Ann Oper Res
DOI 10.1007/s10479-014-1540-7

A flexible iterative improvement heuristic to support creation of feasible shift rosters in self-rostering

E. van der Veen · J. L. Hurink · J. M. J. Schutten · S. T. Uijland

© Springer Science+Business Media New York 2014

Abstract Self-rostering is receiving more and more attention in literature and in practice. With self-rostering, employees propose the schedule they prefer to work during a given planning horizon. However, these schedules often do not match with the staffing demand as specified by the organization. We present an approach to support creating feasible schedules that uses the schedules proposed by the employees as input and that aims to divide the burden of shift reassignments fairly throughout the employees. We discuss computational results and indicate how various model parameters influence scheduling performance indicators. The presented approach is flexible and easily extendable, since labor rule checks are isolated from the actual algorithm, which makes it easy to include additional labor rules in the approach. Moreover, our approach enables the user to make a trade-off between the quality of the resulting roster and the extent to which the planner is able to track the decisions of the algorithm.

keywords Self-rostering · Linear programming · Heuristics · Personnel rostering · Shift rostering

E. van der Veen (✉) · S. T. Uijland
ORTEC, Houtsingel 5, 2719 EA Zoetermeer, The Netherlands
e-mail: egbert.vanderveen@ortec.com

E. van der Veen
Center for Healthcare Operations, Improvement, and Research (CHOIR), University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

J. L. Hurink
Department of Applied Mathematics, University of Twente, P.O. Box 217, Enschede, The Netherlands

J. M. J. Schutten · S. T. Uijland
Department of Management and Governance, University of Twente, P.O. Box 217, Enschede, The Netherlands

Published online: 05 February 2014

1 Introduction

In service industries, such as healthcare and security services, shifts have to be staffed around the clock. Considering the many employee preferences and labor legislation that are implied on schedules from such circumstances, it is often hard to come up with good or fair shift schedules, see Brooks and Swailes (2002). Self-rostering is a way to better cope with employee preferences leading also to an increased job satisfaction and an improved employee commitment and cooperation, see Hung (1992), Kellogg and Walczak (2007), Robb et al. (2003).

Several self-rostering processes exist in practice. The basic structure of these processes is that employees propose a schedule by indicating for each day in the schedule the shift they prefer to work, or whether they would like to have a day-off. These proposed schedules have to comply with labor legislation and meet contract hours. The organization now evaluates the proposed schedules and has to ensure that sufficient employees are assigned to each shift. If the joint schedules of the employees do not meet these bounds, feedback information is provided to the employees. Based on this information, employees may choose to update their proposed schedules, for example by trading shifts with other employees. This process leads to updated schedules that hopefully fulfill the demands. However, it may happen that some shifts are still insufficiently staffed. In this case, the department manager or a human shift planner has to decide which shifts to reassign.

The method proposed in this paper is concerned with the final planning phase. Our method operates independently of the actual self-rostering process applied by the organization. The input used by our method consists of a set of proposed schedules and a shift demand, indicating for each shift the minimum number of employees that have to be assigned to this shift. Given this input, understaffed and overstaffed shifts are identified, whereby we refer to a shift as *understaffed* if less employees have signed up for this shift than specified by the staffing demand and *overstaffed* if the opposite holds. The method now has to resolve this unbalance by reassigning shifts, taking into account several constraints and criteria.

The criteria used within our method are derived from case studies from practice. The goal of our method is to minimize the total understaffing, while satisfying labor legislation. This is accomplished using 'shift swaps'. Shift swaps are defined by an unassignment of a shift of some employee and an assignment of another shift to the same employee. Essentially, our method iteratively selects combinations of shift swaps in order to reduce the total understaffing as much as possible. By including only shift swaps in the method that satisfy labor legislation we ensure that our method does not violate labor legislation. Next to minimizing the total understaffing and satisfying labor legislation, the considered case studies indicate that the method has to be transparent, meaning that the planner should be able to understand the method's decisions. Besides this, a specified fraction of each schedule proposed by the employees has to be retained, since otherwise employees get discouraged to participate in the self-rostering process. These requirements are accomplished in our method by proper choices of parameter values.

Shift roster instances as provided by the case studies are used to evaluate the proposed method. In total, 72 instances were used. We analyze the influence of various parameters on the shift rosters produced by the method. The main outcome is the observed trade-off between the total understaffing on the one hand and the number of schedule changes on the other hand.

Our contribution is twofold. First, the designed method is flexible and easily extendable, since the checking of labor legislation is an isolated component in our approach. Labor

legislation is checked in the method that defines the allowed shift swaps, but not in the optimization method that selects the swaps to be applied. Second, the iterative nature of our method helps planners to understand the decisions made by the algorithm in each iteration, since only a limited number of shift reassignments are performed in each iteration. This is especially the case if in each iteration only one or only a few employees are allowed to get a changed schedule.

This paper is organized as follows. Section 2 discusses related literature. Then, Sect. 3 provides a problem description and outlines our principal solution approach. Next, Sect. 4 presents the mathematical implementation of our solution approach and Sect. 5 discusses results on the instances derived from the case studies. The paper closes with conclusions and discussion in Sect. 6.

2 Literature review

Personnel rostering [see, e.g., the review by Ernst et al. (2004); Van den Bergh et al. (2013)] and nurse rostering in particular [see the review by Burke et al. (2004)] are well-studied problems. Classically, nurse rostering problems are modeled using hard and soft constraints. Hard constraints express strict rules that the schedules must satisfy, such as labor legislation and the number of employees that have to be staffed on each shift. Soft constraints express rostering rules that preferably should be satisfied.

Employee preferences, such as requests for a day-off or to work some specific shift on some specific day, are often quantified using soft constraints. Optimizing the rosters of individual employees, while satisfying hard rostering rules is referred to as *preference rostering*. Examples are presented by, e.g., Bard and Purnomo (2005) and Purnomo and Bard (2007). De Grano et al. (2009) propose an auction model, where employees bid on shifts and rest days using 'points'. Shifts are awarded to employees based on their bids. A mathematical program is used to check for feasibility and awarding shifts.

Another way of dealing with employee preferences is self-rostering. Qualitative aspects of self-rostering are discussed in, e.g., Bailyn et al. (2007) and Teahan (1998). Self-rostering starts with the organization specifying the staffing demand, i.e., specifying per day how many employees have to be staffed on each shift. After that, employees propose their preferred schedules. For each day in the planning horizon, the employees choose the shift they prefer, or whether they like to have a day-off. This proposed schedule has to satisfy labor legislation and other rostering constraints defined by the organization. The staffing resulting from the proposed schedules of all employees is compared to the staffing demand identifying understaffed and overstaffed shifts. In most self-rostering processes, feedback information is provided to the employees. Based on some incentive, such as 'scores' for shifts, employees may choose to change some of the shifts in their schedules, leading hopefully to a decrease in the total understaffing. In some self-rostering processes, employees can negotiate about changing shifts, which Wang and Wang (2009) model as a multi-agent model in which employees are represented by agents. For the remaining understaffed shifts, mostly a human planner decides who works these shifts, since the planner has the final responsibility to create schedules without understaffed shifts.

