Dealing with Perceived Fairness when Planning Doctor Shifts in Hospitals

Renaud De Landtsheer[1], Gaëtan Delannay[2] and Christophe Ponsard[1]
[1]*CETIC Research Centre, Gosselies, Belgium*
[2]*GeezTeem, Malonne, Belgium*

Keywords: User Acceptance, Scheduling, Rostering, Fairness, Metaheuristics.

Abstract: Producing planning of doctors' roles, including night shifts, on-call shifts (doctor can be called back in case of need), and regular working day in a hospital is complex and it is often difficult to effectively address all real world constraints. Furthermore, the produced solution requires a very good user acceptance to be effectively deployed in production. This paper reports on the application of metaheuristics to solve a planning problem in an hospital and focuses on user acceptance aspects of an algorithm. One of the key aspects to ensure user acceptance is to ensure the comprehensibility of the delivered solution. This includes the understanding of both the algorithm itself and its executions. The algorithm is a composition of local search, greedy algorithms, tabu search, and a few additional metaheuristic principles, and proved both good and fast enough.

1 INTRODUCTION

In the hospital environment, efficient coordination is of prime importance, notably to ensure that necessary workforce is available at the required time, with the required competences. This is because an hospital has to cope with possibly large flows of patients that cannot be interrupted, and medical care requires a set of specific competencies. Medical workforce is needed at different roles throughout the week. These include regular working role, on-site night shifts, and on-call duty. Roles planning must obey a given set of rules about resting time and cope with peoples' non-availabilities. Some equity between people is also required to maintain a good working atmosphere.

Setting up such planning is time consuming and can be the source of personal frustrations since planning composition, when is performed by a human colleague, can always be perceived as not objective and thus unfair. Fairness in duty rostering is emphasised in guidelines of many countries (NSW, 2015; NHS, 2016).

This paper presents a dedicated metaheuristic-based algorithm that has been developed to automatically compose such planning. The algorithm has been designed to maximize user acceptance and perceived fairness of the delivered planning. This means the algorithm produces a "transparent solution". Together with the planning it also gives a clear trace explanation of the taken decisions regarding a number of constraints or several priorities including constraints impacting the fairness both at short and long term. To make this trace comprehensible by doctors, the algorithm relies on a relatively simple metaheuristic.

Of course the doctor scheduling problem (DSP) and also the related nurse scheduling problem (NSP) have received a large attention and solutions to those problems have been proposed based on a large spectrum of techniques generally based on heuristic approaches such as tabu search, simulated annealing, stochastic optimization, genetic algorithms or ant colony optimization (Burke et al., 2004; Gendreau et al., 2007; De Causmaecker and Vanden Berghe, 2011). Yet, planning algorithms can always be biased in one way or another by the person in charge of the planning, and well-established generic algorithms such as CP, LNS, or MIP can be perceived by the user as efficient, but obscure black boxes. This might raise suspicions regarding the fairness of the tool. A fundamental design choice of our approach is to not rely on an existing optimization framework but to implement a dedicated search procedure. The risk to reach a sub-optimal solution was balanced with other constraints such as the need to fully justify the resulting planning and also some integration constraints.

This choice of developing an ad-hoc, simple metaheuristic is further discussed in the light of the user feedback collected from real world deployment. The

Landtsheer, R., Delannay, G. and Ponsard, C.
Dealing with Perceived Fairness when Planning Doctor Shifts in Hospitals.
DOI: 10.5220/0006657403200326
In *Proceedings of the 7th International Conference on Operations Research and Enterprise Systems (ICORES 2018)*, pages 320-326
ISBN: 978-989-758-285-1

deployment was conducted by the MedErgo company which triggered this research and has integrated the result in the scope of their product, with name *Nice-Watch* (MedErgo, 2016).

The paper is structured as follows: Section 2 presents the considered planning problem and its constraints with a focus on equity. Section 3 presents the technical approach to solve the problem and the considered metaheuristics. Then, Section 4 discusses on the different mechanisms that have been deployed in order to maximize user acceptance. Section 5 presents related work sharing the same concerns of facilitating the user acceptance. Finally, Section 6 draws some conclusions and research perspectives.

