070w8	10308.0	72908.0	9834.0	12379.0	51.5	37.7	186.3	104.2	0.4	6.6	1.1
n110w4	6183.5	34383.5	6065.0	7495.0	12.1	7.2	42.1	42.9	0.7	3.9	1.0
n110w8	11467.5	113867.5	11211.0	14366.0	78.7	42.8	237.5	130.9	0.9	10.2	1.9
average									0.6		
	Consecutive-days absences										
	Initial infeasible solution		MIP						VND		
Instance Id	NRP OFV	NRP+NRRP OFV	NRP OFV	NRP+NRRP OFV	Opt Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.	Gap(%)	Time(s)	Std. Dev.
n035w4	2175.5	43745.5	2292.0	3167.0	5.3	2.0	9.6	8.0	1.4	1.2	0.3
n035w8	4075.0	117090.0	4548.5	6383.5	7.5	7.8	28.6	48.3	1.9	1.9	0.5
n070w4	4042.5	50032.5	4029.5	5564.5	13.8	23.0	41.9	34.4	0.4	2.5	0.6
n070w8	8856.0	85831.0	8880.0	11595.0	109.2	81.0	208.0	86.1	0.9	6.3	0.9
n110w4	5397.5	52967.5	5059.0	7039.0	28.3	24.1	84.5	57.3	0.5	4.9	0.6
n110w8	9927.0	175977.0	9740.5	13670.5	229.6	267.5	598.9	495.7	1.8	9.7	2.0
average									1.1		

An interesting finding concerns instance *n035w4_2_9-9-2-1* that has one violation of nurses below the minimum coverage. This infeasibility was solved without any changes concerning working days, days off or shift changes. This was only possible since the number of scheduled nurses is higher than the required minimum and the nurses are multi-skilled. The formulation is designed in such a way that the solver was capable of finding a solution by only changing skills of nurses who were already assigned to shifts. Other noteworthy observations concern instances *n070w4_0_3-6-5-1* and *n110w4_2_5-1-3-0*, which were feasible even with the randomly generated disruptions. This occurs because the absent nurses were generated on days where the number of nurses exceeded the minimum coverage, referred to as the preferred number of nurses. The solving times for instances with single-day absences were considerably quicker than for instances with consecutive-days absences, with the time required to reach the optimum solution for instances n110w8 being 78.7 for single-day absences compared to 229.6 seconds for when consecutive-day absences were generated. Using the VND heuristic, the time required to reach relative gaps of 0.6% and 1.1% were considerably shorter compared to the MIP solver, as can be observed in the eleventh column. Despite the MIP solver generating optimum results in tractable time limits, the VND heuristic still provides a good alternative whenever a MIP solver is not affordable or if an urgent change regarding the current roster must be performed online.

These results demonstrate that *robust* rosters can be created by scheduling more nurses than required, provided that these nurses are multi-skilled. Moreover, common disruptions occurring in real-world scenarios, such as employee absenteeism on a sequence of days, are more difficult to solve than single-day disruptions.

6.2.2. Complete scheduling horizon and ignoring the NRP's soft constraints

These experiments consider the complete scheduling horizon, NRRP hard/soft constraints, and NRP hard constraints. In contrast to Section 6.2.1, these experiments ignore the NRP's soft constraints when rerostering. The first block of Table 12 presents the Instance Id, while the second block provides the results obtained considering all NRP and NRRP constraints. The last block presents the results ignoring the NRP's soft constraints.

Table 12's eighth column shows how on average, the NRRP objective value was lower (highlighted in bold) when the NRP soft constraints were dropped. This occurred for 51 of the 60 instances with single-day absences and in 54 instances with consecutive-days absences. The primary advantage associated with ignoring the NRP soft constraints is the required computation time. Considering the single-day absence instances, Table 12 details the average computational time required to reach the optimum value when considering all constraints for the larger instances (n110w8), which was 78.8 seconds, while without the NRP soft constraints the average computation time decreases significantly to just 6.3 seconds. A similar observation occurs for the consecutive-days absences instances where the average times were 229.6 and 7.5 seconds with and without the NRP soft constraints for the *n110w8* instances, respectively.

Table 13 presents the same experiment employing the VND heuristic. The results detail a decrease regarding NRRP constraint violations (eighth column) when the NRP constraints are ignored compared to when all constraints are considered (third column). Moreover, when the results of the VND heuristic's relaxed constraints are compared against the MIP results, a smaller increase of the NRRP+NRP objective value (ninth column) is observed when compared against the ninth column of Table 12. These results are due to the VND heuristic being unable to reach optimum results in all the instances when the NRP constraints are ignored, benefiting in these cases, the NRP constraints.

It may, therefore, be concluded that the original NRP soft constraints (presented in Table 8) are important to consider for generating good quality solutions. However, in some situations it may be useful to ignore them, such as when, for example, an urgent surgery is scheduled and there are also nurse shortages. The surgery should be prioritized over nurse preferences or consecutive working and resting day restrictions/entitlements.

Table 12: MIP - Complete scheduling horizon, NRP soft constraints relaxation.

	Single-day absences											
	All constraints					NRP constraints relaxation						
Instance Id	NRP OFV	NRRP OFV	NRP+NRRP OFV	Opt Time(s)	Opt Prove Time(s)	NRP OFV	NRRP OFV	NRP+NRRP OFV	Opt Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.
n035w4	2546.0	660.0	3206.0	3.7	5.7	4332.0	**620.0**	4952.0	0.8	0.2	0.9	0.2
n035w8	5165.5	1325.0	6490.5	8.5	14.0	7583.5	**1165.0**	8748.5	2.7	0.7	3.7	2.3
n070w4	4665.0	1185.0	5850.0	5.8	16.6	8766.0	**970.0**	9736.0	1.7	1.0	2.0	0.9
n070w8	9829.0	2550.0	12379.0	51.5	186.3	16902.0	**1865.0**	18767.0	5.4	1.8	6.5	2.3
n110w4	6065.0	1430.0	7495.0	12.1	42.1	9643.0	**1240.0**	10883.0	1.5	0.8	2.7	2.6
n110w8	11211.0	3155.0	14366.0	78.7	237.5	16538.5	**2665.0**	19203.5	6.3	6.2	39.6	23.2
	Consecutive-days absences											
	All constraints					NRP constraints relaxation						
Instance Id	NRP OFV	NRRP OFV	NRP+NRRP OFV	Opt Time(s)	Opt Prove Time(s)	NRP OFV	NRRP OFV	NRP+NRRP OFV	Opt Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.
n035w4	2292.0	875.0	3167.0	5.3	9.6	4077.0	**795.0**	4872.0	0.8	0.2	0.8	0.2
n035w8	4553.5	1830.0	6383.5	7.5	28.6	7122.0	**1720.0**	8842.0	2.5	0.3	2.6	0.4
n070w4	4039.5	1525.0	5564.5	13.8	41.9	7926.5	**1340.0**	9266.5	2.0	0.8	2.2	0.8
n070w8	8880.0	2715.0	11595.0	109.2	208.0	14879.5	**2425.0**	17304.5	5.0	2.0	6.3	2.1
n110w4	5059.0	1980.0	7039.0	28.3	84.5	8848.5	**1690.0**	10538.5	2.4	1.7	2.8	1.7
n110w8	9735.5	3935.0	13670.5	229.6	598.9	15089.5	**3470.0**	18559.5	7.5	4.5	16.4	14.0

6.2.3. Scheduling horizon relaxation

These experiments evaluate the impact of rerostering when considering different scheduling horizons. The complete scheduling horizon, the most straightforward approach, is analyzed in addition to restricted horizons which only reroster those days where nurses are absent, from the first absent day until the last absent day, or from the

Table 13: VND - Complete scheduling horizon, NRP soft constraints relaxation.