The main difference between preference rostering and self-rostering is that the proposed schedules of the employees in self-rostering have to satisfy labor legislation, while in preference rostering a schedule that satisfies all preferences specified by the employees does not necessarily imply that this schedule is allowed by labor legislation.

Algorithms that assist the human planner in deciding which employee works which understaffed shift are proposed in Ásgeirsson (2012), Petrovic et al. (2003), and Rönnberg and Larsson (2010). Ásgeirsson (2012) proposes a method that first unassigns overstaffed shifts. This is done in an iterative way by first selecting an overstaffed shift, based on some priority rule, and then selecting an employee to be unassigned from this shift, again based on some priority rule. When this process is finished, understaffed shifts are assigned to employees in an analogous fashion. Next, several shift swapping and shift reassignment steps are applied, which are similar to the shift reassignment steps considered in an analogous approach of Petrovic et al. (2003). Whereas the algorithms proposed by Ásgeirsson (2012) and Petrovic et al. (2003) select at most 2 shifts at a time, the method proposed by Rönnberg and Larsson (2010) considers the complete roster. In addition to proposing schedules, employees prioritize between shifts by specifying either a weak request, a strong request, or a veto for each shift. Based on the proposed schedules and the requests, a mathematical program is used to create a feasible roster. The mathematical program reduces the total understaffing while balancing the number of preferences honored per employee.

The objective of re-rostering problems as studied by Moz and Pato (2004, 2007), Clark and Walker (2011), and Maenhout and Vanhoucke (2011), is also to reassign shifts in order to find a schedule without understaffed shifts. Re-rostering is needed if shifts get understaffed due to unexpected absences, for example due to illness. The heuristics proposed by Moz and Pato (2004) and the genetic algorithm proposed by Moz and Pato (2007) try to find a schedule without understaffed shifts while minimizing the number of shift swaps. Clark and Walker (2011) proposes a mathematical programming approach that minimizes a weighted sum of the total overstaffing and the total understaffing. Maenhout and Vanhoucke (2011) propose a genetic algorithm that also considers shift preferences and distributes the workload after reconstructing the roster.

This paper proposes an iterative method to assist the planner in reducing the total understaffing. We propose an iterative method, as opposed to Rönnberg and Larsson (2010), such that planners can evaluate intermediate results. However, in our method, multiple shift reassignments are allowed in a single iteration, which leads to a more globalized optimization, as opposed to the iterative method proposed by Ásgeirsson (2012) that assigns or unassigns a single shift in every iteration. Moreover, as opposed to other self-rostering and re-rostering literature, the method proposed in this paper guarantees that some given fraction of each employee's schedule is preserved by the algorithm.

3 Problem description and principal approach

In this section, we give a detailed problem description and discuss the principal ideas underlying our approach.

3.1 Problem description

We are given a schedule period with a set of shifts K and a set of employees I. For each shift $k \in K$, a demand d_k is specified that indicates the minimum number of employees that have to be assigned to that shift. Note that a specific shift (e.g., a morning shift) on different days is represented by separate shifts in the set K.

For every employee $i \in I$, we are given a proposed schedule S_i. This schedule specifies, for each day in the planning horizon, either the shift the employee prefers to work or that

the employee prefers to have a day off. We assume that the proposed schedules satisfy all hard constraints implied on the schedules, e.g., labor legislation, and that the overall contract hours of the employee over the planning horizon are fulfilled.

Based on the proposed schedules, the difference between the specified staffing demand d_k and the actual staffing can be calculated for each shift $k \in K$. The differences are denoted by a difference parameter v_k, which indicates how far the proposed schedules are away from the preferred situation. We define v_k to be positive for understaffed shifts and negative for overstaffed shifts. We introduce scores σ_k for each shift $k \in K$, which are given as a function of v_k, i.e., $\sigma_k = f(v_k)$. We define f to be increasing in the value of v_k. So, for overstaffed shifts, σ_k is smaller than for understaffed shifts. Using the scores σ_k, we calculate for each employee $i \in I$ a score s_i by summing up the scores of all shifts the employee has in his schedule S_i. An employee with a relatively high score s_i has chosen relatively many understaffed shifts and is thus doing well for the organization. For employees with relatively low scores s_i, the opposite holds.

The goal now is to change the schedules S_i of the employees such that the total understaffing and the total overstaffing, as expressed by v_k, is reduced. Hereby, we have to ensure that the new schedules satisfy labor legislation and contract hours. Moreover, an employee's schedule should not change too much, since otherwise the employee might get discouraged to propose schedules in the future. In addition, we have to ensure that the burden of shift reassignments is divided fairly throughout the employees.

First, in the next section, we present the principal ideas underlying our approach. A formal specifications of our method is given in Sect. 4.

3.2 Principal approach

To reduce the total understaffing and the total overstaffing, we apply 'swaps' in the proposed schedules. A swap changes an employee's schedule by unassigning one of his shifts and assigning this employee to some other shift, whereby these shifts do not have to be on the same day. To reduce the total understaffing and the total overstaffing, a swap unassigns an overstaffed shift and assigns an understaffed shift. This means, that we do not allow that an overstaffed shift becomes understaffed or that an understaffed shift becomes overstaffed. The reason for this is, that such changes lead to smaller fractions of the proposed schedules of the employees to be retained, but do not contribute directly to the objective of reducing the total understaffing and the total overstaffing. For the same reason we also exclude swaps from understaffed shifts to understaffed shifts and from overstaffed shifts to overstaffed shifts. To make sure that the final schedule satisfies labor legislation, we only allow swaps that comply with labor legislation. Moreover, strict unavailabilities of employees are incorporated in the model by excluding swaps that assign shifts to a day where an employee is strictly unavailable.

The proposed solution approach iteratively selects combinations of swaps, where in each iteration at most one swap per employee is selected. Since each swap satisfies labor legislation and labor legislations are defined per employee and not for groups of employees, this implies that all employee schedules satisfy labor legislation after applying swaps. The reason that we restrict to at most one swap per employee per iteration is that a combination of two swaps that individually are allowed by labor legislation, may not necessarily lead to a feasible schedule. For example, if it is allowed to work at most five shifts consecutively, additional checks are required to make sure that a combination of two swaps satisfies this constraint. Since, in general, labor legislation does not imply constraints on the relation between schedules of different employees, we may apply swaps at

multiple employees simultaneously without having to include labor legislation checks in the swap selection method. Therefore, we consider a subset of employees for whom we simultaneously apply swaps in an iteration. The maximum size of this subset is an input parameter of the approach. Moreover, employees can explicitly be excluded from this subset. For example, employees that already have received a certain number of shift swaps may be excluded from receiving additional shift swaps.

