A hyper-heuristic approach based upon a hidden Markov model for the multi-stage nurse rostering problem*

Ahmed Kheiri[a,*], Angeliki Gretsista[b], Ed Keedwell[c], Guglielmo Lulli[a], Michael G. Epitropakis[d] and Edmund K. Burke[e]

[a] Lancaster University, Department of Management Science, Lancaster, LA1 4YX, UK
[b] Queen Mary University of London, School of Electronic Engineering and Computer Science, Mile End Road, London, E1 4NS, UK
[c] University of Exeter, College of Engineering, Mathematics and Physical Sciences, Streatham Campus, Harrison Building, Exeter, EX4 4QF, UK
[d] The Signal Group, Glyfada, Athens, 16 674, GR
[e] University of Leicester, University Road, Leicester, LE1 7RH, UK

ARTICLE INFO

Keywords:
Hyper-heuristic
Optimisation
Healthcare
Scheduling

ABSTRACT

The importance of the nurse rostering problem in complex healthcare environments should not be understated. The nurses in a hospital should be assigned to the most appropriate shifts and days so as to meet the demands of the hospital as well as to satisfy the requirements and requests of the nurses as much as possible. Nurse rostering represents a challenging and demanding combinatorial optimisation problem. To address it, general and efficient methodologies, such as selection hyper-heuristics, have emerged. In this paper, we will consider the multi-stage nurse rostering formulation, posed by the second international nurse rostering competition's problem. We introduce a sequence-based selection hyper-heuristic that utilises a statistical Markov model. The proposed methodology incorporates a dedicated algorithm for building feasible initial solutions and a series of low-level heuristics with different dynamics that respect the characteristics of the competition's problem formulation. Empirical results and analysis suggest that the proposed approach has significant potential for difficult problem instances.

1. Introduction

Over many years, nurse scheduling problems have attracted extensive attention from the scientific community. The interest has been motivated by the practical importance of nurse rostering, which is strongly related to employee requirements satisfaction, clinical and cost imperatives, and the computational challenges posed by this complex class of optimisation problems. Indeed, rosters must determine a suitable number of qualified nurses to meet the cover requirements arising from patients in the hospital, adhering to regulations, distinguishing between temporary and permanent staff, ensuring fair distribution of shifts and accommodating leave requests and employee preferences [1]. Among staffing and scheduling decision problems, nurse rostering is by far the most popular. In a recent survey on personnel scheduling, Bergh et al. [2] counted seventy-four papers in nurse rostering accounting for more than one quarter of all papers classified per application area.

As further evidence of the interest of the research community in this class of problems, two international competitions have been held over the last decade or so. The First International Nurse Rostering Competition (INRC-I) was run in 2010 [3] with a focus on assigning nurses to shifts in a fixed planning horizon, subject to hard and soft constraints. As an outcome of this competition, significant results were reported [4, 5, 6]. Some of the test instances were solved to optimality, while new best solutions were computed for others. Following the success of INRC-I, the second competition (INRC-II) was launched in 2014 [7, 8]. The focus of INRC-II was to address a multi-stage version of the problem on an extended planning horizon. The multi-stage setting reflects real operational environments more accurately, as the rostering of a 'stage' (a week) is influenced by the assignments of nurses in previous stages [9]. Therefore, the multi-stage nature of the problem is approached by solving single stage (one week) problems with an outlook to the long-term performance required at the end of the planning horizon, while the evaluation of a roster is assessed in the last stage of the planning horizon, where all constraints and goals can be accurately calculated for the desired multi-stage solution [7, 8]. Fifteen computational frameworks (algorithms) participated in the INRC-II competition, including the sequence-based selection hyper-heuristic (SSHH) algorithm presented here.

Hyper-heuristics represent a category of optimisation methods that emerged to address optimisation problems across a wide category of different problem domains. The term hyper-heuristic first originated in 2001 [10] to define high-level approaches that are able to select or generate low-level heuristics without drawing extensively on problem specific information. Selection hyper-heuristics choose heuristics from a predefined set of low-level heuristics within a framework, and use them to carry out a sequence of changes (perturbations) to an evolving solution to improve its quality. Generative hyper-heuristics evolve and identify new heuristics based on an input set of low-level heuristics that exhibit improved search capabilities and performance [11, 12].