2 PROBLEM STATEMENT

This section presents the considered rostering problem, with a focus on aspects that can cause doctor frustration.

A *planning* ranges over a given period of time, and defines, for each day of this period of time and each role, who among the set of available doctors will fill in this role. *Roles* can be regular workday at a given place (anaesthetist in operating theatre no 2, emergencies), on-site night duty, or home based on-call duty.

With respect to rostering, doctors are characterized by **a set of legal and organizational attributes** such as contractual availability, qualification, maximal number of duty per month, degree of seniority and assignable duty roles.

Besides, there is also a set of personal constraints to be considered, including personal days off that are considered as strong constraints as soon as they are granted, and personal preferences regarding duty roles. These can be positive or negative preferences.

Planning must also obey as set of legal rules regarding resting times:

- a duty role at the hospital lasts for 24 or 25 hours. If happening during the week, it starts during the normal working hour starting at 8 am and lasts till 9 am the next day. If it occurs on a Saturday, Sunday or legal holiday, it starts at 9 am and ends at 9 am the next day. A duty role happening between and including Sunday to Thursday is automatically followed by a day off. For a duty role occurring on a Friday, the doctor gets half a day off that can (s)he can place anywhere. For a duty role happening on a Saturday, the next Monday is a day off. Legal holidays are treated as Sundays, and days before legal holidays as Fridays.
- A doctor can only fulfil a single role at a time.
- At any time, among all roles requiring a given qualification, one of them at least must be occupied by a senior doctor. For instance, emergencies and anaesthetist roles.
- A doctor cannot be working or on any form of duty when on holiday or when off its contractual working days in case of part time contract.
- There must be at least five days between two consecutive duties of the same doctor.
- In any period of four weeks, a doctor can have at most one duty occurring during the weekend.
- In case of a part time worker, a resting time following a duty cannot happen when the doctor is not working according to the contract.
- A resting time cannot occur during holiday.

A consequence of the compensation system for duty roles is that some of them are more attractive than others. Duty roles happening on a Thursday are the most attractive, since the doctor gets an extended weekend of three consecutive days. The least attractive roles are the ones happening during the weekend, since the compensation is smaller and they are not covering regular working hours, so that the doctor loses half a day off at the end. Friday and other week days duties are in between. This attractiveness is the main cause of frustration with duty roles assignment. **Planning composition should therefore be fair among doctors about this attractiveness of duties**. Inside a large doctor's team, it's indeed impossible to set up a monthly planning that takes into account all the loads or attractiveness of this peculiar month. One must thus figure out a way to spread out the workload among individuals and over longer time frames.

To summarize and prioritize, a planning must comply with the following elements:

1. first, it must comply with the strong constraints here above;
2. then, the personal preferences must be considered;
3. finally, the attractiveness of the planning must be evenly spread across all doctors.

Some inequities can be tolerated temporarily, but they must be compensated the next month. As discussed in the introduction, the planning must ensure some fairness between doctors, and must propose some mechanism to ensure that doctor have a good perception that the algorithm was fair, even though the planning might trigger some personal frustration.

There is also a requirement that the algorithm must be deterministic. This ensures that the person in charge of triggering the runs of the algorithm does not have the possibility to trigger the algorithm on demand to select a solution that better fits some non-expressed desires. As a consequence, all random functions used in our algorithm, notably to break ties, rely on a deterministic pseudo-random generator.

A last non-functional requirement is the efficiency of the planning engine: it must be able to generate a complete schedule for a single month within a few seconds.

3 SOLUTION DESIGN

3.1 Key Design Choices

It is difficult to ensure a good comprehensibility of the algorithm and its execution using existing scheduling engines relying on state-of-the-art algorithms. While they are efficient and can deal with fairness, they are also quite complex and hard to understand for non computer scientists, and they are not designed to provide a traceability of the resulting solution. In order to deal with this issue, a key decision is to implement a dedicated search engine not relying on any framework such as OscaR, Gecode, OR-tools, LocalSolver or any others (OscaR Team, 2012; Gecode Team, 2017; OR-tools Team, 2017; Benoist et al., 2011).

