Constraint reformulation for nurse rostering problems

Pieter Smet

Abstract Various theoretical and empirical hardness studies have shown that constraints concerning the number of consecutive assignments make nurse rostering problems particularly difficult to solve. The present paper further explores this issue and builds upon recent computational insights by reformulating these constraints as subproblems which are known to have an efficient structure. By combining these subproblems together in a new integer programming formulation, the required calculation time to reach optimal solutions is significantly reduced using a standard integer programming solver. To gain a better understanding of these results, a characterization of problem instances for which the reformulation of challenging constraints works particularly well is provided.

Keywords Nurse rostering · Integer programming · s-t paths

1 Introduction

Nurse rostering problems are encountered in hospitals all over the world. Throughout the last decades, countless studies have been published in the academic literature detailing solution approaches for different problem variants [8]. Although there are variants of the nurse rostering problem which are solvable in polynomial time [1,7], the problem quickly becomes NP-hard when essential constraints are included [2]. Computational experiences have shown that certain nurse rostering constraints have a significant impact on the performance of optimization algorithms [5]. In particular, constraints regarding the number of consecutive assignments have been shown to have a particularly

P. Smet
KU Leuven, Department of Computer Science, CODeS & imec
Gebroeders De Smetstraat 1, 9000 Gent, Belgium
Tel.: +3292658704
E-mail: pieter.smet@cs.kuleuven.be

strong influence. The present paper aims to exploit this knowledge by targeting these constraints in a reformulation of the commonly used time-indexed integer programming model for nurse rostering.

The concept of reformulating challenging constraints in rostering problems has also been explored by Zhang et al. [9] who proposed two reformulations. Their first reformulation embedded constraints on consecutive assignments in a network flow structure, while their second model captured constraints on the total number of assignments in such structures. A computational study demonstrated that the models which managed to efficiently represent the constraints on consecutive assignments performed significantly better than the other models, thereby confirming the assumption that such constraints are the most difficult. The results reported in the present paper support this conclusion and provide further analysis concerning a new reformulation of the consecutive assignment constraints. Furthermore, a direct link is established between the computational hardness of these constraints and the performance gain achieved via their reformulation.

The remainder of this paper is organized as follows. Section 2 introduces the nurse rostering problem and presents the standard integer programming formulation. Section 3 identifies which constraints influence the computational hardness of the problem and proposes a reformulation of these constraints. Section 4 presents computational results on different integer programming formulations. Section 5 provides a discussion and analysis of these results. Finally, Section 6 concludes the paper and identifies areas for future research.

2 The nurse rostering problem

This study addresses a general variant of the nurse rostering problem which considers most of the important constraints relevant in practice [3]. After defining the problem, a standard compact integer programming formulation using time-indexed decision variables is presented.

2.1 Problem definition

Let $D = \{1, ..., |D|\}$ be a scheduling period consisting of $|D|$ consecutive days. Given a set of nurses $N = \{1, ..., |N|\}$ and a set of shifts $S = \{1, ..., |S|\}$, the goal is to find an assignment of shifts to nurses. Each shift s has a duration l_s, expressed in minutes. A minimum amount of rest time is required after each shift meaning that certain shift successions are not permitted. Let R_s be the subset of shifts that cannot be assigned on day $d + 1$ if shift s is assigned on day d.

Each nurse works according to a contract which defines feasible nurse-shift assignments. These time-related constraints are categorized as either counter or series constraints [6]. Counter constraints denote constraints which can be evaluated by counting the occurrence of certain assignments in a roster. This

paper considers the following counter constraints for each nurse n: the maximum number of days working shift s m_{ns}^{max}, the minimum and maximum number of minutes worked b_n^{min} and b_n^{max}, and the maximum number of weekends worked a_n^{max}. Furthermore, each nurse n has a set of days F_n on which they cannot be assigned to any shift.

