# Novel Meta-heuristic Approaches to Nurse Rostering Problems in Belgian Hospitals

CS 537
6/1/2017

Wie Hsing Li
Kushal Patel
Pratik Bhagtani

## Abstract

The problem in assigning nurses to personnel members of the members in hospitals is getting big across all belgium. This problem was overcome by applying a nurse-rostering algorithm in over 40 hospitals across belgium. Here we will discuss the implementation and working of the nurse rostering algorithm mainly for real world regulations and also the requirements of the hospitals. Also, the approach to schedule a nurse also depends on the administrative requirements, nurses personal preferences and their contracts. The main priority is to assure a permanent level of care of the patients.

In this algorithm, a solution framework is presented that applies a modular evaluation function. The model includes minimum and preferred coverage levels, self-definable (overlapping) qualifications and shift types, different contracts with modifiable constraints and other features.It also provides many options for initialising and for formulating various objectives and meta-heuristics for searching solutions. Different local search heuristics are applied in different neighbourhoods and we demonstrate the importance of introducing problem specific characteristics into the algorithms.

## Introduction

Due to the increasing amount of work in the hospitals all around Belgium, there is a serious problem for assigning nurses to the patients such that the patients do not have to wait for long before a nurse is available to assist them. There are several ways to improve the nurse rostering schedule for the Belgium hospitals. One potential way of easing this pressure is to develop better nurse rostering decision support systems that can help to produce rosters which employ resources more efficiently. However, there is more than just one goal when generating personnel rosters in hospitals. Resource efficiency is important but so is the satisfaction level of patients. Personnel rosters also affect the organisational structure of the hospital and they directly influence the private lives of the staff. It is therefore important to provide an interactive system that generates high quality scheduling solutions within a reasonable amount of computing time. Such schedules should cover the hospital requirements while avoiding patterns that are detrimental to the nurses and patients priorities. A manual scheduler will also work but can't maintain consistency and also ensuring the availability of nurse for every coming situation.

The presented solution concentrates on the short-term problem of assigning specific tasks to a sufficient number of qualified nurses. The problem of finding a high quality solution for the personnel scheduling problem in a hospital ward has been addressed by many scientists, personnel managers and schedulers over a number of years. A part of the problem is data, such as the number of personnel in a ward, the required qualifications of the nurses, definition of shift types, etc. is determined at the strategic level. Although these settings are not usually considered to be part of the nurse rostering problem, some longer term strategic decisions can affect the

solution strategies. The model described in this chapter therefore provides several possibilities for flexible problem setting. Examples are: shift types can be divided over several nurses, personnel demands can be expressed in terms of shorter intervals than shift length, night shifts can be assigned to a special category of night nurses, possibilities exist for creating part time work, people can temporarily be assigned to different wards in order to address emergencies, and many more.

Here we are focusing on the different steps during the development of this system for nurse-rostering problem. The steps involved are initial comparison with related system to ours, the modelling of the nurse rostering problem, the setting up of a solution framework and development of appropriate search techniques. We also investigate the applicability of a multi criteria approach for solving the nurse rostering problem. The possibility of assigning weights to certain criteria or conditions guides the search through a different set of solutions and produces interesting results of a very good quality. And Finally to compare the results between the algorithm and related developed algorithms.

## Problem Description

The main goal of the system is to create a schedule by assigning shift types to skilled personnel members, in order to meet the requirements in a certain planning period. Personnel have work regulations limiting their assignments. The software is implemented in over 40 hospitals in Belgium, of which some have as many as 100 wards. The system has helped to save time by replacing the manual system. Although the problem is user-defined to a large extent, the software has to be efficient in different settings. Every specific hospital ward should be able to formulate its problem within the restrictions of the model described further. A shift type is a predefined period with a fixed start and end time in which personnel members can be on or off duty. Different part time contracts require a large variation in start and end times and in duration. Table 1 presents a simplified example of a set of shift types. It is common that hospital schedulers define shift types according to their needs.

| | | Start | End |
|---|---|---|---|
| M | morning shift | 06:45 | 14:45 |
| L | late shift | 14:30 | 22:00 |
| N | night shift | 22:00 | 07:00 |

Table 1: Example of shift types

Planning periods for nurse rostering vary from a couple of days to a few months. Since cyclical rosters are not common at all, it is important for individual employees to know their schedule some time in advance. Long term scheduling, on the other hand, should

not be too detailed because the personnel requirements and preferences fluctuate and are not predictable in the long term.

Short planning periods enable the search algorithms to find good quality results much faster than longer planning periods. However, guaranteeing fairness among personnel members is restricted when the planning period is short. The roster, in which the shift assignments are stored, is called the *schedule*. We define assignment units as entities of minimum allocation in a schedule.

| | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| P1 | M | M | L | L | N | | |
| P2 | N | | N | L | L | | |
| P3 | M | M | M | M | M | M | M |
| P4 | M | | L | N | N | N | |
| P5 | M L | L | L | L | | | |

Figure 1: Roster example for 5 people (P1,. . . , P5) and 1 week; M, L, and N being the shift types introduced in Table 1

Here we illustrate the meaning of assignment units with a simple example. A fragment of a possible personnel roster is presented in Fig. 1. We notice that there are five people in the ward, and the shift types correspond to Table 1. Fig. 2 presents the *schedule* that corresponds to the roster of Fig. 1. Each column in the schedule represents an assignment unit. For every day of the planning period there are 3 columns, one for each shift type. The assignment units are ordered according to the start times of the shift types that they represent. When two shift types have the same start time, the first assignment unit

will match the shift type with the earliest end time. In this way the system assigns nurses for each ward in the hospital.

