Memorandum 1974 (January 2012). ISSN 1874-4850. Available from: http://www.math.utwente.nl/publications
Department of Applied Mathematics, University of Twente, Enschede, The Netherlands

# Shift rostering using decomposition: assign weekend shifts first

Egbert van der Veen *[1,2], Erwin W. Hans[2], Gerhard Post[1,2], and Bart Veltman[1,2]

[1]ORTEC, Gouda, The Netherlands
[2]Center for Healthcare Operations, Improvement, and Research (CHOIR), University of Twente, Enschede, The Netherlands

January 24, 2012

**Abstract**

This paper introduces a shift rostering problem that surprisingly has not been studied in literature: the weekend shift rostering problem. It is motivated by our experience that employees' shift preferences predominantly focus on the weekends, since many social activities happen during weekends. The Weekend Rostering Problem (WRP) addresses the rostering of weekend shifts, for which we design a problem specific heuristic. We consider the WRP as the first phase of the shift rostering problem. To complete the shift roster, the second phase assigns the weekday shifts using an existing algorithm. We discuss effects of this two-phase approach both on the weekend shift roster and on the roster as a whole. We demonstrate that our first-phase heuristic is effective both on generated instances and real-life instances. For situations where the weekend shift roster is one of the key determinants of the quality of the complete roster, our two-phase approach shows to be effective when incorporated in a commercially implemented algorithm.

**Keywords:** Shift rostering, weekend shift rostering, decision support software, heuristics, decomposition

## 1 Introduction

For many people most social activities are scheduled during the weekend, which makes working in the weekend less attractive for them. Various industries, like healthcare and security services, offer services on a 24/7 basis, which implies that certain employees need to work during weekends. Rostering employees during weekends is challenging, since most employers like to consider the preferences of each individual on the one hand, but on the other hand also staffing demands need to be covered. Furthermore, rosters have to respect labor legislation, and shifts have to be distributed in an equitable way among employees, which complicates the matter even further.

*corresponding author: Egbert.vanderVeen@ortec.com

A significant amount of literature is devoted to personnel rostering, see, e.g., the comprehensive review by Ernst et al. [2004]. Most of the early literature (1950s-1970s) focuses on either finding feasible shift rosters under a set of (hard) constraints or minimizing the number of employees needed to cover a given set of shifts. In other words, the focus is on employers' needs. More recent literature (1980s-2000s) additionally considers employee preferences. The models in this literature try to balance or align the goals of employers and employees. Literature reviews by Burke et al. [2004b], Cheang et al. [2003], and Kellogg and Walczak [2007] show that, within this literature, there has been a strong focus on nurse rostering problems. Nurse rostering characterizes itself by employees working 24/7 and taking into account many employee preferences, e.g., about which weekends employees prefer not to work. The literature review by Kellogg and Walczak [2007] particularly focuses on the implementation of theoretical models and algorithms in practice, and shows that there is still a considerable gap between theory and practice.

Existing literature recognizes the importance of finding good or fair assignments of weekend shifts to employees; some papers propose (soft) constraints to cope with preferences related to weekends shifts, see, e.g., Burke et al. [2004b]. Still, to our knowledge, literature does not consider methods specifically designed to construct weekend shifts rosters. Of course, weekend shift rosters can be created using a general rostering method. However, this approach ignores problem specific information. Therefore, we design a weekend shift rostering algorithm that is tailored to use this information.

This research was motivated by practical experiences of customers of ORTEC, the shift rostering software supplier that employs three of this paper's authors. Many planners, when assigning shifts manually, decompose the shift rostering process into two or more steps. They first assign weekend shifts, and secondly they assign the weekday shifts. This shows that many planners consider weekend-related shift rostering preferences as more important than other preferences. We use the same assumption in this paper. Moreover, we apply the same decomposition to assign the weekend and weekday shifts. Of course, other decomposition approaches might be valuable in practice, like first assigning night shifts or first assigning days-off. Section 3.2 discusses decomposition approaches found in shift rostering literature.

In Section 2 we formally introduce the Weekend Rostering Problem (WRP), which we consider as a first phase of the shift rostering problem. A problem specific heuristic for the WRP is designed in Section 3. To complete the roster, in a second phase the weekday shifts are assigned using a commercially implemented algorithm, see Burke et al. [2008], Post and Veltman [2004]. In Section 4 we show the first-phase heuristic to be effective both on artificially generated instances and real-life instances. In addition, Section 4 discusses the effects of our two-phase approach on the weekend shift roster, as well as on the complete roster. Section 5 presents conclusions and discussion.

# 2 Problem assumptions and formulation

This section defines the Weekend Rostering Problem (WRP), discusses the assumptions of the WRP and addresses the constraints that are considered in the WRP. In the WRP we are only interested in assigning weekend shifts, i.e., shifts

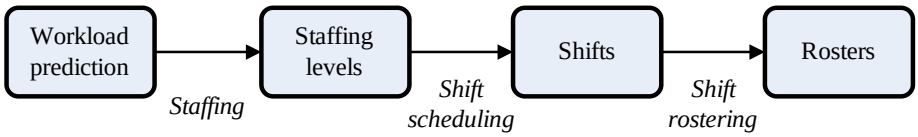


Figure 2.1: The scheduling process

that overlap with weekends. These are the shifts on Saturday and Sunday, and may include (late) Friday and (early) Monday shifts.

