Becker, Tristan

Article — Published Version

A decomposition heuristic for rotational workforce scheduling

Journal of Scheduling

Provided in Cooperation with:
Springer Nature

Suggested Citation: Becker, Tristan (2020) : A decomposition heuristic for rotational workforce scheduling, Journal of Scheduling, ISSN 1099-1425, Springer US, New York, NY, Vol. 23, Iss. 5, pp. 539-554,
https://doi.org/10.1007/s10951-020-00659-2

This Version is available at:
https://hdl.handle.net/10419/288361

Journal of Scheduling (2020) 23:539–554
https://doi.org/10.1007/s10951-020-00659-2

A decomposition heuristic for rotational workforce scheduling

Tristan Becker[1]

Published online: 30 June 2020

Abstract
In rotational workforce planning, a schedule is constructed from a sequence of work and rest periods. Each employee starts at a different part of the schedule, and after a certain amount of time, the schedule repeats. The length of the schedule increases with a higher number of employees. At the same time, various constraints on work sequences and days off have to be considered. For a large number of employees, it is difficult to construct a schedule that meets the requirements. It is important to ensure low solution times independently of the problem instance characteristics. In this work, a novel decomposition approach for rotational shift scheduling is proposed. The decomposition exploits the fact that most constraints in rotational workforce scheduling are imposed on the work shift sequence. By considering a fixed set of blocks to cover the demand, the problem complexity can be greatly reduced. Given a fixed set of blocks, we propose a network model that determines whether a feasible sequence of shift blocks exists. The decomposition approach is applied to the problem structure of the Rotating Workforce Scheduling Problem but may be extended to different problem structures. In a computational study, the decomposition approach is compared to a mathematical formulation and previous exact and heuristic approaches. Computational results show that the decomposition approach greatly outperforms previous heuristics on the standard benchmarks.

Keywords Staff scheduling · Integer programming · Decomposition · Rotating Workforce Scheduling Problem

1 Introduction

Human resources are typically one of the most expensive resources of a company. Shift planning thus plays a vital role for efficient operations. A sufficient staffing level must be ensured while legal and practical constraints are respected for each individual schedule. One of the most common requirements in shift scheduling is to work several consecutive days, as this typically implies multiple consecutive days off.

Generally, it is possible to employ either a manual or rotational schedule. With a rotational schedule, every employee works the same schedule lagged in time. After a certain number of days, the rotational schedule repeats. Rotating schedules are especially well suited for organizations with static workforce demand, a work load that exceeds the working time of a single employee (e.g., 3 shifts per day, seven days a week) and homogeneous skill levels among employees. This structure is often found in manufacturing and organizations from the public sector such as emergency medical services, fire, and police departments (Laporte 1999). Once a suitable rotational schedule has been found, planning is greatly simplified as the manager and employees repeat the same schedule (Becker et al. 2019). An exemplary rotational schedule and the resulting coverage are depicted in Fig. 1. The schedule covers a demand pattern that repeats after seven days. Staffing requirements for each shift type are shown at the bottom. The schedule requires a number of nine employees who start with an offset of seven days, i.e., every employee could start at the beginning of a different row. After a number of 63 days, the schedule repeats from the employee's point of view. In each row of the schedule, the last day is adjacent to the first day of the next row. While rotational schedules reduce planning efforts, negative health effects and employee dissatisfaction are often reported (Åkerstedt 2003).

For shift scheduling, there are different commercial software packages available. Important aspects of computational methods for workforce scheduling comprise the flexibility and computational time required to facilitate an interactive planning process. Standard software often cannot provide the required flexibility for rotational workforce scheduling prob-

✉ Tristan Becker
tristan.becker@om.rwth-aachen.de

[1] RWTH Aachen University, School of Business and Economics, Chair of Operations Management, Kackertstraße 7, 52072 Aachen, Germany

			Day						
			Mon	Tue	Wed	Thu	Fri	Sat	Sun
Employee	1		M	M	N	N	-	-	N
	2		N	N	N	-	-	M	M
	3		A	A	-	-	-	A	A
	4		N	N	-	-	M	M	M
	5		M	-	-	M	M	A	A
	6		-	-	A	A	A	N	N
	7		-	-	M	M	N	N	-
	8		-	M	M	A	A	-	-
	9		A	A	A	N	N	-	-
		M	2	2	2	2	2	2	2
	Cov.	A	2	2	2	2	2	2	2
		N	2	2	2	2	2	2	2

Fig. 1 An exemplary weekly rotational workforce schedule (feasible solution for a benchmark instance from Musliu et al. (2018), M = morning, A = afternoon, N = night)

lems or requires a high amount of computational time (Burke et al. 1999; Musliu 2006).

A standard problem structure for the Rotating Workforce Scheduling Problem (*RWS*) is proposed by Musliu (2006). In addition, a set of benchmark problems is presented that is inspired by problems from practice to evaluate computational methods for the *RWS*. Recently, the benchmark set has been further extended (Musliu et al. 2018).

Several studies propose exact and heuristic methods for the *RWS* (Triska and Musliu 2011; Mutingi and Mbohwa 2015a; Erkinger and Musliu 2017; Musliu et al. 2018). Given the importance of rotational shift scheduling, efficient computational methods are a continuing interest of research. We propose a novel state-of-the-art decomposition heuristic that is applicable for rotational shift scheduling problems and other types of scheduling problems where a cyclic plan is desired. The decomposition is applied to the *RWS* to illustrate the benefits for computational performance. The decomposition approach is very efficient for feasible instances from the *RWS* benchmark set and greatly improves the results of previous heuristics. The contribution of this study is threefold:

- Innovative decomposition approach for rotational scheduling problems
- Novel state-of-the-art heuristic for the *RWS*
- Comparison of exact and heuristic solution approaches for the *RWS* in a computational benchmark study

The remainder of this paper is structured as follows: Sect. 2 provides a literature review on solution approaches for shift scheduling with a focus on the *RWS*. A formal statement and mathematical formulation of the *RWS* are given in Sect. 3. The mathematical formulation serves as the basis for the discussion of the decomposition approach. The decomposition approach consists of a master and a subproblem and is explained in detail in Sect. 4. A decomposition heuristic using a heuristic solution method for the master problem and an exact method for the subproblem is described in Sect. 5. Section 6 compares the performance of the decomposition heuristic with previous results and the mathematical formulation from Sect. 3 using the standard benchmarks for *RWS*. Finally, Sect. 7 gives a conclusion and provides some perspective for future research on the decomposition approach.

2 Literature review

A rich body of literature exists on the theory and practice of shift scheduling. Surveys are provided by Ernst et al. (2004) and Van Den Bergh et al. (2013). Computational methods of shift scheduling have been applied to various problems in practice. Many studies have focused on the problem of nurse rostering. The nurse rostering problem is a complex shift scheduling problem that considers many practical and legal constraints, as well as employee preferences. Typically, employee preferences and the violation of some constraints are considered as objective. Owing to the heterogeneous qualifications among nurses, schedules are mostly generated for a limited amount of time. Due to the complexity and variety of features found in nursing wards, the nurse rostering problem remains one of the most studied problems in shift scheduling literature (Burke et al. 2004). Therefore, numerous papers on solution methodology and decomposition approaches focus on nurse rostering problems. The *RWS* is a more specialized problem. The objective is to find a feasible schedule that provides the required staffing level at all times with a homogeneous workforce. No violation of hard constraints is allowed. We focus this literature review on solution approaches for shift scheduling and the generation of rotational workforce schedules.

2.1 Solution approaches for shift scheduling

Many variants of shift scheduling problems have been investigated by the literature. Solution approaches that have been specifically applied to the *RWS* are discussed in Sect. 2.2. Valouxis and Housos (2000) investigate solution methodology for the monthly creation of nurse rosters. They formulate an integer program that is difficult to solve with standard commercial software. They find that preferences for specific work sequences cannot be efficiently modeled by integer programming. A hybrid solution method that combines local and tabu search is used to obtain acceptable solutions from a

practical perspective. Brucker et al. (2010) propose a decomposition approach for non-cyclic nurse scheduling. In the first stage, work sequences with a high quality regarding the nurse preferences are selected. The second stage iteratively constructs a schedule by combining sequences that meet constraints for the overall roster. Local search techniques are applied to improve the initially constructed roster. Rocha et al. (2014) discuss a constructive heuristic for rotational staff scheduling in the glass industry. Their approach finds a sequence of blocks that each comprise multiple work shifts. They note that the number of breaks between blocks is important to obtain a feasible solution. An approach to assess the feasibility of the problem based on the parameters and the resulting number of shift breaks is presented. Their study focuses the case of a constant workforce demand. They assume a requirement of exactly one shift group per day and shift. Brucker et al. (2011) analyze the complexity of several classes of personnel scheduling problems. Several mathematical models that cover different personnel scheduling cases are presented. The models are differentiated by polynomial solvable and NP-hard problems. Kiermaier et al. (2016) present a stochastic optimization approach for rotational rostering in the service industry. While the days and shifts are fixed, the starting times may be adjusted in the short term to react to changes in demand. A multi-stage stochastic program is proposed, but it turns out to be intractable. Two approximations are presented that reduce the program to a two-stage problem. Various decomposition strategies have been applied to shift scheduling problems. Furthermore, the idea to aggregate sequential work shifts to shift blocks is frequently used in solution approaches.

