Investigating Constraint Programming for Real-Life Rotating Workforce Scheduling Problems

Lucas Kletzander (✉) and Nysret Musliu

Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling
DBAI, TU Wien, Vienna, Austria
{lkletzan,musliu}@dbai.tuwien.ac.at

Abstract. In many professions daily demand for different shifts varies during the week. The rotating workforce scheduling problem deals with the creation of repeating schedules for such demand and is therefore of high practical relevance. This paper describes solving this real-life problem with several new practically relevant features. This includes early recognition of certain infeasibility criteria, complex rest time constraints regarding weekly rest time, and optimization goals to deal with optimal assignments of free weekends. We introduce a state-of-the-art constraint model and evaluate it with different extensions. The evaluation shows that many real-life instances can be solved to optimality using a constraint solver. Our approach is under deployment in a state-of-the-art commercial solver for rotating workforce scheduling. In order to analyse strengths and weaknesses of different solution methods, we use Instance Space Analysis to perform an in-depth evaluation on the problem.

Keywords: Constraint Programming, Workforce Scheduling, Instance Space Analysis

1 Introduction

In many professions, different shifts are required to cover varying requirements including areas like health care, protection services, transportation, manufacturing or call centers. This problem may surface in many shapes, using different demands and constraints.

In several applications it can be beneficial to obtain a rotating schedule where each employee rotates through the same sequence of shifts and days off across several weeks, however, at different offsets within the rotation. As the design of shift schedules highly influences the work-life balance of the employees, such problems are subject to a wide range of constraints, dealing not only with the demand for employees in different shifts, but also legal and organizational constraints that determine allowed shift assignments.

Due to its importance, there has been ongoing research on the rotating workforce scheduling (RWS) problem, and results have found their way into commercial software like the Shift Plan Assistant (SPA) by XIMES GmbH.

This paper presents contributions in two different areas regarding RWS from two recent papers. First [11], we extend the problem in several different ways based on the needs in real-life situations, including fast detection of infeasible instances, complex constraints to respect weekly rest times, as well as soft constraints optimizing free weekends in the schedule, turning the satisfaction problem into an optimization problem. To solve the problem we provide a new constraint model and implement it in the constraint modelling language MiniZinc. Second [12], we use instance space analysis on two different exact models and a metaheuristic approach to show different strong and weak areas in the instance space as well as a good coverage of the instance space when combining the strengths of the algorithms.

2 Related Work

Due to high practical relevance various versions of employee scheduling problems have been investigated for several decades. For an overview of existing literature, refer to surveys like [4, 9, 3, 7].

The rotating workforce scheduling problem can be classified as a single-activity tour scheduling problem with non-overlapping shifts and rotation constraints [1, 21] and is known to be NP-complete [6].

So far the problem has been addressed with a range of different methods. Complete approaches include a network flow formulation [2], integer linear programming [14], several constraint programming formulations [13, 19, 15, 26] and an approach with satisfiability modulo theories [8]. There is also work on heuristic approaches [17, 18], the creation of rotating schedules by hand [13], and using algebraic methods [10].

The current state-of-the-art complete method for standard RWS was introduced by [20]. It uses a solver independent formulation in the MiniZinc constraint language, either with a direct representation or using a regular automaton, and applies both the lazy clause generation solver Chuffed and the MIP solver Gurobi. It is the first complete method able to solve the standard benchmark set of 20 instances and introduces new benchmark instances that we also use for comparison.

Existing work on RWS mostly deals with the standard version of the problem, requiring any feasible solution, or delegates the selection of preferred solutions to the user in an interactive process [19]. While standard RWS already has practical relevance, the extensions allow to deal with more complex issues and provide solutions that are of higher value in real-life applications.