Series constraints are restrictions on successive or consecutive assignments. The current problem considers the following series constraints for each nurse n: the minimum and maximum number of consecutive days worked c_n^{min} and c_n^{max}, and the minimum number of consecutive days off o_n^{min}.

Each nurse is associated with a number of personal preferences regarding their roster, resulting in penalties q_{nds} which are incurred if nurse n is not assigned to shift s on day d, and p_{nds} if nurse n is assigned to shift s on day d. Coverage requirements u_{ds} specify the required number of nurses working shift s on day d. If more nurses than necessary are assigned, a penalty of v_{ds}^{max} multiplied by the number of excess nurses is incurred. Similarly, if fewer nurses than required are assigned, a penalty of v_{ds}^{min} multiplied by the nurse shortage is incurred. The objective is to minimize the sum of nurse preference violations as well as over- and understaffing penalties.

2.2 Time-indexed integer programming formulation

The most commonly used compact integer programming formulation of the nurse rostering problem utilizes time-indexed decision variables. Let x_{nds} equal one if nurse n is assigned to shift s on day d, and zero otherwise. Additionally, a number of auxiliary decision variables are used. Let k_{nw} equal one if nurse n is working during weekend w, and zero otherwise. Let y_{ds} and z_{ds} be the number of nurses under and over the required number of nurses on day d and shift s.

Objective function (1) minimizes the weighted sum of penalties regarding preference violations and under- and overstaffing.

$$min \sum_{n\in N}\sum_{d\in D}\sum_{s\in S}(q_{nds}(1-x_{nsd})+p_{nds}x_{nsd})+\sum_{d\in D}\sum_{s\in S}(v_{ds}^{min}y_{ds}+v_{ds}^{max}z_{ds}) \quad (1)$$

The first group of constraints models the core assignment requirements. Constraints (2) ensure that each nurse is assigned to at most one shift per day. Constraints (3) model the coverage requirements for each day and shift.

$$\sum_{s\in S} x_{nds} \leq 1 \qquad \forall\ n \in N, d \in D \quad (2)$$

$$\sum_{n\in N} x_{nds} - z_{ds} + y_{ds} = u_{ds} \qquad \forall\ d \in D, s \in S \quad (3)$$

The second group of constraints corresponds to the nurses' counter constraints. Constraints (4) restrict the maximum number of days each nurse

is assigned to shift s. Constraints (5) limit the minimum and maximum time worked. Constraints (6)-(7) restrict the maximum number of weekends worked, while Constraints (8) model the hard day off requests.

$$\sum_{d \in D} x_{nds} \leq m_{ns}^{max} \qquad \forall\, n \in N, s \in S \quad (4)$$

$$b_n^{min} \leq \sum_{d \in D} \sum_{s \in S} l_s x_{nds} \leq b_n^{max} \qquad \forall\, n \in N \quad (5)$$

$$k_{nw} \leq \sum_{s \in S} x_{n(7w-1)s} + \sum_{s \in S} x_{n(7w)s} \leq 2k_{nw} \qquad \forall\, n \in N, w \in W \quad (6)$$

$$\sum_{w \in W} k_{nw} \leq a_n^{max} \qquad \forall\, n \in N \quad (7)$$

$$x_{nds} = 0 \qquad \forall\, n \in N, d \in F_n, s \in S \quad (8)$$

The third group of constraints concerns the nurses' series constraints. Constraints (9) model the forbidden shift successions. Constraints (10) and (11) restrict the maximum and minimum number of consecutive days worked, respectively. Constraints (12) limit the minimum number of consecutive days off.