2.2 Rotational workforce schedules

An early survey of rotational workforce scheduling is provided by Baker (1976). They differentiate between shift scheduling and days-off scheduling. Days-off scheduling determines the work days only, while shift scheduling determines the shifts assigned for each work day. Tour scheduling is the integrated assignment of both work days and shifts (Ernst et al. 2004). Laporte et al. (1980) formalize some conditions for rotational schedules and propose an integer programming-based algorithm to create rotational schedules. They note that an essential property of the structure of rotational schedules is that there is an alternating sequence of consecutive work shifts and days off. Bartholdi et al. (1980) introduce the *(k, m)*-scheduling problem. It involves finding a schedule of length m where every employee works a number of k successive days. Millar and Kiragu (1998) present a mathematical model for cyclic and non-cyclic nurse scheduling. They construct a network where each node resembles either a sequence of work shifts or days off. The mathematical formulation finds a shortest path in the network with side constraints to construct a schedule. Rocha et al. (2013) propose a new mathematical formulation for cyclic staff scheduling. A new approach for the formulation of work sequence constraints is incorporated into the model. The practical applicability is illustrated by two case studies. Balakrishnan and Wong (1990) present a network model for the *Rotating Workforce Scheduling Problem (RWS)*.

Musliu et al. (2002) propose a four step decomposition approach for the *RWS*. They begin by choosing a fixed set of allowable lengths for work blocks. Then, a schedule that assigns work days and days off is constructed by selecting a particular sequence of work blocks and days off. The construction of the schedule also takes into account weekend preferences. Next, feasible shift sequences are enumerated given the possible work block sequences and shift sequences are assigned to work blocks so that the staffing requirements are met. It is noted that different schedules can be constructed by rearranging the order of work blocks. Moerz and Musliu (2004) propose a genetic algorithm for the *RWS*. New mutation and crossover operators are presented. Musliu (2006) presents a unified problem structure and benchmark set of 20 test instances, which are inspired by real rotational workforce scheduling applications. Several problem instances from the literature are included in the benchmark set. They propose new heuristic methods for the automatic generation of rotational workforce schedules. Experiments show that a tabu search heuristic with a conflict minimization strategy (*MC-T*) is well suited for solving the *RWS*. The *MC-T* greatly outperforms standard software for shift scheduling. Although many problem instances are solved quickly, there is still a high solution time for several instances.

Various algorithms and methods have been applied to the benchmark set provided by Musliu (2006). These include integer programming, constraint programming, genetic algorithms, iterated local search, and satisfiability modulo theories. In the following, several heuristic and exact approaches for the *RWS* are discussed. Most of these approaches were specifically developed for the problem structure of the *RWS*.

Musliu (2011) apply an iterated local search method to the *RWS*. For some instances, their approach is faster than the state of the art, while the overall performance is comparable. Triska and Musliu (2011) propose a constraint programming formulation for the *RWS*. As a constraint programming formulation, it represents an exact method. For some small problem instances, the solution times are lower than those of the *MC-T*. Overall, it is not competitive with the solution times of the *MC-T*. Mutingi and Mbohwa (2015a) and Mutingi and Mbohwa (2015b) apply variants of an evolutionary algorithm to the *RWS*. The algorithms are inspired by a biological metamorphosis process. In comparison with the *MC-T*, the mean solution time of their best performing algorithm is less than half as high.

Rocha et al. (2013) demonstrate that their general mathematical formulation for cyclic staff scheduling is also applicable to the *RWS*, although it was not specifically developed for the *RWS*. However, many problem instances cannot be solved within the time limit. Erkinger and Musliu (2017) investigate satisfiability modulo theories as an option for an exact method to solve the *RWS*. A number of instances of the test set can be solved quickly. Yet, for some instances no feasible solution is found within the time limit.

Different mathematical and constraint programming formulations for solving the *RWS* are evaluated by Musliu et al. (2018). The mathematical formulations are solved with Gurobi, and the best performance is obtained by the use of a deterministic finite automaton constraint to ensure feasible shift sequences. While this type of constraint is typically used in constraint programming, they note that Gurobi is able to transform the mathematical formulation to a network model which can be solved very efficiently. The best overall performance is obtained by a constraint programming formulation that is solved with Chuffed. In this formulation, a direct modeling approach is pursued, where there is mostly a one-to-one correspondence between the model constraints and practical restrictions. To strengthen the formulation, valid inequalities are added. Furthermore, the benchmark set proposed by Musliu (2006) is extended by an additional 1980 randomly generated problem instances of realistic structure so that an overall of 2000 instances are obtained. Extensive computational experiments are conducted to compare the mathematical and constraint programming formulations to other exact and heuristic methods for the *RWS*. The computational experiments establish their constraint programming formulation as the fastest state-of-the-art method to obtain solutions for the *RWS*. Furthermore, the experiments show that mathematical programming is most robust to detect infeasibility.

Musliu (2013) note that the most efficient solution method is dependent on the specific instance characteristics. Using a larger test set, they apply different machine learning techniques to select the solution method on the basis of the instance characteristics.

2.3 Implications

Many solution approaches for shift scheduling have been proposed. Decomposition approaches were often successfully used for nurse rostering problems. A number of solution approaches have been applied for the *RWS*. Only recently solution approaches were proposed that provide lower solution times in comparison with the heuristic proposed alongside the benchmark set (Musliu 2006; Mutingi and Mbohwa 2015b; Musliu et al. 2018). The computational time required to generate a feasible schedule is an important preference of decision makers (Burke et al. 1999). Thus, it is important for solution methods to generate feasible schedules quickly independent of the problem instance. Moreover, there is further potential to reduce the computational times of the state-of-the-art solution methods. This paper fills the gap by proposing a novel decomposition approach which is applied to the *RWS*. It differs from the existing decomposition approach proposed by Musliu et al. (2002), since different constraints are taken into account by the master and subproblem. In particular, the decomposition approach presented in this paper takes the staffing requirements into account in the master problem. In that way, the master problem has to consider the complete set of feasible work shift sequences. As a result, there is no necessity for an intermediate step which determines the feasible shift sequences for the selected work days and the subproblem does not have to account for the staffing requirements. In addition, an exact approach based on a network model is used to efficiently enumerate the solution space of the subproblem.

The novel decomposition approach proposed in this paper is applied to the *RWS*, and computational results indicate that it is very efficient for feasible instances in comparison with the current state of the art using the recently proposed benchmark set.

3 Problem description and mathematical formulation

In the following, we give a formal statement of the *Rotating Workforce Scheduling Problem (RWS)* (Musliu 2006). Table 1 gives an overview of the parameters used in the problem description and mathematical formulation.

3.1 Problem description

The *RWS* involves the construction of a schedule for a given workforce and number of days that exactly covers all shift demands on each day. Shift demands are specified for each shift on every day separately. Different shift types may comprise morning, afternoon, and night shifts. In a problem instance of the *RWS*, the workforce can be either split into homogeneous groups or each employee can be considered explicitly. More formally, given a set of employees E, a schedule has to be derived over a number of days that can be repeated to cover the demand T_{ds} for each day d and shift type s. The length of the schedule is defined by the number of employees and the offset between the start days of each employee. For example, in Fig. 1 a schedule for 9 employees is shown. The offset between the start days Ω is seven, so that every employee starts with the first shift in her respective row. Looking at Fig. 1 column-wise, all shifts in each column are worked simultaneously so that the workforce demand is covered. The average weekly working time is implicitly defined

Table 1 Declaration of used sets, parameters, and variables

Name	Description
$d \in D$	Days of the schedule
$d \in G$	Days of the planning period
$s \in S$	Shifts
$e \in E$	Employees
Ω	Offset between employee start days
T_{ds}	Staffing requirement on day d in shift s
$\min^d$	Minimum number of consecutive days off required
$\max^d$	Maximum number of consecutive days off allowed
$\min^{\text{work}}$	Minimum number of consecutive work days required
$\max^{\text{work}}$	Maximum number of consecutive work days allowed
$\min_s^{\text{work}}$	Minimum number of consecutive shifts of type s allowed
$\max_s^{\text{work}}$	Maximum number of consecutive shifts of type s allowed
F_2	Set of forbidden shift sequences of length 2
F_3	Set of forbidden shift sequences of length 3
x_{ds}	1, if shift s on day d is part of the schedule, 0 otherwise
y_d	1, if a sequence of $\min^{\text{work}}$ shifts starts on day d, 0 otherwise
z_{ds}	1, if a sequence of $\min_s^{\text{work}}$ shifts of type s starts on day d, 0 otherwise

by the parameters, since the number of employees and days included in the schedule is known, as well as the total number of shifts to be covered. Formally, the schedule comprises a number of $|E|$ time periods with a length of Ω each. Typically, a number of seven days is selected for the length of a single time period Ω. This ensures that the schedule rotates over the same weekdays. After a number of $|E| \cdot \Omega$ days, the schedule repeats. At this point, every employee has worked the complete schedule so that an equal work distribution among the workforce is ensured. The demand T_{ds} is specified for every day $d \in G$, where G is defined as $\{1, \ldots, \Omega\}$. If there is a repeating pattern in the demand that comprises more than seven days, a longer time span has to be selected for the value of Ω.