	Single-day absences									
	All constraints					NRP constraints relaxation				
Instance Id	NRP OFV	NRRP OFV	NRP+NRRP OFV	Time(s)	Std. Dev.	NRP OFV	NRRP OFV	NRP+NRRP OFV	Time(s)	Std. Dev.
n035w4	2565.0	660.0	3225.0	1.0	0.2	2633.0	**625.0**	3258.0	0.6	0.0
n035w8	5208.0	1350.0	6558.0	1.8	0.5	5683.0	**1185.0**	6868.0	0.9	0.2
n070w4	4670.5	1180.0	5850.5	2.8	0.6	5099.5	**970.0**	6069.5	1.3	0.1
n070w8	9962.5	2467.9	12430.4	6.6	1.1	10983.0	**1875.0**	12858.0	1.9	0.1
n110w4	6124.0	1420.0	7544.0	3.9	1.0	6369.0	**1265.0**	7634.0	1.8	0.1
n110w8	11303.0	3194.7	14497.7	10.2	1.9	12172.0	**2725.0**	14897.0	2.7	0.2
	Consecutive-days absences									
	All constraints					NRP constraints relaxation				
Instance Id	NRP OFV	NRRP OFV	NRP+NRRP OFV	Time(s)	Std. Dev.	NRP OFV	NRRP OFV	NRP+NRRP OFV	Time(s)	Std. Dev.
n035w4	2330.5	880.0	3210.5	1.2	0.3	2492.0	**805.0**	3297.0	0.6	0.0
n035w8	4624.6	1878.1	6502.7	1.9	0.5	5048.0	**1800.0**	6848.0	0.9	0.2
n070w4	4130.6	1454.9	5585.5	2.5	0.6	4402.0	**1340.0**	5742.0	1.3	0.1
n070w8	9009.3	2688.8	11698.1	6.3	0.9	9488.0	**2465.0**	11953.0	2.0	0.1
n110w4	5103.1	1969.9	7073.0	4.9	0.6	5559.0	**1695.0**	7254.0	1.9	0.2
n110w8	9912.8	4002.9	13915.7	9.7	2.0	10939.5	**3565.0**	14504.5	2.7	0.2

first absent day until the end of the scheduling horizon. All NRP and NRRP constraints were considered throughout these experiments.

Tables 14 and 15's fifth columns document the results when only rerostering on absent days employing the MIP solver and the VND heuristic, respectively. Whereas this restricted rerostering considers a problem which is more constrained, the computational results indicate only a slight deterioration concerning solution quality compared against the complete scheduling horizon. Values in bold indicate improvements obtained by restricting the rerostering horizon. This rerostering strategy, therefore, provides a good alternative when obtaining a solution is urgent and must be acquired within a very short period of time.

Tables 14 and 15 also present the results when the scheduling horizon is limited from the first absence to the last absence (third block), and until the end of the scheduling horizon (fourth block). The results are very similar to when the complete scheduling horizon is considered. The gaps reported in Table 15 demonstrate consistent performance of the VND heuristic under different strategies regarding scheduling horizon relaxations. For the consecutive-day absences the relative gaps are 1.1%, 1.0%, 1.7% and 1.1% for the complete scheduling horizon, only absent days, first absence to last absence, and first absence to the end of the scheduling horizon, respectively.

It can therefore be concluded that the VND heuristic generates near-optimum results (with gaps less than 2%), providing a good alternative to the MIP solver. When the new schedule has already been communicated to all employees and the new month has not yet begun, then rerostering the complete scheduling horizon provides the best alternative. Nevertheless, it is worthwhile to consider other strategies which restrict the scheduling horizon, given that the NRRP depends on when an employee communicates their absence. For example, if the new month has already begun and some employees communicate unavailabilities, the revised roster should not reconsider assignments from the past and consequently the beginning of the new scheduling horizon should instead be the first absent day.

Table 14: MIP - Comparison of scheduling horizons.

	Single-day absences											
	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)
n035w4	3206.0	3.7	5.7	**3217.0**	0.5	0.6	3206.0	2.3	3.7	3206.0	3.0	4.7
n035w8	6490.5	8.5	14.0	**6504.0**	1.1	1.1	6490.5	6.7	10.9	6490.5	7.3	15.4
n070w4	5850.0	5.8	16.6	**5867.0**	2.4	2.5	**5857.0**	7.4	10.2	**5852.5**	5.6	16.3
n070w8	12379.0	51.5	186.3	**12452.0**	5.0	6.8	**12382.5**	41.0	160.6	12379.0	36.9	148.5
n110w4	7495.0	12.1	42.1	**7501.5**	3.9	5.3	**7496.5**	6.2	20.2	7495.0	9.0	25.1
n110w8	14366.0	78.7	237.5	**14433.5**	5.7	10.5	**14375.0**	67.0	199.3	14366.0	61.8	164.3
	Consecutive-days absences											
	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)
n035w4	3167.0	5.3	9.6	**3177.5**	1.1	1.2	3167.0	4.4	5.5	3167.0	4.1	5.4
n035w8	6383.5	7.5	28.6	**6392.0**	2.2	2.6	6383.5	7.1	29.8	6383.5	6.9	40.9
n070w4	5564.5	13.8	41.9	**5581.0**	3.8	7.8	5564.5	7.0	27.1	5564.5	7.4	31.0
n070w8	11595.0	109.2	208.0	**11623.5**	6.0	11.5	11595.0	85.3	180.0	11595.0	73.3	190.3
n110w4	7039.0	28.3	84.5	**7055.5**	7.1	21.8	7039.0	27.7	65.9	7039.0	29.7	60.8
n110w8	13670.5	229.6	598.9	**13692.5**	54.0	84.4	**13674.0**	188.5	597.2	**13674.0**	178.9	466.9

6.3. Computational results for the Lisbon instances

Since there are no soft constraints in the Lisbon instances, only those strategies concerning the scheduling horizon are analyzed for this dataset. Tables 16 and 17 present the results for the Lisbon instances using the MIP solver and VND heuristic, respectively. In both tables, rerostering the complete scheduling horizon and only a

Table 15: VND - Comparison of scheduling horizons.

	Single-day absences											
	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)
n035w4	3225.0	0.6	1.0	**3233.5**	0.5	0.4	**3332.9**	3.8	1.0	3225.0	0.6	0.9
n035w8	6558.0	1.0	1.8	**6570.0**	1.0	0.6	**6614.0**	1.9	1.7	6558.0	1.0	1.7
n070w4	5850.5	0.0	2.8	**5867.5**	0.0	1.5	**5893.1**	0.6	2.3	**5853.0**	0.0	2.6
n070w8	12430.4	0.4	6.6	**12494.9**	0.3	3.5	**12456.4**	0.6	6.0	12430.4	0.4	6.3
n110w4	7544.0	0.6	3.9	**7550.5**	0.6	2.7	**7565.0**	0.9	3.2	7544.0	0.6	3.6
n110w8	14497.7	0.9	10.2	**14566.0**	0.9	6.2	**14554.2**	1.2	9.1	14497.7	0.9	9.8
		0.6			0.6			1.5			0.6	

	Consecutive-days absences											
	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)
n035w4	3210.5	1.4	1.2	**3219.0**	1.3	0.6	**3223.5**	1.8	1.0	3210.5	1.4	1.1
n035w8	6502.7	1.8	1.9	**6506.4**	1.8	0.9	**6560.6**	2.7	2.0	6502.7	1.8	1.7
n070w4	5585.5	0.4	2.5	**5596.4**	0.3	1.7	**5610.4**	0.8	2.2	5585.5	0.4	2.3
n070w8	11698.1	0.9	6.2	**11716.6**	0.8	3.5	**11711.8**	1.0	5.7	11698.1	0.9	5.9
n110w4	7073.0	0.5	5.0	**7083.5**	0.4	3.9	**7171.6**	1.8	4.7	7073.0	0.5	4.6
n110w8	13915.7	1.8	9.7	**13931.2**	1.7	7.6	**13965.3**	2.1	9.0	**13919.2**	1.8	9.3
		1.1			1.0			1.7			1.1	

limited part is evaluated. In Table 16 the columns labeled *OFV* report the optimum objective values obtained by the MIP solver for each scheduling horizon, while columns *opt time(s)* and *opt prove time(s)* are the times (in seconds) to reach the optimum value and to prove optimality, respectively. In Table 17 the *gap* is relative to the optimum value obtained by the MIP solver for each scheduling horizon.