The motivation behind the development of a sequence-based selection hyper-heuristic (SSHH) for the nurse rostering problem is to build a more forceful search approach of the

* This research was supported by EPSRC under grant EP/K000519/1.
*Corresponding author (a.kheiri@lancaster.ac.uk)

solutions' space. SSHH is one of the first hyper-heuristics that employs sequences of heuristics, while its promising performance gains have been verified over a wide set of problem domains, such as school timetabling [13], vehicle routing [14, 15], inventory routing [16], wind farm layout optimisation [17], and urban transit routing [18] among others. The proposed SSHH algorithm utilises a specific construction procedure to generate an initial feasible solution and exploit the search power of a set of nine dedicated low-level heuristics to effectively search the optimisation landscape and improve the initial solution while respecting feasibility. The proposed low-level heuristics have different diversification and intensification characteristics and cover a wide variety of heuristic types, such as perturbation and ruin and recreate.

The SSHH algorithm placed third among all the competing algorithms of the INRC-II challenge. However, the proposed algorithm was the only participant among the top three entries of the competition that was able to locate a feasible solution to every instance and performed comparably well, in statistical terms, against the winner in the most difficult instances. Moreover, the algorithm exhibited robust computational performance in terms of the resulting solution quality across all the test instances given the available computational time budget.

The remainder of the paper is organised as follows. The recent related work of the problem is presented in Section 2. The multi-stage nurse rostering problem, including its main characteristics and the models used in this study are briefly described in Section 3. The proposed methodology including the low-level heuristics used and the parameter settings defined are presented in Section 4. The paper continues with Section 5, where a thorough experimental evaluation of the proposed framework on a wide set of nurse rostering problem instances is presented. Finally, the paper concludes in Section 6 with a summary of the experimental findings of this work and provides some pointers for future work.

2. Related Work

Several exhaustive reviews demonstrate a wide body of literature on the topic of nurse rostering problems [1, 19, 20]. These surveys describe a large number of nurse rostering problems with very different features and characteristics that are the consequence of different regulations and different organisational practices. To classify the different nurse rostering problems, De Causmaecker and Vanden Berghe [20] presented a notation scheme along the lines of the Graham notation for scheduling problems [21]. The notation helps to position the problem in the vast body of research on the subject.

In addition to different models and/or formulations of the nurse rostering problem, a wide variety of approaches have been proposed in the literature to solve instances of problems, including exact [22, 23], heuristic [24, 25, 26, 27] and hybrid [28, 29] methods. Santos et al. [22] proposed an integer programming technique with improved cut generation procedures and primal heuristics to solve to optimality and provide tight dual bounds for a variety of nurse rostering problem instances. He and Qu [23] studied a hybridisation of a column generation with a constraint programming approach to model and address various benchmark problem instances with complex constraints. The hybridisation exploits the expressiveness of constraint programming to model complex constraints and the effective relaxation and reasoning of linear programming enhanced with novel column generation techniques, resulting in a highly efficient hybrid algorithm. Recently, Rahimian et al. [28] also proposed a hybrid algorithm, which combines integer programming and constraint programming to efficiently solve the problem. In particular, constraint programming is used to generate a feasible solution to be used to start the IP solver.

Variable Neighbourhood Search (VNS), Tabu Search, Iterated Local Search, Genetic Algorithms and Simulated Annealing represent well-studied algorithms that have been applied to address nurse rostering problems. The interested reader may refer to Burke et al. [19] for a classification of the nurse rostering literature - up to 2004 - that is not limited to the problem definition (formulation) but also includes the solution method perspective. Moreover, in this review, the authors provide information on the data used and whether the reviewed approaches were applied in practice or not. More recently, Zheng et al. [26] proposed a simplified VNS approach that employs a randomly combined group of operators to iteratively search for incumbent solutions, and a cycle shift operator to diversify the search space when no improvement is achieved within a certain number of iterations. Knust and Xie [27] presented a simulated annealing approach that proved to be effective in finding good quality and robust solutions in a short amount of time, i.e., its computational performance was not negatively influenced by the specific nature of the problem instances solved.