$$x_{nds} + x_{n(d+1)s'} \leq 1 \qquad \forall\, n \in N, d \in \{1, ..., |D| - 1\}, s \in S, s' \in R_s \quad (9)$$

$$\sum_{j=d}^{d+c_n^{max}} \sum_{s \in S} x_{nds} \leq c_n^{max} \qquad \forall\, n \in N, d \in \{1, ..., |D| - c_n^{max}\} \quad (10)$$

$$\sum_{s \in S} x_{nds} + (i - \sum_{j=d+1}^{d+i} \sum_{s \in S} x_{njs}) + \sum_{s \in S} x_{n(d+i+1)s} > 0$$
$$\forall\, n \in N, i \in \{1, ..., c_n^{min} - 1\}, d \in \{1, ..., |D| - (i+1)\} \quad (11)$$

$$(1 - \sum_{s \in S} x_{nds}) + \sum_{j=d+1}^{d+i} \sum_{s \in S} x_{njs} + (1 - \sum_{s \in S} x_{n(d+i+1)s}) > 0$$
$$\forall\, n \in N, i \in \{1, ..., o_n^{min} - 1\}, d \in \{1, ..., |D| - (i+1)\} \quad (12)$$

Finally, the last group of constraints defines bounds for the decision variables.

$$x_{nds} \in \{0, 1\} \qquad \forall n \in N, d \in D, s \in S \quad (13)$$

$$k_{nw} \in \{0, 1\} \qquad \forall n \in N, w \in W \quad (14)$$

$$y_{ds} \geq 0 \qquad \forall d \in D, s \in S \quad (15)$$

$$z_{ds} \geq 0 \qquad \forall d \in D, s \in S \quad (16)$$

3 Constraint reformulation

A series of experiments was conducted to empirically verify the theoretical results from the academic literature. A set of problem instances was solved with different sets of time-related constraints in order to gain insight into which constraints influence the computational hardness of rostering problems. Table 1 reports the required calculation time in seconds for the full set of time-related constraints and details the effect of removing forbidden shift succession (Constraints (9)), restrictions on the maximum number of consecutive days worked (Constraints (10)) and restrictions on the minimum number of consecutive days worked and days off (Constraints (11)-(12)). The relative change in required computation time is shown in the columns denoted by Δ. Negative values indicate that computation time increased when removing the constraint(s). Gurobi 7.5.2 was used to solve the integer programming problems with a time limit of 7200 seconds on a server with two Intel Xeon E5-2670 processors and 128GB RAM.

Instance	All constraints	Without (9)		Without (10)		Without (11)-(12)	
	Time (s)	Time (s)	Δ	Time (s)	Δ	Time (s)	Δ
I01	0.2	0.2	-8%	0.1	22%	0.0	97%
I02	1.9	1.5	23%	0.0	99%	0.0	99%
I03	3.1	3.3	-7%	0.3	92%	0.1	98%
I04	22.8	29.4	-29%	0.2	99%	0.1	100%
I05	128.8	49.4	62%	0.4	100%	0.1	100%
I06	210.8	13.9	93%	1.0	100%	0.2	100%
I07	134.1	78.8	41%	3.6	97%	0.3	100%
I08	7200.0	1017.1	86%	237.2	97%	3.1	100%
I09	314.8	101.3	68%	153.1	51%	36.1	89%
I10	83.5	319.2	-282%	1.6	98%	0.6	99%
I11	32.5	17.5	46%	5.0	85%	1.0	97%
I12	613.9	174.3	72%	69.2	89%	2.6	100%
I13	7200.4	7200.3	0%	7200.3	0%	872.0	88%
I14	408.8	307.3	25%	7.8	98%	2.1	99%
I15	7200.6	5393.3	25%	2500.6	65%	14.8	100%
I16	7200.0	596.8	92%	1.0	100%	0.5	100%
I17	5453.5	2421.3	56%	4.5	100%	2.2	100%
I18	7200.1	7200.1	0%	2.9	100%	1.5	100%
I19	7200.0	7200.4	0%	69.6	99%	13.0	100%
I20	7200.2	7200.7	0%	61.9	99%	43.5	99%
I21	-	7200.1	-	7200.1	-	305.7	-
I22	-	-	-	1399.1	-	1355.5	-
I23	-	7200.4	-	-	-	7200.8	-
I24	-	-	-	-	-	-	-
Average	2890.5	1966.3	18%	516.0	84%	49.7	98%