Table 16 details the results when employing the MIP solver. All instances were quickly solved to optimality. In the worst case, the MIP solver proved the optimum solution within 2.4 seconds. Both rerostering the complete scheduling horizon and rerostering from first absence until the end of the scheduling horizon generated the best results, while rerostering only the absent days resulted in infeasibility for 8 of the 68 instances. Finally, rerostering from the first absence until the last absent day resulted in 7 instances being infeasible. Note that instance *II7_19* is infeasible for all the scheduling horizons.

Table 17 details the results obtained by the VND heuristic. The best results for the instances with 19 nurses were obtained by restricting the scheduling horizon to only the absent days while the worst solutions were obtained when considering the full scheduling horizon. This indicates that the algorithm's overall performance improves when restricting the available possibilities for rerostering. Increasing the allowed computation time of the VND heuristic to ten minutes did not considerably improve the average relative gaps. For instances with 32 nurses, the chosen strategy does not affect the average gaps significantly. Compared to the MIP solver, solutions within 1% of the optimum solutions are obtained in comparable computation time. Independent of which strategy was applied, the VND heuristic performed significantly better when more nurses are available for rerostering as this allowed for more possibilities to repair the infeasibilities.

6.4. Limits of the solution approaches

The previous results demonstrated that CPLEX was able to find optimum solutions for all feasible instances while requiring very little computation time. This section further challenges the proposed integer programming model by investigating the performance of an alternative MIP solver and analyzing the performance of the solution approaches on large-scale problem instances.

Table 18 compares the performance of CPLEX against that of Coin-OR CBC, one of the leading open-source MIP solver projects (Lougee-Heimer, 2003). The third and seventh columns provide the number of feasible solutions found for each group of ten INRC-II instances. The fourth and fifth columns detail the time required by CPLEX to reach the reported objective value and the time required to prove optimality, while the eighth and tenth columns show these times for Coin-OR CBC. The reported standard deviations are always relative to the times in the preceding column. A dash (-) indicates that no feasible solution was found within the imposed time limit of one hour.

In general, the computation times of Coin-OR CBC were much longer than those of CPLEX. Consequently, Coin-OR CBC could prove optimality only for the smallest instances with 35 nurses and a scheduling horizon of four weeks and was unable to find feasible solutions for the larger instances containing 70 and 110 nurses and a scheduling horizon of eight weeks. However, on instances for which feasible solutions were obtained, Coin-OR CBC performed only slightly worse than CPLEX, indicating that the open-source solver is a suitable alternative when the number of nurses is limited and when short computation times are not crucial.

Table 16: MIP - Experiments employing the Lisbon instances

	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)
I1_19	3	0.3	0.3	3	0.0	0.1	3	0.0	0.1	3	0.0	0.1
I2_19	2	0.2	0.3	2	0.0	0.1	2	0.1	0.1	2	0.0	0.1
I3_19	9	0.5	0.8	9	0.1	0.1	9	0.1	0.1	9	0.1	0.1
I4_19	2	0.2	0.3	2	0.0	0.1	2	0.0	0.1	2	0.1	0.2
I5_19	15	0.3	0.6	17	0.1	0.1	17	0.2	0.2	17	0.1	0.1
I6_19	8	0.3	0.5	8	0.2	0.2	8	0.2	0.2	8	0.2	0.2
I7_19	19	0.3	0.7	20	0.3	0.4	20	0.3	0.3	20	0.3	0.3
I8_19	2	0.2	0.2	2	0.0	0.0	2	0.0	0.0	2	0.1	0.2
II1_19	1	0.2	0.2	1	0.0	0.0	1	0.0	0.0	1	0.1	0.1
II2_19	0	0.0	0.1	0	0.0	0.0	0	0.0	0.0	0	0.0	0.1
II3_19	5	0.2	0.3	5	0.1	0.1	5	0.0	0.2	5	0.1	0.2
II4_19	10	0.3	0.7	∞	0.1	0.2	12	0.1	0.3	12	0.1	0.3
II5_19	6	0.2	0.4	6	0.1	0.1	6	0.1	0.1	6	0.1	0.2
II6_19	16	0.3	0.5	16	0.1	0.3	16	0.1	0.3	16	0.1	0.3
II7_19	∞	-	-	∞	-	-	∞	-	-	∞	-	-
II8_19	3	0.2	0.3	6	0.1	0.2	6	0.1	0.2	5	0.1	0.3
III1_19	7	0.3	0.3	7	0.0	0.1	7	0.0	0.1	7	0.2	0.2
III2_19	9	0.3	0.6	∞	0.0	0.1	∞	0.0	0.1	12	0.5	0.5
III3_19	10	0.2	0.5	∞	0.0	0.1	∞	0.0	0.1	13	0.5	0.5
III4_19	7	0.2	0.4	7	0.1	0.2	7	0.1	0.2	7	0.2	0.3
III5_19	27	1.1	1.1	27	0.6	0.6	27	0.5	0.5	27	0.7	0.8
III6_19	26	0.8	0.8	28	0.3	0.5	26	0.6	0.6	26	0.6	0.6
III7_19	18	0.7	0.9	23	0.3	0.4	19	0.3	0.8	19	0.3	0.7
III8_19	10	0.3	0.7	10	0.1	0.3	10	0.1	0.3	10	0.2	0.3
IV1_19	8	0.2	0.7	∞	0.0	0.1	∞	0.0	0.1	9	0.3	0.4
IV2_19	11	0.3	0.8	∞	0.0	0.1	∞	0.0	0.2	12	0.4	0.4
IV3_19	10	0.2	0.6	∞	0.0	0.1	∞	0.0	0.1	10	0.3	0.3
IV4_19	26	0.3	0.5	∞	0.1	0.1	∞	0.1	0.2	26	0.2	0.5
IV5_19	17	0.3	0.9	19	0.2	0.6	19	0.7	1.0	17	0.3	0.6
IV6_19	21	0.9	1.1	25	0.1	0.3	23	0.3	0.4	23	0.6	0.6
IV7_19	9	0.3	0.6	9	0.1	0.2	9	0.3	0.6	9	0.3	0.6
IV8_19	9	0.2	0.5	9	0.1	0.3	9	0.2	0.4	9	0.2	0.5
I1_32	3	0.3	0.5	3	0.1	0.3	3	0.1	0.4	3	0.1	0.3
I2_32	3	0.3	0.4	3	0.1	0.2	3	0.1	0.2	3	0.1	0.1
I3_32	6	0.3	0.5	6	0.1	0.4	6	0.1	0.4	6	0.1	0.4
I4_32	1	0.3	0.3	1	0.1	0.1	1	0.1	0.1	1	0.1	0.1
I5_32	8	0.4	0.9	8	0.1	0.3	8	0.1	0.3	8	0.1	0.3
I6_32	12	0.3	0.5	12	0.1	0.2	12	0.1	0.2	12	0.1	0.2
I7_32	7	0.3	0.6	7	0.1	0.2	7	0.1	0.2	7	0.1	0.1
I8_32	8	0.3	0.6	8	0.1	0.2	8	0.1	0.1	8	0.1	0.2
II1_32	1	0.3	0.4	1	0.1	0.1	1	0.1	0.1	1	0.1	0.1
II2_32	1	0.3	0.3	1	0.1	0.1	1	0.1	0.1	1	0.1	0.1
II3_32	3	0.4	0.4	3	0.1	0.2	3	0.1	0.2	3	0.1	0.2
II4_32	7	0.4	0.7	7	0.1	0.3	7	0.1	0.4	7	0.2	0.6
II5_32	16	0.3	0.6	16	0.2	0.4	16	0.2	0.3	16	0.2	0.3
II6_32	20	0.3	0.7	20	0.1	0.4	20	0.1	0.4	20	0.1	0.4
II7_32	6	0.3	0.5	6	0.1	0.2	6	0.2	0.2	6	0.2	0.2
II8_32	5	0.3	0.5	5	0.1	0.2	5	0.2	0.3	5	0.2	0.3
III1_32	7	0.4	0.6	7	0.1	0.2	7	0.1	0.2	7	0.2	0.4
III2_32	5	0.3	0.4	5	0.1	0.1	5	0.1	0.1	5	0.2	0.3
III3_32	7	0.4	0.5	7	0.1	0.2	7	0.1	0.2	7	0.2	0.4
III4_32	6	0.3	0.5	6	0.1	0.2	6	0.1	0.2	6	0.3	0.4
III5_32	19	0.4	1.2	19	0.2	0.6	19	0.3	0.8	19	0.3	0.8
III6_32	36	0.4	1.4	36	0.3	1.2	36	0.3	1.3	36	0.3	1.5
III7_32	22	0.4	1.0	22	0.3	1.2	22	0.4	1.1	22	0.3	1.1
III8_32	25	0.4	1.0	25	0.3	1.0	25	0.3	1.1	25	0.3	1.1
IV1_32	4	0.3	0.6	4	0.1	0.2	4	0.1	0.3	4	0.3	0.4
IV2_32	5	0.3	0.6	5	0.1	0.3	5	0.1	0.2	5	0.3	0.5
IV3_32	4	0.3	0.5	4	0.1	0.1	4	0.1	0.2	4	0.3	0.5
IV4_32	12	0.3	0.5	12	0.1	0.2	12	0.1	0.2	12	0.3	0.5
IV5_32	11	0.4	0.9	11	0.2	1.1	11	0.2	1.0	11	0.3	0.9
IV6_32	1	0.3	0.3	1	0.1	0.1	1	0.1	0.1	1	0.3	0.3
IV7_32	10	0.4	1.1	10	0.2	0.8	10	0.2	0.7	10	0.3	0.9
IV8_32	22	0.4	1.8	22	0.4	1.3	22	0.3	1.3	22	0.3	1.4
V1_32	7	0.3	0.5	7	0.3	0.5	7	0.4	0.5	7	0.3	0.5
V2_32	19	0.4	1.7	19	0.4	1.8	19	0.4	1.7	19	0.4	1.7
V3_32	19	0.3	1.0	19	0.3	1.0	19	0.4	1.1	19	0.3	0.9
V4_32	87	1.4	2.4	87	1.4	2.4	87	1.4	2.4	87	1.4	2.4