The schedule has to adhere to various constraints with regard to work shift sequence and days off. The minimum and maximum number of days for each work shift sequence is defined by $\min^{\text{work}}$ and $\max^{\text{work}}$, respectively. That is, each shift sequence has to include at least $\min^{\text{work}}$ and at most $\max^{\text{work}}$ consecutive days. Between shift sequences, there must be at least a number of $\min^d$ days off and at maximum a number of $\max^d$ days off. A shift sequence may consist of different shift types. Within a work sequence, the minimum and maximum number of consecutive shifts of type s is defined by $\min_s^{\text{work}}$ and $\max_s^{\text{work}}$, respectively. Finally, there are forbidden shift sequences of different length. F_2 denotes the set of forbidden shift sequences of successive shifts. These shift sequences may not occur in a shift block. Shift sequences included in F_3 relate to forbidden sequences between two shift blocks with only one day off in between. A forbidden shift sequence can be, e.g., a night shift that is followed by a morning shift on the next day, or an afternoon shift, that is followed by a day off and a morning shift.

3.2 Mathematical formulation

We propose the following mathematical formulation for the *RWS*. The objective is to find a solution that satisfies all constraints. Several mathematical formulations with similar features are proposed by Rocha et al. (2013); Musliu et al. (2018); Becker et al. (2019). It is a direct formulation in the sense that most practical constraints can be directly mapped to mathematical constraints. Thus, to apply this mathematical formulation, a problem must have the characteristics detailed in the previous section.

$$\sum_{d' \in D | ((d'-1) \bmod \Omega)+1=d} x_{d's} = T_{ds} \qquad \forall d \in G, s \in S \tag{1}$$

$$\sum_{d' \in \{d, \ldots, d+\max^d\}} \sum_{s \in S} x_{d's} \geq 1 \qquad \forall d \in D \tag{2}$$

$$1 - \sum_{s \in S} x_{ds} \leq 2 - \sum_{s \in S} (x_{(d-1)s} + x_{(d+1)s}) \qquad \forall d \in D \tag{3}$$

$$\sum_{d' \in \{d,\ldots,d+\max^{\text{work}}\}} \sum_{s \in S} x_{d's} \leq \max^{\text{work}} \quad \forall d \in D \tag{4}$$

$$y_d \leq \frac{\sum_{d' \in \{d,\ldots,d+\min^{\text{work}}-1\}} \sum_{s \in S} x_{d's}}{\min^{\text{work}}} \quad \forall d \in D \tag{5}$$

$$x_{(d+\min^{\text{work}}-1)s} \leq \sum_{d' \in \{d,\ldots,d+\min^{work}-1\}} y_{d'} \forall d \in D, s \in S \tag{6}$$

$$x_{ds_1} \leq 1 - x_{(d+1)s_2} \quad \forall (s_1, s_2) \in F_2, d \in D \tag{7}$$

$$x_{ds_1} \leq 1 + \sum_{s \in S} x_{(d+1)s} - x_{(d+2)s_2} \forall (s_1, s_2) \in F_3, d \in D \tag{8}$$

$$\sum_{s \in S} x_{ds} \leq 1 \quad \forall d \in D \tag{9}$$

$$\sum_{d' \in \{d,\ldots,d+\max_s^{\text{work}}\}} x_{d's} \leq \max_s^{\text{work}} \quad \forall d \in D, s \in S \tag{10}$$

$$z_{ds} \leq \frac{\sum_{d' \in \{d,\ldots,d+\min_s^{\text{work}}-1\}} x_{d's}}{\min_s^{\text{work}}} \quad \forall d \in D, s \in S \tag{11}$$

$$x_{(d+\min_s^{\text{work}}-1)s} \leq \sum_{d' \in \{d,\ldots,d+\min_s^{work}-1\}} z_{ds} \forall d \in D, s \in S \tag{12}$$

$$x_{ds}, y_d, z_{ds} \in \{0, 1\} \quad \forall d \in D, s \in S \tag{13}$$

It follows from the previous discussion that the set of days D in the rotational schedule is defined as $\{1, \ldots, \Omega \cdot |E|\}$ for the set of employees E and an offset Ω between employee start days. It is a cyclic set and repeats after $\Omega \cdot |E|$ days.

When the schedule is implemented, the start day of each employee is defined by $d_e^{\text{start}} = \Omega \cdot (e-1)+1 \ \forall e \in E$. As a consequence, every day d of the schedule provides coverage for day $(d-1 \bmod \Omega)+1 \ \forall d \in D$. Let variable $x_{ds} \in \{0, 1\}$ denote whether shift type s on day d is part of the schedule. Then, constraints (1) ensure that each shift is covered on every day by the required number of employees T_{ds}.

To ensure that no sequence of days off violates the maximum number of consecutive days off, Constraints (2) are introduced. At least one shift needs to be scheduled in every sequence of days of length $\max^d + 1$. For the case $\min^d = 2$, Constraints (3) are introduced. If no shift is scheduled on day d, at least one of the days $\{d-1, d+1\}$ must not contain any shifts. Constraints (4) ensure no shift sequence exceeds a length of $\max^{work}$. In combination, Constraints (3) and (4) imply that shift sequences are separated by at least a single day off.

Each shift sequence needs to include at least $\min^{work}$ shifts. Let variable $y_d \in \{0, 1\}$ denote whether a shift sequence of a length of at least $\min^{work}$ starts on day d. Constraints (5) only allow a value $y_d = 1$ if a number of $\min^{work}$ are scheduled on days $\{d, \ldots, d + \min^{work} - 1\}$. Then, Constraints (6) ensure that every shift is part of a shift sequence that respects the minimum length. Forbidden shift sequences of length two are considered by Constraints (7). Constraints (8) are used to exclude forbidden sequences that span three shifts, where the second day of the sequence is a day off. Only one shift can be worked per day (Constraints (9)).

Further, the minimum and maximum length of shift blocks $\min_s^{work}$ and $\max_s^{work}$ of type s has to be respected. Constraints (10) ensure that no shift block exceeds the maximum length per type. Let value $z_{ds} \in \{0, 1\}$ denote whether a shift sequence of type s with a length of at least $\min_s^{work}$ begins on day d. Then, Constraints (11–12) enforce the minimum shift block length per type s analogously to Constraints (5–6). Finally, Constraints (13) limit the decision variables to their domains.

4 Decomposition approach

In this section, a formal description of the decomposition approach is given. The algorithmic framework is separately described in the next section. Table 2 gives an overview of the symbols used in this section.

A feasible solution to an instance of the *RWS* is characterized by a sequence of shift blocks separated by sequences of days off. This is explained by the fact that employees and in turn decision makers value schedules, where employees work sequences of shifts and recover during sequential days off. Owing to the various constraints regarding the minimum and maximum length of shift blocks and undesirable shift sequences, the set of feasible shift blocks is highly constrained. Thus, the *RWS* is split into two subproblems:

- *Master problem* Find a combination of tuples with a specified start day and shift block (g, b) such that the staffing requirements T_{ds} are covered.
- *Subproblem* Construct a schedule by finding a feasible sequence of block tuples (g, b) based on the solution of the master problem.

4.1 Master problem

The set of feasible shift blocks can typically be easily enumerated. Let $b \in B$ denote the set of shift blocks that respect all constraints on shift sequences with regard to the specific problem instance. The master problem is defined as finding a set of shift blocks that meet the staffing requirements T_{ds} exactly on every day d and in every shift s. Note that coverage for every day of the staffing requirements can be provided by a shift block starting on day $g \in G$.

For example, assume $\Omega = 7$. Given a single shift type 1, that is denoted by M in the following, a shift block could comprise, e.g., the sequence MMM. If this shift block is scheduled on day 1, coverage is provided for days $\{1, 2, 3\}$. With start day 7, coverage would be provided on days $\{7, 1, 2\}$. The problem represents an extension of the exact

Table 2 Declaration of used sets, parameters, and variables

Name	Description
$b \in B$	Feasible shift blocks
$w \in W$	Subtours
$n \in N$	Set of nodes
$g \in G$	Shift block start days
C_{ds}	Set of tuples (g, b) that contain shift s on day d
K	Set of edges (i, j) that must not be used
d_{ij}	Additional employees required if node j succeeds node i
λ_{gb}	Number of times shift block b starts on day g
$v_{ij} \in \{0, 1\}$	1 if node j is visited after node i; 0 otherwise