**Hard and Soft Constraints**

**Hard constraints** are those that must be satisfied at all costs. Soft constraints are those which are desirable but which may need to be violated in order to generate a workable solution. We call a *feasible* solution one that satisfies the following hard constraints:

- all the shift types specified in the personnel requirements have to be assigned to a personnel member
- one person cannot be assigned twice to the same shift on the same day - shifts can only be assigned to people of the right skill category.

The real-world situation addressed in this research incorporates a high number of **soft constraints** on the personal schedules. The soft constraints will preferably be satisfied, but violations can be accepted to a certain extent. It is highly exceptional in practice to find a schedule that satisfies all the soft constraints. The aim of the search algorithms is to minimise the penalties due to violations of these constraints.

The list of soft constraints can be divided into 3 categories.

1. Certain constraints hold for the entire hospital. Examples include:
    - Minimum time between two assignments
    - Allow use of an alternative skill category in certain situations

2. Another set of soft constraints is the same for all the people with the same contract (full-time, half-time, night nurses, etc). Values are set by the users. Examples include:

   - Maximum number of assignments in the planning period
   - Minimum/Maximum number of consecutive days
   - Minimum/Maximum number of hours worked
   - Minimum/Maximum number of consecutive free days
   - Maximum number of assignments per day of the week
   - Maximum number of assignments for each shift type
   - Maximum number of a shift type per week
   - Number of consecutive shift types
   - Assign 2 free days after night shifts
   - Assign complete weekends
   - Assign identical shift types during the weekend
   - Maximum number of consecutive working weekends
   - Maximum number of working weekends in a 4-week period
   - Maximum number of assignments on bank holidays
   - Restriction on the succession of shift types
   - Patterns enabling specific cyclic constraints (e.g. a free Wednesday afternoon every 2 weeks)
   - Balancing the workload among personnel

3. When individual personnel members have an agreement with the personnel manager or head nurse, then certain constraints can be implemented. Examples include:
    - Day off; shifts off
    - Requested assignments
    - Tutorship (people not allowed to work alone)
    - People not allowed to work together

## Solution Framework

The algorithm presented in this paper revolves around being able to constantly evolve and improve given new requirements and challenges. As the needs of the hospitals change or problems and weaknesses of the algorithm become apparent, this method allows for the algorithm to be flexible to change. The solution framework enables this fluidness and allows the

algorithm to be modular in nature as to allow the user, the hospital staff, to change the way it works. Mainly, it changes how the solution framework evaluates the solutions which the algorithm creates and what attributes it uses to evaluate the "correctness" of the solution. As we will see later in this paper, the evaluation of the solution is a core part of the algorithm and it uses this to strengthen the solution.

The algorithm is given a set of nurses and some given requirements for which the algorithm has to create a schedule. The schedule must follow the required requirements. There are two types of requirements, hard constraints and soft constraints. Hard constraints must be satisfied in the solution of the algorithm, other wise it is not considered a solution. Soft constraints, on the other hand, can be left out if a perfect schedule can not be found. Each soft constraint is given a value or weight and then a possible solution is evaluate against other possible solutions to find the best solution. These soft constraints are used as the benchmark in order to evaluate all possible solutions. The algorithm must also use all the nurses optimally and some nurse will have personal requests which the algorithm may also satisfy.

The core of the algorithm works by searching for the best possible solution given the search space. In this case, the search space is the set of all possible solutions. Using the soft constraints as the criteria, the solutions are evaluated against each other. It is important that this evaluation algorithm use very little space and processor time because the solution framework will likely have to compare and evaluate hundreds of combinations of possible solutions. Again, since hard constraints must be satisfied, the solution framework only has to worry about evaluating the soft constraints because any solution which does not satisfy hard constraints is not

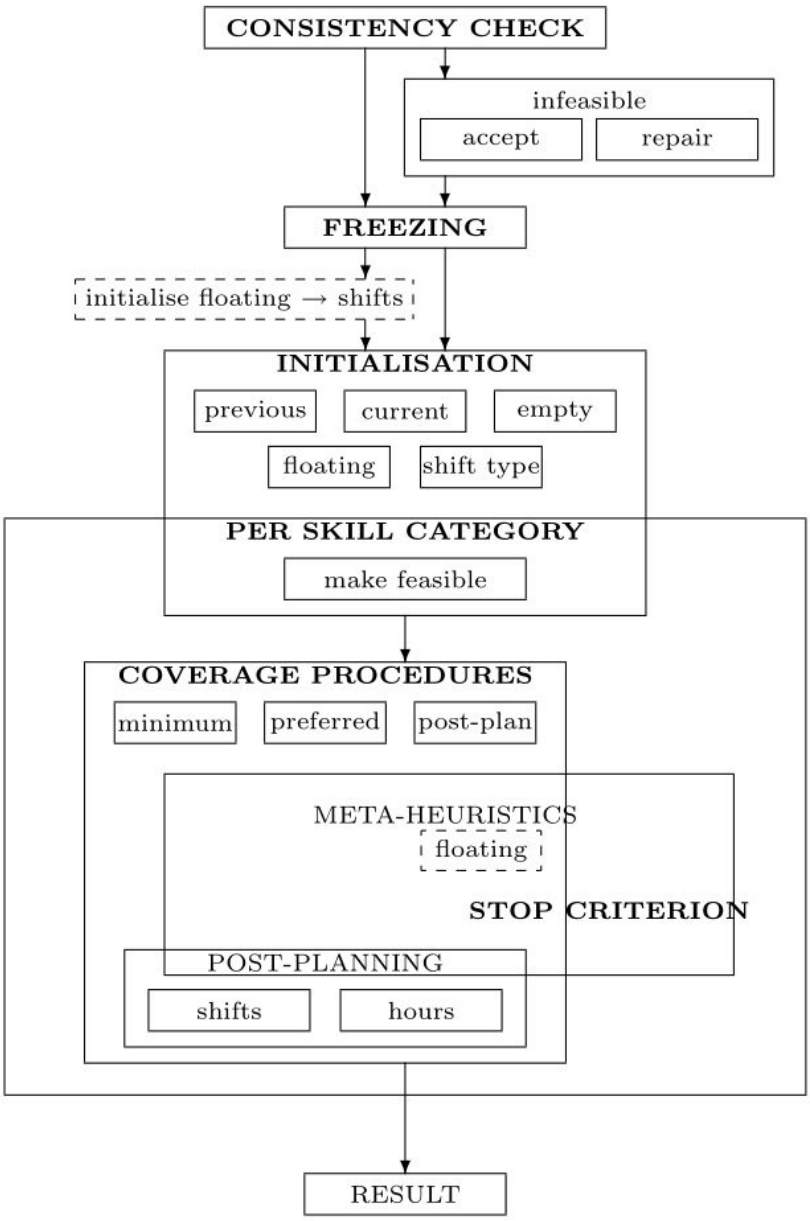
a solution. It is important that in most cases, all soft constraints will never be satisfied because of how complex the problem is.