This approach has two advantages. First, note that labor legislation only influences the selection of possible swaps, but does not interfere with finding a good combination of swaps. In this way, labor legislation is an isolated component in our approach. This component may be changed without the need to adapt the other components. Second, the size of the subset of employees that is selected in an iteration defines a trade-off between transparency of the approach on the one hand and larger improvements in the schedule on the other hand. Applying a single swap for one employee makes the decisions made by the model easier to understand, whereas applying swaps for multiple employees simultaneously leads, in general, to larger improvements in the schedules.

Applying multiple swaps at one employee in one iteration makes decisions made by the model harder to understand for the planner and implies that we have to include labor legislation checks in the mathematical model. As both of these are undesirable consequences, in our model we apply at most one swap per employee in a single iteration.

4 Realization of the approach

To reduce the total understaffing and the total overstaffing, we apply an iterative method, in which in every iteration a mathematical program is solved to select a combination of swaps to apply. First, we discuss the mathematical programming formulation, after which we present the details of our iterative approach.

4.1 Swap selection model

Given is a set of schedules $\{\bar{S}_1, \ldots, \bar{S}_{|I|}\}$ at the start of the current iteration and a vector of shift demands $(d_1, \ldots, d_{|K|})$. For each employee $i \in I$, a score $\bar{s}_i$ is calculated by summing the scores σ_k of the shifts k that are in the current schedule $\bar{S}_i$ of employee i. Note that the scores σ_k only depend on the value of v_k and are not updated during the iterative method.

For the swap selection, we introduce sets J_i, which contain all possible swaps for the employees $i \in I$. Each swap $j \in J_i$ is allowed by labor legislation and unassigns an overstaffed shift and assigns an understaffed shift. For each swap $j \in J_i$, a decision variable x_j is introduced which denotes whether this swap j is selected for employee i ($x_j = 1$) or not ($x_j = 0$), i.e.:

$$x_j \in \{0, 1\} \quad j \in J_i;\ i \in I. \tag{1a}$$

As outlined in Sect. 3.2, in every iteration, at most one swap is selected per employee. This is enforced by the constraint:

$$\sum_{j \in J_i} x_j \leq 1 \quad i \in I. \tag{1b}$$

Also, we do not allow that an overstaffed shift becomes understaffed or the other way around. For this, we introduce the set $K^O \subseteq K$ that denotes the set of overstaffed shifts, i.e.:

$$K^O = \{k \in K | v_k \leq 0\}$$

and the set $K^U \subseteq K$ that denotes the set of understaffed shifts, i.e.:

$$K^U = \{k \in K | v_k \geq 0\}.$$

Next to the sets K^O and K^U, we introduce a variable n_k, which denotes the difference between the staffing demand and the actual staffing *after* swaps have been applied. Hence, n_k is positive if shift k is understaffed after swaps have been applied and negative if k is overstaffed after swaps have been applied. Let the parameter $\bar{v}_k$ denote the difference between the staffing demand and the actual staffing at the start of the current iteration. Let $U_i^k \subseteq J_i$ contain all swaps that unassign shift k from employee i, and let $A_i^k \subseteq J_i$ contain the swaps that assign shift k to employee i. Using the sets U_i^k and A_i^k, we can calculate n_k as follows:

$$n_k = \bar{v}_k + \sum_{i \in I} \left(\sum_{j \in U_i^k} x_j - \sum_{j \in A_i^k} x_j \right) \quad k \in K. \tag{1c}$$

Now, the following constraints ensure that overstaffed shifts do not become understaffed:

$$n_k \leq 0 \quad k \in K^O \tag{1d}$$

and that understaffed shifts do not become overstaffed:

$$n_k \geq 0 \quad k \in K^U. \tag{1e}$$

To evaluate the quality of a solution, we consider an objective function that has two components. The first component is responsible for minimizing the total understaffing. The second component specifies that swaps should be applied in the schedules of employees that have a low score $\bar{s}_i$. The employees that have low scores $\bar{s}_i$ at the start of the iteration are the employees that have relatively many overstaffed shifts in their current schedule. Combining these two components leads to the following objective:

$$\min \lambda_1 \sum_{k \in K^U} n_k + \lambda_2 \sum_{i \in I} \bar{s}_i \sum_{j \in J_i} x_j. \tag{1f}$$

Note that the second component makes the swap selection more fair, since $\bar{s}_i$ influences the swap selection, meaning that employees with a low score are more likely to get a shift reassignment. Also note that if the total overstaffing is smaller than the total understaffing the first component cannot achieve the value 0. The parameters λ_1 and λ_2 are used to define the relative importance of the components.

The resulting model (1a–1f) is an integer linear program (ILP) that can be solved using standard solvers.

4.2 Solution approach

To reduce the total understaffing and the total overstaffing, we apply an iterative solution approach. Each iteration consists of three phases. First, a subset of employees $I' \subseteq I$ is selected. Second, for the employees in I', we determine the set of possible swaps J_i, and third, we select a combination of swaps to be applied using the ILP model presented in Sect. 4.1.

Shift swaps are applied only to the schedules of the employees $i \in I'$. As outlined in Sect. 3.2, the number of employees to include in the set I' is a model parameter. In I', we include the employees with the smallest scores $\bar{s}_i$.The scores $\bar{s}_i$ are calculated as explained in Sect. 4.1. Hence this selection is influenced by the scores σ_k that are determined by the function f. If an employee's score $\bar{s}_i$ is low, this employee has relatively many overstaffed shifts, which is undesirable from an organizational point of view. Note that the scores $\bar{s}_i$ are updated after swaps are applied. Furthermore, employees for which applying a swap implies that less than the specified fraction of their schedule is retained, are excluded from I'.

Next, for each employee $i \in I'$ the set of allowed swaps J_i is determined. As outlined in Sect. 3.2, we only allow swaps from overstaffed to understaffed shifts. This assumption is relaxed later on in Sect. 4.3.

Given I' and the corresponding sets J_i, we use the ILP model (1) to determine the best combination of swaps. Note that, in every iteration, the sets K^O and K^U are based on the *current* roster.