As indicated in Section 1, we consider the WRP as a first phase of the shift rostering problem. In general, shift rostering problems assign shifts to employees. Shifts are time periods specifying working time as opposed to rest time. Furthermore, demand constraints define the number of times each shift should be assigned. Demand constraints are the outcome of the *shift scheduling* phase, which precedes the *shift rostering* phase. Shift scheduling designs shifts to efficiently cover staffing levels, which in turn are the outcome of the *staffing* phase. Based on the predicted workload, staffing levels are determined, indicating how many employees need to be present on any specific time and day. For an overview of this process, see Figure 2.1.

The objective of the WRP is to assign as many weekend shifts as possible, while satisfying a set of hard constraints, and subsequently minimizing the total penalty cost associated with violations of a set of soft constraints. We will now discuss the hard and soft constraints that are used in the WRP.

In practice, many hard constraints are implied on rosters by labor legislation and labor agreements. The following four hard constraints are relevant for the weekend rostering problem, and considered in most shift rostering literature.

**Constraint C1** (Availability)**.** Employees may not work if they are unavailable. Employee are unavailable if they are on leave, for example, or assigned to another shift.

**Constraint C2** (Skills)**.** Shift may only be assigned to employees that have all skills required to work the shift.

**Constraint C3** (Rest between shifts)**.** Between two subsequent shifts there should be a rest period of a given minimum length.

The next constraint implies restrictions on the number of working weekends in a given period, see Burke et al. [2004a], Burke et al. [2008], Burns and Carter [1985], Burns and Koop [1987], Burns et al. [1998], Emmons and Fuh [1997], Hung [1994a], Jaumard et al. [1998], Koop [1986], Miller et al. [1976], and Wright et al. [2006]:

**Constraint C4** (Weekends in $p$ weeks)**.** Employees may only work $k$ out of $p$ weekends, where both $k$ and $p$ may be employee dependent.

Note that constraints on the maximum number of consecutive working weekends $m$, that are considered by Burke et al. [2004a], Jaumard et al. [1998], and Miller et al. [1976], can also be implied via hard constraint C4: working at most $m$ weekends consecutively is the same as working at most $m$ out of $m+1$ weekends.

Next to these hard constraints, the WRP also considers eight soft constraints. This list of soft constraints is supposed to be a complete list of soft constraints that can be implied on the assignment of weekend shifts.

The first soft constraint concerns the scheduling of 'complete' off weekends. Working only Saturday or only Sunday is unattractive for two reasons. First, there are labor rules regarding the number of times employees work in weekends (during a specified period), see hard constraint C4. Hence, working either on both Saturday and Sunday, or not at all, enables organizations to assign more weekend shifts to employees in total. Second, employee preferences are either to work the complete weekend or to have the complete weekend off. We encounter this in the real-life instances of Section 4.2 on which we test the WRP algorithm designed in Section 3.3, and in Berrada et al. [1996], Brucker et al. [2010], Burke et al. [2001], Burke et al. [2004a], Burke et al. [2008], Burns and Carter [1985], Burns and Koop [1987], Gärtner et al. [2001], Ikegami and Niwa [2003], Jaumard et al. [1998], Koop [1986], Miller et al. [1976], Musliu et al. [2002], and Sodhi and Norris [2004]. Therefore, the following soft constraint is used to assign complete weekends off:

**Soft constraint S1** (Complete weekends off)**.** If an employee is off one day of the weekend, the employee should preferably be off the complete weekend.

The next two soft constraints concern organizational preferences.

**Soft constraint S2.** Weekend shifts should be equitably distributed among employees, i.e., proportional to the contract hours of the employees.

**Soft constraint S3.** Weekend shift types, like, e.g., early, late, night, should be equitably distributed among employees, see Brucker et al. [2010], and Burke et al. [2004a]

The next four soft constraints concern employee preferences.

**Soft constraint S4.** Employee preference to (not) work during a specific weekend.

**Soft constraint S5.** Employee preference to (not) work a specific shift on a specific day in a specific weekend.

**Soft constraint S6.** Employee preference to (not) work two specific shifts consecutively, see Brucker et al. [2010], and Burke et al. [2004a].

**Soft constraint S7.** Employee preference to work at most $k$ weekends in $p$ weeks, see Brucker et al. [2010], and Warner [1976].

Note that soft constraint S7 is the equivalent of hard constraint C4.

Each of the soft constraints has a (user definable) penalty cost associated with the violation of the constraint. The penalty cost may differ per employee, and constraints can be added to a problem instance multiple times with different parameter and penalty values. If a shift assignment violates multiple soft constraints penalty costs are incurred for every violation.

Now that we have stated, described, and motivated the constraints used in the WRP, the next section introduces a model to solve the WRP.

# 3 Solution approach and Modeling

## 3.1 Introduction

To assign the weekday *and* weekend shifts, i.e., to solve the shift rostering problem, we apply a decomposition approach. The first phase, the Weekend Rostering Problem (WRP), assigns the weekend shifts. The second phase completes the roster by assigning the weekday shifts. For the second phase we use a hybrid heuristic ordering method as used in commercial software, see Burke et al. [2008], Post and Veltman [2004], which is not discussed in detail in this paper. With the WRP, we introduce a new decomposition approach for shift rostering problems. In Section 3.2, we discuss the use of other decomposition approaches to solve shift rostering problems. In Section 3.3, we explain how we solve the WRP.