Instance Space Analysis is a methodology developed by Smith-Miles and coworkers [23, 24, 16] in recent years, by extending the Algorithm Selection Problem framework of Rice [22, 25]. Instances are represented as a feature vector that captures the intrinsic difficulty of instances for various algorithms (or models or parameter settings). By constructing a 2-d projection of a feature-vector representation of instances, Instance Space Analysis (ISA) allows us to visualize the distribution and diversity of existing benchmark instances, assess the adequacy

of the features, identify and measure the algorithm's regions of strength (footprint) and weaknesses, and distinguish areas of the space where it may be useful to generate additional instances to support greater insights.

3 Problem Definition for Standard RWS

A rotating workforce schedule consists of the assignment of shifts or days off to each day across several weeks for a certain number of employees. Table 1 shows an example for four employees (or four equal-sized groups of employees), assigning the three shift types day shift (D), afternoon shift (A), and night shift (N). Each employee starts their schedule in a different row, moving from row i to row $i \bmod n+1$ (where n is the number of employees) in the following week.

Table 1. Example schedule for 4 employees

Empl.	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	D	D	D	D	N	N	-
2	-	-	A	A	A	A	N
3	N	N	-	-	D	D	D
4	A	A	N	N	-	-	-

3.1 Problem Specification

We start by defining the basic version of the rotating workforce scheduling problem and recall definitions and notation by [19] and [20]. Extensions to the formulation are introduced in the next section. We define:

- n: Number of employees.
- w: Length of the schedule, typically $w = 7$ as the demands repeat in a weekly cycle. The total length of the planning period is $n \cdot w$, as each employee rotates through all n rows.
- $\mathbf{A}$: Set of work shifts (activities), enumerated from 1 to m, where m is the number of shifts. A day off is denoted by a special activity O with numerical value 0 and we define $\mathbf{A}^+ = \mathbf{A} \cup \{O\}$.
- R: Temporal requirements matrix, an $m \times w$-matrix where each element $R_{i,j}$ corresponds to the number of employees that need to be assigned shift $i \in \mathbf{A}$ at day j. The number of employees o_j that need to be assigned a day off on day j can be calculated by $o_j = n - \sum_{i=1}^{m} R_{i,j}$.
- ℓ_w and u_w: Minimal and maximal length of blocks of consecutive work shifts.
- ℓ_s and u_s: Minimal and maximal lengths of blocks of consecutive assignments of shift s given for each $s \in \mathbf{A}^+$.
- Forbidden sequences of shifts: Any sequences of shifts (like $N\ D$, a night shift followed by a day shift) that are not allowed in the schedule. This is

typically required due to legal or safety concerns. In practice it is usually sufficient to forbid sequences of length 2 or sequences of length 3 where the middle shift is a day off. These are also the kind of restrictions used in the benchmark instances for rotating workforce scheduling.
In our model, we use a set $\mathbf{F}_s^2 \subseteq \mathbf{A}$ for each $s \in \mathbf{A}$ to denote forbidden sequences of length 2, such that $x \in \mathbf{F}_s^2$ declares that shift x must not follow shift s.
Forbidden sequences of length 3 are given as a set $\mathbf{F}_3$ of arrays of length 3 containing elements of $\mathbf{A}^+$. This definition could be extended to arbitrary lengths ℓ using corresponding sets $\mathbf{F}_\ell$.

The task is to construct a cyclic schedule S, represented as an $n \times w$-matrix, where each $S_{i,j} \in \mathbf{A}^+$ denotes the shift or day off that employee i is assigned during day j in the first period of the cycle. The schedule for employee i through the whole planning period consists of the cyclic sequence of all rows of S starting with row i.

Instead of the matrix representation, the same schedule can also be represented as an array T which is equal to the schedule of the first employee, where T_i denotes the shift assignment on day i with $1 \leq i \leq n \cdot w$. As the schedule is cyclic, we could choose any day in the schedule to correspond to the first element in T. We define the offset o with $0 \leq o < w$ to denote the position of the first element in T within the schedule, i.e., its day of the week. This representation is beneficial for some of the following constraints.

