A modeling methodology to support nurse rostering practitioners

Bödvarsdottir, Elin Björk; Smet, Pieter; Vanden Berghe, Greet; Stidsen, Thomas Jacob Riis

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

MISTA 2019

A modeling methodology to support nurse rostering practitioners

Elín Björk Böðvarsdóttir · Pieter Smet · Greet Vanden Berghe · Thomas Stidsen

Abstract Researchers have studied the nurse rostering problem (NRP) for decades, producing many efficient algorithms for different formulations of the problem. In spite of these successes, the models and methods developed seldom make it to practical implementation. We believe too much emphasis has been put on solving the problem, and that the research community needs to take a step back and reflect on which problem they are solving. When modeling the NRP, we need to temper the violation of multiple constraints, and researchers tend to formulate the objective as minimizing the weighted sum of these violations. Although a solver may view a solution as optimal, practitioners might reject it if the *combination* of soft constraint violations is not acceptable. In this paper, we discuss some of the major drawbacks of this modeling approach and suggest a paradigm shift that may help us overcome them. We introduce the concepts *targets* and *acceptance thresholds* and discuss how we can integrate them into nurse rostering models and solution methods. We devote some attention to setting accurate acceptance thresholds, i.e., ensuring an *acceptable combination* of violations without the problem becoming infeasible. As this paradigm shift involves terms that practitioners can intuitively understand, it makes the automated scheduling approaches more accessible to them. Furthermore, the resulting approach can be employed for quality control, and we argue that it can yield more reliable rostering systems in practice.

Elín Björk Böðvarsdóttir
Department of Technology, Management and Economics, Technical University of Denmark, Anker Engelundsvej 1, 2800 Kgs. Lyngby, Denmark
E-mail: ebod@dtu.dk

Pieter Smet
KU Leuven, Department of Computer Science, CODeS, Gebroeders De Smetstraat 1, Gent 9000, Belgium
E-mail: pieter.smet@kuleuven.be

Greet Vanden Berghe
KU Leuven, Department of Computer Science, CODeS, Gebroeders De Smetstraat 1, Gent 9000, Belgium
E-mail: greet.vandenberghe@cs.kuleuven.be

Thomas Stidsen
Department of Technology, Management and Economics, Technical University of Denmark, Anker Engelundsvej 1, 2800 Kgs. Lyngby, Denmark
E-mail: thst@dtu.dk

1 Introduction

The nurse rostering problem (NRP) is the task of assigning nurses to shifts to generate a feasible work schedule. This problem belongs to the class of personnel scheduling problems, which consider the assignments of employees to shifts for numerous different professions and scenarios. Even though we could extend this paper to consider personnel scheduling problems in general, the main focus will be on the nurse rostering problem for two reasons. First, it is one of the most widely studied subclass of personnel scheduling problems, and second, the problem requires round-the-clock scheduling, resulting in a more complicated set of constraints than some other types of personnel scheduling problems.

Researchers have studied the NRP for several decades and provided multiple formulations and solution methods (Burke et al, 2004; Van Den Bergh et al, 2013). In 2018, Kingston et al introduced a general format for expressing nurse rostering instances, extending a similar format for high school timetabling. They have converted several publicly available instances to this format, but we have yet to see how the research community will adopt to it.

Despite countless academic papers on nurse rostering, implementation in practice is limited. Kellogg and Walczak (2007) reported that only 30% of published research on nurse rostering had led to implementation, while 86% stated that they had intended for their work to be implemented. More recently, Petrovic (2019) reported that most of the available software for personnel scheduling had not benefited from the results of the research community. This gap from academic research to implementation in practice is intriguing, as the literature includes numerous examples of efficient solution approaches producing optimal or near-optimal solutions in a short time.

We believe that this lack of implementation is not due to our inability to solve the problem, but rather because we are not addressing the right problem. In an ideal world, we could generate perfect rosters with optimal staffing while fulfilling every single requirement for each nurse. However, in the real world, we cannot prevent all constraint violations, and thus, the challenge becomes to generate *high-quality* rosters with an *acceptable combination* of violations.

When Kellogg and Walczak (2007) analyzed the lack of implementation of nurse rostering research they named seven possible explanations, one of which being *lack of acceptance by nurses*. They related this lack of acceptance to two factors: the *lack of trust* in computerized systems and the *time required* to learn and use such systems.

Although this paper is over 10 years old and many have referenced Kellogg and Walczak (2007)'s results, no researchers have focused on the acceptance by nurses in their research since. Nonetheless, many have addressed the concept of *self-scheduling* (i.e., involving the nurses in creating their own schedules), which was another of Kellogg and Walczak (2007)'s possible explanations, along with *preference-scheduling*, which is closely related. Even though one can argue that having increased autonomy will likely lead to increased acceptance, this type of research addresses neither the lack of trust nor the time required. Only a single paper (Mihaylov et al, 2016) has addressed the time required, by introducing a method for automating a process that is currently manual.