Attractiveness of the planning is a key element of doctor satisfaction, so that it must be quantified in order to reason upon it. The approach is to define a score of discomfort for each doctor on a given planning. All duties get a score of discomfort; the less attractive, the higher is this score. The discomfort of a doctor for a given planning is the sum of the discomfort of all duties (s)he is assigned to in this planning.

With this mechanism of discomfort score, we can model attractiveness, and compensate inequities from one month to another one, by accumulating the score of discomfort across months.

To find a solution, a greedy approach is used. Because it might fail to fill in a role, we therefore introduced the notion for a role of being unassigned. The key points are summarised here and detailed in the rest of this section.

- The main loop is a simple loop that allocates doctors to roles. It iterates onto unassigned roles, and assigns them to a doctor.
- The role is selected to be among the unassigned ones, as the one that has the fewest possible doctors, in view of the strong constraints and with an ordering based on the expressed preferences (positive or negative). In case of equality, a random role is selected.
- Relaxation is used in case the system reaches a step with no possible assignment. In this case a role that was assigned to a doctor is unassigned to generate the necessary degree of freedom.
- Diversification and cycle detection are also used to avoid the system iterating over the same set of partial allocations, leading to a dead end.

3.2 Metaheuristic: Greedy Approach

The algorithm itself is a greedy approach, with possibility to undo some of the greedy decisions. At each iteration a role is picked up, and a doctor is selected for this role. The algorithm has the possibility to relax an assignment in case of no doctor can be selected for the considered role. Finally, a tabu component is added to prevent relaxing assignments too quickly, and help escape local impossibilities.

Basically, the greedy search iterates on roles is a well-chosen order, and assign the current role to a doctor. It is summarised in Listing 1.

Listing 1: Greedy Algorithm for Role Assignment.

```
while(roleToAsign is not empty){
  val currentRole =
      select role in roleToAssign
             minimizing degreeOfFreedom(role)
  val doctorToAssign =
      select doctor in admissibleDoctor(currentRole)
             maximizing affinity(doctor,currentRole)
  assign(currentRole,doctorToAssign)
  update degree of freedom and admissibleDoctor
}
```

The iteration on roles is based on the degree of freedom of the role. Roles with the smallest degree of freedom are assigned first. The degree of freedom of a role is the number of doctors that can be assigned to this role, given the strong constraints and the existing assignments. It is updated every time an assignment is performed, or relaxed. A role assignment may impact the degree of freedom of another role because some constraints impose a minimal delay between shifts, notably through a resting period.

Ties for roles as well as doctor selection are broken based on a deterministic pseudo-random selection. We deliberately use a deterministic generator because we want several runs of the algorithm to produce the same output.

The affinity between a doctor and a role is a weighted sum involving:

- the preference (positive or negative) between the role and the doctor.
- the attractiveness of the considered role, based on weighting along the features of the role.

- a cumulated satisfaction score of the considered doctor that sums up the attractiveness of the past and already assigned role to this doctor.

3.3 Relaxing in Case of Impossibility

In case the algorithm reaches a point where the current role cannot be fulfilled by any doctor, one or more assignments are relaxed to provide the necessary freedom to the considered role. These are all related to the same doctor, who is then assigned to that role.

The doctor is selected such that he can be assigned to the role after some other assignments are relaxed. This excludes all doctors that are not available this day, based on their contract, for instance. The selected doctor also minimizes the number of assignments that are to be relaxed. Ties are broken based on a deterministic pseudo-random selection, again to ensure that the algorithm is deterministic.

3.4 Dealing with Allocation and Deallocation Cycles

The relaxation performed in case of a role cannot be assigned can lead to the algorithm oscillating in a closed loop: a role “a” cannot be assigned, so an assignment involving role“b” is relaxed. In turn to assign role “b” the algorithm can relax role “a”, etc. To prevent this, each assignment is added to a “tabu list”.