The solution framework is also built in a modular way to allow the user flexibility. It provides the user with a way to extend the functionality and incorporate any real world problems, issues, or changes that need to be added to the algorithm. It also allows the user to provide feedback which helps improve the results of the algorithm. By allowing the user to provide feedback, the algorithm can improve the end result. Feedback is provided during the intermediate steps of the algorithm, while the data is being processed.

**Planning Procedures**

The overall goal of the solution framework is achieved by splitting up the entire process into smaller procedures. The authors call these the “Planning Procedures” which represent general tasks that the hospitals do in order to manually plan their schedules.

Below is the figure of the diagram which represents the Planning Procedures as a flow chart.

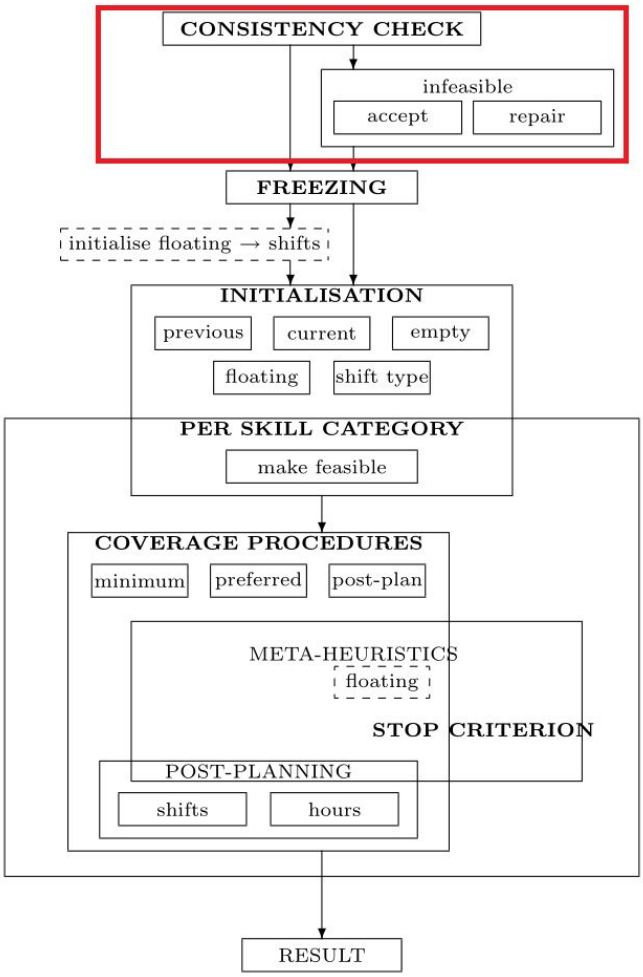



Each box in the diagram represents a procedure and boxes on the same level represent decisions or choices that the hospital can make. In total there are 6 procedures which are outlined below.

1. Consistency check on available people
2. Freezing parts of the schedule
3. Initialisation

4. Planning order of skill categories
5. Time interval requirements
6. Coverage Procedures

Consistency Check




In this procedure, the algorithm checks to see if a solution is even feasible. The algorithm checks to see if it is even possible to create a solution given the hard constraints. In some cases the hard constraints are so strict that a solution that satisfies all of them does not exist. In this case, the procedure will as the user, the hospital staff in charge of running this algorithm, for feed

back. The feedback in this case comes in the form of a decision. The user can choose two ways to solve this issue.