## 3.2 Decomposition in shift rostering

This section discusses some decomposition approaches to solve the shift rostering problem. As a first example, De Causmaecker and Vanden Berghe [2003] proposes decomposition on skills. In a metaheuristic framework the shift roster is iteratively optimized per skill category. Another decomposition example is days-off rostering, studied by, e.g., Burns and Carter [1985], Elshafei and Alfares [2008], Emmons and Burns [1991], and Hung [1994b]. Days-off rostering first decides when employees should work, and next, in a reduced solution space, shifts are assigned. The algorithms used by Aickelin and Dowsland [2000], and Dowsland and Thompson [2000] first decide on day and night assignments, and afterwards assign day to early and late. A third example is the use of shift patterns. Shift patterns specify a sequence of shifts that are worked consecutively. In the methods proposed by Aickelin and Dowsland [2000], Dowsland and Thompson [2000], and Ikegami and Niwa [2003] shift patterns must be created manually, after which they are assigned by scheduling algorithms. The algorithms of Brucker et al. [2010], and Burke et al. [2009] first generate these shift patterns and then assign them to employees. The idea is shift patterns already comply with many constraints and preferences on consecutive shifts, which makes the assignment easier and more efficient.

## 3.3 The weekend rostering problem

The WRP is solved using a heuristic solution approach. A solution is constructed with a greedy 3-step heuristic. Step 1, described in Section 3.3.1, creates 'weekend shift combinations': combinations of Saturday and Sunday shifts for one particular weekend. These are possibly extended with Friday night and Monday morning shifts, dependent on whether these shifts are considered weekend shifts in the particular problem instance. Step 2, described in Section 3.3.2, assigns the weekend shift combinations to employees. The first two steps of our heuristic are illustrated with an example in Section 3.3.3. Finally, in Step 3, described in Section 3.3.4, we apply local search to improve the assignment of Step 2.

### 3.3.1 Weekend shift combinations

We create shift combinations per weekend, by solving a minimum-cost transportation problem. The idea is to choose shift combinations, such that as many shifts as possible can be assigned.

Let $I$ and $J$ be sets of nodes corresponding to the different types of shifts on Saturday and Sunday, respectively. For $i \in I$, let $s_i$ be the number of Saturday shifts of type $i$ that must be covered, and, for $j \in J$, let $d_j$ be the number of shifts of type $j$ that must be covered. Let $N$ be the set of employees, and let $N^{ij} \subset N$ denote the subset of employees that are allowed to work combination $(i, j)$. An employee is not allowed to work combination $(i, j)$ if working it would violate one of the hard constraints of Section 2. Let $c_{ij}$ denote the transportation cost from node $i$ to node $j$. To define $c_{ij}$ we distinguish three cases: either 'many', 'few' or no employees are allowed to work combination $(i, j)$. First, if no employees are allowed to work combination $(i, j)$, we do not want it to be selected. Second, if 'few' employees are allowed to work combination $(i, j)$, the combination may be selected only when it is necessary, i.e., when otherwise there is no solution to the transportation problem. Third, if 'many' employees are allowed to work combination $(i, j)$ we want to relate $c_{ij}$ to the penalty cost associated with violating the soft constraints.

If no employee is allowed to work combination $(i, j)$ we set $c_{ij} = \infty$, hence:

$$c_{ij} = \infty \text{ if } N^{ij} = \emptyset. \tag{3.1}$$

However, when $|N^{ij}| > 0$, we want to let $c_{ij}$ depend on the size of $N^{ij}$. If only 'few' employees are allowed to work combination $(i, j)$ we want $c_{ij}$ to be high. We say that 'few' employees are allowed to work combination $(i, j)$ if:

$$\frac{|N^{ij}|}{|N|} < \frac{\min\{s_i, d_j\}}{\max\{\sum_{i \in I} s_i, \sum_{j \in J} d_j\}}. \tag{3.2}$$

Here, the numerator in the fraction on the right hand side denotes the maximum number of times combination $(i, j)$ can be assigned, the denominator denotes the minimum number of employees that need to work during the particular weekend. If the right hand side in (3.2) is larger than the left hand side, which denotes the fraction of employees that are allowed to work combination $(i, j)$, we say that 'few' employees are allowed to work combination $(i, j)$. In this case we define $c_{ij}$ as $M \cdot |N \setminus N^{ij}|$, where $M$ is a 'big' number. Hence:

$$c_{ij} = M \cdot |N \setminus N^{ij}| \text{ if } \frac{|N^{ij}|}{|N|} < \frac{\min\{s_i, d_j\}}{\max\{\sum_{i \in I} s_i, \sum_{j \in J} d_j\}}. \tag{3.3}$$

If 'many' employees are allowed to work shift combination $(i, j)$, i.e., if (3.2) does not hold, we relate $c_{ij}$ to the penalty costs associated with the soft constraints. Let $S^n_{ij}$ denote the penalty cost associated with the soft constraints of assigning combination $(i, j)$ to employee $n$. Then:

$$c_{ij} = \sum_{n \in N} S^n_{ij} \text{ if } \frac{|N^{ij}|}{|N|} \geq \frac{\min\{s_i, d_j\}}{\max\{\sum_{i \in I} s_i, \sum_{j \in J} d_j\}}. \tag{3.4}$$

Summarizing we have:

$$c_{ij} = \begin{cases} \sum_{n \in N} S_{ij}^n & \text{if } \frac{|N^{ij}|}{|N|} \geq \frac{\min\{s_i, d_j\}}{\max\{\sum_{i \in I} s_i, \sum_{j \in J} d_j\}} \\ M \cdot |N \setminus N^{ij}| & \text{if } \frac{|N^{ij}|}{|N|} < \frac{\min\{s_i, d_j\}}{\max\{\sum_{i \in I} s_i, \sum_{j \in J} d_j\}} \\ \infty & \text{if } N^{ij} = \emptyset. \end{cases} \tag{3.5}$$

The minimum-cost single-commodity transportation problem defined by the described parameters is solved by the network flow formulation of Syslo et al. [1983].