In this paper, we discuss some major drawbacks with the current models in nurse rostering research (Section 2) and present a modeling methodology that is designed to simplify the information flow between the users and the solver (Section 3). Furthermore, we present preliminary results showing the computational potential of this methodology

for 12 real instances from Danish hospitals (Section 4). We believe that the increased transparency that this modeling methodology brings will both increase the trust in computerized rostering methods and reduce the time required to learn and utilize these methods.

2 Problem statement

Most often the NRP is formulated as a set of hard constraints and soft constraints, where the objective function minimizes a weighted sum of soft constraint violations. Despite this standard way of modeling the problem, the constraints that researchers choose to include may vary substantially. Furthermore, some researchers may formulate a given constraint as hard while other researchers argue that the same constraint should be soft, and in Kingston et al (2018)'s general format, each constraint can either be hard or soft, depending on the user's choice. This inconsistency is a clear indicator of how difficult it is to model this problem, as we need to capture different elements, many of which are conflicting.

Furthermore, practitioners are not interested in the objective value of a roster, and may even prefer suboptimal solutions if the combination of violations is better. Figure 1 provides an example, where we consider two soft constraints for overtime and float nurses (i.e., nurses not employed by the ward but hired in for single shifts as needed at an additional cost). The dashed lines indicate *acceptance thresholds*, i.e., the level of violations that practitioners will accept for each constraint. The objective function minimizes a weighted sum of these two constraints, and we indicate the decrease in the objective function along with displaying several feasible solutions marked with `x`. Although the one marked red is optimal within this set of solutions, practitioners would prefer one of the subpotimal solutions marked green as those are within the acceptance thresholds.

We argue that the complex nature of the weighted sum objective function is an important reason for the lack of successful implementation in practice. For the solution methods to generate high-quality rosters, accurate objective weights are essential (Burke et al, 2008). However, setting these weights is a difficult task, and Mihaylov et al (2016) revealed a substantial inconsistency between the weights that practitioners claim to be correct and their actions when manually generating rosters.

Both researchers and practitioners have raised concerns regarding this objective formulation and provided concrete examples of challenges related to choosing the weights (Gärtner et al, 2018; Petrovic and Vanden Berghe, 2012). When diving deeper into specific examples, we note that practitioners often increase weights in attempts to remove inevitable constraint violations, e.g., due to some hard constraints. This behavior is worrying, as it shows that practitioners are unaware of the types of solutions they can and cannot obtain. Moreover, one can see how these difficulties may lead to a very time-consuming process, while concurrently causing the rosters to be of unnecessarily poor quality.

Obtaining an in-depth understanding of the relationship between different constraints of the NRP is not trivial. For some pairs of constraints, we can intuitively predict the impact they have on each other, for example, that requiring a minimum number of days off does not support a constraint for a minimum number of workdays. Despite the conflict between these two constraints, we might be able to satisfy both depending on the bounds that have been set, along with how these two constraints

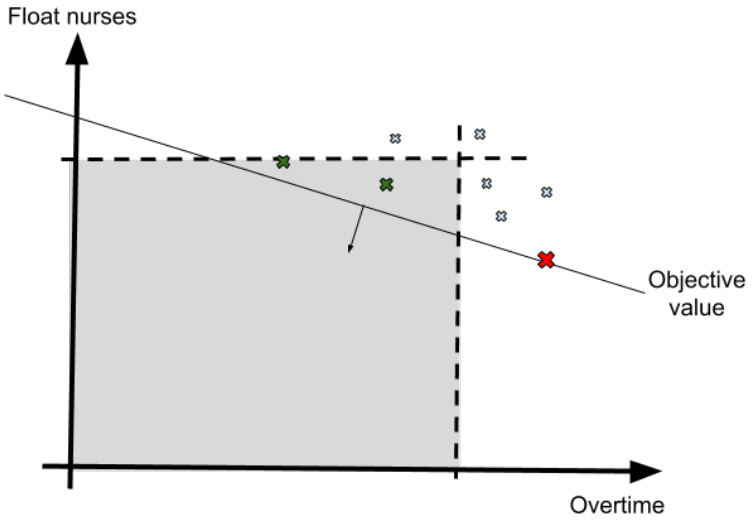

Fig. 1 Example showing how suboptimal solutions may include a better combination of violations.

interact with the remaining constraints in the formulation. Furthermore, the relation between two constraints might be more complicated than either supportive or conflicting, depending on the scenario. For example, a nurse's request for a day off can either support or conflict with a constraint for a minimum number of consecutive workdays, depending on where it is placed. Overall, the relationship between different constraints can be viewed as a spider web, i.e., a set of different threads tangled together, making it difficult to understand the role every single thread plays in the whole.