Table 1: Required computation time for different sets of time-related constraints

Constraints on the number of consecutive assignments clearly have the biggest impact on the problem's computational hardness. Removing the constraints regarding the maximum number of consecutive days worked decreased

the required calculation time by 84%, while removing the minimum number of consecutive assignments constraints resulted in a decrease of 98%. Removing forbidden shift succession, on average, reduced computation time by only 18%, even increasing the required time for some instances. The proposed model therefore focuses on reformulating the constraints related to consecutive assignments in an efficient structure, namely as the problem of finding a feasible path in a directed graph. All other time-related constraints remain formulated using time-indexed decision variables which are linked to the feasible path decision variables using a set of linking constraints.

3.1 Graph representation of series constraints

For each nurse n, a directed layered graph $G_n = (V, A)$ is used to model Constraints (10)-(12) with V the set of nodes and A the set of arcs. Each layer in the graph represents a single day of the scheduling period and consists of two nodes for modeling a day worked and a day off which are called a work node and rest node, respectively. The set of nodes also contains a source and sink node.

A feasible path from the source node to the sink node, also referred to as an s-t path, corresponds to a set of assignments for nurse n which respect the constraints concerning the minimum and maximum number of consecutive days worked and days off. If the path contains an arc between a work node on day d and a rest node on day $d' > d$, it indicates that the nurse begins a stretch of days worked on d which ends on day $d'-1$. Similarly, an arc between a rest node on day d and a work node on day $d' > d$ indicates that the nurse had a continuous period of days off from day d until $d'-1$.

There are arcs between the work node on day d to all rest nodes on days $d' \in \{d+c_n^{min}, d+c_n^{max}\}$. If $d' > |D|$, the work node on day d is connected to the sink node, thereby indicating that the stretch of days worked starting at day d does not end within the current scheduling period. The rest node on day d is connected to the sink node and to all work nodes on days $d' \in \{d+o_n^{min}, |D|\}$. To complete the set of arcs, the source node is connected to the work and rest nodes on the first day of the scheduling period.

As an example, consider the graph shown in Figure 1. In this example, the scheduling period consists of seven days and the minimum and maximum number of consecutive days worked is set to three and four, respectively. The minimum number of consecutive days off is two. To simplify the presentation, only a subset of the arcs are drawn which model the stretches of days worked starting on days 1, 6 and 7, and the stretches of days off starting on days 4 and 5.

Note that the proposed graph configuration assumes that constraints on assignments near the boundaries of the current scheduling period are always satisfied by assignments in the preceding and next scheduling period. For constraints on the minimum number of consecutive assignments, it is assumed that assignments at the start of the current scheduling period continue an

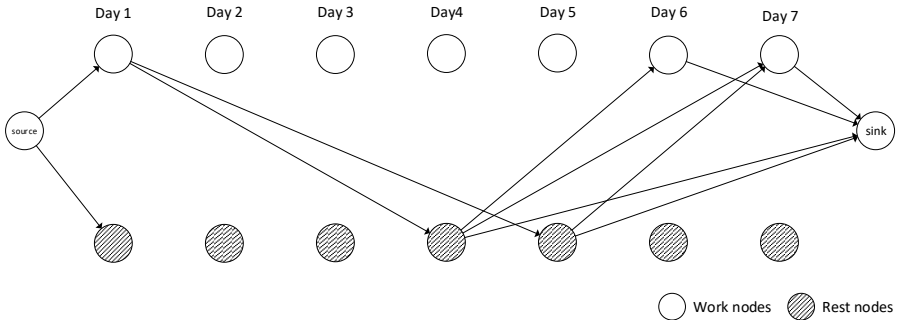


Fig. 1: Example of G_n which models restrictions on the total number of consecutive days worked and days off

existing stretch from the previous scheduling period. Similarly, at the end of the current scheduling period, it is assumed that the current stretch of assignments may be extended in the following scheduling period to ensure the minimum number of consecutive assignments is reached. For constraints on the maximum number of consecutive assignments, it is assumed that the current stretch is preceded or followed by at least one assignment which ends the current stretch. For example, the last stretch of days worked in the current scheduling may consist of c_n^{max} assignments since it is assumed that for this constraint, the first assignment in the next scheduling period is a day off. If the previous or following scheduling periods contain other assignments, the arcs in the first and last layers of the current graph should be modified appropriately.