The solution $x_{ij}$ of the transportation problem indicates the number of times we should assign shift combination $(i, j)$. These shift combinations are extended with the remaining Friday shifts by solving additional transport problems. To this end, the Friday shifts are considered as the set of $I$ shifts whereas the combinations of Saturday and Sunday shifts are considered as the set of $J$ shifts. This problem is then initialized and solved as described above. For Monday shifts we apply an analogous procedure.

### 3.3.2 Assign shift combinations

Now that we have the set of shift combinations per weekend that we want to assign, we describe how these shifts are assigned to employees.

The heuristic first selects a shift combination (part 1) and then an employee (part 2). The shift combination is selected using the following scheme:

1a. Select the "least flexible" shift combination, i.e., the combination $(i, j)$ for which the ratio between "the number of times combination $(i, j)$ should be assigned" $(x_{ij})$ and "the number of employees allowed to work combination $(i, j)$" $(|N^{ij}|)$ is the largest. These shift combinations are presumed to be the hardest to assign to an employee. In case of a tie, go to 1b.

1b. Of the remaining combinations, select the one for which $x_{ij}$ is the smallest. These are presumed to be the hardest to assign. In case of a tie, go to 1c.

1c. Of the remaining combinations, select the combination that is earliest in time. Since some of the hard constraints consider assignments in previous weekends, shifts that are earlier in time are presumed to be harder to assign. In case of a tie, go to 1d.

1d. Randomly select from the remaining combinations.

When a shift combination is selected, an employee is selected via the following scheme:

2a. Select the "least busy" employee, i.e., the employee working the fewest number of weekends relative to his contract hours. This way, all employees will work (approximately) the same number of weekends. In case of a tie, go to 2b.

2b. Select the employee for which $S_{ij}^n$ is the smallest. In case of a tie, go to 2c.

2c. Select the "least flexible" employee, i.e., select the employee that has the least number of remaining shift combinations that the employee is allowed to work. In case of a tie, go to 2d.

2d. Randomly select from the remaining employees.

### 3.3.3 Illustrative example

The example in this section consists of two parts. The first part illustrates the creation of weekend shift combinations, and the second part illustrates the assignment of the weekend shift combinations.

**Create weekend shift combinations for one weekend** We have one $A$ and one $B$ shift on both Saturday and Sunday, so $I = J = \{A, B\}$. We have 4 employees, so $N = \{1, 2, 3, 4\}$. Employee 1 is not allowed to work shift $B$ at all, employee 3 is not allowed to work shift $B$ on Saturday. We let $S_{AA} = S_{BB} = 1$ and $S_{AB} = S_{BA} = 2$. We omit the superscript of the penalty costs, since in this example they are the same for all employees.

First, note that:

$$\frac{\min\{s_i, d_j\}}{\max\{\sum_{i\in I} s_i, \sum_{j\in J} d_j\}} = \frac{1}{2} \quad \text{for } i \in I, j \in J. \tag{3.6}$$

Second, $|N^{AA}| = 4$, $|N^{AB}| = 3$, $|N^{BA}| = 2$, $|N^{BB}| = 2$, and $|N| = 4$. Hence, for $(A, A)$ we have:

$$\frac{|N^{AA}|}{|N|} = \frac{4}{4} = 1 > \frac{\min\{s_A, d_A\}}{\max\{\sum_{i\in I} s_i, \sum_{j\in J} d_j\}} = \frac{1}{2}. \tag{3.7}$$

So, for combination $(A, A)$ we say that 'many' employees are available, hence:

$$c_{AA} = S_{AA} = 1. \tag{3.8}$$

Similar, for $(A, B)$ we find that 'many' employees are available, hence $c_{AB} = S_{AB} = 2$, and for $(B, A)$ and $(B, B)$ we find that 'few' employees are available, hence $c_{BA} = M \cdot |N \setminus N^{BA}| = M \cdot 2$, and $c_{BB} = M \cdot |N \setminus N^{BB}| = M \cdot 2$.

The solution of the transportation problem is then $x_{AA} = x_{BB} = 1$, $x_{AB} = x_{BA} = 0$. So, we create one shift combination $(A, A)$ and one shift combination $(B, B)$, but no shift combinations $(B, A)$ or $(A, B)$.

**Assign weekend shift combinations** After shift combinations are created they are assigned to employees in the assignment phase. To illustrate this: assume we have the shift combinations and available employees as in Table 1.

Table 1: Assigning shift combinations

| Weekend | Combination | $x_{ij}$ | $N$ | $S_{ij}$ |
|---|---|---|---|---|
| 1 | $(A, A)$ | 1 | $\{1, 2, 3, 4\}$ | 1 |
| 1 | $(B, B)$ | 1 | $\{2, 4\}$ | 1 |
| 2 | $(B, A)$ | 1 | $\{1, 4\}$ | 2 |
| 2 | $(A, B)$ | 2 | $\{1, 2, 3, 4\}$ | 2 |

The $x_{ij}/|N^{ij}|$ ratios of the shift combinations are $\frac{1}{4}$, $\frac{1}{2}$, $\frac{1}{2}$, and $\frac{2}{4}$, respectively, hence shift combinations 2, 3, and 4, are the "least flexible" (Step 1a). We have