4.3 Extensions and discussion

In addition to swaps that unassign overstaffed shifts and assign understaffed shifts, to which we refer as *primary swaps*, we optionally extend the approach with *secondary swaps*. A secondary swap is performed at two employees. The first employee is unassigned from an overstaffed shift and assigned to a shift for which staffing demand is *exactly* matched; we refer to such a shift as a *matching* shift. The second employee is unassigned from the same matching shift and assigned to an understaffed shift. Note that secondary swaps explicitly assign from and to matching shifts, since otherwise we could have applied a primary swap that leads to the same reduction in overstaffed shifts, but retains a larger fraction of the employees' proposed schedules. Secondary swaps make it possible to reduce the total overstaffing and understaffing in situations where primary swaps are not able to do so.

To include secondary swaps in the method, swaps from overstaffed shifts to matching shifts and swaps from matching shifts to understaffed shifts are included in the sets J_i. Let the set $K^M \subseteq K$ be the set of matching shifts, hence $K^M = K^O \cap K^U$. Constraints (1d) and (1e) ensure that n_k remains 0 for $k \in K^M$. Note that secondary swaps are a kind of ejection chains (with a length of 2). If we would also include swaps from matching shifts to matching shifts in the sets J_i, we would allow the model to select even larger 'swap-chains'.

Finally, it is interesting to include shift pattern preferences. For example, employees often do not prefer to have isolated working days, i.e., a single shift with days off on either side. Violations of these preferences and the corresponding penalty weights can be calculated when creating swaps. In the objective function, a trade-off between the minimization of understaffed shifts and violated preferences can then be made. However, how this trade-off should be done is not straightforward and we leave this for future research.

5 Case studies and results

This section studies the effects of various input parameters on the shift rosters produced by our approach. For this, we apply the developed approach to shift roster instances based on

data from practice. In Sect. 5.1, we describe some important criteria for the solution approach that were stated by these organizations. After that, Sect. 5.2 describes some characteristics of the provided shift rosters and Sect. 5.3 analyzes the performance of our approach on these rosters.

5.1 Criteria from practice

We have asked a general hospital, a center for forensic psychiatry, a transportation company, and a service company to indicate criteria that a solution approach has to meet. This resulted in the following set of criteria:

1. Minimize the total understaffing.
2. Do not violate labor legislation.
3. At least 80 % of each proposed shift roster must be preserved.
4. Decisions made by the approach have to be transparent.
5. The computation time has to be short, i.e., at most a couple of minutes.

Criterion 1, 2, and 5, have straightforward explanations. Criterion 3 might imply that understaffed shifts remain that could have been solved if a smaller percentage than 80 % would have been permitted. However, the organizations indicate that if an insufficient part of the proposed schedules is preserved, employees are discouraged to propose their preferred shift rosters in the future. The organizations indicate that 80 % is a suitable threshold. Note that, in our experimental results we evaluate the effect of decreasing this percentage to 70 %. Criterion 4 implies that the algorithm should be such that planners are able to understand the decisions made by the algorithm and if necessary to explain them to managers or employees. The first helps to stimulate the implementation success; the organizations expect that planners will more easily accept a system that they understand. The second may help to convince the employees to accept the system.

All these criteria can be handled in the proposed approach. Criterion 1 coincides with the objective (1f) of the mathematical program, which is used in the iterations, if we set the values of λ_1 and λ_2 such that the first component of objective (1f) dominates the second component. Furthermore, Criterion 2 can be satisfied by only including swaps that do not violate labor legislation. Criterion 3 can be realized by excluding employees from I' for which an additional swap would imply that less than 80 % of their schedules is preserved. Next to this, Criterion 4 may be handled by including only one or a few employees in I' and Criterion 5 asks for setting a time limit on the computation time.

5.2 Experimental setup

We are provided with final shift rosters as executed by two departments of the general hospital and by a department of the center for forensic psychiatry. In total, these organizations provided 36 shift rosters with a one-month planning horizon and between 14 and 79 employees. The resulting shift occupancy determines the shift demand used in the computational experiments. Since the executed shift rosters obviously match this demand exactly, we apply randomizations to the executed shift rosters to create shift rosters that can be used to evaluate our approach. Randomization is applied in two ways: to the executed shift rosters, and to the shift demand. Details of this generation process can be found in Uijland (2012).

Looking at the overstaffed and understaffed shifts, the randomization leads to two classes of shift rosters:

1. [Demand randomization] Weekly alternating overstaffed and understaffed shifts. In these rosters, the following pattern is repeated every two weeks: the first week contains overstaffed shifts, but no understaffed shifts. The second week contains understaffed shifts, but no overstaffed shifts. On average, these rosters contain a total understaffing of 30.1 shifts.
2. [Roster randomization] Overstaffed and understaffed shifts in each week of the roster. On average, these rosters contain a total understaffing of 92.0 shifts.

Hence, these randomizations imply two totally different ways of distributing the understaffed and overstaffed shifts throughout the shift rosters, which enables us to evaluate whether our method is able to effectively handle these different classes of shift rosters. To analyze the results and assess the quality of the rosters produced by our approach, we consider two performance indicators: the remaining total understaffing and the average fraction of the rosters of the employees that is retained.

For the organization, it is important that the total understaffing is minimized, since understaffed shifts imply that an insufficient number of employees are staffed on these shifts. If $\sum_{k\in K^U} n_k > \sum_{k\in K^O} |n_k|$, we correct the remaining total understaffing by the difference, since in this case this difference is a lower bound to the minimum $\sum_{k\in K^U} n_k$ and thereby, after this correction, the natural lower bound on the total understaffing is 0. For the two classes of shift rosters that we consider, the mentioned total understaffing is the corrected total understaffing that, in theory, can thus be reduced to 0.

For the employees, it is important that the fraction of their rosters that is retained is as high as possible, since this implies that they get to work a large fraction of the schedule they proposed. We calculate this fraction as the number of days in the schedule of the employee where the assignment is unchanged (including days-off) divided by the length of the planning horizon (in days).

5.3 Experimental results

We investigate the influence of the following five parameters on the results achieved by the proposed method: swap strategy, constraints on the minimum remaining fraction, the size of I', the values of λ_1 and λ_2, and the function $f(v_k)$. For a detailed results analysis we refer the reader to Uijland (2012). These results indicate that the latter three parameters have no noticeable influence on the performance indicators for reasonable choices of the parameters. The values we use for $|I'|$ are $|I'| = 1$ and $|I'| = 3$. For (λ_1, λ_2), we compare $(\lambda_1, \lambda_2) = (1, 0)$, which implies that the objective only considers minimizing the total understaffing, with $(\lambda_1, \lambda_2) = (1, \varepsilon)$ for some small $\varepsilon > 0$, which implies that the minimization of the total understaffing dominates the objective function, but that this minimization is guided by the scores $\bar{s}_i$ of the employees. For the score function $f(v_k)$ we compared:

$$f(v_k) = \begin{cases} 3 & \text{if } v_k > 0 \\ 2 & \text{if } v_k = 0 \\ 1 & \text{if } v_k < 0 \end{cases}, \tag{2}$$

with:

$$f(v_k) = \frac{v_k + d_k}{d_k} = \frac{\text{``employees that have chosen } k\text{''}}{\text{staffing demand shift } k}. \tag{3}$$

In this section, we focus our analysis on the effect of the minimum remaining percentage and the swap strategy. For this, each of the 72 randomized rosters is solved for each combination of parameter values. Hence, each roster is solved $2^5 = 32$ times. We have made these instances available for others to challenge our results, see Self-Rostering Instances (2013).