3.2 Constraint Model

Our main model for standard RWS uses a direct representation of the constraints based on [20]. Some aspects are modelled in a different way, most notably the different way to deal with cyclicity using the offset o.

For any array or matrix the indices are modulo its dimension. Within this description, such modulo operations are omitted for better readability. We define $\mathbf{N} = \{1, \ldots, n\}$, $\mathbf{W} = \{1, \ldots, w\}$ and $\mathbf{NW} = \{1, \ldots, n \cdot w\}$.

The following equations model the demand.

$$\sum_{i=0}^{n-1} (T_{d+w \cdot i} = s) = R_{s,d+o} \qquad \forall d \in \mathbf{W}, s \in \mathbf{A} \quad (1)$$

$$\sum_{i=0}^{n-1} (T_{d+w \cdot i} = O) = n - \sum_{i=1}^{m} R_{i,d+o} \qquad \forall d \in \mathbf{W} \quad (2)$$

Equation (1) models the demand for each day d and each shift type s. The left side counts occurrences of s on day d, the right side uses the offset to access the correct column of the demand matrix. Equation (2) is a redundant constraint

that counts the number of day-off assignments for each day d. This is not necessary to obtain a complete model of the problem, however, constraint satisfaction solvers can benefit from such redundant definitions.

The next equations introduce symmetry breaking constraints which are also used to make dealing with the cyclic nature of the problem easier in following constraints.

$$T_1 \neq O \tag{3}$$

$$T_{n \cdot w} = O \tag{4}$$

Equations (3) and (4) declare that the first element of T has to hold a working shift, while the last element of T has to hold a day off. In principle any day of the planning period could be used as the first day as it is cyclic. Taking into account that every reasonable rotating workforce scheduling problem contains at least one working day and at least one day off, we can set the first element of T to align with the beginning of a working block.

This has two advantages. First, it eliminates several symmetric versions of the same solution, reducing cyclic occurrences of the same solution from $n \cdot w$ possible notations to the number of working blocks within the solution. Second, this guarantees that blocks of the same shift type or working blocks (consecutive days without day-off assignments) can never cycle across the end of T, eliminating the need to deal with cyclicity in their definition.

Next, constraints for the lengths of shift blocks and working blocks are defined. Note that according to [20] the regular constraint does not improve results in combination with the solver Chuffed, which is why we stick with the direct representation.

$$\forall j \in \{1, \ldots, \ell_s - 1\} : T_{i+j} = s \qquad \forall s \in \mathbf{A}^+, i \in \mathbf{NW}, T_i = s, T_{i-1} \neq s \tag{5}$$

$$i + u_s > n \cdot w \vee \exists j \in \{\ell_s, \ldots, u_s\} : T_{i+j} \neq s \qquad \forall s \in \mathbf{A}^+, i \in \mathbf{NW}, T_i = s, T_{i-1} \neq s \tag{6}$$

$$\forall j \in \{1, \ldots, \ell_w - 1\} : T_{i+j} \neq O \qquad \forall i \in \mathbf{NW}, T_i \neq O, T_{i-1} = O \tag{7}$$

$$i + u_w > n \cdot w \vee \exists j \in \{\ell_w, \ldots, u_w\} : T_{i+j} = O \qquad \forall i \in \mathbf{NW}, T_i \neq O, T_{i-1} = O \tag{8}$$

Equation (5) defines the minimum block length for all shift types including day-off assignments. For all elements T_i containing shift s, where the block starts at i (corresponding to $T_{i-1} \neq s$), the next elements of T until the minimum length must also contain shift s. Equation (6) defines the maximum block length, stating that no later than u_s elements after the block start a different shift type has to occur. Additionally, if i is too close to the end of T, the block will end anyway, giving rise to the inequality part.

Equations (7) and (8) define the same constraints for working blocks, using ℓ_w and u_w as bounds and checking for any working shift ($\neq O$) instead of a specific shift type s.