Table 17: VND - Experiments employing the Lisbon instances

	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)
I1_19	3.0	0.0	0.2	3.0	0.0	0.1	3.0	0.0	0.1	3.0	0.0	0.1
I2_19	2.0	0.0	0.2	2.0	0.0	0.1	2.0	0.0	0.1	2.0	0.0	0.1
I3_19	9.8	8.9	0.5	9.6	6.7	0.1	9.6	6.7	0.2	9.6	6.7	0.2
I4_19	2.0	0.0	0.2	2.0	0.0	0.1	2.0	0.0	0.0	2.0	0.0	0.2
I5_19	19.5	30.0	0.5	19.5	14.7	0.2	19.0	11.8	0.2	19.5	14.7	0.2
I6_19	9.5	18.8	0.6	9.5	18.8	0.2	9.5	18.8	0.3	9.5	18.8	0.1
I7_19	27.9	46.8	10.9	27.0	35.0	10.5	27.0	35.0	2.1	27.0	35.0	0.1
I8_19	2.0	0.0	0.2	2.0	0.0	0.1	2.0	0.0	0.1	2.0	0.0	0.1
II1_19	1.0	0.0	0.2	1.0	0.0	0.1	1.0	0.0	0.0	1.0	0.0	0.2
II2_19	0.0	0.0	0.2	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.2
II3_19	6.5	30.0	0.5	5.5	10.0	0.9	∞	-	-	6.5	30.0	0.2
II4_19	16.5	65.0	0.8	∞	-	-	17.0	41.7	0.6	16.5	37.5	0.3
II5_19	12.0	100.0	0.8	9.0	50.0	2.6	11.0	83.3	1.6	12.0	100.0	0.3
II6_19	25.6	60.0	0.9	25.6	60.0	0.4	25.7	60.6	0.6	25.6	60.0	0.2
II7_19	∞	-	-	∞	-	-	∞	-	-	∞	-	-
II8_19	7.1	136.7	0.7	7.1	18.3	0.2	∞	-	-	7.1	42.0	0.3
III1_19	7.0	0.0	0.2	7.0	0.0	0.1	∞	-	-	7.0	0.0	0.3
III2_19	12.2	35.6	23.7	∞	-	-	∞	-	-	14.0	16.7	0.4
III3_19	14.3	43.0	24.3	∞	-	-	∞	-	-	14.8	13.8	0.3
III4_19	9.8	40.0	0.6	9.8	40.0	0.1	7.9	12.9	0.2	9.8	40.0	0.4
III5_19	34.9	29.3	35.3	33.7	24.8	11.2	∞	-	-	33.0	22.2	0.4
III6_19	33.2	27.7	34.5	32.4	15.7	66.2	32.3	24.2	5.7	33.5	28.8	0.3
III7_19	25.0	38.9	1.6	24.7	7.2	329.6	25.7	35.3	1.4	25.2	32.6	10.1
III8_19	15.3	53.0	0.7	12.1	21.0	7.0	13.6	36.0	1.8	15.3	53.0	0.3
IV1_19	11.9	48.8	0.7	∞	-	-	∞	-	-	11.9	32.2	0.6
IV2_19	14.7	33.6	0.8	∞	-	-	∞	-	-	15.6	30.0	0.4
IV3_19	13.9	39.0	0.8	∞	-	-	∞	-	-	14.1	41.0	0.4
IV4_19	39.2	50.8	6.9	∞	-	-	∞	-	-	39.2	50.8	0.4
IV5_19	30.6	80.0	0.8	23.8	25.3	9.5	21.8	14.7	4.9	30.6	80.0	0.5
IV6_19	34.9	66.2	8.1	27.9	11.6	75.7	35.6	54.8	2.0	34.9	51.7	0.4
IV7_19	10.2	13.3	0.6	10.2	13.3	0.3	10.2	13.3	0.8	10.2	13.3	0.4
IV8_19	10.2	13.3	0.6	10.2	13.3	0.3	10.0	11.1	0.7	10.2	13.3	0.4
average		35.8			16.1			21.9			27.9	
I1_32	3.0	0.0	0.5	3.0	0.0	0.1	3.0	0.0	0.1	3.0	0.0	0.1
I2_32	3.0	0.0	0.5	3.0	0.0	0.1	3.0	0.0	0.1	3.0	0.0	0.1
I3_32	6.0	0.0	0.6	6.0	0.0	0.2	6.0	0.0	0.2	6.0	0.0	0.2
I4_32	1.0	0.0	0.5	1.0	0.0	0.1	1.0	0.0	0.1	1.0	0.0	0.2
I5_32	8.0	0.0	0.5	8.0	0.0	0.2	8.0	0.0	0.2	8.0	0.0	0.2
I6_32	12.0	0.0	0.5	12.0	0.0	0.1	12.0	0.0	0.3	12.0	0.0	0.1
I7_32	7.0	0.0	0.5	7.0	0.0	0.1	7.0	0.0	0.2	7.0	0.0	0.1
I8_32	8.0	0.0	0.5	8.0	0.0	0.1	8.0	0.0	0.1	8.0	0.0	0.1
II1_32	1.0	0.0	0.5	1.0	0.0	0.1	1.0	0.0	0.1	1.0	0.0	0.2
II2_32	1.0	0.0	0.5	1.0	0.0	0.1	1.0	0.0	0.1	1.0	0.0	0.2
II3_32	3.0	0.0	0.5	3.0	0.0	0.1	3.0	0.0	0.1	3.0	0.0	0.2
II4_32	7.0	0.0	0.5	7.0	0.0	0.2	7.0	0.0	0.3	7.0	0.0	0.3
II5_32	16.0	0.0	0.5	16.0	0.0	0.3	16.0	0.0	0.5	16.0	0.0	0.3
II6_32	20.0	0.0	0.5	20.0	0.0	0.2	20.0	0.0	0.3	20.0	0.0	0.2
II7_32	6.0	0.0	0.5	6.0	0.0	0.2	6.0	0.0	0.2	6.0	0.0	0.3
II8_32	5.0	0.0	0.5	5.0	0.0	0.2	5.0	0.0	0.3	5.0	0.0	0.3
III1_32	7.0	0.0	0.5	7.0	0.0	0.1	7.0	0.0	0.2	7.0	0.0	0.4
III2_32	5.0	0.0	0.5	5.0	0.0	0.1	5.0	0.0	557.1	5.0	0.0	0.3
III3_32	7.0	0.0	0.5	7.0	0.0	0.1	7.0	0.0	0.2	7.0	0.0	0.4
III4_32	6.0	0.0	0.5	6.0	0.0	0.1	6.0	0.0	0.2	6.0	0.0	0.4
III5_32	19.0	0.0	0.5	19.0	0.0	0.3	19.0	0.0	0.4	19.0	0.0	0.3
III6_32	36.9	2.5	14.4	37.5	4.2	10.1	36.6	1.7	9.3	37.5	4.2	10.1
III7_32	22.0	0.0	0.5	22.0	0.0	0.4	22.0	0.0	0.3	22.0	0.0	0.3
III8_32	27.0	8.0	0.9	27.0	8.0	0.6	27.0	8.0	0.5	27.0	8.0	0.6
IV1_32	4.0	0.0	0.5	4.0	0.0	0.1	4.0	0.0	0.1	4.0	0.0	0.4
IV2_32	5.0	0.0	0.5	5.0	0.0	0.1	5.0	0.0	0.1	5.0	0.0	0.4
IV3_32	4.0	0.0	0.5	4.0	0.0	0.1	4.0	0.0	0.1	4.0	0.0	0.4
IV4_32	12.0	0.0	0.5	12.0	0.0	0.1	12.0	0.0	0.2	12.0	0.0	0.5
IV5_32	11.0	0.0	0.5	11.0	0.0	0.3	11.0	0.0	0.4	11.0	0.0	0.4
IV6_32	1.0	0.0	0.5	1.0	0.0	0.1	1.0	0.0	0.1	1.0	0.0	0.4
IV7_32	10.0	0.0	0.5	10.0	0.0	0.3	10.0	0.0	0.4	10.0	0.0	0.4
IV8_32	22.0	0.0	0.5	22.0	0.0	0.4	22.0	0.0	0.4	22.0	0.0	0.4
V1_32	7.0	0.0	0.4	7.0	0.0	0.4	7.0	0.0	0.4	7.0	0.0	0.4
V2_32	20.0	5.3	12.5	20.0	5.3	12.1	21.0	10.5	11.0	20.0	5.3	11.6
V3_32	19.7	3.7	9.3	19.7	3.7	9.6	19.9	4.7	8.7	19.7	3.7	9.1
V4_32	102.0	17.2	126.3	102.0	17.2	119.7	102.0	17.2	67.4	102.0	17.2	120.5
average		1.0			1.1			1.2			1.1	