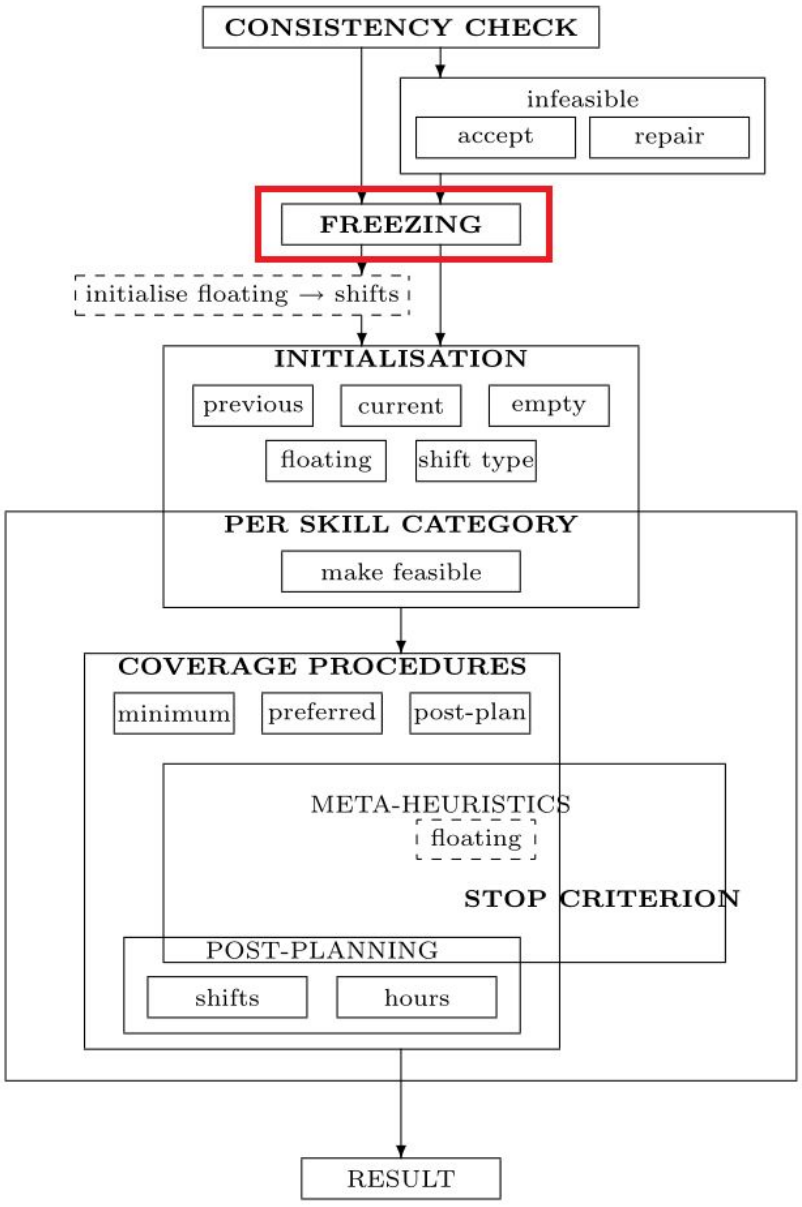
1. Relax the hard constraints
2. Violate a hard constraint

In the first of those two choices, the user can relax one or some or even all of the hard constraints to an acceptable point. The second choice is to try the procedure again by completely violating a hard constraint. This means that this particular hard constraint is effectively removed from the procedure. The user can also choose to remove or relax a soft constraint but there is no guarantee that this will cause the procedure to succeed.

Once the decision has been made, the algorithm will either proceed with the algorithm with the corrected constraints or the algorithm will stop.

Freeze Schedule

The procedure is to determine if any parts of the schedule needs to be frozen before proceeding. The hospital staff can choose to freeze parts of the schedule before starting the algorithm or during this procedure. Freezing the schedule or parts of the schedule tells the algorithm to proceed with creating the solution without changing certain parts of the schedule. These parts are specified by the user, the hospital staff.



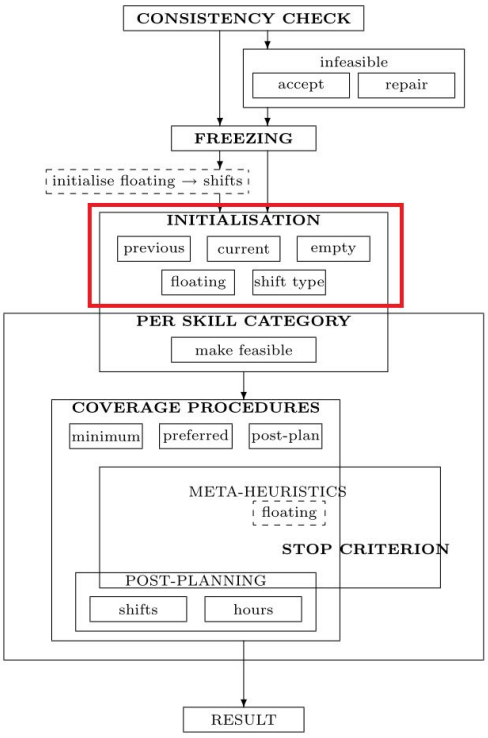

There are various reasons why the hospital would want to freeze some parts of the schedule. We have identified 3 of those reasons below.

1. Some hospitals prefer to manually schedule the schedule of some nurses.
2. To accommodate odd start and end times of some nurses.
3. To accommodate for emergencies for urgent matters.

One important thing to note is that no matter what part of the schedule is frozen or how many parts of the schedule are frozen, the solution framework will always take into account the schedule as a whole when evaluating it. This means that when a hospital planner manually schedules a schedule, the manually created schedule also get evaluated and assessed along with the algorithm generated schedule. Although the algorithm can not change this manually created schedule, it must evaluated it.

## Initialisation

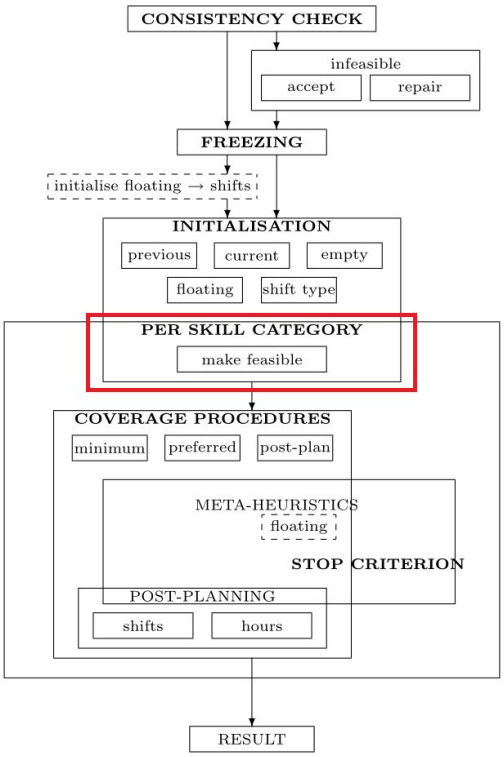
The next procedure is the initialisation, which consists of two phases. In the first phase, the schedule is loaded and a "seed" is chosen. The user can choose to pick a random schedule as the seed or the use can choose to use a previous schedule as the seed.