In an attempt to solve the nurse rostering problem efficiently, hybrid approaches that combine exact and heuristic methods have been recently proposed to exploit the advantages of both worlds. For instance, Rahimian et al. [29] proposed a hybridisation scheme, closely related to the work presented in [28], that combines integer programming and the VNS heuristic.

Most, if not all, the approaches described so far suffer from lack of generality across different problem domains. A new category of methods emerged in response to this need, which are referred to as hyper-heuristics [11]. The nurse rostering and the more generic field of personnel scheduling problem, has attracted a significant number of hyper-heuristics researchers. Representative examples of effective hyper-heuristics for addressing nurse rostering and personnel scheduling problems include but are not limited to [30, 24, 31, 25] and [32, 33] respectively.

The multi-stage nurse rostering problem formulation proposed in the INRC-II competition attracted several teams to address the challenge. Römer and Mellouli [34], the winning team, applied a mixed-integer linear program methodology to solve the problem that was formulated as a multi-commodity network flow model. ORTEC [35] employed their commercial solver and enhanced it with an ejection chain method to improve its performance. Legrain et al. [36] developed a dynamic math-heuristic based on a primal-dual algorithm and embedded it into a sample average approximation algorithm. The weekly "static (scenario)" version of the problem is solved with a novel branch-and-price algorithm. The branch-and-price approach is used as a routine to reconstruct the rosters of some nurses within an adaptive large neighbourhood search. The adaptive large neighbourhood search algorithm, in the attempt to find a better solution iteratively destroys and repairs a part of the current solution [36]. A detailed overview of all the competing team algorithms is available in Ceschia et al. [8].

Other algorithms have been proposed for the multi-stage nurse rostering problem formulation of INRC-II. Mischek and Nysret [37] and Thi Thanh Dang et al. [38] developed hybrid methods between integer programming and various local search algorithms. More recently, Ceschia et al. [39] proposed an approach that employs a composition of large neighbourhoods guided by the Simulated Annealing metaheuristic to solve the static version of the problem. As part of the study, the authors investigated different neighbourhood structures to obtain a better exploration of the search space.

3. Problem Description

In this section, we present the nurse rostering problem proposed in the INRC-II challenge. The focus of the challenge is to capture the dynamic aspects of the problem by considering a planning horizon of a given number of weeks. At the beginning of each stage (week), the "solver" has to compute the roster for the current week without having any information about the future but relying exclusively on the information up to the current stage (week). The objective of the solver is to provide rosters for all the weeks that are "globally" optimal or near-optimal, i.e., optimal for the complete planning horizon. A detailed description of the problem studied in this work can be found on the INRC-II competition website. For the sake of completeness, we here provide a mathematical description of the problem.

The multi-stage problem formulation requires solving a set of stages $\mathscr{W} = \{w_1, \dots, w_{|\mathscr{W}|}\}$, each corresponding to one week. At each stage, it involves deciding at which shifts $S = \{s_1, \dots, s_{|S|}\}$ and on which days $D = \{d_1, \dots, d_{|D|}\}$ each nurse $N = \{n_1, \dots, n_{|N|}\}$ should work. Each nurse may have multiple skills $K = \{k_1, \dots, k_{|K|}\}$, and for each skill, different coverage constraints are required, where a coverage constraint is defined as the minimum required (or preferred) number of nurses of each skill $k \in K$ at any time in the planning horizon [8, 40].