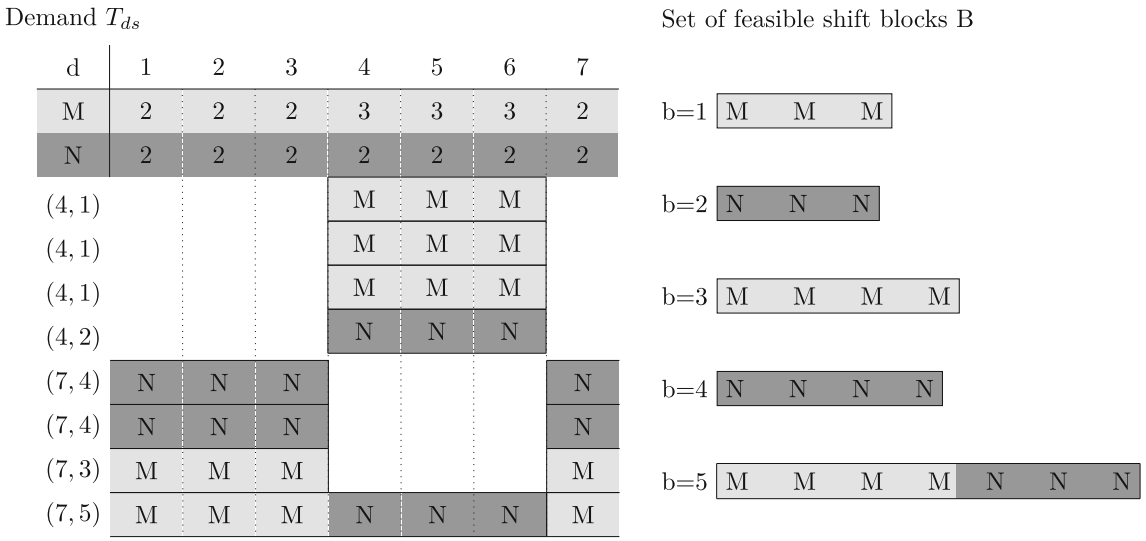

Fig. 2 Exemplary covering of staffing requirements by use of a set of shift blocks

covering problem, where each element has to be covered a specified number of times and each set can be selected multiple times. Coverage requirements are specified by T_{ds}. The sets are defined by every combination of start day and block (g, b). Figure 2 shows an example for the master problem. Workforce requirements T_{ds} are shown on the upper left side. An exemplary subset of feasible shift blocks B that is defined by the constraints of the *RWS* is shown on the right side. One possible solution to the master problem is shown on the left side. In each row, a combination of start day g and block b is shown. In terms of the mathematical formulation from section 3.2, the master problem ensures that the staffing requirements (1), constraints on the length of shift blocks (4–6), constraints on forbidden shift sequences in shift blocks (7), and constraints on the length of shift sequences of each type (10–12) are satisfied. The master problem neglects only constraints on consecutive days off and sequences of shift blocks.

Formally, the master problem can be stated as follows: Let $\lambda_{gb} \in \mathbb{N}_0$ denote the number of times shift block b starts on day g. Set C_{ds} contains the tuples of start day and shift block (g, b) that provide coverage for shift s on day d. The set of solutions of the master problem has to satisfy the following equations:

$$\sum_{(g,b)\in C_{ds}} \lambda_{gb} = T_{ds} \quad \forall d \in G, s \in S \tag{14}$$

In case the master problem is infeasible, the overall *RWS* instance is also infeasible.

Typically, there are many solutions to the master problem for a feasible instance of the *RWS*. However, there may not be a feasible sequence of shift blocks for a solution of the master problem, as it may not be possible to find a sequence that satisfies constraints on minimum and maximum number of days off and conflicts between shift blocks.

Solutions also may vary greatly in structure. Consider, e.g., a solution for the master problem with shift block $AAANNN$. The shift block could always be split into two parts AAA and NNN, while maintaining feasibility for the first problem if the work shift sequence constraints are satisfied for the two parts. Generally, a single block with many shifts will lead to a shorter schedule compared to two single blocks, as there are no days off in between. Given conflicts between shift blocks, a lower number of blocks reduce the potential number of conflicts, but the number of breaks is also reduced. Consequently, it is important to find solutions of different structures to search the solution space effectively.

4.2 Subproblem

For a given solution of the master problem, the subproblem determines whether a feasible sequence of shift blocks exists. In terms of the mathematical formulation from Sect. 3.2, the subproblem has to only consider constraints on consecutive days off (2–3) and constraints on forbidden sequences of shift blocks (8). Some examples for feasible and infeasible sequences of start day and shift blocks are illustrated in Fig. 3. In a feasible sequence of shift blocks (g, b) each combination of start day and block (g, b) needs to be included λ_{gb} times. Furthermore, the number of days off has to satisfy the constraints on minimum and maximum number of consecutive days off (2–3) and no forbidden sequences of shift blocks must occur (8).

It is important to note that the number of employees required for the schedule may vary depending on the sequence of shift blocks (g, b). A schedule that uses a lower or higher number of employees implies a higher or lower weekly workload, respectively, since the same staffing demand is covered by a different number of employees. To obtain a rotational schedule for a workforce with a number of $|E|$ employees, the length of the sequence has to be exactly $|E|\cdot\Omega$. If the subproblem is feasible for a given solution of the master problem, a solution for the *RWS* is obtained.

The subproblem can be seen as the problem of finding a Hamiltonian cycle [1] with a sum of edge weights that is exactly $|E|$ in a graph with node precedence constraints. Let $\mathcal{G}$ be a graph that contains a node $n \in N$ for every shift block (g, b) in the solution of the master problem. If $\lambda_{gb} > 1$, multiple nodes are added for shift block (g, b) according to the value of λ_{gb} in the solution of the master problem. The cardinality of $\mathcal{G}$ is thus defined as $\sum_{g\in G}\sum_{b\in B}\lambda_{gb}$, i. e., the number of blocks in the master problem solution. To ensure coverage of the staffing requirements (Eq. 14), every shift block (g, b) has to be included into the schedule. The subproblem can be stated as follows:

$$\sum_{i\in N}\sum_{j\in N} d_{ij}v_{ij} = |E| \tag{15}$$

$$\sum_{j\in N|i\neq j} v_{ij} = 1 \qquad \forall i \in N \tag{16}$$

$$\sum_{i\in N|i\neq j} v_{ij} = 1 \qquad \forall j \in N \tag{17}$$

$$\sum_{i\in w}\sum_{j\in w} v_{ij} \leq |w| - 1 \qquad \forall w \in W \tag{18}$$

$$v_{ij} = 0 \qquad \forall (i, j) \in K \tag{19}$$

$$v_{ij} \in \{0, 1\} \qquad \forall i \in N, j \in N \tag{20}$$

Constraints (15) enforce a length of $\Omega\cdot|E|$ for the schedule, so that the schedule is feasible for the available number of employees. In the following, an example is given to illustrate how the number of employees varies depending on the block sequence and how edge weights d_{ij} are defined. We assume that the staffing demand is $T = \binom{2\,2\,2\,2\,2\,2\,2}{2\,2\,2\,2\,2\,2\,2}$. The first shift type is a morning shift, denoted as M, and the second shift type is a night shift, denoted by N. For simplicity, we assume that only a single shift block $MMNN$ is allowed. Figure 4 shows two rotational schedules that cover the workforce requirements. In the first column, the number of the employee starting at the beginning of each row is described. It can be seen that Ω has a value of seven since each row includes seven days. The last shift of each row is adjacent to the first shift of the next row. In the last column, the set of start days of blocks which begin in the respective row are shown. Note that the example involves seven shift blocks with start days $\{1, \ldots, 7\}$. Both schedules use exactly the same set of shift block and start day tuples (g, b) and provide the same coverage. Figure 4 shows that the length of the schedule varies depending on the sequence of shift block and start day tuples (g, b). As the length of the schedule needs to match the number of employees in the *RWS*, a sequence is a feasible solution to the second subproblem only if $\frac{|D|}{\Omega} = |E|$.

In the example, the shift block starts exactly once on each $g \in \{1, \ldots, 7\}$. Thus, the graph for the subproblem comprises seven nodes and we will refer to each node by the number of the start day. Now, let d_{ij} denote the number of employees required additionally for the schedule if node i succeeds node j. Then, $\sum_{i\in N}\sum_{j\in N} d_{ij}v_{ij}$ equals the number of employees required for the rotational schedule defined by the block sequence. The graph associated with the exemplary schedules is shown in Fig. 5. The top graph corresponds to the top schedule in Fig. 4, and the bottom graph corresponds to the bottom schedule in Fig. 4. Edge weights show the value of d_{ij}. For example, if edge (4, 5) is selected, i.e., the shift block with start day 4 is succeeded by the shift block with start day 5, a number of two employees are required

[1] A *Hamiltonian cycle* is a cycle that contains all nodes (Bollobas 2013).

