Refining self-scheduled rosters with limited changes and blocks

Qing Chuan Ye[0000−0002−8249−9890]

ORTEC B.V., Houtsingel 5 2719 EA Zoetermeer, The Netherlands
charlie.ye@ortec.com

Abstract. Self-rostering allows healthcare workers to propose their own work schedules, increasing autonomy and engagement, but often results in rosters that do not meet operational requirements without substantial manual intervention. We present an optimization-based method that supports planners by adjusting rosters within individual change limits. The approach incorporates shift blocks and a quantified system of change points to ensure minimal modifications. To determine the smallest distance between the initial and modified rosters, we use a minimum-cost flow formulation. A pilot with a Dutch hospital will evaluate the ability of our optimizer to produce good quality rosters more efficiently.

Keywords: Nurse scheduling problem · Self-rostering · Shift blocks

1 Introduction

Self-rostering has become a popular approach in workforce scheduling in healthcare. It allows employees to influence their own work patterns, giving them a sense of control and increasing their engagement in the planning process. It also has a positive effect on their work-life balance [1] and job satisfaction [4]. Employees can create feasible rosters with regard to labor laws and agreements, as the system indicates violations and prevents the employee from submitting a roster with violations. However, it is difficult for employees to construct a roster that satisfies all duty demands without external intervention. In current practice, a planner must manually adjust the submitted individual rosters to ensure that the overall schedule meets operational requirements. This manual correction process is time-consuming, often suboptimal, and might incorporate the planner's bias.

To help the planner address these issues, we introduce an optimization-based approach that aims to satisfy the duty demands as much as possible while ensuring that the changes to each employee's self-constructed roster do not exceed predefined acceptable limits. In addition, we take into account shift blocks in the rosters, as these are important for the recognizability of an employee's schedule.

2 Shift blocks

A shift block is defined as a sequence of consecutive shifts in which the rest between the planned shifts is not more than 32 hours. Employees prefer to plan

shift blocks, as it makes their roster more predictable and stable. They do not like working single shifts or highly variable sequences of shifts, especially when it comes to night shifts [3]. Having the shifts blocked together may also help alleviate circadian disruption and minimize fatigue [2]. Therefore, we try to respect shift blocks in an employee's roster and aim to make only small changes. We enforce the following properties on shift blocks:

- *Minimum/maximum block length*: number of consecutive shifts in a block needs to be at least the minimum and at most the maximum block length;
- *Maximum shift types*: number of different shift types that can be present in a block. Example of shift types based on start time: *Early*, *Late*, *Night*;
- *Mandatory block*: specifies for a given shift type a collection of weekdays that needs to be included together in any block that includes one of those days. For example, if *Night* shifts have a mandatory block of [Friday, Saturday, Sunday], then if a *Night* shift is planned on a Friday, the adjacent Saturday and Sunday also must have a *Night* shift planned;
- *Extend block with shift*: adding a new shift to an existing block is preferred over adding shifts that would create a new shift block.

Furthermore, if an employee plans a block that is smaller or larger than the minimum or maximum block length, we assume that this is a deliberate choice, and we will allow this. These blocks may be changed, but if they are still present in the modified roster, we consider it to be a feasible roster.

3 Roster changes limit

Each employee has their own limit to the changes that can be made to their submitted roster. This limit is based on how much an employee has helped the overall schedule by taking on less desired shifts. Therefore, each shift in the self-rostering phase has a certain score. The less desired a shift is, the higher the score. When an employee submits their individual roster, the scores are summed up, and the planner maps the total score to a *maximum number of change points* (MP). This mapping is inverse, meaning that high total scores will lead to lower MP, and low total scores to higher MP. If an employee has submitted an infeasible roster with respect to their contract hours, the MP will be increased to accommodate the changes needed for the roster to be feasible again.

A change in the roster can be classified into different categories, each with an associated number of *change points* (CP). For example, a planner could use the following categories based on the starting times of the shifts:

- *Small change (1 CP)*:
  - changing a shift to an adjacent shift ($\leq$ 8 hours difference), e.g., an *Early* shift was changed to a *Late* shift on the same day;
  - removing a shift from an existing block;
- *Medium change (2 CP)*:

  - changing a shift to a non-adjacent shift, up to and including the same shift a day earlier or later (≤ 24 hours difference), e.g., an *Early* shift in an employee's roster was changed to a *Night* shift on the same day or to an *Early* shift on the next day;
  - adding a shift to an existing block;
- *Large change (3 CP)*:
  - adding a shift to a non-existing block, e.g., creating a new block, or adding a shift to a newly created block.

Note that any change consisting of a shift moving more than 24 hours, e.g., changing *Early* on day d to *Late* on day $d+1$ would be considered as removing (1 *CP*) and adding a shift (2 *CP* if adjacent to an existing block, 3 *CP* if not).

