Self-Rostering applied to case studies

An ILP method to construct a feasible schedule

Suzanne Uijland · Egbert van der Veen ·
Johann Hurink · Marco Schutten ·

Received: date / Accepted: date

Abstract We discuss the self-rostering problem, a concept receiving more and more attention from both theory and practice. We outline our methodology and discuss its application to a number of practical case studies.

Keywords Personnel Rostering · Shift Rostering · Self Rostering · Linear Programming

1 Introduction

In service industries, like healthcare and security services, people work around the clock. Considering the many employee preferences and labor legislation that are implied on rosters, it is, both in theory, [L. De Grano et al (2009); Rnnberg and Larsson (2010)] and practice, often hard to come up with good or fair shift rosters. A way to better cope with employee preferences, and to

Suzanne Uijland
Department of Management and Governance, University of Twente, P.O. Box 217, Enschede, The Netherlands, ORTEC, Groningenweg 6k, 2803 PV, Gouda, The Netherlands

Egbert van der Veen
Presenting author
ORTEC, Groningenweg 6k, 2803 PV, Gouda, The Netherlands
Tel.: +31182540500
Fax.: +31182540540
Center for Healthcare Operations, Improvement, and Research (CHOIR), University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands
E-mail: Egbert.vanderVeen@ortec.com

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

Marco Schutten
Department of Management and Governance, University of Twente, P.O. Box 217, Enschede, The Netherlands,

increase job satisfaction, is self-rostering. The main idea in self-rostering is that employees can propose their own shift rosters, and if they do this in a 'good' way, they get to work most of their shifts as in their preferred schedule.

Bailyn et al (2007) argue that self-rostering increases job satisfaction by an improved work-life balance, increased predictability and flexibility in schedules, and enhancing the communication and interaction to stimulate cooperative community building. Especially the latter is typical for self-rostering, since self-rostering stimulates employees to propose schedules that are 'good' for the organization, see Section 3.

However, the shift rosters employees propose in general do not match up with a shift staffing demand, and, therefore, staffing *shortages* for shifts for specific days occur. The goal of this paper is to design a method that unassigns some of the proposed shifts in order to solve the shortages.

In the next sections, we will subsequently describe the self-rostering process, our solution technique, and discuss some results.

2 Self-rostering process

For a given rostering period (of e.g., a month)the self-rostering problem can be decomposed in 5 phases:

1. The organization defines the *staffing demand*, i.e., specifies, per day, the number of employees that need to perform a certain shift.
2. The employees propose their preferred schedule. These schedule have to obey labor legislation, and other scheduling constraints specified by the organization, like forward rotations or planning homogeneous work blocks.
3. The employees' preferred schedules are combined, after which the staffing demand is subtracted from these, indicating surpluses and shortages on each shift.
4. The information of Phase 3 is returned to the employees, offering them the opportunity to adjust their schedules.
5. The planner solves the remaining shortages after Step 4.

In this paper we focus on the fifth phase and propose a method to solve the remaining shortages. Note that the presented method is independent of the self-rostering process itself and the way employees create and adjust their proposed schedules. In fact our method can be applied to any given schedule where for each employee a preferred schedule is given and where a mismatch between the shift staffing and shift demand is given.

3 Method

The objective of our method is to solve as many shortages as possible, whereby making sure that a specified fraction of an employee's proposed schedule remains.

To resolve the remaining shortages, we use an iterative method. In every iteration an integer linear program (ILP) is solved, that mainly considers two things: the *score* of every employee's schedule, and *swaps*.

The *score* of an employee's schedule is calculated as follows. First, we assign 'points' to shifts using a specified point system. For example, understaffed shifts are assigned 3 points, overstaffed shifts get 1 point, and matching shifts (shifts where the staffing demand is exactly matched) get 2 points. Second, we calculate the score of every employee, by calculating his total points, possibly multiplied by some factor to, e.g., take part time percentages into account. As a consequence, employees with a large score work many relatively unpopular shifts, whereas for employees with a small score the opposite holds.

In a *swap* we unassign an employee from one of his overstaffed shifts, and assign this employee to a shift that is currently understaffed. A swap can be performed on two shifts on the same day, but also on two shifts on different days. A swap may only be performed if it obeys labor legislation and other practical scheduling constraints specified by the organization. This can be precalculated before running the ILP.

Given the *scores* and *swaps* in every iteration, the ILP is used to select a subset of swaps is selected, with at most one swap per employee. We select at most one swap per employee to make sure we do not violate labor legislation or other constraints the organization implies on the schedules. The objective makes a trade-off between two factors. First, we want to maximize the number of swaps selected, in order to maximize the number of shortages solved in an iteration. By maximizing the number of shortages solved in an iteration we expect to solve more shortages in total. Secondly, we preferably perform swaps at employees that have a low score, in order to reward employees that choose relatively many unpopular shifts.

4 Results

We applied our method to 168 schedules based on data from real life. The range of the number of employees is 15-80, and the range of the number shortages is 16-212. Labor legislation is dealt with by including only swaps in the model that do not violate labor legislation.

From the experimental results we observe that the mean computation time of our method is 2.8 seconds, the mean number of shortages left is 0.90, and on average 92.2% of the schedules is retained. The number of remaining shortages is calculated as the positive difference between the number of remaining shortages and the number of remaining excesses. Depending on, among others, the point system used, parameter values of the ILP, and swaps, there is a trade-off in the model between either the number of shortages solved and fraction of the proposed schedules that is preserved.

References

Bailyn L, Collins R, Song Y (2007) Self-scheduling for hospital nurses: an attempt and its difficulties. Journal of Nursing Management 15(1):72–77, DOI 10.1111/j.1365-2934.2006.00633.x, URL http://dx.doi.org/10.1111/j.1365-2934.2006.00633.x

L De Grano M, Medeiros D, Eitel D (2009) Accommodating individual preferences in nurse scheduling via auctions and optimization. Health Care Management Science 12:228–242, URL http://dx.doi.org/10.1007/s10729-008-9087-2, 10.1007/s10729-008-9087-2

Rnnberg E, Larsson T (2010) Automating the self-scheduling process of nurses in swedish healthcare: a pilot study. Health Care Management Science 13:35–53, URL http://dx.doi.org/10.1007/s10729-009-9107-x, 10.1007/s10729-009-9107-x