Due to the significant effect that poor rostering can have on nurses' health and happiness (Gärtner et al, 2018), the perceived quality of a roster is of utmost importance. Nonetheless, defining an overall measure for *quality* is impossible in this context, as many different factors impact it. By bundling everything up in a weighted sum objective, we lose control over individual components. As weights are subjective, a solution may be optimal (or near-optimal) while still including an unacceptable combination of constraint violations. Therefore, we stress the importance of addressing the right problem, not only by including relevant constraints but also by ensuring that they produce an acceptable roster.

3 Moving modeling towards a common language

Nurse rostering research could be strengthened with increased transparency and by simplifying the information flow between the user and the solver. Therefore, we adopt the shifted paradigm when formulating nurse rostering problems, from *objectives* and *objective weights* to *targets* and *acceptance thresholds*. For example instead of assigning a *weight* (penalty) to an *objective* for nurses working overtime, we instead set an *acceptance threshold* for an overtime-*target*, describing the amount we will accept.

This paradigm shift introduces some new challenges, namely how to set accurate acceptance thresholds for targets, and how to include these aspects in nurse rostering models and solution approaches. We will present a methodology to address these challenges in the following sections, with Section 3.1 focusing on the estimation of attainable acceptance thresholds and Section 3.2 focusing on how to apply these thresholds.

3.1 Estimating acceptance thresholds

When manually generating rosters, practitioners apply various rules of thumb, many of which could be translated into acceptance thresholds. Nonetheless, practitioners often need to bend these rules to generate feasible rosters, and one may even see that the *hardest* of constraints become violated under some circumstances. Thus, finding universal values for these acceptance thresholds is unrealistic. Instead, we should focus on deriving attainable acceptance thresholds based on the instance at hand.

Acceptance thresholds are related to bounds on the number of violations of soft constraints. For example, for a constraint k with a bound of n_k violations, a corresponding acceptance threshold could be accepting $n_k + \delta_k$ violations. In this case, δ_k represents some slack that is intended to ensure that a feasible solution exists. Nonetheless, estimating accurate bounds presents a challenge as the acceptance thresholds should neither be too tight (causing infeasibility) nor too loose (unnecessarily compromising the quality of the rosters).

An important early work in this area is by Burke et al (2002), where they used bounds for soft constraint violations to introduce an alternative objective function formulation. They estimated the bounds for each constraint separately, where the lower bound represented the lowest possible number of violations. They mapped the lower bounds to an *ideal point* in the search space, and acknowledged that it would likely lie in an infeasible part of the solution space. Nevertheless, allowing an appropriate slack from the ideal point may translate into attainable acceptance thresholds.

When estimating acceptance thresholds, we should not only consider a single constraint. Instead, we should pragmatically choose relevant components of the problem. For example, considering float nurses and overtime separately will not produce reliable bounds as they complement one another in a roster. As setting acceptance thresholds for multiple constraints simultaneously often becomes a question of trade-off, we need to incorporate practitioners' priorities in the procedure. Theoretically, we could assign multiple float nurses to completely avoid overtime, or vice versa, meeting the staffing requirements without float nurses by assigning the nurses to significant overtime. These examples represent the two extremes, but in practice the desired outcome lies somewhere between. Table 1 provides an example where the first two rows represent the two extremes, while the remaining rows present some different combinations of acceptance thresholds. The accurate combination depends on each instance, for example, the ward's budget for float nurses or how well the overtime spreads among different nurses.

Estimating bounds for a component of the problem is closely related to the concept of *minimal unsatisfiable subsets* (i.e., a set of constraints such that no feasible solution exists but all strict subsets have a feasible solution). When solving a minimization problem with several terms in the objective function, we obtain a feasible *combination* of bounds, given the objective weights (similar to those presented in Table 1). Reformulating these objective terms as hard constraints, using the combination of bounds

Table 1 Different acceptance thresholds for float nurses and overtime, depending on practitioners priorities.

Float nurses	Overtime [in hours]
10	0
0	70
5	30
3	50
8	10

as an upper bound for each term, would yield the same result. However, if we would make one of the bounds tighter (without changing other bounds) it would result in an infeasible problem, where the corresponding constraint would become unsatisfiable. Thus, we need to set the acceptance thresholds for relevant components of the problem simultaneously to avoid unsatisfiable subsets.

In some cases, acceptance thresholds may correspond to organizational targets, e.g., fulfilling a minimum percentage of requests for days off. Nonetheless, we might need to relax this minimum if infeasible for the instance considered. Thus, we may base some acceptance thresholds on rules of thumb, but meanwhile, we need to ensure that those rules are adapted to each instance as needed.

The following sections present two examples exhibiting the methodology for adapting organizational targets to a specific instance, resulting in attainable acceptance thresholds. The former example considers only a single target, while the latter considers three targets simultaneously.