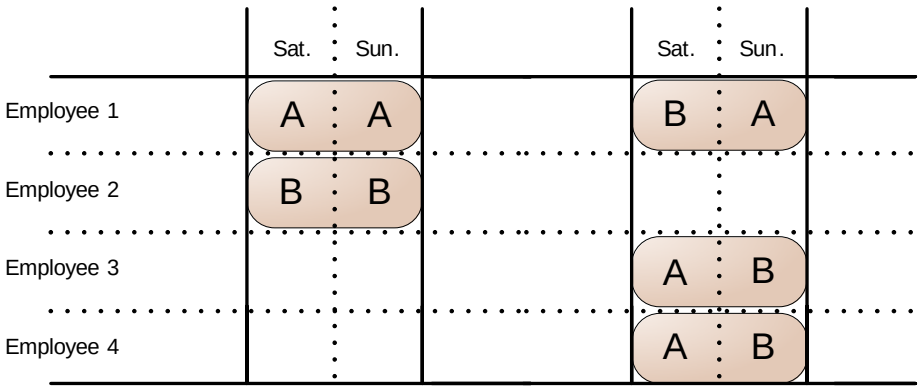


Figure 3.1: Example - Resulting roster

$x_{BB} = 1, x_{BA} = 1$, and $x_{AB} = 2$, so $(B, B)$ and $(B, A)$ have to be assigned the least number of times (Step 1b). Since $(B, B)$ is a shift combination of the first weekend, and $(B, A)$ of the second, $(B, B)$ is earliest in time (Step 1c), so $(B, B)$ is to be assigned.

Since there are no shift combination assigned yet, employees 2 and 4 (the employees available for $(B, B)$) are equally busy (Step 2a). For both these employees $S^n_{BB} = 1$, so Step 2b gives no conclusion on which employee to select. However, since employee 2 has only 3 remaining shift combinations (including $(B, B)$) and employee 4 has 4, Step 2c assigns shift combination $(B, B)$ to employee 2.

The other shift combinations are assigned to employees in an analogous way. $(B, A)$ is assigned randomly to 1 or 4, say it is assigned to 1. Then $(A, B)$ is assigned randomly to 3 or 4, say it is assigned to 3. Next, the other $(A, B)$ combination is assigned to 4, and, finally, $(A, A)$ is assigned randomly to 1, 3 or 4, say it is assigned to 1. We then get the roster as in Figure 3.1.

### 3.3.4 Local search

The initial solution, created via the heuristics described in Section 3.3.1 and Section 3.3.2, has a cost implied by violations of the soft constraints. We try to improve this solution via two local search techniques: 2-opt search, and a Very Large Neighborhood Search (VLNS).

The VLNS that we apply calculates whether swapping shift combinations, not necessarily in the same weekend, improves the total penalty cost, see Figure 3.2. These options include, assigning an assigned shift combination to another employee, swapping two shift combinations, or cyclically swapping three shift combinations. Swaps can be applied both on shift combinations in the same weekend, and in different weekends.

Every iteration of 2-opt search calculates per shift whether the total penalty cost improves if this shift is swapped with another shift. That is, if employee 1 is assigned to shift A, and employee 2 to shift B, 2-opt calculates whether the total penalty cost decreases when employee 1 is assigned to shift B, and employee 2 to shift A, see Figure 3.3. The swap is accepted if the total penalty cost decreases. Note that 2-opt also considers swapping not assigned shifts with

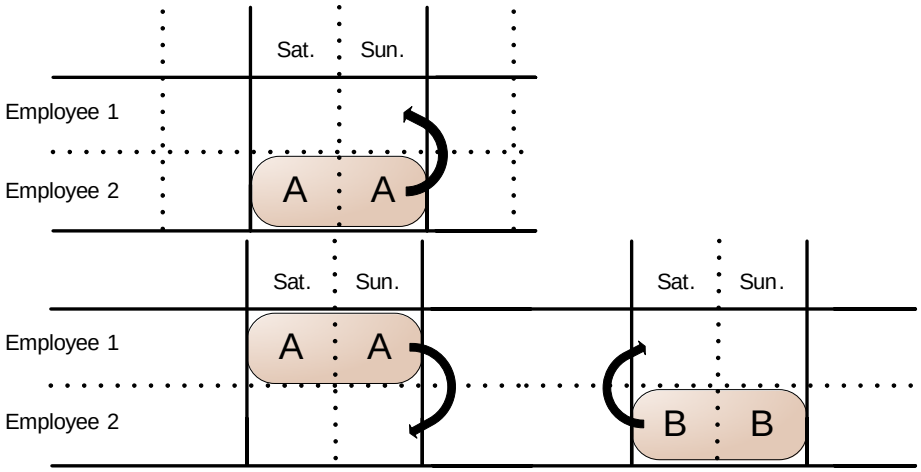


Figure 3.2: Local search: Very Large Neighborhood Search (VLNS). Assigning a shift combination to another employee (top), and swapping two shift combinations in (possibly) different weekends (bottom)

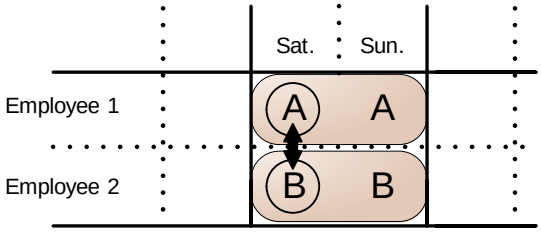


Figure 3.3: Local search: 2-opt search

assigned shifts.