We start by investigating the influence of secondary swaps. For this, we apply our approach with and without secondary swaps. Next, we check the influence of the constraint on the fraction of the schedule that has to be retained. The organizations indicated that this fraction equals 0.8, which we are going to compare with a fraction of 0.7.

First, we look at results for the first class of shift rosters, where overstaffed and understaffed shifts occur in alternating weeks. Table 1 shows the effects of the bound on the minimum remaining fraction and the swap strategy on the average remaining total understaffing (ARU) and the average remaining fraction (ARF) of the rosters. Moreover, average and maximum computation times are shown in Table 1. These averages were calculated from the various shift roster instances that were solved using the various parameter values. From Table 1, we observe that in all cases the ARU is reduced from 30.1 shifts to less than 1 shift. Furthermore, the influence of secondary swaps on the achieved results is larger than that of the bound on the minimum remaining fraction: using secondary swaps reduces the ARU by almost 0.6 shifts compared to only using primary swaps, whereas reducing the bound from 0.8 to 0.7 leads only to an additional reduction of around 0.1 understaffed shifts. Furthermore, note that if the constraint on the remaining fraction is relaxed to 0.7, the resulting ARF is still far above 0.8 (around 93.5 %). However, there are a few employees that now retain less than 80 % of their schedules; on average this holds for less than 2 % of the employees. We see that the average and maximum computation times are small and thus satisfy Criterion 5. In the appendix, Tables 3, 4 and 5 present detailed results per instance.

For the second class of shift rosters, where overstaffed and understaffed shifts occur in the same week, we summarize the corresponding results in Table 2. Table 2 shows that the total understaffing is again reduced heavily: from on average 92 shifts to less than 1.5 shifts. For this class of shift rosters, the bound on the minimum remaining fraction has a larger influence on the achieved results than secondary swaps have: the ARU is reduced by over 1.1 shift if the bound is relaxed from 0.8 to 0.7, whereas using secondary swaps in addition to primary swaps leads only to an additional reduction of around 0.15 shifts. Furthermore, if the constraint on the remaining fraction is set to 0.7, the ARF is again still far above 0.8 (around 89.1 %). However, in that case some employees retain less than 80 % of their schedule; on average this was the case for about 9.5 % of the employees. Again, we see that the average and maximum computation times are small and thus meet Criterion 5. In the appendix, Tables 6, 7 and 8 present detailed results for this class of roster instances.

In general, we observe that the method performs well. For both classes of shift rosters, the remaining total understaffing is small, while on average 89 % or more of each employee proposed schedule is retained. For the first class of shift rosters, secondary swaps have the largest influence on the key performance indicators: the remaining total understaffing and the average fraction of each proposed schedule that is retained. We think the effect of secondary swaps is larger here, since overstaffing and understaffing do not occur in the same week. Then, swaps have to be applied that unassign a shift in one week, and assign a shift in another. Since labor legislation implies restrictions on, for example, the number of shifts in a week and the number of consecutive shifts, secondary swaps offer a source of flexibility that turns out to be useful. To the contrary, for the second class of shift rosters, the bound on the minimum remaining fraction has the largest influence, which is caused by the larger understaffing in

Table 1 Results for first class of shift rosters (demand randomization)

Min. remaining	0.8				0.7			
	ARU	ARF (%)	Time (s)		ARU	ARF (%)	Time (s)	
			Avg	Max			Avg	Max
Only primary swaps	0.85	94.0	1.6	4.7	0.72	93.9	1.6	4.8
Prim. & sec. swaps	0.25	93.6	1.7	5.2	0.14	93.5	1.7	5.7

Table 2 Results for second class of shift rosters (roster randomization)

Min. remaining	0.8				0.7			
	ARU	ARF (%)	Time (s)		ARU	ARF (%)	Time (s)	
			Avg	Max			Avg	Max
Only primary swaps	1.49	89.7	3.4	9.7	0.30	89.1	3.3	11.5
Prim. & sec. swaps	1.31	89.6	3.4	10.2	0.19	89.0	3.3	11.9

these rosters. Also swaps that swap shifts on the same day are very often allowed by labor legislation, which implies a smaller need for secondary swaps.

6 Conclusions

In this paper, we considered a problem occurring within self-rostering processes. In particular, we studied how to reduce the total understaffing and overstaffing resulting from rosters proposed by employees. For this, we designed an iterative solution approach. In each iteration, the total understaffing is reduced by selecting shift swaps, i.e., an unassignment of an overstaffed shift and an assignment of an understaffed shift to one and the same employee. The swaps are selected using a mathematical programming model. Our method is flexible in that labor legislation is an isolated component that can be adapted without having to change the mathematical programming model. Moreover, the decisions made in our approach can be made very easy to trace and understand by the planner and the employees. We interviewed four organizations to determine important criteria for our self-rostering method. These organizations stated that the method has to incorporate labor legislation, be understandable, and enforce that at least 80 % of an employee proposed schedule is preserved.

We applied our approach to 72 shift rosters with a one-month planning horizon, which are based on shift rosters provided by two organizations. These 72 shift rosters, which we subdivided into two classes, contain on average a total understaffing of 30.1 and 92 shifts, respectively. On average, our method reduces this understaffing to less than 0.5 and 0.9 shifts, respectively. The remaining total understaffing is mainly affected by two model parameters: the swap strategy and the constraint on the minimum fraction of each roster that must be retained. If overstaffed shifts and understaffed shifts occur in alternating weeks, we observe that the total understaffing decreases by more than 1 shift if we include so-called secondary swaps in our swap strategy. However, if both overstaffed shifts and understaffed shifts occur in each week of the roster, the minimum remaining fraction has the largest influence on the remaining total understaffing.

For further research, we suggest to study metaheuristic approaches, since the greedy local search character of our solution approach may cause that we block possible future swaps. Note, however, that this influences the transparency of the method, which was an important criterion in the design. From a practical point of view, it is interesting to incorporate specific employee preferences that are not necessarily honored by the proposed approach, such as 'work 8 hour shifts'.

Acknowledgements This research is supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs. The authors thank the anonymous referees for their valuable comments that have helped to improve the paper.