3.1.1 Maximum number of consecutive workdays

To show how we can adapt acceptance thresholds to a specific instance we consider the maximum number of consecutive workdays. This is a common constraint in many personnel scheduling problems, as it is often imposed by legislation, for example, that there should at most be n consecutive workdays. Nonetheless, this constraint is not always formulated as a hard constraint as legislation may leave room for exceptions, to better accommodate the employees' individuality. Thus, the objective function often includes a term that penalizes for exceeding n consecutive workdays.

Algorithm 1 displays how to extend the general threshold n for nurse i to day-specific acceptance thresholds n_d^i for all days of the rostering horizon D. In line 2, we consider when we should assign the next day off. When generating rosters in practice, the nurses are often involved in the process, e.g., by requesting which days they would and would not like to work. Furthermore, some shift assignments may be fixed beforehand (e.g., due to administrative meetings) and the feasibility of scheduling a day off may depend on these assignments. Finally, we emphasize that this estimation does not consider any other constraints of the problem, but is solely based on pre-processing the input data for the nurse in question.

This extension may be used to manage where these type of constraint violations occur, i.e., only when an employee has specifically allowed ignoring this constraint. Although the number of violations of the general threshold might be the same (and thus considered equally good by the objective function), the combination of violations would be more acceptable. Estimating the acceptance thresholds in this manner can

Algorithm 1 Adapting the acceptance threshold for the maximum number of consecutive workdays to a specific nurse i.

1: **for** $d \in D$ **do**
2: Estimate $\mu_d :=$ the number of days from d until a day off is feasible and the employee has not specifically requested a work assignment
3: Set $n_d^i = \max(n, \mu_d)$ as the acceptance threshold for number of consecutive workdays from day d for nurse i
4: **end for**

be highly valuable in practice, as it enables adapting general rules to the uniqueness of individual employees.

3.1.2 Available resources

This section indicates how we can estimate acceptance thresholds for a group of constraints simultaneously. We provide two examples of how to control the available resources in a rostering environment where the nurses have the opportunity to influence their rosters, similar to self-scheduling. If we assume that the staffing requirements are hard constraints, then the available resources refer to the ward's nurses along with float nurses needed to fulfill the minimum staffing. We would like to minimize the deviation from the nurses' contractual hours, corresponding to two targets: one for *overtime* and another for *undertime*. Additionally, we have a third target as we would like to minimize the number of *float nurses* needed to meet the staffing requirements.

Algorithm 2 displays how to employ a nurse rostering model along with any optimization method to estimate bounds for the targets, and then translate these bounds into acceptance thresholds. In line 3, we let the objective weights represent the priorities of these targets, ensuring that those priorities are transferred into the acceptance thresholds. The results are not very sensitive when it comes to the weights, and preliminary results have shown that setting the weight for float nurses to 10 and the other two to 1 produces accurate bounds (assuming that all objectives penalize based on the same measurement unit). When translating the bounds n_k into acceptance thresholds λ_k we add a small slack δ_k, that should give the flexibility to ensure feasibility w.r.t. other targets.

Algorithm 2 Estimating bounds for available resources using any solver.

1: Consider a full nurse rostering model
2: Deactivate objectives not directly corresponding to these targets
3: Set the objective weights for the corresponding targets to represent the priorities
4: Optimize to obtain bounds n_{float}, n_{over}, n_{under}
5: Add appropriate slack to obtain acceptance thresholds
$\lambda_k = n_k + \delta_k$ for $k \in \{float, over, under\}$

Instead of using a full nurse rostering model to obtain these bounds, we can also analyze the rostering data prior to the optimization to derive acceptance thresholds. We assume some general thresholds have been set, e.g., employing zero float nurses and allowing a deviation from contractual hours corresponding to half a shift. After analyzing the data we loosen these general thresholds to ensure feasibility.

Algorithms 3-4 display how to derive nurse-specific acceptance thresholds for overtime and undertime, respectively. Mainly, two factors control whether overtime or undertime is necessary, namely the surplus or deficit from the previous rostering horizon along with the shifts and days off that have already been fixed with self-scheduling.

In Algorithm 4 (line 1) we consider the days that have not yet been fixed during self-scheduling and estimate a lower bound on the number of working hours that we can assign on those days. In this estimation we take into account various factors concerning the already fixed days, including the number of such days and whether they include work or a day off. Furthermore, we distinguish between weekdays and weekends, as we often have organizational constraints that restrict weekend work.

Algorithm 3 Deriving acceptance thresholds for overtime for a single nurse i based on the specific instance.

1: Calculate `Fixed working hours` from input data
2: `Minimum assigned hours` = `Fixed working hours` + `Surplus from previous horizon`
3: **if** `Minimum assigned hours` > `Contractual hours` + `General threshold` **then**
4: λ^i_{over} = `Minimum assigned hours - Contractual hours`
5: **else**
6: λ^i_{over} = `General threshold`
7: **end if**

Algorithm 4 Deriving acceptance thresholds for undertime for a single nurse i based on the specific instance.