Once the schedule seed is picked, the schedule is made feasible by randomly adding and removing assignments in the schedule until the hard constraints are satisfied.

Planning order of skill categories

The next procedure is to schedule each skill category individually. As mentioned in the above sections, each wing or ward of the hospital needs to have a specific number of qualified nurses. This means that there must be a certain number of experienced nurses and a specific number of nurses with expertise of a certain kind in a given ward or wing. These criteria can vary from hospital to hospital or even from ward to ward. Specialized wards like the ICU or a ward for cancer patients can have individual requirements as well.

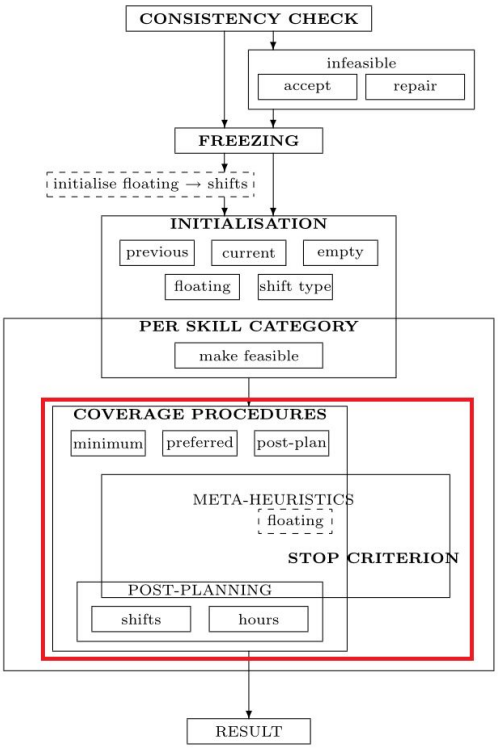



Time interval requirements

The next procedure is the time interval. In some cases the default or standard time intervals will not satisfy the needs of some hospitals. This is because hospitals can offer many different types of work hour types for their nurses. There hours can include part time work, 24 hour shifts, 12 hour shifts, split shifts, and many more combinations. Because of this large variation of work time intervals, the algorithm must account for this by including a way for the user to adapt the algorithm to their needs. This procedure allows for that.

Coverage Procedures

This procedure is the largest and the most consuming procedure of all of them. It is also the final procedure.




This procedure allows the user to adapt the algorithm to any special needs which their hospital could want. For example the number of actual nurses at a given ward can be lower than

the required nurses because of special rules. These rules could be that if there are many experienced nurses in a ward, then the total number of nurses can be lower.

**Meta-Heuristic and Hybrids**

In this section, we explore a method of dynamically modifying environments of search heuristics to address certain problem characteristics. The exploration of the search space can be improved by merging short sighted search along with greedy search in a wider environment.

The meta-heuristics that was developed to solve nurse roster problem change neighborhoods if a better solution cannot be found after a number of iterations. For modified environments to be considered as a solution, it must satisfy hard constraints. The cost function, which is the driving force for the heuristics, remains blind for improvements if it cannot interpret specific problem characteristics.

For short sighted improvements of environments, we will focus more on day to day shifts of the nurses. Such environments include Single shift-day neighborhood, Soft constrained related neighborhoods, and swapping large sections of personal schedules.

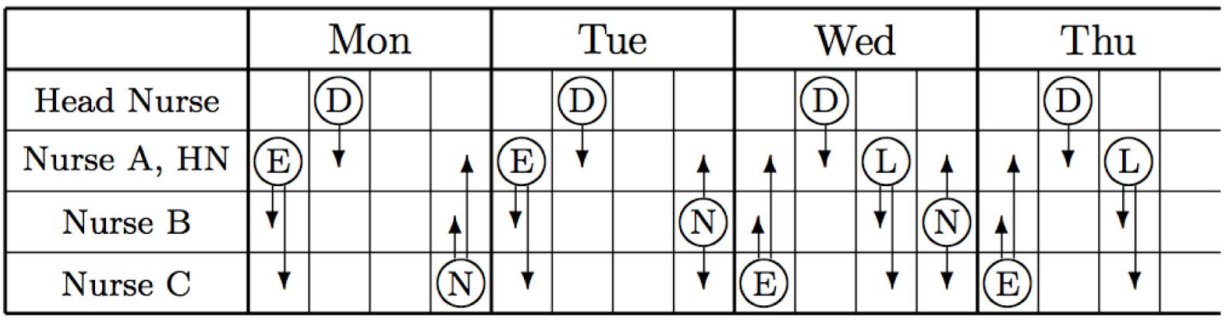


*Figure 2: Single shift-day neighborhood. Each day is separated into columns to signify the different shifts in a day; Early (E), Day (D), Late (L), Night (N).*

- Single shift day Neighborhood – One of the simplest environments to create because it is almost identical to the current solution differing by moving one assignment to a different nurse. Figure 2 demonstrates a given neighborhood with 4 nurses and 4 different shift types. Nurse A is able to replace the head nurse for the day shift ("D") while nurse's B and C are unable to. The arrows show the only possible moves the shifts are able to move to as to not violate any hard constraints. Moving the head nurse's day shift to nurse B would not serve a viable solution as does moving nurse A's E shift to the head nurse's.
- Soft constraint related neighborhoods – Nurses with practical experience were worried about violating soft-constraints rather than the overall solution. This would result in creating environments in a completely different way. The environments would then attempt to improve the current solution by only consider the soft constraints and are blind to the global effect. Using this approach would oppose the idea of the general evaluation function that abstracts from individual constraints. Therefore, the search procedure would not end after exploring this neighborhood, but be combined with another neighborhood for a more beneficial environment.
- Swapping large sections of personal schedules – Unlike the previous to neighborhoods solutions which differ from the current solution by altering a single shift type, this neighborhood explores schedules that differ considerably. This environment considers switching a part of the worst personal schedule (according to the evaluation function)

with any other schedule. Unlike moving single shift types (like the previous two), all assignments in a longer period are switched. Reallocating large groups of assignments is often less harmful for the quality of the resulting schedule than simply moving shifts around. However, the main drawback of using this method is that the number of neighboring solutions is very large thus requiring large amounts of computation time.