Finally the forbidden sequences need to be modelled.

$$T_i \neq s \vee T_{i+1} \notin \mathbf{F}_s^2 \qquad \forall s \in \mathbf{A}, i \in \mathbf{NW} \quad (9)$$

$$\exists j \in \{1, \ldots, \ell\} : X_j \neq T_{i+j-1} \qquad \forall X \in \mathbf{F}_\ell, i \in \mathbf{NW} \quad (10)$$

Equation (9) models sequences of length 2, denoted by the set $\mathbf{F}_s^2$ of shift types not allowed to follow shift type s. Forbidden sequences of arbitrary length ℓ are modelled in (10), where for each possible match of each forbidden sequence at least one element must differ from the forbidden sequence.

Further symmetry breaking constraints might be applied to determine the offset o if certain conditions hold.

4 Problem Extensions

When using RWS in practical applications, the need for additional constraints and optimization goals arises. Therefore, [11] introduces new extensions to RWS in order to apply it to more real-life scenarios. This section summarizes the main contributions.

4.1 Detecting Infeasible Instances

The standard benchmark data set consists of 20 instances derived from real life scenarios, all of them admitting feasible solutions. However, the larger instance data set by [20] also includes infeasible instances. The results show that the solver Chuffed also used by us has difficulties identifying those instances. However, in practice it is important to provide fast feedback to the user when they give infeasible settings so that they can correct their input.

Two particular infeasibility tests are introduced. As these are defined on input parameters only, several infeasible instances can be detected already while compiling the instance for the solver, while there is still one consistent formulation of the problem.

The first constraint is based on the fact that some combinations of weekly demand profiles for individual shifts in combination with minimum and maximum block lengths for this shift already lead to infeasibility.

Another observation is the fact that in a cyclic schedule the number of work blocks and the number of free blocks is equal. On the other hand, both for work blocks and free blocks a minimum and maximum number of blocks can be calculated from the required number of shifts and the allowed block lengths. We introduce two possibilities to use the block bounds for redundant constraints. The first uses a global cardinality constraint (`EXT1`), the second possibility uses counting arrays for the current number of blocks at each position in the array (`EXT2`).

4.2 Weekly Rest Time

While forbidden sequences of shifts can be used to handle minimum free time between consecutive shifts, work regulations often contain different, more complex regulations for free time. Real-world scenarios often need to consider a weekly rest time. Typically once a week a certain minimum amount of time has to be free without interruption. Further, it might be possible to have exceptions once every few weeks where the weekly rest time might be shorter according to certain rules.

This can be included by maintaining the current amount of rest time at the start of each shift and using this to enforce the required rest periods each week.

4.3 Optimizing Free Weekends

In the previous SPA implementation the user was presented a choice in several stages of the algorithm, potentially selecting from a large number of feasible solutions. However, defining properties of beneficial solutions beforehand and including these definitions in the model allows to transform the satisfaction problem into an optimization problem and to shift the selection process to the solver.

Shift work can be very disruptive to the social life of employees, e.g., social interactions with friends and family might be hard to schedule as free time is arranged in various different patterns compared to employees with regular free weekends.

Therefore, we chose to optimize the free time on weekends and provide different measurements of desirable weekend schedules. The first optimizes the number of weekends where both Saturday and Sunday are free, optionally enforcing no night shift on Friday as well. Further options deal with minimizing the maximum distance between free weekends or minimizing the sum of squared distances between free weekends in order to incorporate a better distribution of free weekends.

4.4 Results

We use the lazy clause generation solver Chuffed [5] as the solver for our models. The results show `EXT1` and `EXT2`, especially in combination, to be very efficient in detecting infeasible instances in very little computational time. When adding new real-life requirements in the form of complex weekly rest time constraints we are still able to solve the majority of the benchmark instances in short computational time. Introducing new objectives to optimize the scheduling of free weekends, we show that for the majority of instances the optimum can be found and proven in short computational time. Moreover, the output of intermediate solutions allows the user to decide about the trade-off between runtime and quality while running the solver.