1: Estimate `Lower bound for the addition` to the hours that have already been fixed
2: `Possible hours` = `Fixed working hours` + `Lower bound for the addition`
3: **if** `Possible hours - Deficit from previous horizon` < `Contractual hours - General threshold` **then**
4: λ^i_{under} = `Contractual hours - Possible hours`
5: **else**
6: λ^i_{under} = `General threshold`
7: **end if**

Algorithm 5 displays how we estimate the acceptance threshold for float nurses in three steps. First, we consider each specific coverage constraint, composed of a day, a shift and a skill (lines 1-5). Second, we consider the accumulated coverage required for all shift and skill combinations on a specific day (lines 6-12). Third, we consider the accumulated coverage required during the entire horizon for a specific shift and skill combination (lines 13-20).

In the third step (line 14), we estimate an upper bound on the total number of times we can assign a nurse of a given skill to a given shift. This estimation is based on either a fixed maximum for how often each nurse can be assigned to the specific shift or calculated based on the `Possible hours` (from Algorithm 4). Thus, an inaccurate estimation for one threshold may impact the estimate for other thresholds. Therefore, we need to identify which targets are related and take it into consideration when estimating the thresholds.

Algorithm 5 Deriving acceptance threshold for float nurses based on the specific instance.

```
for coverage constraints do
    if the number of available nurses for the shift and skill on the day is less than the
       required coverage then
        Increase acceptance threshold for float nurses
    end if
end for
for days of the rostering horizon do
    if the total number of available nurses is less than the accumulated coverage needed
       for the day then
        if this difference does not arise due to a specific combination then
            Increase acceptance threshold for float nurses
        end if
    end if
end for
for (shift,skill)-combinations do
    Estimate upper bound for the number of nurses with skill assigned to shift during entire
    rostering horizon
    if the upper bound is lower than the total required coverage then
        if this difference has not been considered on a specific day then
            Increase acceptance threshold for float nurses
        end if
    end if
end for
```

3.2 Applying acceptance thresholds

By estimating attainable acceptance thresholds, we provide practitioners with beneficial guidelines on the types of solution that they can obtain. When considering the common weighted sum objective function, these thresholds support the practitioners in making educated decisions when setting the weights. Nonetheless, we gain even more by incorporating acceptance thresholds directly into automated scheduling methods. Unfortunately, this is not trivial and we will discuss some of the new challenges that it may bring.

If we are certain that at least one feasible solution complying with all acceptance thresholds exists, then an obvious approach would be to enforce the acceptance thresholds as hard constraints in the formulation. However, one needs to tread carefully, as claiming to have certainty when considering all future rostering horizons is rather unreasonable. Thus, we need to introduce the acceptance thresholds to the formulations without introducing new challenges concerning infeasibility.

Even though obtaining an intuitive understanding of objective weights is difficult, they currently play an important role in trying to capture the priorities of different constraints. Thus, future research should investigate simpler alternatives for capturing these priorities. Another crucial question is how to directly include acceptance thresholds in nurse rostering models while ensuring to avoid infeasibility. Obtaining answers to these questions will aid in simplifying the objective function, resulting in nurse rostering models becoming more accessible for practitioners.

In the following sections, we present two methodologies for employing targets and acceptance thresholds in automatic scheduling. The former is a supplement for any model with a weighted sum objective, while the latter is a reformulation of the problem, which removes the complexity of the weights.

3.2.1 Weight Tuning

Although objective weights are an abstract concept, they are still used in a far majority of nurse rostering formulations. We believe that making weighted sum formulations more accessible for practitioners can increase the implementation of already developed nurse rostering approaches. Therefore, we introduce a general methodology, *Behind-the-Scenes Weight Tuning*, that employs targets and acceptance thresholds to automatically set the weights when working with a weighted sum objective.

Starting from an initial set of weights, the methodology evaluates the quality of the resulting roster by comparing the constraint violations to the corresponding targets and their acceptance thresholds. If the roster does not reach some target(s), we automatically adjust the supporting weight(s) to penalize such violations more heavily. With the new set of weights, we re-optimize and perform the same evaluation and (eventual) adjustments to further improve the perceived quality of the roster.

This methodology does not alter the complex objective function but instead provides automatic assistance with the associated challenges. While the set of feasible solutions remains the same, *Behind-the-Scenes Weight Tuning* guides the optimization method to more promising solutions by altering the objective weights. The methodology is independent of the formulation and functions as a supplement on current nurse rostering models and solution methods. Thus, it is ideal to facilitate the paradigm shift.

3.2.2 Goal programming

The concepts of targets and acceptance thresholds fit quite naturally in the framework of goal programming. For every target, we should stay within the corresponding acceptance threshold if feasible, and if not, then we should minimize the deviation from the threshold. As a standard goal programming reformulation has everything bundled together in a single objective, we would still need the complex concept of objective weights to capture the priorities of different targets.