Table 18: Open-source solver - complete scheduling horizon NRP+NRRP constraints.

	CPLEX				Coin-OR CBC					
Instance Id	OFV	#Feasible	Opt Time(s)	Opt Prove Time(s)	OFV	#Feasible	Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.
n035w4	3167.0	10	5.3	9.6	3167.0	10	183.1	113.9	200.5	135.2
n035w8	6383.5	10	7.5	28.6	6420.5	10	1742.6	724.8	-	-
n070w4	5564.5	10	13.8	41.9	5565.0	10	1299.4	472.0	-	-
n070w8	11595.0	10	109.2	208.0	-	0	-	-	-	-
n110w4	7039.0	10	28.3	84.5	7062.0	10	2406.4	599.1	-	-
n110w8	13670.5	10	229.6	598.9	-	0	-	-	-	-

To investigate the performance of the proposed solution approaches on large problem instances, ten additional larger instances containing 150, 200, 300, 400 and 500 nurses were generated based on the INRC-II constraints and problem characteristics. Table 19 presents the results using CPLEX and the VND heuristic for these much larger instances. For each instance, the complete scheduling horizon and all NRP and NRRP constraints were considered. Note that Coin-OR CBC is not included in this comparison as Table 18 already demonstrated that instances with 110 nurses are beyond its capabilities.

The first column in Table 19 describes the instance size ranging from 150 to 500 nurses and scheduling horizon of four and eight weeks. The second and seventh columns show the percentage of instances for which a feasible solution was found. The third and eight columns provide the objective function values, while the fourth and ninth columns detail the gap relative to the lower bound obtained by CPLEX. The fifth and tenth columns provide the required computation time in seconds. The sixth and eleventh columns present the standard deviation relative to the computation time. Infeasible solutions were not taken into account for these calculations.

CPLEX manages to consistently find feasible solutions for problems with up to 400 nurses and a scheduling horizon of four weeks. Even for the instances with 400 nurses and scheduling horizon of eight weeks, a feasible solution was found for the majority of instances (7 out of 10). An interesting observation was that when CPLEX can solve the initial infeasibility, it quickly found (near-)optimum solutions in very limited computation time. For the instances with 500 nurses and scheduling horizon of eight weeks, CPLEX was unable to find any feasible solutions within the time limit.

Table 19: Large instances - complete scheduling horizon NRP+NRRP constraints.

	CPLEX					VND				
Instance Id	Feasible(%)	OFV	Gap(%)	Time(s)	Std. Dev.	Feasible(%)	OFV	Gap(%)	Time(s)	Std. Dev.
n150w4	100	126550.5	0.0	318.1	146.2	100	126889.3	0.3	39.1	10.3
n150w8	100	317429.5	0.0	2317.2	897.5	100	318278.3	0.3	107.7	32.1
n200w4	100	177228.0	0.0	557.9	165.4	100	177844.5	0.4	79.6	22.6
n200w8	100	442882.5	0.6	3421.7	234.3	100	444443.5	1.0	275.8	94.3
n300w4	100	270278.0	0.0	1925.7	991.6	100	270989.6	0.3	176.0	51.9
n300w8	100	689322.0	1.3	3241.5	444.0	100	690517.9	1.5	722.6	237.3
n400w4	100	358380.0	0.0	2233.5	678.7	100	359583.0	0.3	489.3	63.7
n400w8	70	903917.1	1.8	2385.3	1069.8	100	904319.2	1.9	1877.2	253.3
n500w4	100	453763.6	0.1	3394.9	283.3	100	454904.0	0.3	527.4	53.6
n500w8	0	-	-	3600.0	-	100	1375360.4	-	3790.7	589.2

By contrast, the VND heuristic generated feasible solutions for all these instances in considerably shorter running time, with exception of the instances with 500 nurses and eighth weeks where the running time was on average 3791 seconds. The solutions obtained by the VND heuristic were near-optimum with an average gap of only 0.7%, demonstrating how it provides the best solution approach for large-scale problems with hundreds of nurses if low computation times are required.