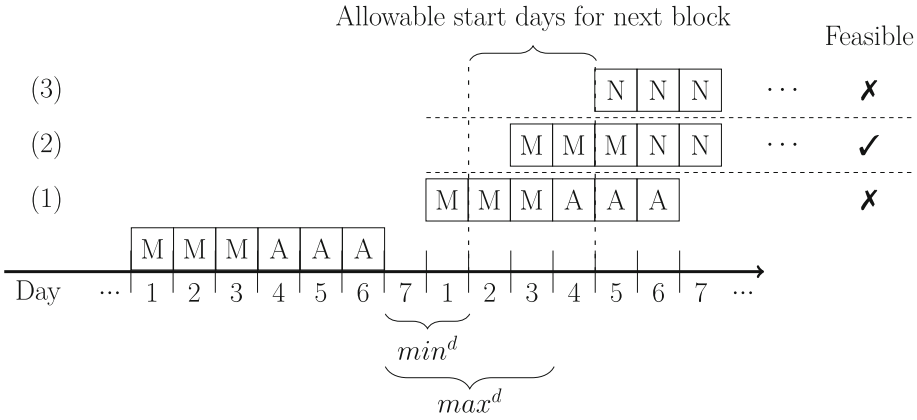

Fig. 3 Illustration of block sequences infeasible owing to minimum and maximum consecutive days off

day

employee	1	2	3	4	5	6	7	block start days
1	**M**	M	N	N				{1}
2		**M**	M	N	N			{2}
3			**M**	M	N	N		{3}
4				**M**	M	N	N	{4}
5					**M**	M	N	{5}
6	N					**M**	M	{6}
7	N	N					**M**	{7}
8	M	N	N					∅

day

employee	1	2	3	4	5	6	7	block start days
1	**M**	M	N	N		**M**	M	{1, 6}
2	N	N					**M**	{7}
3	M	N	N		**M**	M	N	{5}
4	N		**M**	M	N	N		{3}
5				**M**	M	N	N	{4}
6		**M**	M	N	N			{2}

Fig. 4 Varying the block sequence changes the length of the rotational schedule

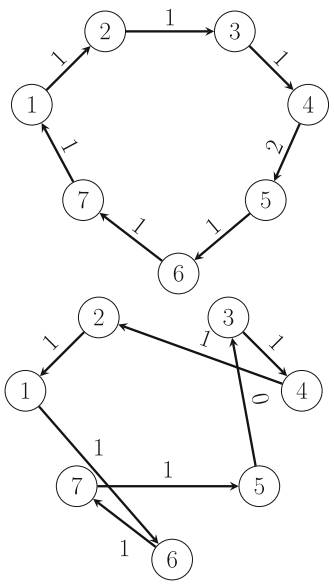

Fig. 5 Graphs of block sequence for the exemplary rotational schedules

$$d_{ij} = \begin{array}{c} \\ i{=}1 \\ i{=}2 \\ i{=}3 \\ i{=}4 \\ i{=}5 \\ i{=}6 \\ i{=}7 \end{array} \begin{array}{c} \begin{array}{ccccccc} j{=}1 & j{=}2 & j{=}3 & j{=}4 & j{=}5 & j{=}6 & j{=}7 \end{array} \\ \begin{pmatrix} 1 & 1 & & & & 1 & 1 \\ 1 & 1 & 1 & & & & 1 \\ 1 & 1 & 1 & 1 & & & \\ & 1 & 1 & 1 & 2 & & \\ & & 0 & 0 & 1 & 1 & \\ & & & 1 & 1 & 1 & 1 \\ 1 & & & & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Fig. 6 Values of matrix d_{ij} with shift block $MMNN$ and $\min^d = 1$, $\max^d = 4$ for start days $\{1, \ldots, 7\}$

additionally and $d_{45} = 2$. Looking at the top schedule in Fig. 4, this sequence occurs between rows 4 to 6. The shift block with start day 4 finishes on the last day of row 4. If the shift block with start day 5 is scheduled in succession, it must begin in row 5 to obtain a feasible schedule. The last shift of the block ends on the first day of row 6. Since both rows 5 and 6 are added to the schedule, an additional two employees are required if edge $(4, 5)$ is selected. Further, consider the case in the bottom schedule of Fig. 4 where the block with start day 5 is succeeded by the block with start day 3. Since the block with start day 5 ends in row 4 and the block with start day 3 ends in the same row, no additional employees are necessary and $d_{53} = 0$. Figure 6 shows matrix d_{ij} for the exemplary schedules in Fig. 4. Empty values indicate that node i cannot succeed node j because the respective blocks do not meet constraints regarding the minimum and maximum number of days off in between.

As the number of shift blocks selected in the master problem corresponds to the nodes in $\mathcal{G}$, every node has to be visited exactly once. Constraints (16–17) of the subproblem formulation ensure that there are exactly one predecessor and successor for each node. In order to avoid subcycles, Constraints (18) forbid every cycle in $\mathcal{G}$ as part of the solution that does not visit every node. Many nodes cannot succeed

each other, owing to the number of days off in between the underlying shift blocks, or forbidden shift sequences. These edges are excluded from the solution by Constraints (19). Figure 3 shows examples how block sequences may be feasible depending on the number of days off in between.

The mathematical problem above cannot be solved directly for a realistic problem size, as the number of subcycles $w \in W$ is too large to explicitly add a constraint for each one. As a result, a well-known branch-and-cut procedure is applied (Sect. 5.2). The solution approaches for the master and subproblem are discussed in more detail in the next section.

5 Solution approach

In this section, an algorithmic framework using the decomposition from the previous section to quickly obtain feasible solutions for the *RWS* is presented. For the master problem, a greedy local search procedure is proposed (Sect. 5.1). Standard solvers are not well suited to solve the master problem, as many solutions of different structures are required. The subproblem is solved with a branch-and-cut procedure (Sect. 5.2).

The **d**ecomposition **h**euristic for the **Rota***ting Workforce Scheduling Problem* (*DH-Rota*) is shown in Algorithm 1. For a specified number of iterations (Algorithm 1, l. 2), a solution to the master problem (Sect. 4.1) is constructed using a greedy local search heuristic with different block inclusion criteria (Algorithm 1, l. 2–6). If a feasible solution for the master problem is found, the subproblem (Sect. 4.2) is solved (Algorithm 1, l. 7–8). Given a feasible solution to the subproblem, the algorithm terminates successfully (Algorithm 1, l. 9–10). Every time a feasible solution for the master problem is obtained, the search strategy α is changed. The symbols are adopted from the previous section. For simplicity of presentation, we assume that they are available to each function as global parameters and do not state the input data separately.

Algorithm 1: Decomposition heuristic for the *RWS*

```
1  α = 1;
2  while iteration limit not reached do
3  |  λ = initialization(α);
4  |  improvement = true;
5  |  while improvement = true do
6  |  |  improvement = two_opt(λ);
7  |  if coverage(λ) = T_ds then
8  |  |  if solve_subproblem(λ) then
9  |  |  |  feasible solution found;
10 |  |  |  terminate;
11 |  |  α = 1 + α mod 3;
```

5.1 Master problem

In every iteration of the algorithm, a solution λ for equations (Sect. 4.1, 14) is constructed (Algorithm 1, l. 3–7). First, function *initialization* (Algorithm 2) greedily chooses shift blocks to improve coverage according to the search strategy α (Algorithm 1, l. 2). Three search strategies are used to improve search diversification:

$$\text{obj}^1(\text{cov}^\delta, \gamma, b) = \begin{cases} \frac{1}{f_b}\text{cov}^\delta, & \text{if } F_3 \neq \varnothing \\ \gamma_{\text{No}_b}\text{cov}^\delta, & \text{otherwise} \end{cases} \tag{21}$$

$$\text{obj}^2(\text{cov}^\delta, \gamma, b) = \text{round}(\gamma_{\text{No}_b}\frac{\text{cov}^\delta}{\text{No}_b}) \tag{22}$$

$$\text{obj}^3(\text{cov}^\delta, \gamma, b) = \text{round}(\gamma_{\text{No}_b}\frac{\text{cov}^\delta}{(\text{No}_b)^2}) \tag{23}$$

Each search strategy selects the next block to include in the solution differently. Once no further improvement in coverage is possible by including additional shift blocks, local search is applied. Function *two_opt* (Algorithm 3) evaluates whether the coverage can be improved by replacing a shift block with a different one (Algorithm 3, l. 5–7). After no further improvement is possible by local search, the solution is checked for feasibility (Algorithm 1, l. 7). The combination of block tuples (g, b) has to exactly cover the staffing requirements T_{ds} for feasibility. If the solution is feasible, it is passed to the subproblem and the block inclusion criterion α is changed.

5.1.1 Initialization

Algorithm *initialization* begins with picking a random weight γ for each block length from the interval $[lb, ub]$ (Algorithm 2, l. 1). A higher spread of weights indicates more blocks of homogeneous length in the solution. Next, a local copy T_{ds}^{copy} of the staffing requirements is defined to capture the missing coverage given the current solution λ (Algorithm 2, l. 2). Every block tuple (g, b) is evaluated regarding the impact on the coverage (Algorithm 2, l. 4–8). The number of shifts included in the block tuple that are not fully covered by the current solution λ is denoted by coverage$^+$. The number of shifts included in the block tuple where additional coverage leads to surplus coverage is denoted by coverage$^-$. We now define the marginal coverage provided by a block tuple as $\text{cov}^\delta = \text{coverage}^+ - \text{coverage}^-$ (Algorithm 2, l. 6–7).