Thus, we suggest reformulating the problem as a lexicographic goal programming model, with a distinct priority for each target. For each target, we solve a simple goal program, where the only objective is to minimize the deviation from the corresponding acceptance threshold. If the objective value is zero, then we have a feasible solution satisfying the acceptance threshold and may introduce it as a hard constraint. If the objective value is greater, then we must adjust the acceptance threshold accordingly before introducing it as a hard constraint.

With this approach, we iteratively introduce the acceptance thresholds as hard constraints, but only after confirming their feasibility. Furthermore, when reaching a target that cannot comply with the threshold, we adapt the threshold knowing that it will not be feasible without violating the acceptance thresholds for targets of a higher priority.

Figure 2 presents how to apply this methodology to a small nurse rostering example, where we have four targets and the only hard constraints relate to coverage. The targets (in prioritized order) correspond to the number of float nurses, exceeding the maximum number of consecutive workdays, the amount of overtime and the amount of undertime. As shown, we move through the targets in prioritized order and introduce them as hard constraints once we have verified their feasibility. If we need to adjust a threshold, the appropriate adjustment comes directly from the value of the objective function and

thus, we do not need to reiterate in any step. When solving the goal program for the last target we obtain an optimal roster w.r.t. the priorities and the original acceptance thresholds.

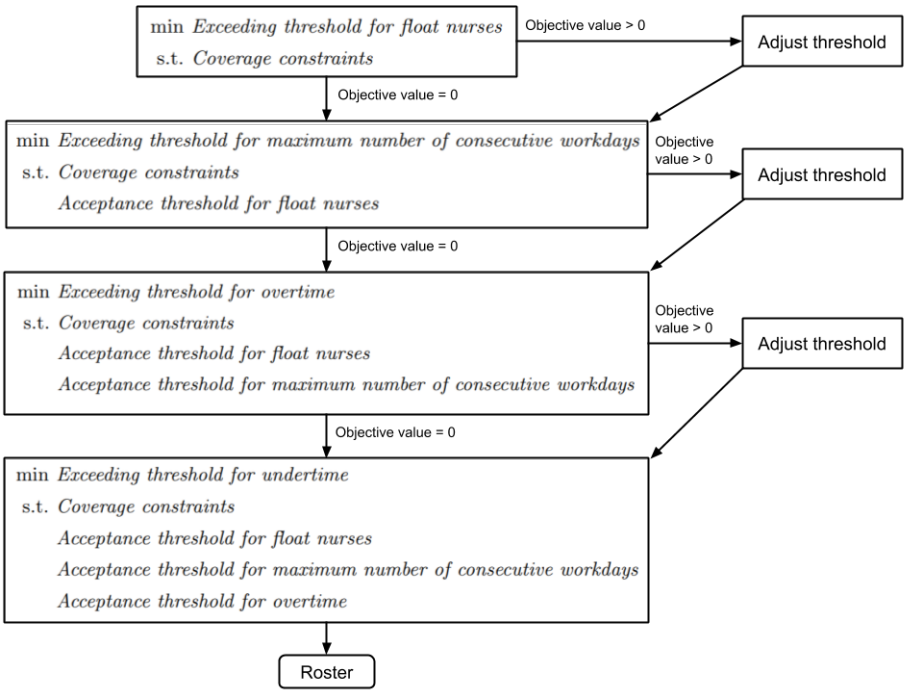

Fig. 2 Example of the lexicographic goal programming methodology.

The most valuable advantage from working with accurate acceptance thresholds is that we can exclude targets of a lower priority when optimizing each target, as they have implicitly been incorporated in the acceptance thresholds. For comparison, if we use a goal of zero violations at each level, the subsequent hard constraints would presumably be too tight to enforce them at lower priority levels. As an example, when minimizing the number of float nurses considering only the coverage constraints, we may obtain a solution that uses no float nurses. However, when introducing the remaining targets, we will need five float nurses to arrive at a reasonable roster. If we possess this knowledge beforehand, we can set the initial acceptance threshold (and thus the bound on the corresponding hard constraint) accordingly. However, without the knowledge we would enforce a hard constraint forbidding all float nurses, resulting in obscure violations for other targets when moving down the prioritized list.

4 Preliminary results

This section includes some preliminary results for the estimation techniques described in Section 3.1. We provide results for 12 real-world nurse rostering instances published

by Böðvarsdóttir et al (2019b), that have a rostering horizon of 28 days and include 40-50 nurses.

We consider the four targets presented throughout this paper, with the following priority ordering: the number of float nurses, exceeding the maximum number of consecutive workdays, the amount of overtime and the amount of undertime. In addition to these targets, we include numerous hard constraints, for example, coverage constraints and a maximum number of assignments to specific shifts. A complete overview of the hard constraints is presented by Böðvarsdóttir et al (2019a).