Appendix

Detailed results

See Tables 3, 4, 5, 6, 7, and 8.

Table 3 Demand randomization—Case Forensic Psychiatry

Instance	Swaps	Min. remaining = 80 %				Min. remaining = 70 %			
		ARU	ARF	Time (s)		ARU	ARF	Time (s)	
				Avg	Max			Time	Max
1	Prim.	2.50	92.6	0.8	0.9	2.75	92.7	0.7	0.9
	Prim.&sec.	1.63	91.7	0.8	1.2	1.75	91.7	0.8	0.9
2	Prim.	2.25	90.2	1.0	1.6	2.25	90.2	0.9	1.1
	Prim.&sec.	0.50	88.5	1.2	1.6	0.38	88.4	1.1	1.5
3	Prim.	1.75	90.0	0.8	1.1	1.75	90.0	0.8	0.9
	Prim.&sec.	0.38	88.5	1.0	1.2	0.38	88.6	1.0	1.3
4	Prim.	1.00	89.0	0.9	1.3	0.75	88.9	0.9	1.2
	Prim.&sec.	0.00	88.0	0.9	1.1	0.00	88.1	0.9	1.1
5	Prim.	0.13	90.4	0.7	1.0	0.13	90.4	0.6	0.8
	Prim.&sec.	0.00	90.2	0.6	0.9	0.00	90.2	0.6	0.8
6	Prim.	0.38	89.3	0.9	1.1	0.25	89.1	0.8	1.1
	Prim.&sec.	0.00	88.9	0.9	1.5	0.00	88.8	1.0	1.2
7	Prim.	2.13	90.2	0.7	1.1	1.13	89.6	0.8	1.2
	Prim.&sec.	1.63	89.7	0.9	1.2	0.38	88.9	1.0	1.2
8	Prim.	3.25	88.9	0.9	1.1	2.63	88.6	1.0	1.6
	Prim.&sec.	1.38	87.1	1.1	1.4	0.75	87.0	1.1	1.5
9	Prim.	0.13	92.4	0.6	1.1	0.13	92.4	0.7	1.1
	Prim.&sec.	0.00	92.4	0.7	1.0	0.00	92.3	0.8	1.4
10	Prim.	2.13	90.2	0.8	0.9	1.00	89.6	0.9	1.0
	Prim.&sec.	0.25	88.3	1.0	1.3	0.00	88.6	1.0	1.2
11	Prim.	3.50	90.6	0.9	1.1	2.75	90.2	0.9	1.0
	Prim.&sec.	1.63	88.9	1.1	1.4	0.63	88.2	1.2	1.5
12	Prim.	2.63	89.6	1.0	1.5	2.38	89.5	0.9	1.2
	Prim.&sec.	0.75	88.1	1.2	1.3	0.50	87.9	1.1	1.3

Table 4 Demand randomization—General Hospital Department 1

Instance	Swaps	Min. remaining = 80 %				Min. remaining = 70 %			
		ARU	ARF	Time (s)		ARU	ARF	Time (s)	
				Avg	Max			Time	Max
1	Prim.	0.13	95.8	0.9	1.5	0.13	95.8	1.1	2.4
	Prim.&sec.	0.00	95.8	0.9	2.0	0.00	95.8	1.0	2.1
2	Prim.	0.13	95.4	1.2	1.9	0.13	95.4	1.1	1.7
	Prim.&sec.	0.00	95.4	1.2	1.9	0.00	95.4	1.2	2.1
3	Prim.	0.63	94.5	1.8	2.2	0.25	94.4	1.7	2.3
	Prim.&sec.	0.00	94.3	1.9	2.5	0.00	94.4	1.9	2.7
4	Prim.	0.38	94.3	1.7	2.5	0.38	94.3	1.7	2.7
	Prim.&sec.	0.00	94.2	1.7	2.7	0.00	94.3	1.8	2.7
5	Prim.	0.25	94.4	2.0	2.5	0.25	94.4	2.0	2.4
	Prim.&sec.	0.00	94.3	2.1	2.9	0.00	94.3	2.1	2.7
6	Prim.	1.88	95.3	2.4	2.7	1.63	95.2	2.4	2.7
	Prim.&sec.	0.25	94.8	2.7	3.1	0.13	94.8	2.7	3.1
7	Prim.	0.00	96.0	0.9	1.2	0.00	96.0	1.0	1.2
	Prim.&sec.	0.00	96.0	0.9	1.2	0.00	96.0	0.9	1.3
8	Prim.	1.13	95.5	2.7	3.8	1.13	95.5	2.6	3.4
	Prim.&sec.	0.00	95.3	3.0	4.2	0.00	95.3	2.9	3.8
9	Prim.	0.00	96.3	1.0	1.3	0.00	96.3	1.0	1.3
	Prim.&sec.	0.00	96.3	1.0	1.3	0.00	96.3	0.9	1.3
10	Prim.	1.00	96.0	2.3	2.5	0.88	96.0	2.2	2.4
	Prim.&sec.	0.13	95.7	2.5	3.1	0.00	95.7	2.5	2.8
11	Prim.	0.63	95.8	2.4	2.9	0.63	95.8	2.4	3.1
	Prim.&sec.	0.00	95.6	2.7	3.5	0.00	95.6	2.6	3.7
12	Prim.	0.00	95.1	2.2	2.7	0.00	95.1	2.1	2.6
	Prim.&sec.	0.00	95.1	2.1	2.7	0.00	95.1	2.1	2.7

Table 5 Demand randomization—General Hospital Department 2

Instance	Swaps	Min. remaining = 80 %				Min. remaining = 70 %			
		ARU	ARF	Time (s)		ARU	ARF	Time (s)	
				Avg	Max			Time	Max
1	Prim.	0.13	95.6	1.9	3.6	0.13	95.6	2.3	4.1
	Prim.&sec.	0.00	95.6	2.0	4.0	0.00	95.6	2.2	4.3
2	Prim.	0.00	96.6	1.2	1.6	0.00	96.6	1.1	1.4
	Prim.&sec.	0.00	96.6	1.2	1.5	0.00	96.6	1.1	1.4
3	Prim.	0.00	96.4	2.2	2.7	0.00	96.4	2.2	2.6
	Prim.&sec.	0.00	96.4	2.2	2.7	0.00	96.4	2.3	2.6
4	Prim.	0.13	96.5	1.3	2.8	0.13	96.5	1.4	2.8
	Prim.&sec.	0.00	96.5	1.4	3.6	0.00	96.5	1.5	3.8