The tabu list is determined by setting a number of iterations during which the assigned role cannot be relaxed by the relaxation procedure.

4 PROTOTYPING AND VALIDATION

The proposed algorithm was implemented for MedErgo, a Belgian company provides a web-based software application for doctor planning, called *NiceWatch* (MedErgo, 2016). The system is composed of a global database containing the actual planning, and a web-based user interface in which doctors can post their own constraints, query their planning, interact with other doctors to barter duty roles, etc. A specific interface is also available to the coordinator to visualize all individual constraints, and set up the planning. This interface is depicted in Figure 1.

A corollary of the need to implement the full search procedure is that the algorithm could actually be developed by a skilled developer not specialist in optimisation with only some support from an expert. The implementation was carried out in Python, an

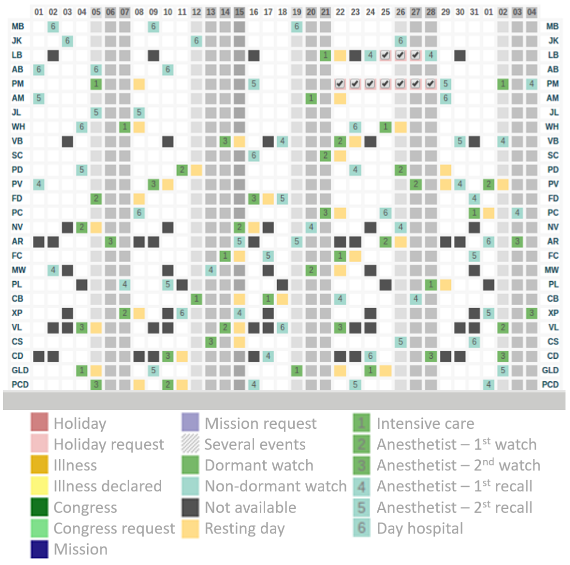

Figure 1: Partial with of the planning interface with possible constraints and allocations types.

Open Source multi-paradigm programming language focusing more on expressiveness and ease of integration than on performance as it is usually interpreted (van Rossum, 1991). The implementation was performed following an Agile approach. This allowed us to quickly validate that the algorithm performance would not be a bottleneck and then to progressively improve that performance to deal with more complex constraints and to produce the justification reports.

Shifts are typically scheduled each month based on the available doctor staff. The validation example presented here is composed of about 50 doctors which needs to ensure 5 simultaneous watch roles, some at every day, some only on week days. Table 1 shows the staff request for a typical month of 31 days.

Table 1: Typical Role Request for Different Watches.

	Watch role	# days
1	Intensive care	31
2	Anesthetist 1st watch	31
3	Anesthetist 2nd watch	31
4	Anesthetist 1st watch (recall)	31
5	Anesthetist 2nd watch (recall)	22
6	Day hospital	22

Figure 1 shows the planning interface displaying both the constraints (coloured square without number) and the proposed allocation (coloured square with number). The legend details the full set of possible constraints and allocations. During the planning process a full allocation trace is generated and available for checking the allocation process. A typical trace is displayed in Listing 2 for the first two slots of some allocated day.

Listing 2: Justification Trace.

```
4/0/4  Pref=1  P=6.53  Av=30  DDC|  Unavailable  (off)
1/0/1  Pref=1  P=5.90  Av=27  MAF
4/0/4  Pref=1  P=5.74  Av=31  MH
4/0/5  Pref=1  P=8.03  Av=31  DL
4/0/4  Pref=1  P=8.15  Av=31  KS
...
Day X - Slot 1 - Allocated to: MAF

4/0/4  Pref=1  P=8.72  Av=30  CC  |  Unavailable  (off)
4/0/4  Pref=1  P=9.11  Av=31  KB  |  minDistanceKO
1/0/2  Pref=1  P=3.94  Av=21  RCA|  minUnfavDistanceKO
1/0/2  Pref=1  P=4.15  Av=27  SC  |  minDistanceKO
1/0/2  Pref=1  P=4.44  Av=18  WM
1/0/2  Pref=1  P=4.52  Av=22  VN
2/0/3  Pref=1  P=4.86  Av=31  VRP
...
Day X - Slot 2 - Allocated to: WM
```

For each slot, a list reviewing possible doctor allocations is compiled. The list starts with doctors that cannot be allocated with a justification code whose explanation is detailed in Table 2. It is followed by a prioritised list of doctors using the ranking procedure described in the previous section. The first available doctor is the allocated one.