Finally, we consider a modified roster feasible if the total number of change points associated with the changes made in the roster for each employee is less than or equal to the employee's maximum number of change points ($\sum CP \leq MP$), in addition to remaining feasible with regard to labor laws and agreements.

4 Determining changes

Now that we have defined how many *change points* each change costs, we need to determine what the changes are between the employee's submitted roster and the modified roster. It can be difficult to determine whether a shift should be considered as added, removed, or changed from an existing shift (and from which shift). Even more so considering blocks of shifts, as a block that has moved one day later could be viewed as either all shifts have moved one day, or the first shift has been removed, and a new shift has been added at the end. We want to identify the changes that result in the fewest total change points. Therefore, we make use of a minimum-cost flow formulation, which we solve using a successive shortest augmenting paths algorithm.

For a single employee, the minimum-cost flow is defined by the directed graph $G = (V, E)$ with nodes V and edges E, see Figure 1. We call shifts in the employee's submitted roster *initial shifts* $(i_1, i_2, \ldots, i_m)$, and shifts in the modified roster *current shifts* $(c_1, c_2, \ldots, c_n)$, where m is the total number of initial shifts and n the total number of current shifts, respectively. An employee also has a limited number of jokers that they can put on a specific shift or a day off, which indicates that these are very important to the employee and, therefore, *must* be adhered to. Jokers on shifts are incorporated as *fixed* shifts, which will be present in both the initial and current shifts. Jokers on days off are incorporated as days on which shifts cannot be planned.

The RA node mediates removals and additions. An edge $i \to RA$ indicates a removal of the initial shift i, while an edge $RA \to c$ indicates an addition of the current shift c. A directed edge $i \to c$ signifies that the current shift c is obtained from the initial shift i. The source s distributes the flow to the initial shifts, and the sink t collects the flow from the current shifts. All edges have capacity 1, except the edge $s \to RA$ with capacity $p_r = \max(0, m - n)$, and $RA \to t$ with capacity $p_a = \max(0, n - m)$. This means that when there are more *current* than

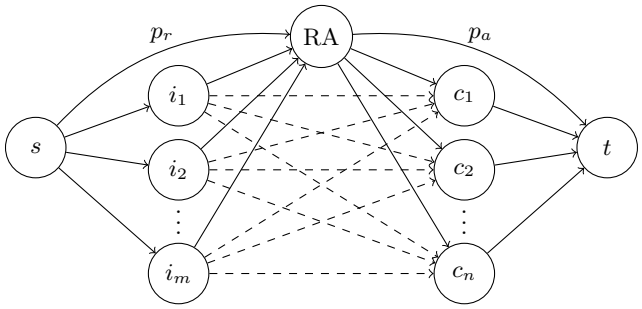

Fig. 1. Directed graph used for minimum-cost flow to determine changes between initial and current roster for a single employee.

initial shifts ($m > n$), we need to remove p_r shifts. And vice versa, when there are more *initial* than *current shifts* ($n > m$), we need to add p_a shifts.

The cost of each edge from s and to t is 0. The cost of the edges from and to the shift nodes corresponds to the change points (CP) defined in Section 3. The edges $i \to RA$ have cost 1, and $RA \to c$ have cost 2 if c is in an existing block, or cost 3 if c is in a new block. The edges $i \to c$ have cost 0 if the start time is the same, cost 1 if there is ≤ 8 hours difference in start time, or cost 2 if the difference is ≤ 24 hours. If the difference is > 24 hours, there will be no edge.

5 Customer pilot

The details of this work were established in cooperation with planners from a Dutch hospital. We are starting a pilot in 2026 using our optimizer that takes into account both shift blocks and the roster changes limit. We will compare our results to the planner's manually produced roster. Our goal is to produce a feasible roster with as few changes as possible, so that it can help the planner create a feasible roster from the employees' submitted rosters in less time.

References

1. Albertsen, K., Garde, A. H., Nabe-Nielsen, K., Hansen, Å. M., Lund, H., Hvid, H. (2014). Work-life balance among shift workers: results from an intervention study about self-rostering. *International Archives of Occupational and Environmental Health*, 87(3), 265-274.
2. Booker, L. A., Mills, J., Bish, M., Spong, J., Deacon-Crouch, M., & Skinner, T. C. (2024). Nurse rostering: understanding the current shift work scheduling processes, benefits, limitations, and potential fatigue risks. *BMC nursing*, 23(1), 295.
3. Petrovic, S., Vanden Berghe, G. (2008, August). Comparison of algorithms for nurse rostering problems. In *Proceedings of the 7th International Conference of Practice and Theory of Automated Timetabling*.
4. Webster, B., Archibald, D. (2022). Self-rostering, work-life balance and job satisfaction in UK nursing: a literature review. *Nursing Management*, 29(6).