In total this provides several improvements to the state-of-the-art modelling of the problem that are currently being integrated as a core component of the next iteration of the Shift Plan Assistant software.

5 Instance Space Analysis

In order to explain what causes hardness in instances and which models work best on which instances, we use Instance Space Analysis (ISA). For this we need a diverse set of instances and features that are able to explain the problem hardness.

In [12], we present a set of features aiming to describe the information required for algorithm selection and instance space analysis. We perform ISA using the toolkit MATILDA, revealing gaps in the coverage of the range of real-life scenarios for a set of 2000 instances for the problem. Therefore, 4000 new instances are created and extended analysis is performed on a more diverse selection of instances.

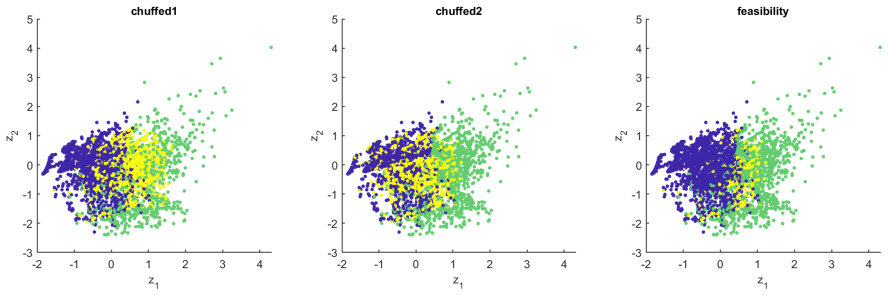

Fig. 1. Algorithm results for the extended instance set. Feasible solutions in blue, infeasible in green, timeout in yellow.

The results in Fig. 1 of the two extended models `EXT1` (chuffed1) and `EXT2` (chuffed2) on the extended instance set show different strong and weak areas in the instance space as well as a transition from feasible to infeasible instances including a more challenging area of instances at this transition. They also show that combining both models gives fast results on the majority of instances as their strong and weak areas complement well.

6 Conclusion

In this paper we presented work on the RWS problem, a real-world scheduling problem, that is solved using constraint programming. We presented several extensions for the original problem that are included in our constraint programming models and can improve results as well as incorporate more complex real-life constraints. Further, we applied instance space analysis on the problem in order to extend an existing instance set and investigate the performance of different models and algorithms.