To avoid surplus coverage in the initial solution, the number of surplus shifts covered is penalized if no additional coverage is necessary. Let No_b denote the number of shifts included in block b. The value of δ_{gb} denotes the score of including block tuple (g, b). It is derived on the basis of cov_δ and the length of the shift block No_b. If conflicts between

Algorithm 2: initialization

```
1  γ = Pick random weight from {lb, ..., ub} for each block length;
2  T_ds^copy = Copy staffing requirements T_ds;
3  repeat
4      for g ∈ G do
5          for b ∈ B do
6              coverage^+ = Useful coverage provided by (g, b) according to T_ds^copy;
7              coverage^- = Surplus coverage by adding block (g, b) according to T_ds^copy;
8              δ_gb = obj^α(coverage^+-coverage^-, γ, b);
9      δ^best = max({δ_gb ∀g ∈ G, b ∈ B});
10     if δ^best ≥ 0 then
11         g^best, b^best = Pick random (g, b) where δ_gb = δ^best;
12         λ_{g^best b^best} += 1;
13         Update T_ds^copy by coverage;
14 until δ^best ≤ 0;
15 return λ;
```

Algorithm 3: two_opt

```
   Data: λ
1  for (g1, b1) where λ_{g1 b1} ≥ 0 do
2      for g2 ∈ G do
3          for b2 ∈ B do
4              δ = Change in coverage if (g1, b1) is replaced by (g2, b2);
5              if δ > 0 then
6                  λ_{g1 b1} -= 1;
7                  λ_{g2 b2} += 1;
8                  return true;
9  return false;
```

shift blocks exist, the number of conflicts of a shift block f^b is taken into account. An inclusion should only be carried out for a value of $\delta_{gb} > 0$. Depending on the block inclusion strategy α, a different formula is used to derive δ_{db}. To diversify the search, the marginal coverage cov^{δ} is multiplied with γ_{No_b}. The search strategies $\text{obj}^1 - \text{obj}^3$ are shown by Eqs. (21–23). The first strategy favors the inclusion of long shift blocks, since the marginal coverage is only multiplied by γ. If conflicts between shift blocks exist, the marginal coverage is divided by the number of conflicts f_b instead. To ensure a score that is independent of block length, the score is divided by block length for the second strategy. For the third strategy, δ_{gb} is divided by the square of the block length. This favors the inclusion of shorter blocks.

Once δ_{gb} has been computed for every block tuple, let δ^{best} denote the maximum score for a block inclusion. If $\delta_{best} > 0$, a random block tuple (g, b) where $\delta_{gb} = \delta^{\text{best}}$ is selected (Algorithm 2, l. 11). Rounding the score of δ_{gb} ensures a randomization of the inclusion process, since the size of the set of block tuples (g, b) where $\delta_{gb} = \delta^{\text{best}}$ is increased. The selected block tuple is included in the solution λ and the demand T_{ds}^{copy} is reduced by the additional coverage of (g, b) (Algorithm 2, l. 12–13). Negative values for T_{ds}^{copy} indicate a surplus coverage. Otherwise, $\delta_{best} \leq 0$ and there is no further improvement possible by including an additional block tuple. The initial solution is returned (Algorithm 2, l. 14–15). Typically, the solution that is returned will not be feasible, as a feasible solution needs to exactly cover each shift on every day according to the demand.

5.1.2 Local search

Starting with the initial solution λ, a local search procedure is applied to obtain a feasible solution (Algorithm 1, l. 5–7). Algorithm *two_opt* evaluates for all block tuples (g, b) where $\lambda_{gb} > 0$ whether there is a replacement (g, b) such that the number of shifts that are over- and undercovered, is reduced (Algorithm 3, l. 4). The value of δ counts the reduction in over- and undercoverage. The first such replacement is accepted (Algorithm 3, l. 5–8). The procedure is called by the main algorithm until no replacement can improve the coverage. Next, it is evaluated if a feasible solution was found, i.e., the solution λ exactly covers the staffing requirements T_{ds} (Algorithm 1, l. 7). If a feasible solution was found, the subproblem is solved (Algorithm 1, l. 8-10). Otherwise, the iteration failed and a new iteration starts.

5.2 Subproblem

A subproblem is solved for every feasible solution of the master problem (Algorithm 1, l. 8–11). If the subproblem is feasible, a solution for the problem instance of the *RWS* has been found. The procedure *solve_subproblem* is illustrated in Fig. 7. A mathematical program is constructed with Constraints (15–17) and (19–20) from Sec. 4.2. Constraints (18) are relaxed, and the mathematical program is solved. Once a feasible integer solution is found, Constraint (18) is added for the longest subcycle of the solution and the mathematical program is resolved. Infeasibility of the subproblem indicates that no sequence of block tuples included in the solution of the master problem exists, such that all constraints of the complete problem are satisfied. If a solution without subcycles is obtained, a feasible solution for the overall problem is reached. The algorithm terminates successfully (Algorithm 1, l. 9–10). For search diversification, the block inclusion criterion of the master problem is changed at the end of each iteration of the subproblem (Algorithm 1, l. 11). In the next section, the algorithmic framework is applied to the benchmark set of the *RWS* and evaluated in comparison with other exact and heuristic approaches.

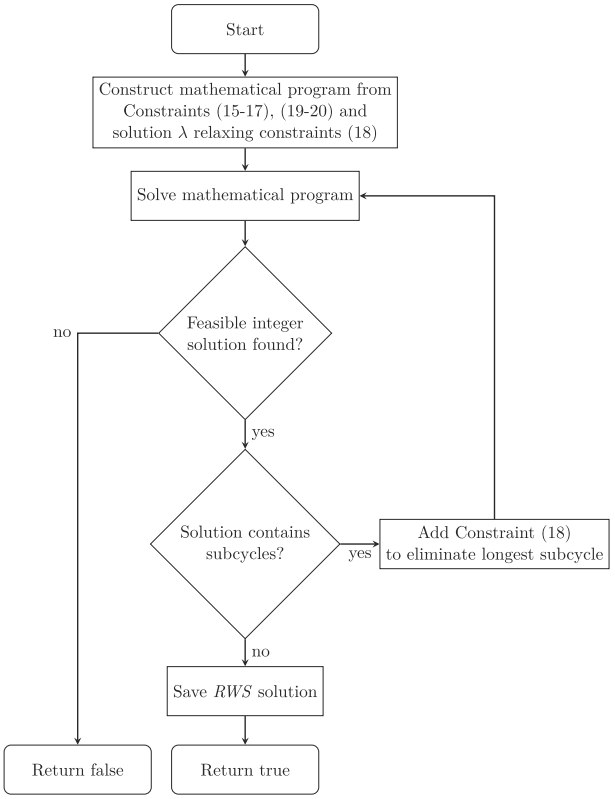

Fig. 7 Flowchart for function *solve_subproblem*(λ)

6 Computational experiments

This section compares the performance of the proposed decomposition heuristic (*DH-Rota*) with selected previous approaches to the *RWS*. Results for our mathematical formulation from Sect. 3.2 are provided as an additional benchmark. Results are reported for the standard benchmarks of the *RWS* and compared to the state-of-the-art solution methods presented by Musliu et al. (2018). The test set is publicly available online[2]. It comprises 2000 test instances of various sizes. Most problem instances were randomly generated to obtain realistic shift scheduling test instances.

In all instances, there is an offset $\Omega = 7$ days between employee start days, such that the schedule comprises a total of $|E| \cdot 7$ days. This is a typical assumption in rotational workforce scheduling, as it ensures that the rotational schedule always rotates over the same weekdays. The test set includes both instances with two and three daily shifts. These comprise a morning, afternoon, and night shift. For an exemplary solution, see Fig. 1 in introduction. The schedule is a feasible solution for a simple problem instance from the benchmark set (instance 2). Table 3 provides some additional detail on the characteristics of the test set.

Table 3 Characteristics of the *RWS* benchmark instances proposed by Musliu et al. (2018)

Parameter	Min.	Mean	Max.
Employees	7	29.9	163
Shifts per week	3.6	4.8	5.6
Feasible shift blocks	3	17.5	56
$\min^{\text{work}}$	2	3.5	4
$\max^{\text{work}}$	5	6	7
$\min^{\text{off}}$	1	1.5	2
$\max^{\text{off}}$	2	3	4
$\lvert F^2 \rvert$	1	2.7	3
$\lvert F^3 \rvert$	0	1.9	4

The table specifies the minimum, mean, and maximum value for several parameters of the *RWS*. For a definition of the notation, see Table 1. It can be seen that there is a large differentiation of key parameter values across the benchmark set. For instance, the number of employees varies between 7 and 163. This parameter has an important influence on the instance size and difficulty. In a mathematical formulation, the number of decision variables typically increases with the number of employees. For *DH-Rota*, the number of shift blocks that need to be considered also tends to increase with the number of employees. The number of shifts per week was computed by dividing the sum of all staffing requirements by the number of employees. It is evident from the minimum and maximum number of shifts per week that the employee workload is highly varied throughout the test set. The number of feasible shift blocks is also specified. Although the benchmark instances do not specify the set of feasible shift blocks B explicitly, it can be easily constructed from the problem instance parameters. Furthermore, different sets of forbidden shift sequences are considered. Note that the staffing requirements also vary for each instance, whereas many different structures are included in the benchmark set.

The *DH-Rota* was implemented in *C++* and compiled with the Visual Studio 2017 compiler under Windows 10. Formulation [(1)–(13), Sect. 3.2] was solved with *Gurobi 8.0*. The experiments were carried out with an *intel i7-8550U* processor and 16 GB of RAM. In Musliu et al. (2018), experiments were conducted on a different processor. However, according to benchmarks[3] which have been previously used for scaling run times to a comparable level, the processor used in this paper has a lower performance (Carrabs et al. 2018). Experiments for the mathematical formulation proposed by Musliu et al. (2018) were implemented and rerun

[2] https://www.dbai.tuwien.ac.at/staff/musliu/benchmarks/.