3.2 Linking constraints

Finding an s-t path in G_n is modeled and solved as an integer programming problem using the decision variables w_{nij} which equal one if the arc from node i to node j is part of the path in graph G_n. Let $\delta^+(i)$ and $\delta^-(i)$ be the sets of incoming and outgoing arcs of node i. The subproblem for each nurse n is formulated as a linear programming problem. Constraints (17) ensure flow conservation while Constraints (18) define bounds on the decision variables.

$$\sum_{(i,j)\in\delta^+(i)} w_{nij} - \sum_{(j,i)\in\delta^-(i)} w_{nji} = \begin{cases} 1 & \text{if } i = \text{source} \\ -1 & \text{if } i = \text{sink} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in V \quad (17)$$

$$0 \leq w_{nij} \leq 1 \qquad \forall (i,j) \in A \quad (18)$$

Linking the w_{nij} variables to the main x_{nds} decision variables requires a set of additional constraints. Let A_d^{work} be the subset of arcs which represent a stretch of days worked that includes day d. This set is constructed by including

arc (i, j) if node i is a work node and the day associated with rest node j is greater than d. Similarly, let A_d^{rest} be the subset of arcs which represent a stretch of days off that includes day d. Constraints (19) and (20) now link the flow variables to the main assignment variables.

$$\sum_{s \in S} x_{nds} = \sum_{(i,j) \in A_d^{\text{work}}} w_{nij} \qquad \forall\, n \in N, d \in D \quad (19)$$

$$\sum_{s \in S} x_{nds} = 1 - \sum_{(i,j) \in A_d^{\text{rest}}} w_{nij} \qquad \forall\, n \in N, d \in D \quad (20)$$

4 Computational study

The proposed model is evaluated in a computational study which compares its performance against that of the standard time-indexed formulation and a branch and price approach from the literature [4].

4.1 Benchmark instances and experimental setup

Table 2 provides the characteristics of the used benchmark data set [3]. The number of decision variables and constraints is also reported for both the time-indexed formulation and the s-t path (FP) reformulation. For the latter, the average number of nodes and arcs in the graph G_n is also provided.

Instance	Weeks	Employees	Shifts	Time-indexed formulation		FP reformulation			
				Variables	Constraints	Variables	Constraints	Nodes	Arcs
I01	2	8	1	156	462	1300	662	30	143
I02	2	14	2	476	926	2574	1372	30	167
I03	2	20	3	964	2012	3874	2522	30	155
I04	4	10	2	712	1526	5572	1916	58	486
I05	4	16	2	1072	2552	8836	3032	58	484
I06	4	18	3	1752	3798	10568	4458	58	512
I07	4	20	3	1928	4154	11768	4944	58	512
I08	4	30	4	3704	8686	18504	9922	58	538
I09	4	36	4	4400	9856	22382	11884	58	538
I10	4	40	5	6040	15020	25480	16580	58	486
I11	4	50	6	8936	23208	33216	24918	58	484
I12	4	60	10	17600	67480	46740	69580	58	486
I13	4	120	18	61968	461414	120398	465984	58	512
I14	6	32	4	5904	13002	38890	14632	86	1065
I15	6	45	6	12114	37407	58299	39492	86	1065
I16	8	20	3	3856	8568	39056	9908	114	1760
I17	8	32	4	7872	19042	64088	21184	114	1708
I18	12	22	3	6312	14116	89988	16202	170	3738
I19	12	40	5	18120	46588	170196	50060	170	3818
I20	26	50	6	58084	154082	918709	163742	366	17018
I21	26	100	8	151112	582736	1872362	602056	366	17018
I22	52	50	10	191880	753240	3567780	771990	730	67518
I23	52	100	16	599248	4213414	7351033	4250924	730	67521
I24	52	150	32	1778296	25516528	11916796	25580048	730	67521