**Tabu Search**

To understand the following neighborhood search methods, we must first explore Tabu search as it is one of the main components in hybrid search algorithms.

Tabu Search is a Global Optimization algorithm and a Metaheurisitic or Meta-strategy for controlling an embedded heuristic technique [1]. The objective of the Tabu Search algorithm is to constrain an embedded heuristic from returning to recently visited areas of the search space. In our case this would be the scheduling rosters for our nurses. The strategy behind Tabu Search is to maintain a short term memory of the small changes of recent moves within the search space and preventing future moves from undoing those changes. If the changes are too large, this may introduce bias moves toward promising areas of the search space, as well as longer-term memory structures that promote a general diversity in the search across the search space.

```
Input: TabuList_size
Output: S_best
S_best ← ConstructInitialSolution()
TabuList ← ∅
While (¬ StopCondition())
  CandidateList ← ∅
  For (S_candidate ∈ Sbest_neighborhood)
    If (¬ ContainsAnyFeatures(S_candidate, TabuList))
      CandidateList ← S_candidate
    End
  End
  S_candidate ← LocateBestCandidate(CandidateList)
  If (Cost(S_candidate) ≤ Cost(S_best))
    S_best ← S_candidate
    TabuList ← FeatureDifferences(S_candidate, S_best)
    While (TabuList > TabuList_size)
      DeleteFeature(TabuList)
    End
  End
End
Return (S_best)
```

*Figure 3: Pseudocode for Tabu Search*

From figure 3, the listing shows a simple Tabu Search algorithm with a short term memory, without intermediate and long term memory management.

**Algorithms**

Different scenarios are likely in variable neighborhood search. The most efficient application to different neighborhoods is to search them in order of increasing size such as,

starting from a single-day shift environment and ending with swapping large sections of assignments. When a better solution is found, the search procedure restarts from the smallest size neighborhood. For large size environments it is nearly impossible for experience planners to make manual improvements thus making it redundant to modify such end states by applying small size neighborhoods afterwards.

**Hybrid Tabu Search**

One of the benefits of Tabu Search is its ability to escape from local optima. This is because of the searches ability to prevent results from going through the same areas. As long as there is some sport of improvement, Tabu Search performs like a steepest descent algorithm. The efficiency of Tabu Search can be further increased by applying diversification. The basic algorithm is the combination of Tabu Search with a single shift-day neighborhood (described earlier). Instead of using random diversification when the basic algorithm cannot generate better solutions (after a number of iterations), different neighborhoods can be used. This is also useful when considering problems such as: soft constraints on full weekends, improving the worst personal roster, and large improvements between two personal rosters.

Using this as a base, we can offer planners a choice between a fast algorithm that generates schedules of satisfactory quality in a short amount of time, and a more thorough schedule when a final solution is required.