7. Conclusions

The primary contribution of this work is the evaluation of novel rerostering strategies such as the relaxation of the NRP soft constraints and various rerostering scheduling horizons. Additionally, a general integer programming formulation considering multi-skilled nurses and a large set of constraints from both the NRP and the NRRP was introduced. A third contribution is a VND heuristic, which provides an alternative to commercial solvers and significantly reduces the required computational time at the expense of very small reductions in solution quality. Furthermore, a new set of instances derived from those proposed by Moz & Pato (2007) and Ceschia et al. (2014), which are used throughout the computational experiments, have been made publicly available online[2].

Besides the NRRP constraints, the computational study has revealed that main-

[2] `http://www.inf.ufrgs.br/~tiwickert/download/2017/reroster`

taining the original NRP's constraints is important for obtaining high-quality NRRP solutions. However, ignoring the NRP's soft constraints provides a good alternative when urgent demands require online changes to the current roster, such as when, for example, it is necessary to cover a shortage of professionals for a surgery. Rerostering the complete scheduling horizon generates the best solutions, but requires longer runtimes. If only the days on which absenteeism occurs are evaluated, less time is required to reach solutions and this, therefore, represents a good alternative strategy when very little time is available for rerostering. Only considering the period from the first absent day until the last absent day or until the end of the scheduling horizon generated similar results, but both scheduling horizons are important to consider when an absence is communicated during the current month.

Results also demonstrated that some solutions remain valid despite nurse absenteeism, with this type of roster robustness being a desirable solution property given how it minimizes the impact when personnel shortages occur. Future research should consider the introduction of robustness which facilitates the repair of disruptions in terms of employee absences and the preparation of rosters which are less susceptible to disruptions in the first place.

Acknowledgement

The authors would like to thank Margarida Moz, Margarida Vaz Pato and Jan Dvořák who kindly have answered our e-mails with questions about the Lisbon instances and their research related to the Nurse Rerostering Problem. Editorial consultation provided by Luke Connolly (KU Leuven).

Appendix A. General integer programming formulation for the nurse rostering problem

Table A.20 provides the sets, decision and auxiliary variables employed in the formulation. The objective function minimizes the cost associated with the violation of the soft constraints.

Table A.20: Indices, sets, decision and auxiliary variables employed in the problem formulation.

Symbol	Definition
Input Data	
$n \in N$	n is the index of the nurse, and N is the set of nurses;
$d \in D$	d is the index of the day, and D is the set of days;
$s \in S$	s is the index of the shift, and $\{1,2,3,4\} \in S$ is the set of shifts, where 1 corresponds to Early, 2 to Day, 3 to Late and 4 to Night;
$k \in K$	k is the index of the skill, and $\{1,2,3,4\} \in K$ is the set of skills, where 1 corresponds to HeadNurse, 2 to Nurse, 3 to Caretaker and 4 to Trainee;
l_{nk}	is the skill of the nurse n at position k of a vector with dimension $\|K\|$, where a zero in position k means that nurse n does not have skill k. Consider the case in which *Nurse1* has only two skills, and the problem's input has four skills. The vector of skills for *Nurse1* is (1, 2, 0, 0) representing the *Nurse1* has the skills *HeadNurse* and *Nurse*, while does not have skills *Caretaker* and *Trainee*. This way, the decision variable can assume 1 on Equation A.5 in the first two cases, because, $(1-1)x_{ndsk} = 0$ implies *Nurse1* can assume a day or shift with skill *HeadNurse*; $(2-2)x_{ndsk} = 0$ implies *Nurse1* can assume a day or shift with skill *Nurse*; $(3-0)x_{ndsk} = 0$ implies *Nurse1* cannot assume a day or shift with skill *Caretaker*; $(4-0)x_{ndsk} = 0$ implies *Nurse1* cannot assume a day or shift with skill *Trainee*;
$(n,d,s) \in U$	vector containing a triple with the undesired working day d, shift s for nurse n. For example, $(1,2,4) \in U$ means *Nurse1* prefers to avoid working on day 2, shift 4 (Night shift).
$r_{dsk} \in \mathbb{N}_0$	number of required nurses on day d, shift s, having skill k;

$(s1,s2) \in \hat{S}$	contains the pairs of invalid shift sequences, for example, $(4,1) \in \hat{S}$ means that a Night shift cannot be followed by an Early shift;
T^w	set of patterns $T^w = \{T_t^w : t \in \{1,2,\ldots,p^w\}\}$, where p^w is the minimum number of consecutive working days - 1. T_t^w is a binary vector of dimension $t+2$, with one zero in the first position and one zero in the last position, being t the number of ones that appear in vector T_t^w. For example, considering 4 as the minimum number of working days, the patterns to search are $T^w = \{T_1^w = (0,1,0), T_2^w = (0,1,1,0), T_3^w = (0,1,1,1,0)\}$. If the first pattern is found in the schedule, it represents three violations, the second pattern two violations, and the third pattern a single violation.
T^r	follows the same idea of T^w, and represents a set of patterns $T^r = \{T_t^r : t \in \{1,2,\ldots,p^r\}\}$, where p^r is the minimum number of consecutive days off - 1.
T^s	follows the same idea of T^w, and represents a set of patterns $T^s = \{T_{t_s}^s : t_s \in \{1,2,\ldots,p_s\}\}$, where p_s is the minimum number of consecutive working days - 1 at shift s.
$w \in W$	w is a Saturday index and W the set of all Saturdays indexes;
$M_h \in \{5,6\}$	set of maximum working days every 7 days. $M_1 = 5$ and $M_2 = 6$ depending of the nurses' contract;
α_{dsk}^1	preferred number of nurses for day d, shift s, skill k;
β_n^i	limit of soft constraint $2,\ldots,5$ and $10,\ldots,12$ for nurse n, that is, minimum/maximum consecutive working days, minimum/maximum consecutive days off, minimum/maximum number of assignments over the scheduling period and total working weekends;
γ_s^i	limit of soft constraint 6 and 7 for shift s, that is, minimum/maximum consecutive assignments to the same shift;

ω^i weight for violating the lower and/or upper limits of soft constraint i.

Decision Variables

$x_{ndsk} \in \{0,1\}$ 1 if nurse n is allocated on day d, shift s with skill k, 0 otherwise;

$y_{nw} \in \{0,1\}$ 1 if nurse n works at weekend w, 0 otherwise.

Auxiliary Variables

$a^1_{dsk} \in \mathbb{N}_0$ preferred number of nurses violations for day d, shift s, skill k;

$b^i_{ndt} \in \mathbb{N}_0$ minimum number of consecutive working days and days off violations, $i \in 2,4$ for nurse n on day d, pattern t;

$c^i_{nd} \in \mathbb{N}_0$ maximum number of consecutive working days and days off violations, $i \in 3,5$ for nurse n on day d;

$e^6_{ndst} \in \mathbb{N}_0$ minimum number of consecutive assignments to the same shift violations, for nurse n on day d, shift s, pattern t;

$f^7_{nds} \in \mathbb{N}_0$ maximum number of consecutive assignments to the same shift violations, for nurse n on day d, shift s;

$g^8_{nds} \in \mathbb{N}_0$ number of nurse's undesired working day/shift violations, for nurse n on day d, shift s;

$h^9_{nw} \in \mathbb{N}_0$ number of complete weekends violations, for nurse n on weekend w;

$j^i_n \in \mathbb{N}_0$ minimum/maximum number of assignments over the scheduling period violations, maximum number of worked weekends violations, $i \in \{10,11,12\}$ for nurse n.