The local search techniques are applied as in Algorithm 3.1:

**Algorithm 3.1.** *Local search scheme*

1. Apply VLNS, until no improvements are found anymore.
2. Apply 2-opt, until no improvements are found anymore.

# 4 Results

This section discusses experimental results. Section 4.1 analyzes the quality of the weekend rostering algorithm, as described in Section 3, on generated instances. Section 4.2 describes two case studies in which we tested the weekend rostering algorithm in practice and benchmarked it against a hybrid heuristic ordering method as used in commercial software, see Burke et al. [2008], Post and Veltman [2004].

## 4.1 Algorithm results

We have studied the quality of the shift combination assignment phase in a preliminary research Versteegh [2009]. We summarize the main results of the preliminary research here.

In total 400 instances were generated, of which, by construction, the optimal objective values were known. Furthermore, for each of these instances, all shifts are assigned in the optimal solution. We used 400 instances, since preliminary tests showed that after 400 instances the average deviation from the optimum stabilizes.

The quality of the algorithm was assessed by counting the number of shifts the algorithm was unable to assign, and determining the penalties resulting from violating the soft constraints. Two soft constraints were considered: (1) equitable division of shifts among employees, and (2) equitable division of shift types among employees.

For 89% of the instances all weekend shifts were assigned. The average number of not assigned shifts per instance is 0.85 shift (calculated over all 400 instances). From the experiments we observed that for 270 (67.5%) instances the optimal solution is found. The average devation from the optimum is 3.8%. It appeared that equitably dividing the shift *types* among employees was harder than equitably dividing the number of shifts. We expect to further improve on these results by extending the local search.

## 4.2 Practical results

This section presents the practical results. We first describe the experimental setup in Section 4.2.1. In Sections 4.2.2 and 4.2.3, presents the case studies in which we test the weekend rostering algorithm, and in Section 4.2.4 we discuss the results.

### 4.2.1 Experimental setup

We want to study the effect that weekend rostering has on the complete roster, i.e., the roster of the entire week. We compare two approaches, called WRP+CA and CA. WRP+CA first assigns the weekend shifts using the weekend rostering heuristic. Then, it fixes these shift, and assigns the weekday shift using the commercial algorithm (CA). CA assigns all shifts (week and weekend) using the same commercial algorithm. Both approaches use the same hard and soft constraints.

WRP+CA and CA are compared using the following performance indicators:

- Number of complete weekends assigned (Complete On)
- Number of half weekends assigned
- Number of off weekends assigned (Complete Off)
- Number of weekend shifts not assigned
- Total number of shifts not assigned

#### 4.2.2 Case 1: Belgian Police

The first case is provided by a Belgian Police department. This department consists of 60 employees, and it uses two-month rosters. In each roster, approximately 2000 shifts of the following four types need to be assigned: morning shifts (7h-13h), afternoon shifts (13h-22h), night shifts (22h-7h), and days-off. Shifts are assigned on both week and weekend days.

Before this Belgian Police department started to use the WRP+CA approach, as proposed in this paper, they manually assigned the weekend shifts before constructing the complete roster. Note that WRP+CA uses exactly the same decomposition. All hard and soft constraints defined in Section 2 are implied on the rosters.

#### 4.2.3 Case 2: Dutch care provider

The second case is provided by a Dutch care provider for visually impaired people that offers intramural and extramural care. We test the weekend planner on two departments called D1 and D2. These have 12 and 42 employees, respectively, and both use monthly rosters. D1 has two shift types: morning shifts (8h-15h) and afternoon shifts (15h-22h), and shifts must be scheduled on week and weekend days. In every month, approximately 135 shifts need to be assigned. D2 has three shift types: morning shifts (8h-15h), day shifts (10h-14h), and afternoon shifts (15h-22h), and again shifts must be assigned on week and weekend days. For D2 approximately 450 shifts must be assigned every month.

All hard and soft constraints defined in Section 2 are implied on the rosters, except soft constraint S3.

#### 4.2.4 Experimental Results

This section presents our experimental results for the cases presented in Section 4.2.2 and Section 4.2.3. For the Belgian Police case we tested the algorithms on 6 instances. For the Dutch care provider we also tested the algorithm on 6 instances; 3 for each department. For both approaches a time limit of 1 hour was set. Table 2 summarizes the results.

In Table 2 we observe for all instances of the Belgian Police case that both the fraction of weekends Complete On (fourth column), and the fraction of weekends Complete Off (sixth column) is larger when the WRP+CA approach is applied. This implies that, for all instances, the fraction of half weekends (fifth column) is smaller if WRP+CA is applied. We say that an employee works a half weekend if the employee works either on Saturday or on Sunday, but not on both days. In fact, on average WRP+CA assigns 55.8% weekends Complete On, 37.1% weekends Complete Off, and 7.1% half weekends. CA assigns on average 44.4% weekends Complete On, 26.7% weekends Complete Off, and 28.9% half weekends. Furthermore, the number of not assigned weekend shifts (seventh column) and the total number of not assigned shifts (eighth column) are approximately the same for WRP+CA and CA. Hence, WRP+CA outperforms CA on the weekend shift assignment, while the number of assigned shifts is approximately equal.