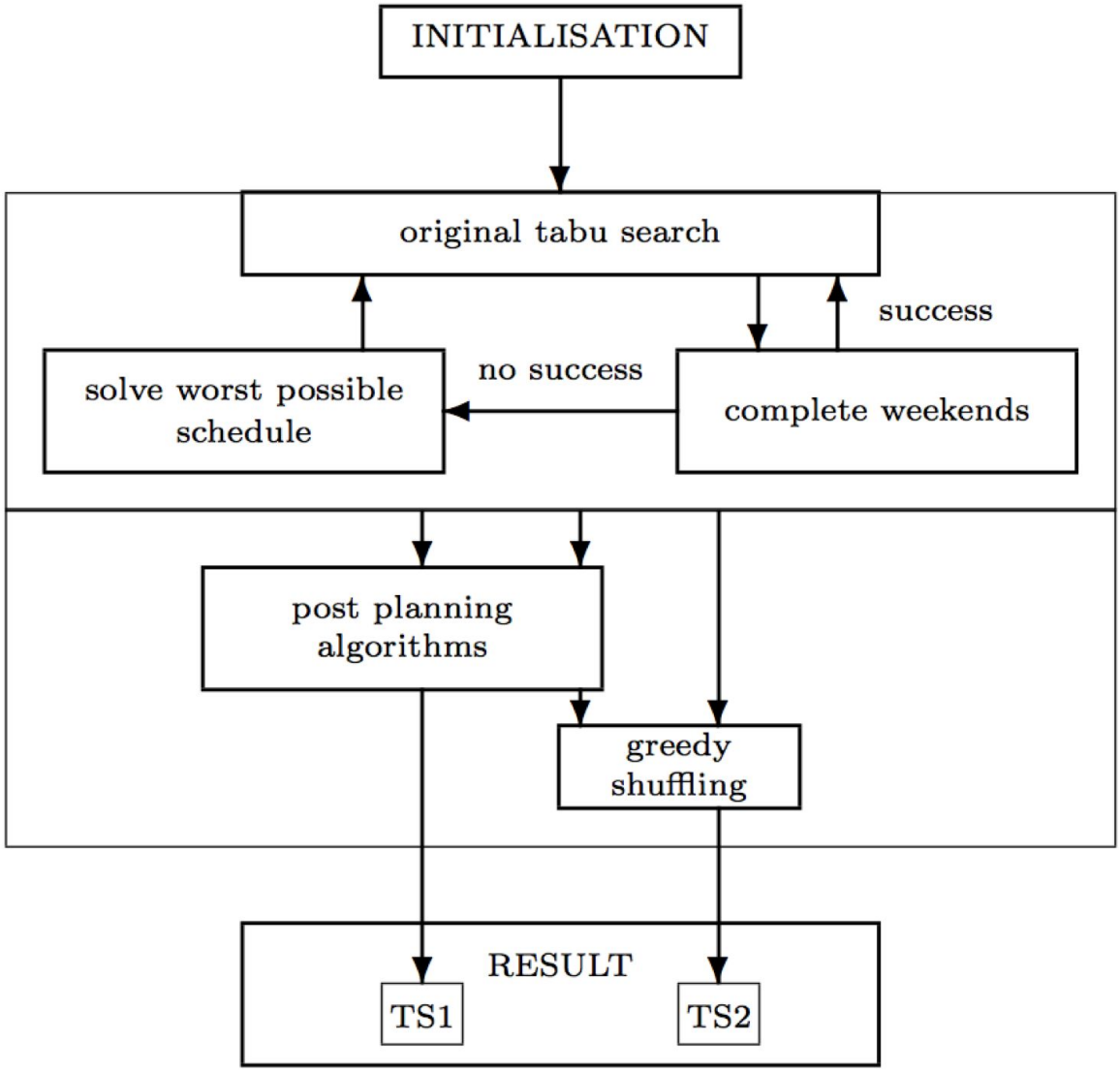



*Figure 4: Hybrid Tabu Search algorithms for the nurse rostering problem.*

Following Figure 4, we consider two versions of the Hybrid Tabu Search: TS1 Fast Planning and TS2 through planning.

- TS1: Fast Planning – When the basic Tabu Search algorithm cannot find improvements, the diversification incorporates single shift day neighborhoods normally consisting of

searching the weekend environment as well as the shuffle environment. This process is then repeated until the global stop criterion is reached.

- TS2: Thorough planning – Looking at Figure 3, TS2 consists of the same process as TS1 but with the added greedy shuffle. This combination requires more time but outputs higher quality results. The drawback with TS2 is that it requires a large amount of computation time.

**Memetic Algorithms**

Population based techniques can overcome problems that occur when a single solution evolves. Memetic algorithms apply local search on every individual of a population. This makes use of cross-over which copies good characteristics from the previous generations.

It is quite complex to implement cross-over operators to ANROM since standard operations to not maintain feasibility. Combining parts of a nurse schedules rarely outputs good quality rosters.

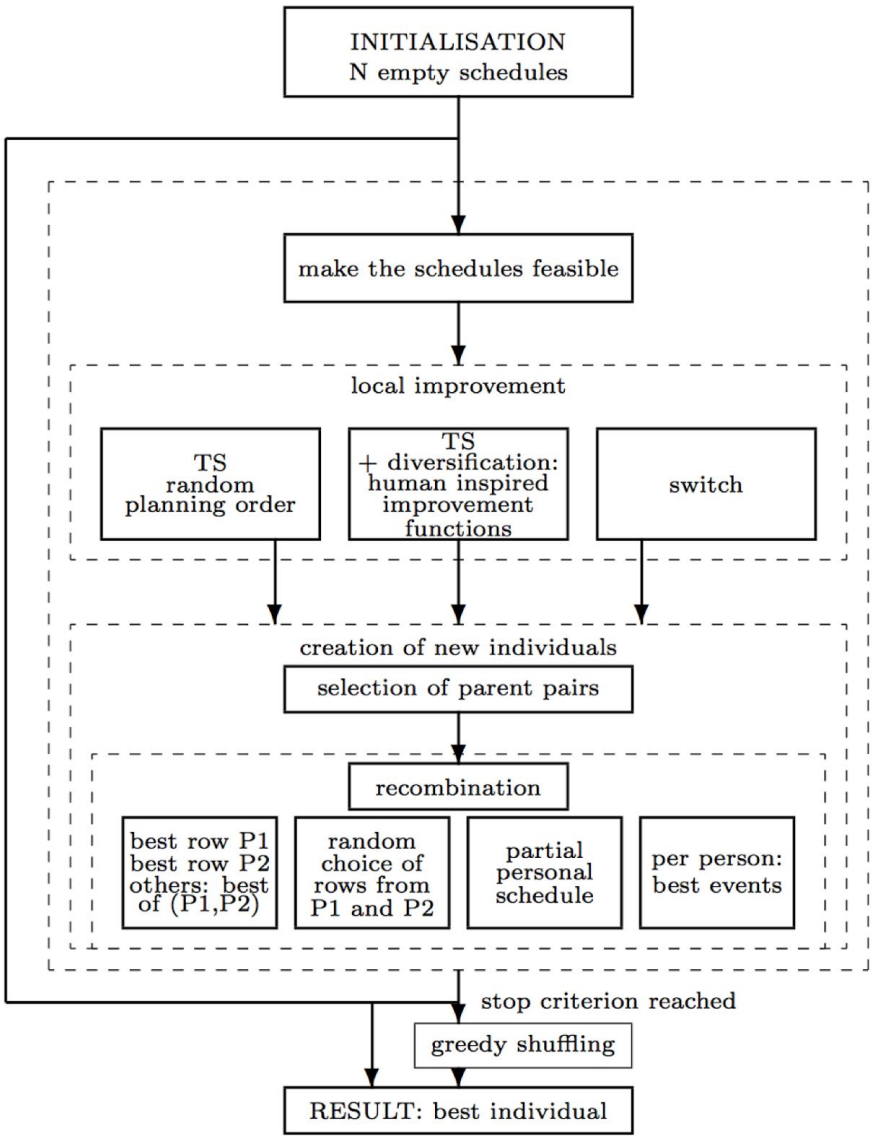



*Figure 5: Components of the genetic and memetic algorithms for nurse rostering problem*

Following Figure 5, this demonstrates how the memetic algorithms for ANROM is implemented.