Table 5 continued

Instance	Swaps	Min. remaining = 80 %				Min. remaining = 70 %			
		ARU	ARF	Time (s)		ARU	ARF	Time (s)	
				Avg	Max			Time	Max
5	Prim.	0.00	95.4	2.3	2.9	0.00	95.4	2.3	3.0
	Prim.&sec.	0.00	95.4	2.2	3.0	0.00	95.4	2.2	2.9
6	Prim.	0.00	96.6	1.8	2.4	0.00	96.6	1.9	2.4
	Prim.&sec.	0.00	96.6	1.7	2.2	0.00	96.6	1.8	2.2
7	Prim.	0.00	96.8	1.1	1.6	0.00	96.8	1.0	1.5
	Prim.&sec.	0.00	96.8	1.1	1.8	0.00	96.8	1.1	1.7
8	Prim.	0.63	96.6	2.9	3.8	0.38	96.6	2.8	3.8
	Prim.&sec.	0.13	96.5	3.1	4.1	0.00	96.5	2.9	4.3
9	Prim.	0.00	96.4	1.3	1.7	0.00	96.4	1.2	1.7
	Prim.&sec.	0.00	96.4	1.4	1.8	0.00	96.4	1.3	1.8
10	Prim.	0.25	96.2	3.1	4.5	0.25	96.2	3.1	4.8
	Prim.&sec.	0.00	96.1	3.3	4.9	0.00	96.1	3.3	5.1
11	Prim.	1.00	96.4	4.0	4.7	1.00	96.4	3.9	4.7
	Prim.&sec.	0.13	96.3	4.5	5.2	0.13	96.3	4.5	5.7
12	Prim.	0.75	96.8	3.5	4.5	0.75	96.8	3.6	4.4
	Prim.&sec.	0.13	96.7	3.8	5.0	0.13	96.7	3.7	4.7

Table 6 Roster randomization—Forensic Psychiatry

Instance	Swaps	Min. remaining = 80 %				Min. remaining = 70 %			
		ARU	ARF	Time (s)		ARU	ARF	Time (s)	
				Avg	Max			Time	Max
1	Prim	1.88	82.5	1.3	1.6	1.25	82.2	1.4	1.7
	Prim.&sec.	1.63	82.2	1.4	1.9	1.00	82.0	1.7	2.0
2	Prim.	10.25	81.0	1.5	2.0	1.38	76.3	1.8	2.6
	Prim.&sec.	10.13	80.9	1.5	2.0	0.63	75.5	1.9	2.3
3	Prim.	1.75	83.8	1.2	1.5	0.00	82.7	1.0	1.4
	Prim.&sec.	1.38	83.4	1.2	1.6	0.00	82.7	1.0	1.4
4	Prim.	0.00	90.4	0.5	0.7	0.00	90.4	0.5	0.7
	Prim.&sec.	0.00	90.4	0.5	0.7	0.00	90.4	0.6	0.8
5	Prim.	0.50	84.9	1.1	1.3	0.00	84.6	0.9	1.2
	Prim.&sec.	0.13	84.6	1.2	1.5	0.00	84.6	0.9	1.2
6	Prim.	5.13	83.7	1.3	1.7	0.38	80.8	1.5	1.9
	Prim.&sec.	4.00	82.8	1.4	1.9	0.00	80.5	1.7	2.3
7	Prim.	1.50	85.0	1.1	1.5	0.00	84.3	1.0	1.4
	Prim.&sec.	1.38	84.9	1.2	1.6	0.00	84.3	1.1	1.5
8	Prim.	0.88	84.1	1.3	1.8	0.00	83.6	1.0	1.4
	Prim.&sec.	0.75	84.1	1.2	1.4	0.00	83.6	1.1	1.5

Table 6 continued

Instance	Swaps	Min. remaining = 80 %				Min. remaining = 70 %			
		ARU	ARF	Time (s)		ARU	ARF	Time (s)	
				Avg	Max			Time	Max
9	Prim.	5.88	83.7	1.8	2.3	0.13	80.8	2.0	2.5
	Prim.&sec.	5.38	83.3	1.8	2.1	0.00	80.7	2.0	2.5
10	Prim.	11.38	82.9	1.7	2.0	4.75	78.9	1.9	2.5
	Prim.&sec.	11.38	82.9	1.7	2.2	4.13	78.3	2.0	2.4
11	Prim.	0.50	85.0	1.4	1.8	0.00	84.4	1.2	2.0
	Prim.&sec.	0.50	85.0	1.4	1.6	0.00	84.4	1.1	1.4
12	Prim.	1.00	84.2	1.3	1.7	0.13	83.7	1.2	1.7
	Prim.&sec.	0.50	83.8	1.4	1.9	0.00	83.5	1.2	1.7

Table 7 Roster randomization—Case General Hospital Department 1

Instance	Swaps	Min. remaining = 80 %				Min. remaining = 70 %			
		ARU	ARF	Time (s)		ARU	ARF	Time (s)	
				Avg	Max			Time	Max
1	Prim.	5.25	87.5	5.2	6.4	1.75	86.8	7.1	8.0
	Prim.&sec.	4.00	87.1	5.5	7.3	0.38	86.4	7.2	8.1
2	Prim.	0.00	97.9	0.8	1.0	0.00	97.9	0.8	1.1
	Prim.&sec.	0.00	97.9	0.7	1.0	0.00	97.9	0.8	1.2
3	Prim.	0.25	90.8	3.5	3.9	0.00	90.7	3.4	4.1
	Prim.&sec.	0.25	90.8	3.5	4.0	0.00	90.7	3.5	4.1
4	Prim.	0.00	92.4	3.8	4.9	0.00	92.4	4.0	5.6
	Prim.&sec.	0.00	92.4	3.8	4.9	0.00	92.4	3.9	5.1
5	Prim.	3.50	89.1	8.4	9.7	0.75	88.6	9.9	11.5
	Prim.&sec.	3.50	89.1	8.5	10.2	0.75	88.6	10.3	11.9
6	Prim.	0.00	90.3	3.7	4.2	0.00	90.2	3.7	4.2
	Prim.&sec.	0.00	90.3	3.8	4.4	0.00	90.2	3.6	4.2
7	Prim.	0.00	92.3	3.7	4.7	0.00	92.2	3.7	4.7
	Prim.&sec.	0.00	92.3	3.8	4.6	0.00	92.2	3.6	4.6
8	Prim.	0.00	91.2	4.9	5.2	0.00	91.2	4.2	4.8
	Prim.&sec.	0.00	91.2	5.0	5.4	0.00	91.2	4.1	4.5
9	Prim.	0.00	94.5	2.2	2.9	0.00	94.5	2.2	3.1
	Prim.&sec.	0.00	94.5	2.2	3.1	0.00	94.5	2.3	3.0
10	Prim.	0.00	92.2	3.9	5.4	0.00	92.2	3.7	4.9
	Prim.&sec.	0.00	92.2	3.8	5.2	0.00	92.2	3.6	5.0
11	Prim.	0.00	92.8	4.0	5.2	0.00	92.8	3.6	4.6
	Prim.&sec.	0.00	92.8	3.9	5.2	0.00	92.8	3.5	4.6
12	Prim.	0.00	91.9	3.5	4.6	0.00	91.8	3.4	4.5
	Prim.&sec.	0.00	91.9	3.6	4.7	0.00	91.8	3.3	4.4