Constant

C constant with value 10.

$$\mathbf{Min} \quad \left[\sum_{d\in D}\sum_{s\in S}\sum_{k\in K} a^1_{dsk}\omega^1\right]+ \left[\sum_{n\in N}\sum_{d\in D}\sum_{t\in T_t}\sum_{i\in\{2,4\}} b^i_{ndt}\omega^i\right]+ \left[\sum_{n\in N}\sum_{d\in D}\sum_{i\in\{3,5\}} c^i_{nd}\omega^i\right]+$$
$$\left[\sum_{n\in N}\sum_{d\in D}\sum_{s\in S}\sum_{t\in T_t} e^6_{ndst}\omega^6\right]+ \left[\sum_{n\in N}\sum_{d\in D}\sum_{s\in S} f^7_{nds}\omega^7\right]+ \left[\sum_{n\in N}\sum_{d\in D}\sum_{s\in S} g^8_{nds}\omega^8\right]+$$
$$\left[\sum_{n\in N}\sum_{w\in W} h^9_{nw}\omega^9\right]+ \left[\sum_{n\in N}\sum_{i\in\{10,11,12\}} j^i_n\omega^i\right] \tag{A.1}$$

Subject to

$$\sum_{s\in S}\sum_{k\in K} x_{ndsk} \le 1 \qquad \forall n\in N, d\in D \tag{A.2}$$

$$\sum_{n\in N} x_{ndsk} \ge r_{dsk} \qquad \forall d\in D, s\in S, k\in K \tag{A.3}$$

$$\sum_{k\in K}(x_{nds1k} + x_{n(d+1)s2k}) \le 1 \qquad \forall n\in N, d\in D\setminus\{|D|\}, (s1,s2)\in\hat{S} \tag{A.4}$$

$$(k - l_{nk})x_{ndsk} = 0 \qquad \forall n\in N, d\in D, s\in S, k\in K \tag{A.5}$$

$$\sum_{d'=d}^{6+d}\sum_{s\in S}\sum_{k\in K} x_{nd'sk} \le M_h \qquad \forall n\in N, d\in D \tag{A.6}$$

$$\sum_{n\in N} x_{ndsk} + a^1_{dsk} \ge \alpha^1_{dsk} \qquad \forall d\in D, s\in S, k\in K \tag{A.7}$$

$$S1_{ndt} + b^2_{ndt} \ge \beta^2_n \qquad \forall n\in N, t\in\{1,2,\dots p^w\}, d\in\{1,2,\dots,|D|-(t+2)\} \tag{A.8}$$

$$\begin{aligned} S1_{ndt} = &\sum_{d'=d}^{t+d+1}\sum_{s\in S}\sum_{k\in K} x_{nd'sk} + \\ &\sum_{d'\in d,t+d+1}\sum_{s\in S}\sum_{k\in K} x_{nd'sk}C + \\ &\sum_{d'=d+1}^{t+d}(1 - \sum_{s\in S}\sum_{k\in K} x_{nd'sk})C \end{aligned} \tag{A.9}$$

$$\sum_{d'=d}^{\beta^3_n+d}\sum_{s\in S}\sum_{k\in K} x_{nd'sk} - c^3_{nd} \le \beta^3_n \qquad \forall n\in N, d\in\{1,\dots,|D|-\beta^3_n\} \tag{A.10}$$

$$S2_{ndt} + b^4_{ndt} \ge \beta^4_n \qquad \forall n\in N, t\in\{1,2,\dots p^r\}, d\in\{1,2,\dots,|D|-(t+2)\} \tag{A.11}$$

$$\begin{aligned} S2_{ndt} = &\sum_{d'=d}^{t+d+1}(1 - \sum_{s\in S}\sum_{k\in K} x_{nd'sk}) + \\ &\sum_{d'\in d,t+d+1}(1 - \sum_{s\in S}\sum_{k\in K} x_{nd'sk})C + \\ &\sum_{d'=d+1}^{t+d}\sum_{s\in S}\sum_{k\in K} x_{nd'sk}C \end{aligned} \tag{A.12}$$

$$\sum_{d'=d}^{\beta^5_n+d}(1 - \sum_{s\in S}\sum_{k\in K} x_{nd'sk}) - c^5_{nd} \le \beta^5_n \qquad \forall n\in N, d\in\{1,\dots,|D|-\beta^5_n\} \tag{A.13}$$

$$S3_{ndst} + e^{6}_{ndst} \geq \gamma^{6}_{s} \qquad \forall n \in N, s \in S, t_s \in \{1, 2, \ldots p_s\}, d \in \{1, 2, \ldots, |D| - (t_s + 2)\} \tag{A.14}$$

$$\begin{aligned} S3_{ndst} = & \sum_{d'=d}^{t_s+d+1} \sum_{k \in K} x_{nd'sk} + \\ & \sum_{d' \in d, t_s+d+1} \sum_{k \in K} x_{nd'sk} C + \\ & \sum_{d'=d+1}^{t_s+d} (1 - \sum_{k \in K} x_{nd'sk}) C \end{aligned} \tag{A.15}$$

$$\sum_{d'=d}^{|\gamma^{7}_{s}|+d} \sum_{k \in K} x_{nd'sk} - f^{7}_{nds} \leq \gamma^{7}_{s} \qquad \forall n \in N, s \in S, d \in \{1, \ldots, |D| - \gamma^{7}_{s}\} \tag{A.16}$$

$$g^{8}_{nds} - \sum_{k \in K} x_{ndsk} = 0 \qquad \forall (n, d, s) \in U \tag{A.17}$$

$$\sum_{s \in S} \sum_{k \in K} (x_{nwsk} + x_{n(w+1)sk}) \leq 2y_{nw} \qquad \forall n \in N, w \in W \tag{A.18}$$

$$\sum_{s \in S} \sum_{k \in K} (x_{nwsk} + x_{n(w+1)sk}) + h^{9}_{nw} \geq 2y_{nw} \qquad \forall n \in N, w \in W \tag{A.19}$$

$$\sum_{d \in D} \sum_{s \in S} \sum_{k \in K} x_{ndsk} + j^{10}_{n} \geq \beta^{10}_{n} \qquad \forall n \in N \tag{A.20}$$

$$\sum_{d \in D} \sum_{s \in S} \sum_{k \in K} x_{ndsk} - j^{11}_{n} \leq \beta^{11}_{n} \qquad \forall n \in N \tag{A.21}$$

$$\sum_{w \in W} y_{nw} - j^{12}_{n} \leq \beta^{12}_{n} \qquad \forall n \in N \tag{A.22}$$

$$x_{ndsk} \in \{0, 1\} \qquad \forall n \in N, d \in D, s \in S, k \in K \tag{A.23}$$

$$y_{nw} \in \{0, 1\} \qquad \forall n \in N, w \in W \tag{A.24}$$

Constraints (A.2) ensure a single shift per day. Constraints (A.3) ensure the minimum number of nurses per days, shift, and skill. Constraints (A.4) ensure that a shift succession must be valid. Constraints (A.5) ensure a nurse can only be scheduled on a shift if they have the required skill. Constraints (A.6) ensure maximum M_h worked days, every 7 days. Constraints (A.7) calculate the preferred coverage violations. Constraints (A.8) and (A.9) calculate the minimum consecutive assignments (working days) violations. In the equations, $S1$ is calculated as the *(sum of the working days) + (two border bits × C) + (complement of middle bits × C)*. Constraints (A.10) calculate the maximum number of consecutive assignments (working days) violations. Constraints (A.11) and (A.12) calculate the minimum number of consecutive days off violations. $S2$ is evaluated similarly to Equations (A.8) and (A.9), however, the bits are inverted and the sum is related to free days instead of working days. Constraints (A.13) calculate the maximum number of consecutive days off violations. Constraints (A.14) and (A.15) calculate the minimum number of consecutive assignments to the same shift violations. $S3$ is evaluated similarly to Equations (A.8) and (A.9), however, the violations are stored by nurse/day/shift/pattern. Constraints (A.16) calculate the maximum of consecutive assignments to the same shift violations. Constraints (A.17) calculate the undesired day/shift assignments violations. Constraints (A.18) calculate if nurse n works on weekend w. Constraints (A.19) calculate the complete weekend violation. Constraints (A.20) calculate the minimum number of total working days violations over the whole scheduling period. Constraints (A.21) calculate the maximum number of total working days violations over the whole scheduling period. Constraints (A.22) calculate the total number of working weekends violations. Constraints (A.23) and (A.24) define the decision variables as binary.