Given initial population consists of N individual schedules that match the hard constraints. The quality of a roster is determined by the sum of the qualities of all individual

rosters. Therefore, cross-over operators will copy the good characteristics from personal rosters to the next generations. Different memetic algorithms exist such as: the simple memetic algorithm, diverse memetic algorithm, diverse memetic algorithm with random selection, memetic algorithm with cross-over, and memetic algorithm that copies the best x assignments.

- Simple Memetic Algorithm – This applies the steepest descent on each newly created person. A simple tournament would select the best parents. The first child is then obtained by copying the best personal schedule from the first parent, along with the best one (for another individual) from the other parent. A personal schedule is the work schedule for one individual. For the second child, we would start with the best solution of the other parent. Normally new individuals a re normally infeasible, but this can be remedied by randomly adding or removing assignments until the constraints are met.
- Diverse Memetic Algorithm – This is based on the Simple Memetic Algorithm, however the planning order of the skill categories are randomly chosen in the local search step.
- Diverse Memetic Algorithm with Random Selection – This applies to the general features of the Diverse Memetic Algorithm but all the personal schedules are randomly selected from both parent's pairs.
- Memetic Algorithm with Cross-over – This generates a random assignment unit for each person in the schedule, then personal schedules from 2 parents are combined. This also makes use of the steepest descent algorithm for each individual.
- Memetic Algorithm that Copies the Best x Assignments – This algorithm selects the assignments that induce the highest amounts of violations when removed for each personal schedule. The assignment(s) is/are then carried on to the new individual.

Diversity is also achieved by randomly making the schedules feasible. Later, the local search algorithms improve the individuals using Tabu Search.

**Results Using Hybrid Algorithms**

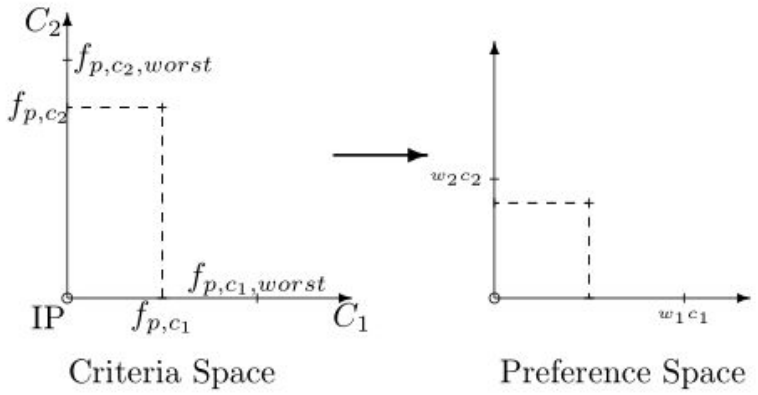
Copying entire parts of the parent schedules using the various Memetic Algorithms did not prove to be efficient. However, steepest descent does product acceptable solutions for individuals. In contrast, by copying very tiny partial schedules with good qualities (using cross-over), there was a much higher diversity and more freedom to improve the children rosters with local search. An assumption can also be made that the best x assignments of a schedule can influence the rest so much that solutions evolve towards quality schedules. The Memetic-Tabu hybrid Memetic Algorithm that copies the Best x Assignments behaves very well for complex problems. It is able to outperform all other algorithms and demonstrates the benefits of the hybrid approach.

## Multi Criteria Approach

The authors also implemented a multi-criteria approach which applies ‘compromise programming’, which is based on the concept of the distance to an ideal point.

This is done by creating a best possible result which the algorithm can produce and

mapping it on a chart. Then comparing it to the actual result generated by the algorithm and measuring the distance between the two on the chart.




The smaller the distance, the better the schedule.

## Conclusion

It is clear that to create a schedule for anything is much more complex and we have to remember a lot of constraints to be satisfied while creating the final schedule. This same applies with the nurse rostering problem in Belgian hospitals. Its main objective is to understand and automatically generate comfortable shift schedules for personnel members in order to meet the

staff coverage. We captured an extensive set of realistic constraints, and integrated them, together with explicit and implicit objectives, in a general, flexible model. After considering all of this the best suitable schedule is selected which satisfies maximum number of constraints. A set of meta-heuristics and hybrids are included in that solution framework, as the central search force for solving nurse rostering problems. We gained insight into the behaviour of applying heuristics and in making use of different problem specific neighbourhoods.

Implementing Nurse-Rostering system in belgian hospitals turned out to be huge beneficial as it helped reduce the scheduling time and effort as compared to the manual approach. The scheduling generated does not prefer any individual needs for something, instead it sets a schedule creating the best case scenario for both, the patients as well as for the nurses. Future research will certainly build upon the promising early findings of testing our multi criteria approach on nurse rostering. It opens perspectives for releasing the planners from setting the cost parameters. It is more realistic and increases the flexibility in setting the weights and thus modifying the relative priority of the constraints.

## References:


[1] "Tabu Search," Clever Algorithms: Nature-Inspired Programming Recipes. [Online]. Available: http://www.cleveralgorithms.com/nature-inspired/stochastic/tabu_search.html. [Accessed: 31-May-2017].

[2] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe: A Multi Criteria Meta-heuristic Approach to Nurse Rostering, Proceedings of Congress on Evolutionary Computation, CEC2002, Honolulu, IEEE Press, 2002, 1197-1202

[3] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe: Floating Personnel Requirements in a Shift Based Timetable, working paper KaHo Sint-Lieven, 2003

[4] P. De Causmaecker, G. Vanden Berghe: Relaxation of Coverage Constraints in Hospital Personnel Rostering, E.K. Burke, P. De Causmaecker (Eds.), Selected papers of 4th International Conference on Practice and Theory of Automated Timetabling, LNCS

[5] Edmund Kieran Burke, Patrick De Causmaecker,and Greet Vanden Berghe: Novel Meta-heuristic Approaches to Nurse Rostering Problems in Belgian Hospitals