Table 2: List of Justification in Allocation Traces.

Justification	Description
Unavailable	Date is within the strong constraints of the doctor (according to work contract and vacations)
Unwanted	Date is within the doctor's wishes not to be on duty
MaxFrequency Reached	Maximum quota is reached for doctor's wanted watches
MaxFrequency ReachedOutside	Maximum quota is reached for doctor's unwanted watches
RecoveryRule Broken	This day is already assigned or is a recovery day
MinDistanceKO	The minimal delay between two wanted watches cannot be respected
MinDistance UnwantedKO	The minimal delay between two unwanted watches cannot be respected
minUnfavourable DistanceKO	The minimal delay between two unfavourable dormant watches cannot be respected
minUnfavourable DistanceOutside KO	The minimal delay between two unfavourable dormant watches cannot be respected
noSenior	No senior doctor would have been assigned to a set of paired roles
BlackListed	This doctor cannot be assigned here because it will lead to an impossibility to complete the schedule later

Our tool was deployed in production in August 2016 in different Belgian hospitals, as part of the *NiceWatch* web-based platform (MedErgo, 2016). Although our solver is not has been developed using standard data structures, the overall performance is quite good as scheduled can be produced within a few seconds and allow the user to wait for the result.

5 RELATED WORK

In DSP, fairness constraints are identified along other constraints and typically formulated as the fair distribution of different types of shifts among doctors with the same experience in (Gendreau et al., 2007). Fairness received specific attention in the emergency room context (Ferrand et al., 2011; Santos and Eriksson, 2014; Devesse et al., 2016).

MIP based heuristics have been used to create balanced scheduling from the set of doctors (Devesse et al., 2016). Integer programming has also been used to take into account constraints of the schedule, different preference ranks w.r.t. shifts, and the historical data of previous schedule periods to maximize the global satisfaction about the proposed shift schedule (Lin et al., 2014). The resulting shifts and days-off were fair and met the staff satisfaction.

In local search, an objective function is expressed as a weighted sum of soft constraint violations. Such an objective function has the advantage of being both easy to understand and to implement. However, they can produce unfair solutions because some high quality allocations can compensated low quality ones. A solution proposed by (Smet et al., 2012) is to use a function where the quality of the worst individual allocation will directly impact the overall solution quality. In doing so, a planing will not be improved at the expense of the worst individual case. Experimental results have confirmed the resulting solution is more fair, nevertheless a drawback is that the search seems less efficient given the new structure of the function. In addition to the lack of explanation traceability, this reinforces us about our dedicated approach.

A complete overview of techniques for NSP with some hint about how to come with personal constraints is presented in (Burke et al., 2001). Evolutionary algorithms are quite commonly used and an approach for the formulation of the fitness function has proven to be very powerful both to enable extendibility and to provide a quick and explanatory mechanism. We achieved the same results using our own approach and our belief is that the technique used is not the key point but rather the ability to take into account the right set of constraints, including historical data as well as the ability to produce justifications. The requirement for traceability also favour better architecture which in turn ease the ability to deal with more complex real-world constraints.

6 CONCLUSION

This paper reports on our research to develop a meta-heuristic that intertwines mechanisms borrowed from various technical origins (local search, greedy, tabu search, etc.) with a specific attention to enable a good fairness of the produced planning. A key point is that the transparency on fairness constraints is more important than the level of optimality of the solution for user acceptance. While a off-the self solution would certainly achieve better results (i.e. less discomfort) than our approach and also cope with fairness, it is hard to achieve a good level of transparency with them and hence there is a risk of early rejection. Our approach on the contrary is able to achieve transparency about "even discomfort". It also has the capacity to evolve to reduce the level of discomfort. In the end, the overhead of having to implement the algorithms without relying on a framework is also not so high when balanced with those advantages.

