Efficient constraint evaluation for the nurse rostering problem

Robin Tourlamain, Pieter Smet, and Greet Vanden Berghe

KU Leuven, Gebroeders de Smetstraat 1, 9000 Gent, Belgium
`robin.tourlamain@kuleuven.be`

Keywords: Nurse rostering, Constraint evaluation, Delta-evaluation

1 Introduction

Local search algorithms, which are often utilized to generate solutions for the nurse rostering problem, rely heavily on the value the evaluation function produces at each iteration. The speed of the evaluation process directly correlates with the number of iterations such algorithms can perform and, consequently, the quality of the solutions that are ultimately obtained. While the evaluation can be performed in a number of different ways, almost no publications report how they do so. Moreover, the significance of the evaluation approach extends beyond performance, with the reproducibility of algorithms and their results negatively affected when this design choice goes unreported.

Burke et al. [1] introduced a generalized constraint evaluation model for nurse rostering. It is one of the few counterexamples to the lack of reporting on evaluation functions. The model focuses on a flexible and user-friendly way to handle the evaluation of rosters. It re-evaluates the entire roster of at least one nurse after each update. We believe we can further increase the evaluation performance.

We introduce an efficient approach to constraint evaluation in local search algorithms, and leverage our results to highlight the importance of disclosing evaluation techniques. By accelerating the evaluation process, we aim to achieve better results with existing algorithms in time-constrained scenarios. We will demonstrate how it is worth treating one's evaluation approach as a design decision, given the tangible impact it has on the effectiveness of an algorithm.

2 Problem context

To evaluate nurse rostering solutions appropriately and efficiently, it is necessary to consider both the (soft) constraint type and the nature of the modification. The constraint types we take into consideration are those described by Bilgin et al. [2], which accurately model many real-world restrictions:

- Counters: constrain the number of occurrences of a specified set of assignments.
- Series: constrain the number of consecutive occurrences of a specified set of assignments.
- Successive series: constrain the consecutive occurrences of disjoint series.

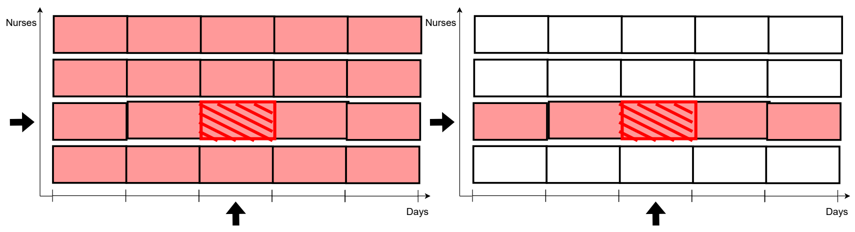
Note that we treat these constraints as soft constraints and that the degree of their violation determines the significance of the penalty. The evaluation function is a weighted sum of all penalties.

The modifications we consider are single assignment modifications: changing, adding, or removing a single shift on a day in a nurse's roster. Compound moves such as swaps can be created by simply linking together multiple modifications.

It is possible to assess the impact of one's evaluation function on instances from various existing datasets. The instances and constraints from Bilgin et al. [2], often referred to as the KaHo instances, and those from the international nurse rostering competition [3] all exclusively contain constraints that can be expressed as one of the three aforementioned general types and will therefore be used for the benchmarking process.

3 Evaluation methods

The most straightforward approach to constraint evaluation is to re-evaluate the entire solution after each update. This method is easy to implement and requires no extra logic besides the evaluation itself. However, upon closer inspection, the method requires many unnecessary computations. The evaluation can therefore easily be improved by recalculating only the penalties in the roster of the affected nurse. Figure 1 provides a side by side comparison of these two approaches.



(a) Full roster evaluation. (b) Partial roster evaluation.

Fig. 1: Evaluation comparison

While evaluating only a single nurse's roster is a step in the right direction, modifications seldom impact all constraints associated with a nurse's roster. In pursuit of greater efficiency, we consequently propose a delta-evaluation method that only iterates over both the constraints and the days that are affected by the modification. The method has the potential to dramatically accelerate iterative improvement algorithms for nurse rostering and other timetabling problems. Figure 2 provides a visual example of our proposed delta-evaluation method. The method will be detailed at the conference.

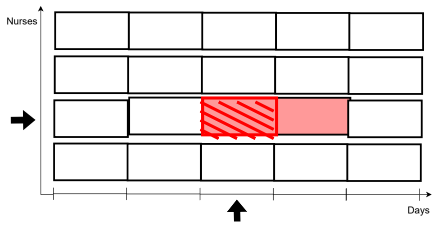


Fig. 2: Delta evaluation.

4 Preliminary results

The number of iterations per time unit impacts the number of iterations one can perform and, ultimately, the final solution quality attained. For the following benchmarking excercise, a single iteration consists of applying a random feasible modification and recalculating the roster penalty. There is no overarching search algorithm present in these experiments because we are simply counting the number of completed random iterations. We compare our delta-evaluation with the speed of evaluating the roster of a single nurse. By way of example, Table 1 documents the average evaluation speedup over ten 1-second runs per instance from the KaHo dataset.

Table 1: Evaluation speedup per instance.

Instance	**# Days**	**# Nurses**	**Speedup**
Hospital1-Emergency-Absence	28	27	x4.35
Hospital1-Emergency-Normal	28	27	x4.40
Hospital1-Emergency-Overload	28	27	x4.40
Hospital1-Geriatrics-Absence	28	21	x3.73
Hospital1-Geriatrics-Normal	28	21	x4.00
Hospital1-Geriatrics-Overload	28	21	x3.99
Hospital1-Meal Preparation-Absence	29	32	x3.34
Hospital1-Meal Preparation-Normal	29	32	x3.25
Hospital1-Meal Preparation-Overload	29	32	x3.57
Hospital1-Psychiatry-Absence	31	19	x5.89
Hospital1-Psychiatry-Normal	31	19	x5.91
Hospital1-Psychiatry-Overload	31	19	x6.19
Hospital1-Reception-Absence	42	19	x7.32
Hospital1-Reception-Normal	42	19	x7.37
Hospital1-Reception-Overload	42	19	x7.56
Hospital2-Palliative Care-Absence	91	27	x13.65
Hospital2-Palliative Care-Normal	91	27	x13.90
Hospital2-Palliative Care-Overload	91	27	x13.34

Evaluating a single nurse's entire roster leaves room for improvement, as the delta-evaluation outperforms it on every instance in the dataset. Indeed, our worst case performance would equal that of the partial roster evaluation. As the time horizon of an instance lengthens, the delta-evaluation eliminates the need to evaluate a greater proportion of days and therefore results in more significant speedups. The new delta-evaluation method can be applied within any iterative algorithm for roster optimization.

References

1. Burke, E. et al. (2003). Fitness Evaluation for Nurse Scheduling Problems. Proceedings of the IEEE Conference on Evolutionary Computation, CEC. 2. 10.1109/CEC.2001.934319.
2. Bilgin, B. et al. Local search neighbourhoods for dealing with a novel nurse rostering model. Annals of Operations Research **194**, 33–57 (2012). https://doi.org/10.1007/s10479-010-0804-0
3. Haspeslagh, S. et al. (2010). First international nurse rostering competition 2010. Annals of Operations Research. 218. 10.1007/s10479-012-1062-0.