[3] https://setiathome.berkeley.edu/cpu_list.php.

also using *Gurobi 8.0* for the computational experiments. Since the specific implementation of the other solution methods is unknown, the results are directly compared. To be conservative, no scaling to the advantage of this paper is conducted. Finally, to maintain comparability with their results, the run time is limited to 200 s in all experiments.

6.1 Comparison with exact and heuristic methods

Table 4 presents a comparison of the run time of *DH-Rota* and selected exact and heuristic approaches for the *RWS* benchmark instances. In the first column, the name of the solution approach is provided. The second column specifies the type of solution approach, while four exact and two heuristic methods are compared. The third and fourth columns specify the number of instances where a feasible schedule was identified and the average run time. The fifth and sixth columns indicate the number of instances proven infeasible and the average run time. In the last column, the number of instances is stated where neither a feasible schedule nor a proof of infeasibility was achieved within the time limit of 200 s. The results in the last column show that no approach could find a feasible schedule or prove infeasibility for the complete test set within the time limit. Furthermore, the results of the exact mathematical programming approach proposed by Musliu et al. (2018) show that a high number of instances are infeasible. Their exact mathematical programming approach has solved the highest number of instances in the sense that either a feasible schedule or a proof of infeasibility is obtained. It was able to identify a feasible schedule for 1321 instances and provided a proof of infeasibility for 667 instances, leaving only 12 instances that were not solved within the time limit. Their constraint programming formulation exhibits a lower run time for feasible instances, but it could only proof infeasibility for 523 instances with a significantly higher average run time.

Looking at the results, it turns out that the mathematical formulation from Sect. 3.2 is not competitive. While the average run time for instances where a feasible schedule has been obtained is lower than the run time of the MathSAT and MC-T, both approaches have provided a feasible schedule for a higher number of problem instances. Furthermore, all other exact approaches have solved more infeasible instances. The mathematical formulation from Musliu et al. (2018) detected the highest number of infeasible instances within the time limit and also has the lowest average run time for detecting infeasibility.

As a heuristic approach, *DH-Rota* does not provide a proof of infeasibility. It will either find a feasible schedule or terminate when the time limit is reached. *DH-Rota* has obtained a feasible schedule for 1322 instances, which is the same number as the constraint programming formulation from Musliu et al. (2018). However, it is interesting to note that *DH-Rota* is able to find a solution for many feasible instances extremely quickly. For the constraint programming formulation from Musliu et al. (2018), an average run time of 5.0 s is reported across the set of instances where a feasible schedule has been obtained. The constraint programming formulation has found a feasible solution for 1322 test instances, and it is the fastest exact method for feasible instances of the *RWS*. In comparison, the average run time of *DH-Rota* across the set of feasible instances solved is 0.18 s. For the 1322 feasible instances solved by *DH-Rota*, a solution was found in less than 0.1 s for more than 90%. Further, 97.7% of these problem instances were solved within a second.

Thus, while *DH-Rota* does not improve the state of the art of the exact methods from Musliu et al. (2018), it improves the results of heuristic methods and it is able to provide solutions for feasible instances very efficiently. In the following, we provide a more detailed analysis of *DH-Rota* run times on feasible instances.

6.2 Analysis of run times of *DH-Rota*

This section provides a more detailed comparison of *DH-Rota* with the other solution methods executed during our experiments on a set of feasible instances, to further investigate the efficiency of the proposed decomposition approach. Figure 8 shows the distribution of run times on a set of feasible instances for *DH-Rota* and the mathematical formulation from Musliu et al. (2018). In order to ensure comparability, the figure only includes run times for feasible instances solved by both solution approaches. The set of feasible instances solved by each approach differs slightly, while the intersection contains a number of 1317 instances. Thus, each part of the figure shows run time results for the same set of instances. The results were divided into multiple categories according to the number of employees. Each category and box plot include the results of more than 300 test instances. A logarithmic scale is used to improve the readability of the box plots at low run time values. Further, to visually distinguish multiple outliers with similar run time values, some outliers were shifted horizontally.

Looking at Fig. 8, the number of employees clearly has an impact on the run time of both solution methods, whereas the run time increases with a higher number of employees. It is evident from the box plot graphs that *DH-Rota* solves many feasible instances very efficiently in comparison with the mathematical formulation. The upper whiskers of the box plots for DH-Rota are strictly below the lower whiskers of the box plots for the mathematical formulation. However, there are more outliers for *DH-Rota* in comparison with the mathematical formulation. The feasible space of the master problem changes with respect to the specific problem instance characteristics. There might be a different number of feasible solutions for the master problem, and further-

Table 4 Comparison of selected exact and heuristic methods for the *RWS*

Solution method	Type	Feasible	Avg. run time	Infeasible	Avg. run time	Unknown
MP (This paper, Sect. 3.2)	Exact	1109	12.85	237	27.07	654
MP (Musliu et al. 2018, Gurobi)	Exact	1321	11.50 s	**667**	**8.35 s**	**12**
CP (Musliu et al. 2018, Chuffed)*	Exact	**1322**	5.0 s	523	47.8 s	155
MathSat (Erkinger and Musliu 2017)*	Exact	1198	32.1 s	272	126.7 s	530
MC-T (Musliu 2006)*	Heuristic	1212	23.3 s	–	tl	788
DH-Rota (This paper)	Heuristic	**1322**	**0.18 s**	–	tl	678

The bold values indicate the best solution approach with respect to the criterion of the column
MP mathematical programming, *CP* constraint programming, *tl* time limit
*Run times reported by Musliu et al. (2018)

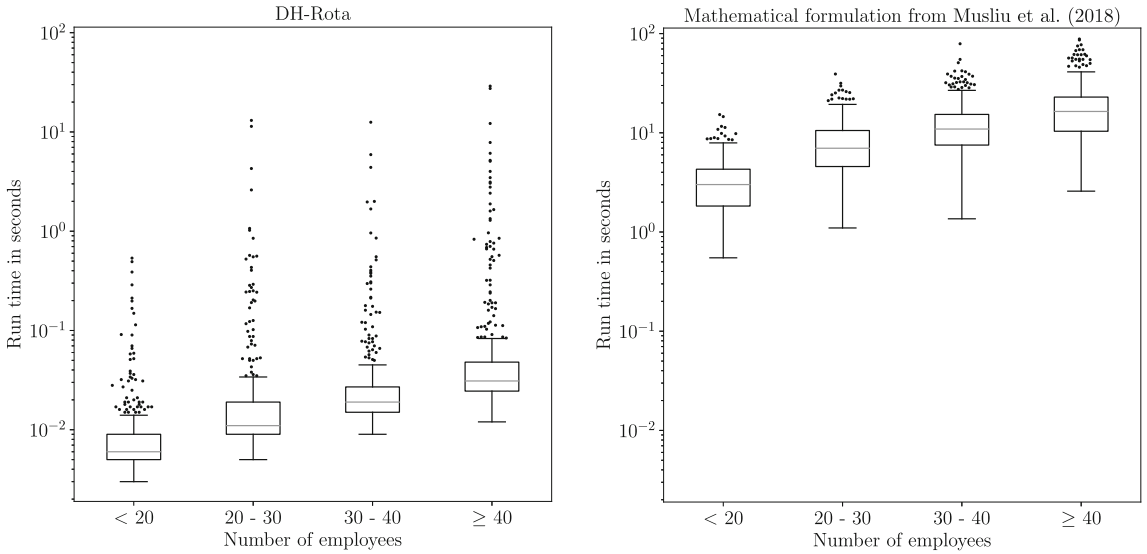

Fig. 8 Comparison of run times for the set of instances where a feasible schedule has been identified by both solution methods (intervals for number of employees include the lower bound and exclude the upper bound)

Table 5 Number of feasible instances solved for each category by the respective solution method

#Shifts	2				3			
#Employees*	< 20	20–30	30–40	≥ 40	< 20	20–30	30–40	≥ 40
DH-Rota	38	47	46	57	274	271	271	318
MP (Sect. 3.2)	34	39	40	46	262	214	224	250
MP (Musliu et al. 2018)	38	47	46	57	274	271	269	319

*Intervals include the lower bound and exclude the upper bound

more, the number of master problem solutions for which the subproblem is feasible also varies. Since *DH-Rota* utilizes a sampling-based heuristic, the unknown probability to sample a block combination for which the subproblem is feasible is different for each problem instance. The number of required iterations and run time of *DH-Rota* increases with a lower probability of sampling a feasible block combination. This leads to a large range of run times for *DH-Rota*. However, it is evident that most feasible instances included in the box plots are solved very efficiently by *DH-Rota*. Furthermore, in each category the maximum outlier of *DH-Rota* lies below the upper whisker of the box plot for the mathematical formulation.