Table 2: Instance characteristics and model size

All experiments were performed on a server with two Intel Xeon E5-2670 processors and 128GB RAM. Gurobi 7.5.2 was used to solved the integer programming models configured to use 16 threads with a time limit of five hours.

4.2 Computational results

Table 3 reports the required computation time in addition to the lower and upper bounds on the objective value obtained by the different formulations. Three formulations are compared: the time-indexed formulation from Section 2.2, the *s*-*t* path reformulation from Section 3 and a formulation with an exponential number of variables which is solved using branch and price [4]. A dash (-) indicates that no feasible solution was found within the time limit, or that the available memory was exceeded. The best solution obtained for each instance is highlighted in bold.

	Time-indexed formulation			FP reformulation			Branch and price [4]		
	Time (s)	LB	UB	Time (s)	LB	UB	Time (s)	LB	UB
I01	0.2	607	**607**	0.5	607	**607**	0.3	558	**607**
I02	1.5	828	**828**	1.6	828	**828**	0.1	828	**828**
I03	3.0	1001	**1001**	1.4	1001	**1001**	0.5	1001	**1001**
I04	20.8	1716	**1716**	4.9	1716	**1716**	1.5	1716	**1716**
I05	117.9	1143	**1143**	8.0	1143	**1143**	25.6	1141	1160
I06	200.4	1950	**1950**	12.0	1950	**1950**	10.5	1949	1952
I07	99.4	1056	**1056**	43.8	1056	**1056**	93.7	1055	1058
I08	17616.5	1300	**1300**	2347.0	1300	**1300**	11831.1	1297	1308
I09	343.0	439	**439**	1010.5	439	**439**	77.0	406	**439**
I10	76.4	4631	**4631**	128.9	4631	**4631**	113.4	4631	**4631**
I11	30.2	3443	**3443**	67.3	3443	**3443**	19.1	3443	**3443**
I12	580.8	4040	**4040**	584.1	4040	**4040**	1336.4	4040	4046
I13	18000.3	1348	1658	18000.3	1346	**1652**	-	-	-
I14	756.7	1278	**1278**	744.7	1278	**1278**	-	-	-
I15	18000.1	3820	3858	18000.1	3820	**3853**	-	-	-
I16	8685.4	3225	**3225**	1689.1	3225	**3225**	265.0	3224	3323
I17	5573.0	5746	**5746**	1467.1	5746	**5746**	-	-	-
I18	18000.1	4404	**4459**	18000.2	4366	4460	-	-	-
I19	18000.1	3144	**3154**	18005.3	2946	3204	-	-	-
I20	18000.5	4765	**4769**	18002.7	4765	5078	-	-	-
I21	18000.1	21122	**261409**	-	-	-	-	-	-
I22	-	-	-	-	-	-	-	-	-
I23	-	-	-	-	-	-	-	-	-
I24	-	-	-	-	-	-	-	-	-

Table 3: Primary computational results. Best solutions are highlighted in bold.

With the time-indexed formulation, feasible solutions were found for 21 out of 24 instances, more than both the FP reformulation (20 out of 24) and the branch and price approach (13 out of 24). The reason for this is the increased memory consumption by both the FP reformulation and branch and price approach, which make them unsuitable for some of the larger problem instances.