We set the general acceptance thresholds to zero for the number of float nurses, while the general threshold for overtime and undertime is 5 hours, corresponding to roughly half a shift. Furthermore, we have a fixed maximum on the number of consecutive workdays, and set a general threshold for never exceeding this maximum. We adapt these acceptance thresholds to each instance using Algorithm 1 (Section 3.1.1) and Algorithms 3-5 (Section 3.1.2).

Table 2 shows the violation of these thresholds when solving the problem using the lexicographic goal programming methodology. For 11 out of the 12 instances, the acceptance thresholds for all four targets are transformed into feasible hard constraints, and for the remaining instance, we only need to extend the acceptance threshold for a single target. These results clearly exhibit how relatively simple pre-processing techniques can yield attainable acceptance thresholds.

For instance B04, we need to extend the acceptance threshold for overtime by almost nine hours in total, distributed among two nurses. The main reason for this extension is a conflict between the targets for float nurses and overtime on a single day. Thus, we assign the two nurses to work during the day to avoid employing float nurses, resulting in the overall working time for these nurses exceeding their requested overtime. By analyzing the roster that was employed in practice, we can confirm that the individual rosters for these two nurses are identical, meaning that the managers accepted the extension of this threshold.

Table 2 The violations obtained with the lexicographic goal programming framework.

Instance	Float nurses [number]	Maximum number of consecutive workdays [days]	Overtime [hours]	Undertime [hours]
A01	0	0	0	0
A02	0	0	0	0
A03	0	0	0	0
A04	0	0	0	0
A05	0	0	0	0
A06	0	0	0	0
A07	0	0	0	0
B01	0	0	0	0
B02	0	0	0	0
B03	0	0	0	0
B04	0	0	8.6	0
B05	0	0	0	0

Accurate acceptance thresholds should not only be feasible but also sufficiently tight to prevent us from accepting a solution with an unnecessarily high number of violations. We estimate the slack for the different targets using Equation (1) for the

number of float nurses and Equation (2) for the remaining targets, as the acceptance thresholds for those are nurse-specific. Table 3 displays the unused slack for the four targets.

$$\frac{\text{Acceptance threshold} - \text{Amount assigned}}{\text{Acceptance threshold}} \tag{1}$$

$$\frac{\sum_{\text{nurses}}(\text{Extension of general threshold} - \text{Use of extension})}{\text{Number of nurses with extension}} \tag{2}$$

Table 3 The unused slack when extending the general acceptance thresholds. A dash (-) indicates that the acceptance threshold was too tight (not feasible).

Instance	Float nurses [%]	Maximum number of consecutive workdays [days/nurse]	Overtime [hours/nurse]	Undertime [hours/nurse]
A01	0	0.36	0.00	5.28
A02	0	2.07	0.00	3.50
A03	0	3.00	1.00	10.19
A04	0	1.24	7.60	6.05
A05	0	0.44	0.94	2.50
A06	0	2.67	1.00	42.41
A07	0	1.15	1.00	0.00
B01	0	4.00	0.00	13.16
B02	0	1.21	4.00	34.85
B03	0	1.33	1.00	56.23
B04	0	1.75	-	9.03
B05	0	2.67	1.00	13.75

The acceptance thresholds for float nurses are accurate, as the number of float nurses we assign when employing the lexicographic goal programming methodology is exactly equal to the estimated acceptance threshold for all 12 instances.

The unused slack for exceeding the maximum number of consecutive workdays differs between the instances, ranging from almost no slack and up to a slack of 4 days per nurse. As we only extend the general threshold for specific days according to the nurses' requests, we will not see any unwanted violations of the general threshold even though the acceptance threshold is rather loose. Thus, having additional slack for this target is not a cause for worry.

The extension of the acceptance threshold for overtime is tight, with the unused slack being at most 1 hour for a majority of the instances. The two exceptions (A04 and B02) both include the Christmas holidays, where a part of the roster was fixed by the manager due to individual agreements with the nurses, resulting in more `Fixed working hours` (Algorithm 3, line 2). Furthermore, we must emphasize that the slack for all instances is less than the length of a single shift.

The unused slack for undertime is below two shifts per nurse for 75% of the instances. For the three remaining instances, the slack is quite substantial or ranging from 4 to 7 shifts per nurse. For all those instances, one or more nurses have significant undertime at the start of a rostering horizon, as a result of working too few shifts in previous horizons. As we cannot guarantee that we will be able to correct for the original deviation we need to set the acceptance thresholds accordingly. As a result, we see a lot of unused slack for nurses where we can reduce the deviation to some extent.