Table 6 Average run time and standard deviation (in s) across the set of feasible instances that were solved by the respective solution method for each category

#Shifts	2								3							
#Employees*	<20		20–30		30–40		≥ 40		<20		20–30		30–40		≥ 40	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
DH-Rota	0.03	0.09	0.29	1.64	0.03	0.07	0.09	0.24	0.02	0.05	0.11	0.85	0.15	0.9	0.44	2.45
MP (Sect. 3.2)	2.22	4.79	2.27	4.98	7.53	18.18	10.31	28.73	5.38	13.85	11.15	23.95	20.41	36.72	19.79	28.36
MP (Musliu et al. 2018)	1.33	0.64	2.47	1.23	4.04	1.89	5.53	2.33	3.75	2.24	9.4	5.68	14.84	9.51	21.82	14.01

*Intervals include the lower bound and exclude the upper bound.

In addition to the number of employees, the number of shifts is an influencing factor of the run time. The number of decision variables considered by the exact approaches directly depends on the number of shifts. In Table 5, the number of feasible instances solved by the solution methods executed during the computational experiments is shown, differentiated by the number of shifts and employees. Table 6 shows the average run time and standard deviation for each category of instances. The data for each solution approach only include results for the run time on feasible instances solved, as specified in Table 5. The data from Table 6 show that the average run time of the mathematical formulation proposed by Musliu et al. (2018) strictly increases with a higher number of shifts or employees. A similar trend is evident for the mathematical formulation (Sect. 3.2) and *DH-Rota*. However, comparing the average run time of the mathematical formulation (Sect. 3.2) for the case of 3 shifts between the categories 20–30 and ≥ 40 employees, the average run time slightly decreases from 20.41 to 19.79 s. It is important to note that there are a large number of instances where the mathematical formulation (Sect. 3.2) did not obtain a feasible solution in both categories. Additionally, the mathematical formulation (Sect. 3.2) exhibits the highest standard deviation among all solution approaches in each category. Furthermore, the influence of the number of shifts on the run times of *DH-Rota* does not appear very consistent from the data in Table 6. While the number of shifts increases the decision variables considered by exact approaches, it does not have such a direct influence on *DH-Rota*. Looking at the standard deviation, *DH-Rota* exhibits the lowest values in most categories. Relative to the average run time, the standard deviation of *DH-Rota* is higher than that of the mathematical formulation of Musliu et al. (2018). This is also evident from the higher number of outliers for *DH-Rota* and the large range of run times shown by the box plots in Fig. 8.

Overall, the results indicate that the new decomposition approach efficiently exploits the problem structure and provides a major reduction in run times for feasible *RWS* problem instances. While *DH-Rota* does not improve the results of the state-of-the-art exact approaches presented in Musliu et al. (2018), it is able to solve feasible instances of the *RWS* very efficiently and thus confirms the benefits of the new decomposition approach.

7 Conclusion

Rotational workforce scheduling is an important tool for organizations with homogeneous skills among employees and staffing requirements. This study has presented a novel decomposition approach (*DH-Rota*) for the Rotating Workforce Scheduling Problem (*RWS*). The heuristic first constructs a fixed set of shift blocks that covers the staffing requirements. Then, a network model is used to determine whether a feasible sequence of shift blocks exists. By considering a fixed set of shift blocks to cover the demand, the problem complexity can be greatly reduced. We have proposed a new greedy heuristic to find combinations of blocks that cover the staffing requirements and an exact branch-and-cut algorithm that determines whether there exists a feasible sequence such that the constraints on days off and forbidden blocks sequences are satisfied.

Computational experiments confirm the efficiency of the decomposition approach and heuristic. The heuristic approach very quickly provides solutions for feasible instances. Although it does not improve the state of the art for exact methods, the average run time across the set of feasible instances solved by *DH-Rota* is only 0.18s. This greatly improves the results of previous heuristics for the *RWS*. Thus, the proposed approach is able to quickly generate feasible schedules for real-world Rotating Workforce Scheduling Problems.

The decomposition approach is not limited to the *RWS*. Various future studies that build on the ideas of the decomposition are apparent. It can be extended by additional practical constraints. Further studies could thus investigate the applica-

bility and performance for different application areas. These include various scheduling problems where cyclic schedules are desired, such as nurse rostering. Another possible area of future research would be a more systematic approach to finding combinations of blocks that cover the staffing requirements. If the set of solutions to this covering problem is systematically enumerated, an exact approach is obtained.

Acknowledgements Open Access funding provided by Projekt DEAL. The author would like to thank three anonymous referees for their valuable and detailed comments. Their comments have helped to greatly improve the manuscript.

References

Åkerstedt, T. (2003). Shift work and disturbed sleep/wakefulness. *Occupational Medicine*, *53*(2), 89–94.

Baker, K. R. (1976). Workforce allocation in cyclical scheduling problems: A survey. *Journal of the Operational Research Society*, *27*(1), 155–167.

Balakrishnan, N., & Wong, R. (1990). A network model for the rotating workforce scheduling problem. *Networks*, *20*(1), 25–42.

Bartholdi, J. J., Orlin, J. B., & Ratliff, H. D. (1980). Cyclic scheduling via integer programs with circular ones. *Operations research*, *28*(5), 1029–1257.

Becker, T., Steenweg, P. M., & Werners, B. (2019). Cyclic shift scheduling with on-call duties for emergency medical services. *Health Care Management Science*, *22*, 676–690.

Bollobas, B. (2013). *Modern graph theory*., Graduate texts in mathematics New York: Springer.

Brucker, P., Burke, E. K., Curtois, T., Qu, R., & Berghe, G. V. (2010). A shift sequence based approach for nurse scheduling and a new benchmark dataset. *Journal of Heuristics*, *16*(4), 559–573.

Brucker, P., Qu, R., & Burke, E. (2011). Personnel scheduling: Models and complexity. *European Journal of Operational Research*, *210*(3), 467–473.

Burke, E., De Causmaecker, P., & Van den Berghe, G. (1999). A hybrid tabu search algorithm for the nurse rostering problem. In B. McKay, X. Yao, C. S. Newton, J. H. Kim, & T. Furuhashi (Eds.), *Simulated Evolution and Learning* (pp. 187–194). Berlin: Springer.

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

Carrabs, F., Cerulli, R., Pentangelo, R., & Raiconi, A. (2018). A two-level metaheuristic for the all colors shortest path problem. *Computational Optimization and Applications*, *71*(2), 525–551.

Erkinger, C., & Musliu, N. (2017) Personnel scheduling as satisfiability modulo theories. In *IJCAI international joint conference on artificial intelligence* (pp. 614–621).

Ernst, A. T., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, *153*(1), 3–27.

Kiermaier, F., Frey, M., & Bard, J. F. (2016). Flexible cyclic rostering in the service industry. *IIE Transactions (Institute of Industrial Engineers)*, *48*(12), 1139–1155.

Laporte, G. (1999). The art and science of designing rotating schedules. *The Journal of the Operational Research Society*, *50*(10), 1011–1017.

Laporte, G., Nobert, Y., & Biron, J. (1980). Rotating schedules. *European Journal of Operational Research*, *4*(1), 24–30.

Millar, H., & Kiragu, M. (1998). Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming. *European Journal of Operational Research*, *104*(3), 582–592.

Moerz, M., Musliu, N. (2004) Genetic algorithm for rotating workforce scheduling. In *Proceedings of second IEEE international conference on computational cybernetics* (pp. 121–126).

Musliu, N. (2006). Heuristic methods for automatic rotating workforce scheduling. *International Journal of Computational Intelligence Research*, *2*(4), 309–326.

Musliu, N. (2011) Constructing cyclic staff schedules by iterated local search. In *MIC 2011: The IX metaheuristics international conference* (pp. 3–5).

Musliu, N. (2013) Applying machine learning for solver selection in scheduling: A case study (pp. 6–8).

Musliu, N., Gärtner, J., & Slany, W. (2002). Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*, *118*(1–2), 85–98.

Musliu, N., Schutt, A., & Stuckey, P. J. (2018). Solver independent rotating workforce scheduling. In W. J. van Hoeve (Ed.), *Integration of constraint programming, artificial intelligence, and operations research* (pp. 429–445). Cham: Springer.

Mutingi, M., & Mbohwa, C. (2015a). Nurse scheduling: A fuzzy multi-criteria simulated metamorphosis approach. *Engineering Letters*, *23*(3), 222–231.

Mutingi, M., & Mbohwa, C. (2015b) A multi-criteria approach for nurse scheduling fuzzy simulated metamorphosis algorithm approach. In *IEOM 2015—Proceeding of 5th international conference on industrial engineering and operations management*.

Rocha, M., Oliveira, J. F., & Carravilla, M. A. (2013). Cyclic staff scheduling: Optimization models for some real-life problems. *Journal of Scheduling*, *16*(2), 231–242.

Rocha, M., Oliveira, J. F., & Carravilla, M. A. (2014). A constructive heuristic for staff scheduling in the glass industry. *Annals of Operations Research*, *217*(1), 463–478.

Triska, M., & Musliu, N. (2011). *A constraint programming application for rotating workforce scheduling* (pp. 83–88). Berlin: Springer.

Valouxis, C., & Housos, E. (2000). Hybrid optimization techniques for the workshift and rest assignment of nursing personnel. *Artificial Intelligence in Medicine*, *20*(2), 155–175.

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