Both the time-indexed formulation and the FP reformulation were able to prove optimality of 15 out of 24 instances[1]. Computation times vary widely between the instances. However, averaged over the first 20 instances, the FP reformulation required 4906 seconds, while the time-indexed formulation required 6205 seconds for the same subset of instances. Moreover, examining only those instances for which both the time-indexed formulation and the FP reformulation did not exceed the time limit, the time-indexed formulation required 2274 seconds while the FP reformulation required only 541 seconds. Overall, these results demonstrate that by embedding the constraints which were identified as computationally hard in an s-t path subproblem, the resulting integer programming problems may be solved in less computation time by a general purpose solver compared to the standard time-indexed formulation. This improved performance comes at the cost of increased model dimensions which makes the proposed reformulation currently not viable for large problem instances.

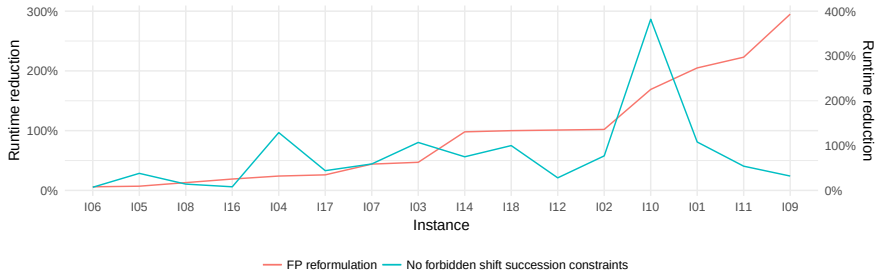
5 Analysis and discussion

As discussed in Section 4.2, the proposed s-t path reformulation does not outperform the time-indexed formulation for all instances. This section attempts to characterize those instances for which the reformulation results in a significant performance increase. This characterization is established empirically by analyzing the relation between the decrease in computation time for the FP reformulation and that of the relaxations of the time-related constraint set discussed in Table 1.

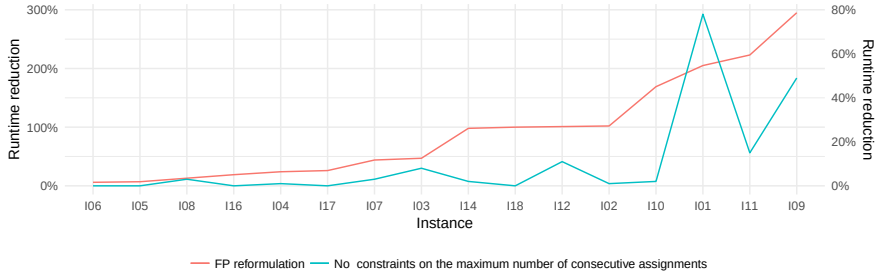
The decrease in computation time is quantified as a ratio $100 \times \frac{t_\alpha}{t_{\text{TI}}}$, with t_α the computation time of the considered approach (either the FP reformulation or one of the relaxations of the time-indexed forumulation), and t_{TI} the computation time of the time-indexed formulation including all time-related constraints. This is a non-negative relative indicator whose value is low when the reduction in computation time is large and vice versa. A value of 100% indicates that the proposed approach requires the same computation time as solving the time-indexed formulation with all time-related constraints. For the different relaxations, these values are plotted in Figure 2. Each graph also plots the reduction of the FP reformulation so as to enable a direct comparison against the reductions obtained by the relaxations. The horizontal axes of these graphs denote the different benchmark instances for which both approaches obtained the optimal solution within the time limit.

Figure 2a plots the reduction in computation time for the FP reformation and the relaxation of forbidden shift succession constraints. This graph does not reveal a strong correlation between the two lines, implying that the presence of the shift succession constraints are not directly related to the improved performance of the FP relaxation. However, such a correlation does exist in