Overall, the thresholds for undertime were intentionally set to be sufficiently loose, as the undertime target conflicts with numerous other constraints and targets. For example, we may need increased undertime to ensure that we do not exceed the maximum number of consecutive workdays, along with ensuring a sufficient number of days off in general. Thus, when estimating the `Possible hours` (Algorithm 4, line 1) we must avoid too tight of a bound to ensure an overall feasible roster exists for the nurse. In many cases, we will never reach the acceptance threshold, because other constraints (such as coverage constraints) prevent such amount of undertime from being feasible for the problem as a whole (explaining the substantial slack in Table 3).

In general, the tightness of each acceptance threshold differs from instance to instance, and from employee to employee. Nonetheless, we confidently state that none of the acceptance thresholds are too loose, meaning that we will not see a practitioner rejecting a solution because of inaccurate thresholds.

5 Conclusion

In spite of a great deal of nurse rostering research throughout the years, the advances made in the research community rarely reach implementation in practice. The complexity of the current formulations is beyond the understanding of most practitioners, making them resistant to adopt them. Furthermore, an overall estimation of quality using a single objective function is far from the intricate decisions that practitioners face.

We have presented a shift in paradigm for nurse rostering research, designed to address these major disadvantages of the current state-of-the-art. This shift puts increased emphasis on the desired outcome, not thinking about the objective function's details but rather about what it can produce. This thought process is essential when working with personnel scheduling and other similar problems, where the choice of objective is debatable. At the conference, we will discuss the implications of working with targets and acceptance thresholds and present computational evidence of its potential for real-world instances.

We see many advantages arising from this modeling methodology. By adaptively setting acceptance thresholds we can better accommodate the unique needs and preferences of individual employees. Furthermore, providing practitioners with information about attainable acceptance thresholds is highly valuable, as we have seen that they frequently try to resolve inevitable constraint violations. Additionally, with accurate acceptance thresholds, we have an opportunity to increasingly automatize the process, reducing the overall need for manual interventions.

By moving away from the abstract concept of objective weights and determining attainable acceptance thresholds, we significantly increase the transparency. As we are moving towards a language that all can understand, we give practitioners a greater opportunity to understand and trust the models. Finally, by simplifying the information flow, we eliminate some critical challenges that practitioners have faced and thus reduce the time and effort needed to generate rosters. Therefore, this shift in paradigm can aid in increasing acceptance by practitioners, conceivably resulting in an increased adaption and implementation in practice.

Acknowledgements We thank the practitioners we have worked with, along with all practitioners that have collaborated with researchers throughout the years to bridge the gap from

research to practice. Furthermore, Elín Björk Böðvarsdóttir acknowledges Data and Development Support at Region Zealand, Denmark, for financial support and collaboration in her PhD project. Research partially supported by Data-driven logistics (FWO-S007318N).

References

Burke EK, De Causmaecker P, Petrovic S, Vanden Berghe G (2002) A multi criteria meta-heuristic approach to nurse rostering. Proceedings of the 2002 Congress on Evolutionary Computation, Cec 2002 2:1004,413, 1197–1202, DOI 10.1109/CEC.2002.1004413

Burke EK, De Causmaecker P, Vanden Berghe G, Van Landeghem H (2004) The state of the art of nurse rostering. Journal of Scheduling 7(6):441–449, DOI 10.1023/B:JOSH.0000046076.75950.0b

Burke EK, Curtois T, Post G, Qu R, Veltman B (2008) A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. European Journal of Operational Research 188(2):330–341

Böðvarsdóttir EB, Bagger NCF, Høffner LE, Stidsen T (2019a) A comprehensive integer programming formulation of the nurse rostering problem in Denmark. Technical report

Böðvarsdóttir EB, Bagger NCF, Høffner LE, Stidsen T (2019b) Data for research on nurse rostering in Denmark [Data set]. https://doi.org/10.5281/zenodo.3374636

Gärtner J, Bohle P, Arlinghaus A, Schafhauser W, Krennwallner T, Widl M (2018) Scheduling matters-some potential requirements for future rostering competitions from a practitioner's view. In: PATAT 2018 - Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling, pp 33–42

Kellogg DL, Walczak S (2007) Nurse scheduling: From academia to implementation or not? Interfaces 37(4):353–369, DOI 10.1287/inte.1060.0247

Kingston JH, Post G, Vanden Berghe G (2018) A unified nurse rostering model based on xhstt. In: PATAT 2018 twelfth international conference on the practice and theory of automated timetabling, Vienna, August, pp 81–96

Mihaylov M, Smet P, Van Den Noortgate W, Vanden Berghe G (2016) Facilitating the transition from manual to automated nurse rostering. Health Systems 5(2):120–131

Petrovic S (2019) "You have to get wet to learn how to swim" applied to bridging the gap between research into personnel scheduling and its implementation in practice. Annals of Operations Research 275(1):161–179

Petrovic S, Vanden Berghe G (2012) A comparison of two approaches to nurse rostering problems. Annals of Operations Research 194(1):365–384, DOI 10.1007/s10479-010-0808-9

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, DOI 10.1016/j.ejor.2012.11.029