For department D1 of the Dutch Care Provider we observe no difference between both approaches. We believe this is caused by the relatively small size of this department. For D2 we observe that WRP+CA outperforms CA on

Table 2: Experimental results

| Case | Instance No. | Approach | Weekend Complete On | Weekend Half | Weekend Complete Off | Not assigned shifts Weekend | Not assigned shifts Total |
|---|---|---|---|---|---|---|---|
| Police | 1 | WRP+CA | 70.0% | 10.8% | 19.2% | 0.5% | 1.8% |
| | | CA | 66.9% | 18.3% | 14.8% | 0.9% | 2.0% |
| | 2 | WRP+CA | 57.8% | 8.5% | 33.7% | 0.3% | 0.3% |
| | | CA | 48.5% | 26.9% | 24.6% | 0.3% | 0.4% |
| | 3 | WRP+CA | 52.5% | 7.2% | 40.3% | 0.1% | 0.1% |
| | | CA | 38.8% | 33.9% | 27.3% | 0.2% | 0.3% |
| | 4 | WRP+CA | 54.2% | 4.9% | 40.9% | 0.0% | 0.1% |
| | | CA | 38.8% | 25.4% | 35.8% | 0.0% | 0.0% |
| | 5 | WRP+CA | 51.8% | 2.4% | 45.8% | 0.1% | 0.1% |
| | | CA | 34.8% | 35.6% | 29.6% | 0.3% | 0.5% |
| | 6 | WRP+CA | 48.6% | 8.7% | 42.7% | 0.4% | 1.1% |
| | | CA | 38.6% | 33.5% | 27.9% | 0.4% | 0.9% |
| Care | D1-1 | WRP+CA | 33.3% | 0.0% | 66.7% | 0.0% | 0.0% |
| | | CA | 33.3% | 0.0% | 66.7% | 0.0% | 0.0% |
| | D1-2 | WRP+CA | 33.3% | 0.0% | 66.7% | 0.0% | 0.0% |
| | | CA | 33.3% | 0.0% | 66.7% | 0.0% | 0.0% |
| | D1-3 | WRP+CA | 26.7% | 6.7% | 66.7% | 0.0% | 0.0% |
| | | CA | 26.7% | 6.7% | 66.7% | 0.0% | 0.0% |
| | D2-1 | WRP+CA | 27.0% | 6.5% | 66.5% | 0.0% | 0.0% |
| | | CA | 23.7% | 13.0% | 63.3% | 0.0% | 0.0% |
| | D2-2 | WRP+CA | 25.0% | 7.9% | 67.1% | 2.4% | 7.3% |
| | | CA | 23.8% | 9.8% | 66.5% | 2.0% | 7.0% |
| | D2-3 | WRP+CA | 26.8% | 8.3% | 64.9% | 0.0% | 0.0% |
| | | CA | 25.0% | 11.9% | 63.1% | 0.0% | 0.0% |

the weekend shift assignment. On average WRP+CA assigns 26.3% weekends Complete On, 66.2% weekends Complete Off, and 7.6% half weekends. CA assigns on average 24.2% weekends Complete On, 64.3% weekends Complete Off, and 11.6% half weekends. Again the number of weekend shifts not assigned and the total number of shifts not assigned are approximately equal. For case D2-2, note that CA assigns only one shift more than WRP+CA does.

# 5 Conclusions and discussion

This paper introduces the Weekend Rostering Problem (WRP), a rostering problem focused on weekend shift assignment. It is motivated by our experience that employee preferences predominantly focus on the weekends, since many social activities happen during the weekend. Despite of its practical relevance, the WRP is underexposed in both literature and decision support software.

In this paper, we introduce a two-phase heuristic to solve the WRP. The first phase assigns weekend shifts, the second phase assigns the remaining, weekday, shifts. For the first phase we design a special-purpose heuristic, whereas for the second phase we use a hybrid heuristic ordering method as used in com-

mercial software. This decomposition approach is inspired by our experience of how shift rosters are created in practice. An algorithm specifically designed to create weekend shift rosters supports the natural planning process. Planners often start assigning the 'hard' shifts, like weekend shifts. Furthermore, the decomposition algorithm allows the planner to adjust the automatically generated weekend rosters, before the rest of the roster is constructed. When the complete roster is already generated, it is harder for the planner to improve the weekend roster manually, since in a complete roster the re-assigning of weekend shifts is constrained by shifts assigned to weekdays Ikegami and Niwa [2003].

We encourage research to improve the algorithm proposed in this paper, since it is the first algorithm we know of that solves the WRP. Furthermore, we encourage further research into alternative decomposition approaches, like decomposition on night shifts or skills.

Experiment results show that the heuristic designed for the weekend shift assignment performs well on a broad range of generated instances. When we use this heuristic as a first phase in the shift rostering problem, results obtained via this decomposition approach look promising. On a set of practical problem instances, our decomposition outperforms the commercial algorithm on all of our weekend roster quality defining performance indicators, while both approaches have the same performance on the weekday shift roster. This proves that our decomposition is valuable when weekend related performance indicators are key determinants of the quality of rosters.

We incorporated the proposed algorithm in commercial software Post and Veltman [2004], and the algorithm is currently used to create rosters for the Belgian Police department case discussed in this paper.

## Acknowledgments

The authors would like to thank Frédérique Versteegh for initiating and taking the first steps in this research, and Monique Hoogstrate for helpful discussions.

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.

## References

Uwe Aickelin and Kathryn A. Dowsland. Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, 3(3):139–153, 2000. ISSN 1099-1425.

Ilham Berrada, Jacques A. Ferland, and Philippe Michelon. A multi-objective approach to nurse scheduling with both hard and soft constraints. *Socio-Economic Planning Sciences*, 30(3):183–193, 1996. ISSN 0038-0121.

Peter Brucker, Edmund Burke, Tim Curtois, Rong Qu, and Greet Vanden Berghe. A shift sequence based approach for nurse scheduling and a new benchmark dataset. *Journal of Heuristics*, 16:559–573, 2010. ISSN 1381-1231.