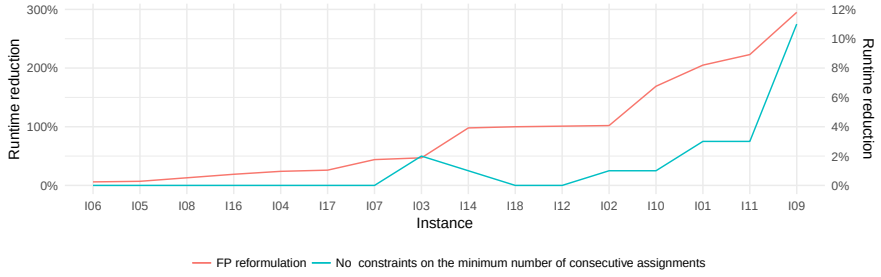
[1] All new best solutions are available at `http://www.cs.nott.ac.uk/~tec/NRP`



(a) No forbidden shift succession constraints



(b) No constraints on the maximum number of consecutive assignments



(c) No constraints on the minimum number of consecutive assignments

Fig. 2: Comparison of reductions in computation time for different models

the graphs of Figures 2b and 2c, which plot the relaxations of the maximum and minimum number of consecutive assignments. These observations are supported by the correlation coefficients of the different runtime reductions. For the data in Figure 2a, the Pearson's correlation coefficient is $r = 0.271$, for Figure 2b $r = 0.701$, and for Figure 2c $r = 0.804$. The results confirm the primary principle of the proposed reformulation: problem instances whose computational hardness is significantly affected by the presence of series constraints, benefit most from reformulating these constraints in an alternative structure.

6 Conclusions and future work

Theoretical and empirical studies have indicated that a particular class of time-related constraints, those concerning the number of consecutive assignments, have a significant influence on the computational complexity and hardness of nurse rostering problems. The present paper focused on these challenging constraints by embedding them in an s-t path subproblem. Integrating these subproblems in a larger integer programming formulation resulted in reduced computation time of a general purpose integer programming solver for small to medium-sized instances. The computational study produced optimal solutions for several previously unsolved benchmark instances using a standard solver. Further analysis of the results characterized problem instances for which the proposed reformulation results in the most significant performance gain as those whose computational hardness is most influenced by the presence of series constraints.

The proposed reformulation was unable to address large problem instances as the number of variables in the FP reformulation increased dramatically compared to the time-indexed formulation. The reach of this approach may be extended by using an exponential reformulation of a subset of the time-related constraints in which subproblem variables (or constraints) are added only when necessary.

Acknowledgements Editorial consultation provided by Luke Connolly (KU Leuven).

References

1. Brucker, P., Qu, R., Burke, E.K.: Personnel scheduling: Models and complexity. European Journal of Operational Research **210**(3), 467 – 473 (2011)
2. Brunner, J.O., Bard, J.F., Köhler, J.M.: Bounded flexibility in days-on and days-off scheduling. Naval Research Logistics **60**(8), 678–701 (2013)
3. Curtois, T.: Employee shift scheduling benchmark data sets (2017). Available online at `http://www.cs.nott.ac.uk/~tec/NRP`
4. Curtois, T., Qu, R.: Computational results on new staff scheduling benchmark instances. Tech. rep., University of Nottingham (2014)
5. Messelis, T., De Causmaecker, P., Vanden Berghe, G.: Algorithm performance prediction for nurse rostering. In: Proceedings of the 6th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2013), pp. 21–38 (2013)
6. Smet, P., Bilgin, B., De Causmaecker, P., Vanden Berghe, G.: Modelling and evaluation issues in nurse rostering. Annals of Operations Research **218**(1), 303–326 (2014)
7. Smet, P., Brucker, P., De Causmaecker, P., Vanden Berghe, G.: Polynomially solvable personnel rostering problems. European Journal of Operational Research **249**(1), 67 – 75 (2016)
8. Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., De Boeck, L.: Personnel scheduling: A literature review. European Journal of Operational Research **226**(3), 367–385 (2013)
9. Zhang, H., Jackson, L., Tako, A., Liu, J., Dunnett, S.: Network based formulations for roster scheduling problems. In: Proceedings of the 11th International Conference on Practice and Theory of Automated Timetabling (PATAT 2016), pp. 403–419 (2016)