Appendix B. VND iterations

Table B.21: VND - Average number of iterations

	VND iterations - single-day absences			
	Complete scheduling	Only absent days	First absence to last absence	First absence to end scheduling
n035w4	1.1	1.0	1.1	1.1
n035w8	1.4	1.0	1.4	1.4
n070w4	1.7	1.5	1.7	1.7
n070w8	1.9	1.8	1.9	1.9
n110w4	1.4	1.4	1.4	1.4
n110w8	1.9	1.8	1.9	1.9
	VND iterations - consecutive-days absences			
	Complete scheduling	Only absent days	First absence to last absence	First absence to end scheduling
n035w4	1.4	1.2	1.4	1.4
n035w8	1.5	1.5	1.5	1.5
n070w4	1.5	1.4	1.5	1.5
n070w8	1.9	1.4	1.9	1.9
n110w4	2.0	1.9	2.0	2.0
n110w8	1.8	1.8	1.8	1.8
	VND iterations - Lisbon instances			
	Complete scheduling	Only absent days	First absence to last absence	First absence to end scheduling
19 nurses	12.9	38.1	32.8	15.3
32 nurses	4.8	4.8	4.8	4.8

References

Aguirre, A., & Kerin, A. (2014). Shiftwork practices 2005. URL: `https://create.piktochart.com/output/2598161-absenteeism` [Online; accessed 20-July-2018].

Bäumelt, Z., Dvořák, J., Šůcha, P., & Hanzálek, Z. (2016). A novel approach for nurse rerostering based on a parallel algorithm. *European Journal of Operational Research*, *251*, 624 – 639.

Burke, E. K., & Curtois, T. (2014). New approaches to nurse rostering benchmark instances. *European Journal of Operational Research*, *237*, 71 – 81. doi:`10.1016/j.ejor.2014.01.039`.

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

Burke, E. K., De Causmaecker, P., Vanden Berghe, G., & Van Landeghem, H. (2004). The state of the art of nurse rostering. *Journal of scheduling*, *7*, 441–499.

Ceschia, S., Dang, N. T. T., De Causmaecker, P., Haspeslagh, S., & Schaerf, A. (2014). Second international nurse rostering competition. URL: `http://mobiz.vives.be/inrc2/wp-content/uploads/2014/10/INRC2.pdf` [Online; accessed 19-July-2018].

Clark, A., Moule, P., Topping, A., & Serpell, M. (2015). Rescheduling nursing shifts: scoping the challenge and examining the potential of mathematical model based tools. *Journal of Nursing Management*, *23*, 411–420. doi:`10.1111/jonm.12158`.

De Causmaecker, P., & Vanden Berghe, G. (2011). A categorisation of nurse rostering problems. *Journal of Scheduling*, *14*, 3–16.

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

Forbes (2013). The causes and costs of absenteeism in the workplace. URL: https://www.forbes.com/sites/investopedia/2013/07/10/the-causes-and-costs-of-absenteeism-in-the-workplace [Online; accessed 16-July-2018].

Gomes, R. A., Toffolo, T. A., & Santos, H. G. (2017). Variable neighborhood search accelerated column generation for the nurse rostering problem. *Electronic Notes in Discrete Mathematics*, *58*, 31 – 38. doi:10.1016/j.endm.2017.03.005. 4th International Conference on Variable Neighborhood Search.

Ingels, J., & Maenhout, B. (2015). The impact of reserve duties on the robustness of a personnel shift roster: An empirical investigation. *Computers & Operations Research*, *61*, 153 – 169. doi:10.1016/j.cor.2015.03.010.

Legrain, A., Bouarab, H., & Lahrichi, N. (2014). The nurse scheduling problem in real-life. *Journal of Medical Systems*, *39*, 160. doi:10.1007/s10916-014-0160-8.

Legrain, A., Omer, J., & Rosat, S. (2017). A rotation-based branch-and-price approach for the nurse scheduling problem. URL: https://hal.archives-ouvertes.fr/hal-01545421 [Online; accessed 19-July-2018].

Lougee-Heimer, R. (2003). The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, *47*, 57–66.

Maenhout, B., & Vanhoucke, M. (2011). An evolutionary approach for the nurse rerostering problem. *Computers & Operations Research*, *38*, 1400 – 1411.

Maenhout, B., & Vanhoucke, M. (2013). Reconstructing nurse schedules: Computational insights in the problem size parameters. *Omega*, *41*, 903 – 918.

Mischek, F., & Musliu, N. (2017). Integer programming model extensions for a multi-stage nurse rostering problem. *Annals of Operations Research*, . doi:10.1007/s10479-017-2623-z.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*, 1097 – 1100. doi:10.1016/S0305-0548(97)00031-2.

Moz, M., & Pato, M. V. (2003). An integer multicommodity flow model applied to the rerostering of nurse schedules. *Annals of Operations Research*, *119*, 285–301.

Moz, M., & Pato, M. V. (2004). Solving the problem of rerostering nurse schedules with hard constraints: New multicommodity flow models. *Annals of Operations Research*, *128*, 179–197.

Moz, M., & Pato, M. V. (2007). A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research*, *34*, 667 – 691. Logistics of Health Care ManagementPart Special Issue: Logistics of Health Care Management.

Pato, M. V., & Moz, M. (2008). Solving a bi-objective nurse rerostering problem by using a utopic pareto genetic heuristic. *Journal of Heuristics*, *14*, 359–374.

SD Worx (2013). 2012: A record year for absenteeism. URL: https://www.sdworx.be/nl-be/sd-worx-r-d/publicaties/persberichten/2013-03-27-recordjaar-ziekteverzuim [Online; accessed 11-July-2018].

Tassopoulos, I. X., Solos, I. P., & Beligiannis, G. N. (2015). A two-phase adaptive variable neighborhood search approach for nurse rostering. *Computers & Operations Research*, *60*, 150 – 169. doi:10.1016/j.cor.2015.02.009.

Valouxis, C., Gogos, C., Goulas, G., Alefragis, P., & Housos, E. (2012). A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, *219*, 425 – 433. doi:10.1016/j.ejor.2011.12.042.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research*, *226*, 367 – 385.

Williams, J. C., Lambert, S., & Kesavan, S. (2017). How the gap used an app to give workers more control over their schedules. URL: https:

//hbr.org/2017/12/how-the-gap-used-an-app-to-give-workers-more-control-over-their-schedules [Online; accessed 20-July-2018].

Zheng, Z., Liu, X., & Gong, X. (2017). A simple randomized variable neighbourhood search for nurse rostering. *Computers & Industrial Engineering*, *110*, 165 – 174.