The proposed solution already proved quite useful and could establish a good level of trust and peace among users. Of course it can be improved. For example, it should be noted that not every discomfort can be assigned to a single shift but can also result from a sequence/set of shifts/tasks can also lead to discomfort. Our present work does not consider this and could be extended in this direction. In this process, the current data structures will probably show their limits by slowing down the computation. To cope with this, we could rely on data structures enabling incremental evaluation for faster exploration of the search space as done by local search solvers (OscaR Team, 2012). At this point it is interesting to consider switching to such a framework as users are less challenging the system fairness. In the process we will also be able to carry out computational comparison between both approaches. We also plan to work on a traceability feature for this framework.

The proposed approach could also be applied to other areas of scheduling where fairness and user acceptance are important issues (e.g. care pathways) with however the drawback that the underlying framework is not generic and must thus be revisited for each new problem.

ACKNOWLEDGEMENTS

This research was partly funded by the Walloon region as part of the PRIMa-Q CORNET project (nr. 1610019). We warmly thanks MedErgo for allowing us to share this case.

REFERENCES

Benoist, T., Estellon, B., Gardi, F., Megel, R., and Nouioua, K. (2011). Localsolver 1.x: a black-box local-search solver for 0-1 programming. *4OR*, 9(3):299 – 316.

Burke, E. K., De Causmaecker, P., Berghe, G. V., and Van Landeghem, H. (2004). The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499.

Burke, E. K. et al. (2001). Fitness evaluation for nurse scheduling problems. In *Proc. of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1139–1146 vol. 2.

De Causmaecker, P. and Vanden Berghe, G. (2011). A categorisation of nurse rostering problems. *Journal of Scheduling*, 14(1):3–16.

Devesse, V., Santos, M. O., and Toledo, C. (2016). Fairness in Physician Scheduling Problem in Emergency Rooms. In *Revista de Sistemas de Informao da FSMA*, pages 9–20.

Ferrand, Y. et al. (2011). Building cyclic schedules for emergency department physicians. *Interfaces*, 41(6):521–533.

Gecode Team (2017). Gecode - an open, free, efficient constraint solving toolkit. Available under the MIT licence from http://www.gecode.org/.

Gendreau, M. et al. (2007). Physician scheduling in emergency rooms. In *Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling VI*, PATAT'06, pages 53–67, Berlin, Heidelberg. Springer-Verlag.

Lin, C.-C., Kang, J.-R., Liu, W.-Y., and Deng, D.-J. (2014). Modelling a Nurse Shift Schedule with Multiple Preference Ranks for Shifts and Days-Off. Mathematical Problems in Engineering.

MedErgo (2016). NiceWatch - Complex Schedules within Seconds. http://www.nicewatch.net.

NHS (2016). Good practice guide: Rostering. https://improvement.nhs.uk/uploads/documents/Rostering_Good_Practice_Guidance_Final_v2.pdf.

NSW (2015). Principles of rostering. http://www.health.nsw.gov.au/Performance/rostering/Pages/principles.aspx.

OR-tools Team (2017). OR-tools: Operations research tools developed at Google. Available from https://code.google.com/p/or-tools/.

OscaR Team (2012). OscaR: Operational Research in Scala. Available under the LGPL licence from https://bitbucket.org/oscarlib/oscar.

Santos, M. and Eriksson, H. (2014). Insights into Physician Scheduling: a case study of public hospital departments in Sweden. *International journal of health care quality assurance/MCB University Press*, 27(2).

Smet, P., Martin, S., Ouelhadj, D., Ozcan, E., and Berghe, G. V. (2012). Investigation of fairness measures for nurse rostering. In *Practice and Theory of Automated Timetabling (PATAT), Son, Norway*.

van Rossum, G. (1991). The python programming language. https://www.python.org.