Table 8 Roster randomization - General Hospital Department 2

Instance	Swaps	Min. remaining = 80 %				Min. remaining = 70 %			
		ARU	ARF	Time (s)		ARU	ARF	Time (s)	
				Avg	Max			Time	Max
1	Prim.	0.00	97.1	1.6	1.8	0.00	97.1	1.6	2.0
	Prim.&sec.	0.00	97.1	1.6	2.1	0.00	97.1	1.6	2.0
2	Prim.	0.00	93.5	4.2	5.3	0.00	93.4	4.3	5.4
	Prim.&sec.	0.00	93.5	4.2	5.1	0.00	93.4	4.3	5.6
3	Prim.	0.00	97.3	1.4	1.8	0.00	97.3	1.5	1.8
	Prim.&sec.	0.00	97.3	1.5	1.7	0.00	97.3	1.5	1.9
4	Prim.	0.00	90.3	6.1	7.4	0.00	90.3	5.4	6.6
	Prim.&sec.	0.00	90.3	6.2	7.7	0.00	90.3	5.3	6.1
5	Prim.	0.75	90.8	6.8	9.1	0.00	90.7	5.7	7.0
	Prim.&sec.	0.25	90.7	7.1	10.0	0.00	90.7	5.6	7.0
6	Prim.	0.00	92.9	4.1	4.7	0.00	92.8	3.8	4.8
	Prim.&sec.	0.00	92.9	4.1	4.9	0.00	92.8	3.9	4.8
7	Prim.	3.38	91.1	8.1	8.5	0.25	90.5	7.5	9.6
	Prim.&sec.	2.13	90.8	8.7	9.2	0.00	90.5	7.5	10.2
8	Prim.	0.00	91.8	5.1	6.1	0.00	91.7	4.3	5.8
	Prim.&sec.	0.00	91.8	5.0	6.0	0.00	91.7	4.4	6.3
9	Prim.	0.00	91.6	5.6	6.3	0.00	91.5	5.5	6.9
	Prim.&sec.	0.00	91.6	5.6	6.5	0.00	91.5	5.5	6.6
10	Prim.	0.00	90.0	6.9	8.0	0.00	90.0	6.1	7.4
	Prim.&sec.	0.00	90.0	7.1	8.1	0.00	90.0	6.2	7.5
11	Prim.	0.00	94.0	4.9	6.1	0.00	94.0	4.7	5.3
	Prim.&sec.	0.00	94.0	4.9	6.2	0.00	94.0	4.7	5.8
12	Prim.	0.00	94.0	3.5	4.7	0.00	94.0	3.6	4.7
	Prim.&sec.	0.00	94.0	3.5	4.7	0.00	94.0	3.6	4.8

References

Ásgeirsson, E. (2012). Bridging the gap between self schedules and feasible schedules in staff scheduling. *Annals of Operations Research (to appear)*. doi:10.1007/s10479-012-1060-2

Bailyn, L., Collins, R., & Song, Y. (2007). Self-scheduling for hospital nurses: An attempt and its difficulties. *Journal of Nursing Management, 15*(1), 72–77.

Bard, J. F., & Purnomo, H. W. (2005). Preference scheduling for nurses using column generation. *European Journal of Operational Research, 164*(2), 510–534.

Brooks, I., & Swailes, S. (2002). Analysis of the relationship between nurse influences over flexible working and commitment to nursing. *Journal of Advanced Nursing, 38*(2), 117–126.

Burke, E., de Causmaecker, P., vanden Berghe, G., & van Landeghem, H. (2004). The state of the art of nurse rostering. *Journal of Scheduling, 7*(6), 441–499.

Clark, A. R., & Walker, H. (2011). Nurse rescheduling with shift preferences and minimal disruption. *Journal of Applied Operational Research, 3*(3), 148–162.

De Grano, M. L., Medeiros, D., & Eitel, D. (2009). Accommodating individual preferences in nurse scheduling via auctions and optimization. *Health Care Management Science, 12*(3), 228–242.

Ernst, A., Jiang, H., Krishnamoorthy, M., Owens, B., & Sier, D. (2004). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research, 127*(1), 21–144.

Hung, R. (1992). Improving productivity and quality through workforce scheduling. *Industrial Management, 34*(6), 4.

Kellogg, D., & Walczak, S. (2007). Nurse scheduling: From academia to implementation or not? *Interfaces, 37*(4), 355–369.

Maenhout, B., & Vanhoucke, M. (2011). An evolutionary approach for the nurse rerostering problem. *Computers & Operations Research, 38*(10), 1400–1411.

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*(1), 179–197.

Moz, M., & Pato, M. V. (2007). A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research, 34*(3), 667–691.

Petrovic, S., Beddoe, G., & Vanden Berghe, G. (2003). Storing and adapting repair experiences in employee rostering. In E Burke & P Causmaecker (Eds.), *Practice and theory of automated timetabling IV lecture notes in computer science, vol 2740* (pp. 148–165). Berlin: Springer.

Purnomo, H. W., & Bard, J. F. (2007). Cyclic preference scheduling for nurses using branch and price. *Naval Research Logistics (NRL), 54*(2), 200–220.

Robb, E. A., Determan, A. C., Lampat, L. R., Scherbring, M. J., Slifka, R. M., & Smith, N. A. (2003). Strategies at work: Self-scheduling: Satisfaction guaranteed?. *Nursing Management, 34*(7), 16–18.

Rönnberg, E., & Larsson, T. (2010). Automating the self-scheduling process of nurses in swedish healthcare: A pilot study. *Health Care Management Science, 13*(1), 35–53.

Self-Rostering Instances. (2013). Retrieved September, 2013 from http://www.utwente.nl/mb/iebis/staff/schutten/SelfRostering/Instances.zip.

Teahan, B. (1998). Implementation of a self-scheduling system: A solution to more than just schedules. *Journal of Nursing Management 6*, (6):361–368

Uijland, S. (2012). Creating feasible schedules in the last step of the self rostering process. http://essay.utwente.nl/62123/. Accessed Feb 2014.

Vanden Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, Z. L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research, 226*(3), 367–385.

Wang Z., & Wang C. (2009). Automating nurse self-rostering: A multiagent systems model. In *IEEE international conference on systems, man and cybernetics*, (pp. 4422–4425)