Edmund Burke, Peter Cowling, Patrick De Causmaecker, and Greet Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied Intelligence*, 15(3):199–214, 2001. ISSN 0924-669X.

Edmund Burke, Patrick De Causmaecker, Sanja Petrovic, and Greet Vanden Berghe. *Variable neighborhood search for nurse rostering problems*, pages 153–172. Kluwer Academic Publishers, Norwell, MA, USA, 2004a. ISBN 1-4020-7653-3.

Edmund Burke, Jingpeng Li, and Rong Qu. A pareto-based search methodology for multi-objective nurse scheduling. *Annals of Operations Research*, pages 1–19, 2009. ISSN 0254-5330.

Edmund K. Burke, Timothy Curtois, Gerhard Post, Rong Qu, and Bart Veltman. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2): 330–341, 2008. ISSN 0377-2217.

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

R. N. Burns and M. W. Carter. Work force size and single shift schedules with variable demands. *Management Science*, 31(5):599–607, 1985. ISSN 00251909.

R. N. Burns and G. J. Koop. A modular approach to optimal multiple-shift manpower scheduling. *Operations Research*, 35(1):100–110, 1987. ISSN 0030364X.

Richard N. Burns, Rangarajan Narasimhan, and L. Douglas Smith. A set-processing algorithm for scheduling staff on 4-day or 3-day work weeks. *Naval Research Logistics (NRL)*, 45(8):839–853, 1998.

B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems–a bibliographic survey. *European Journal of Operational Research*, 151(3):447–460, 2003. ISSN 0377-2217.

Patrick De Causmaecker and Greet Vanden Berghe. Relaxation of coverage constraints in hospital personnel rostering. In Edmund Burke and Patrick De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 129–147. Springer Berlin / Heidelberg, 2003. ISBN 978-3-540-40699-0.

K.A. Dowsland and J.M. Thompson. Solving a nurse scheduling problem with knapsacks, networks and tabu search. *Journal of the Operational Research Society*, 51(7):825–833, 2000.

Moustafa Elshafei and Hesham K. Alfares. A dynamic programming algorithm for days-off scheduling with sequence dependent labor costs. *Journal of Scheduling*, 11(2):85–93, 2008.

Hamilton Emmons and Richard N. Burns. Off-day scheduling with hierarchical worker categories. *Operations Research*, 39(3):484–495, 1991.

Hamilton Emmons and Du-Shean Fuh. Sizing and scheduling a full-time and part-time workforce with off-day and off-weekend constraints. *Annals of Operations Research*, 70(0):473–492, 1997. ISSN 0254-5330.

A.T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1):21–144, 2004. ISSN 0254-5330.

Johannes Gärtner, Nysret Musliu, and Wolfgang Slany. Rota: a research project on algorithms for workforce scheduling and shift design optimization. *AI Communications*, 14(2):83–92, 2001. ISSN 0921-7126.

Rudy Hung. Multiple-shift workforce scheduling under 3-4 the workweek with different weekday and weekend labor requirements. *Management Science*, 40 (2):280–284, 1994a. ISSN 0025-1909.

Rudy Hung. Single-shift off-day scheduling of a hierarchical workforce with variable demands. *European Journal of Operational Research*, 78(1):49–57, 1994b. ISSN 0377-2217.

A. Ikegami and A. Niwa. A subproblem-centric model and approach to the nurse scheduling problem. *Mathematical Programming*, 97(3):517–541, 2003. ISSN 0025-5610.

Brigitte Jaumard, Frederic Semet, and Tsevi Vovor. A generalized linear programming model for nurse scheduling. *European Journal of Operational Research*, 107(1):1–18, 1998.

D.L. Kellogg and S. Walczak. Nurse Scheduling: From Academia to Implementation or Not? *Interfaces*, 37(4):355–369, 2007.

G. J. Koop. Cyclic scheduling of offweekends. *Operations Research Letters*, 4 (6):259–263, 1986. ISSN 0167-6377.

Holmes E. Miller, William P. Pierskalla, and Gustave J. Rath. Nurse scheduling using mathematical programming. *Operations Research*, 24(5):857–870, 1976.

Nysret Musliu, Johannes Gärtner, and Wolfgang Slany. Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*, 118(1-2):85–98, 2002. ISSN 0166-218X.

Gerhard Post and Bart Veltman. Harmonious personnel scheduling. In *Proceedings of the 5th international conference on the Practice and Theory of Automated Timetabling*, pages 557–559, 2004.

ManMohan S. Sodhi and Stephen Norris. A flexible, fast, and optimal modeling approach applied to crew rostering at london underground. *Annals of Operations Research*, 127(1):259–281, 2004. ISSN 0254-5330.

Maciej M. Syslo, Narsingh Deo, and Janusz S. Kowalik. *Discrete optimization algorithms: with Pascal programs*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

Frédérique Versteegh. Let the weekend begin! a solution for solving the weekend scheduling problem for ortec harmony. Master's thesis, University of Twente, The Netherlands, 2009. URL `http://essay.utwente.nl/60656`.

D. Michael Warner. Scheduling nursing personnel according to nursing preference: A mathematical programming approach. *Operations Research*, 24(5): 842–856, 1976.

P. Daniel Wright, Kurt M. Bretthauer, and Murray J. Ct. Reexamining the nurse scheduling problem: Staffing ratios and nursing shortages*. *Decision Sciences*, 37(1):39–70, 2006. ISSN 1540-5915.