Acknowledgements The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

References

1. Baker, K.R.: Workforce allocation in cyclical scheduling problems: A survey. Journal of the Operational Research Society **27**(1), 155–167 (1976)
2. Balakrishnan, N., Wong, R.T.: A network model for the rotating workforce scheduling problem. Networks **20**(1), 25–42 (1990)
3. Van den Bergh, J., Belin, J., De Bruecker, P., Demeulemeester, E., De Boeck, L.: Personnel scheduling: A literature review. European Journal of Operational Research **226**(3), 367–385 (May 2013). https://doi.org/10.1016/j.ejor.2012.11.029
4. Burke, E.K., De Causmaecker, P., Berghe, G.V., Van Landeghem, H.: The State of the Art of Nurse Rostering. Journal of Scheduling **7**(6), 441–499 (Nov 2004). https://doi.org/10.1023/B:JOSH.0000046076.75950.0b
5. Chu, G., Stuckey, P.J., Schutt, A., Ehlers, T., Gange, G., Francis, K.: Chuffed, a lazy clause generation solver. https://github.com/chuffed/chuffed (2018)
6. Chuin Lau, H.: On the complexity of manpower shift scheduling. Computers & operations research **23**(1), 93–102 (1996)
7. De Bruecker, P., Van den Bergh, J., Belin, J., Demeulemeester, E.: Workforce planning incorporating skills: State of the art. European Journal of Operational Research **243**(1), 1–16 (May 2015). https://doi.org/10.1016/j.ejor.2014.10.038
8. Erkinger, C., Musliu, N.: Personnel scheduling as satisfiability modulo theories. In: International Joint Conference on Artificial Intelligence – IJCAI 2017, Melbourne, Australia, August 19-25, 2017. pp. 614–621 (2017). https://doi.org/10.24963/ijcai.2017/86
9. Ernst, A., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research **153**(1), 3–27 (Feb 2004). https://doi.org/10.1016/S0377-2217(03)00095-X
10. Falcón, R., Barrena, E., Canca, D., Laporte, G.: Counting and enumerating feasible rotating schedules by means of Gröbner bases. Mathematics and Computers in Simulation **125**, 139–151 (2016)
11. Kletzander, L., Musliu, N., Grtner, J., Krennwallner, T., Schafhauser, W.: Exact methods for extended rotating workforce scheduling problems. In: Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling. American Association for Artificial Intelligence (AAAI) (2019), to appear
12. Kletzander, L., Musliu, N., Smith-Miles, K.: Instance space analysis for a personnel scheduling problem. In: Workshop Data Science Meets Optimization, IJCAI (2019)
13. Laporte, G.: The art and science of designing rotating schedules. Journal of the Operational Research Society **50**, 1011–1017 (9 1999)
14. Laporte, G., Nobert, Y., Biron, J.: Rotating schedules. European Journal of Operational Research **4**(1), 24–30 (1980)
15. Laporte, G., Pesant, G.: A general multi-shift scheduling system. Journal of the Operational Research Society **55**(11), 1208–1217 (2004)
16. Muñoz, M., Smith-Miles, K.: Performance analysis of continuous black-box optimization algorithms via footprints in instance space. Evol. Comput. **25**(4), 529–554 (2017)

17. Musliu, N.: Combination of local search strategies for rotating workforce scheduling problem. In: International Joint Conference on Artificial Intelligence – IJCAI 2005, Edinburgh, Scotland, UK, July 30 - August 5, 2005. pp. 1529–1530 (2005), http://ijcai.org/Proceedings/05/Papers/post-0448.pdf
18. Musliu, N.: Heuristic methods for automatic rotating workforce scheduling. International Journal of Computational Intelligence Research **2**(4), 309–326 (2006)
19. Musliu, N., Gärtner, J., Slany, W.: Efficient generation of rotating workforce schedules. Discrete Applied Mathematics **118**(1-2), 85–98 (2002)
20. Musliu, N., Schutt, A., Stuckey, P.J.: Solver independent rotating workforce scheduling. In: International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 429–445. Springer (2018)
21. Restrepo, M.I., Gendron, B., Rousseau, L.M.: Branch-and-price for personalized multiactivity tour scheduling. INFORMS Journal on Computing **28**(2), 334–350 (2016)
22. Rice, J.: The algorithm selection problem. In: Advances in Computers, vol. 15, pp. 65–118. Elsevier (1976). https://doi.org/10.1016/S0065-2458(08)60520-3
23. Smith-Miles, K., Baatar, D., Wreford, B., Lewis, R.: Towards objective measures of algorithm performance across instance space. Comput. Oper. Res. **45**, 12–24 (2014). https://doi.org/10.1016/j.cor.2013.11.015
24. Smith-Miles, K., Bowly, S.: Generating new test instances by evolving in instance space. Comput. Oper. Res. **63**, 102–113 (2015). https://doi.org/10.1016/j.cor.2015.04.022
25. Smith-Miles, K.A.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys (CSUR) **41**(1), 6 (2009)
26. Triska, M., Musliu, N.: A constraint programming application for rotating workforce scheduling. In: Developing Concepts in Applied Intelligence, Studies in Computational Intelligence, vol. 363, pp. 83–88. Springer Berlin / Heidelberg (2011)