An instance of the problem is identified by three types of information:

- **Scenario information** that is global to all weeks (stages) and includes:
  - Planning horizon in terms of weeks $|\mathscr{W}|$.
  - List of available skills K, such as, head and trainee.
  - List of contracts $C = \{c_1, c_2, \dots, c_{|C|}\}$, e.g. full time and part time. Each contract $c \in C$ specifies the minimum and maximum total number of assignments in the planning horizon (Q_c^ρ and Q_c^ν, respectively); the minimum and maximum number of consecutive working days (G_c^ρ and G_c^ν, respectively); the minimum and maximum number of consecutive days-off ($\mathscr{G}_c^\rho$ and $\mathscr{G}_c^\nu$, respectively); the maximum number of working weekends in the planning horizon (B_c^ν); and whether the complete weekend constraint to the nurse is expected to be satisfied (W_c).
  - List of nurses N.
  - Each nurse $n \in N$ is associated with a single contract $c^n \in C$.
  - Each nurse $n \in N$ is associated with a set of skills $K^n \subset K$.
  - List of shift types S such as early, late and night.
  - List of forbidden shift type successions $\mathscr{S}$.
  - Minimum and maximum number of consecutive assignments of each shift type (G_s^ρ and G_s^ν, respectively).
- **Week data information** that is specific to a single week (Monday - Sunday). This information includes minimum and optimal coverage requirements for each shift type, for each week day and for each skill ($\mathscr{R}_{s,d,k}^\rho$ and $\mathscr{R}_{s,d,k}^\nu$, respectively), and nurse preferences for specific days. Two types of preferences are considered: $U_{n,d,s}$ and $V_{n,d}$. If $U_{n,d,s} = 1$, then nurse n has requested to not work at shift s on day d. If $V_{n,d} = 1$, then nurse n has requested to take a day off on day d.
- **History information** which is carried over from the preceding stage. This information includes border data (four data sets) and the "Total worked shifts" and "Total number of worked weekends" counters. History information includes the following:
  - Last assigned shift of each nurse s_μ^n. For example, if $s_\mu^n = night$ then nurse n is assigned to 'night' shift on Sunday of the week before the planning period. Note that the forbidden shift type successions must be avoided at the beginning of stage, as well as that this can be empty at the beginning of the process.
  - Number of consecutive worked shifts l_μ^n. For example, if $l_\mu^n = 2$ then there is a shift assigned to nurse n on Saturday and another shift on Sunday of the week before the planning period.
  - Number of consecutive days-off f_μ^n. As an example, if $f_\mu^n = 1$ then there is no shift assigned to nurse n on Sunday of the week before the planning period.
  - Number of consecutive worked shifts of the last shift type $l_\mu^{s^n}$. For example, if $s_\mu^n = night$ and $l_\mu^{s^n} = 3$ then nurse n is assigned to 'night' shift on Friday, Saturday and Sunday of the week before the planning period.
  - Total worked shifts. At the first week, the counter is assigned to zero.

– Total number of worked weekends. At the first week, the counter is assigned to zero.

The INRC-II problem contains four types of hard constraints (H) and eight types of soft constraints (S). A solution must achieve feasibility and minimise the "evaluation function", i.e., a weighted sum of the soft constraints' violations (Equation 1):

$$Evaluate(R) = \sum_{i \in \{1,2,\dots,|S|\}} W_{Si} \times V_{Si}(R) \tag{1}$$

where R is a given roster, W_{Si} indicates the weight associated to constraint Si, V_{Si} indicates the amount of violation of constraint Si for the given roster R. The weight values of the soft constraints are provided in Table 1.

We proceed with the description of the set of hard constraints that has to be necessarily satisfied. To avoid repetition, we first define the decision variable $x_{n,s,d,k}$ as follows:

$$x_{n,s,d,k} = \begin{cases} 1 & \text{if nurse } n \text{ is assigned to shift } s \text{ on day } d \\ & \text{utilising skill } k; \\ 0 & \text{otherwise.} \end{cases}$$

and we continue with the description of the set of hard constraints:

- **H1 Single assignment per day:** A nurse can cover at most one shift per day. This is represented as:

$$\sum_{s \in S} \sum_{k \in K} x_{n,s,d,k} \leq 1, \quad \forall n \in N, d \in D \tag{2}$$

- **H2 Under-staffing:** Minimum requirements for each shift and for each skill per each day must be satisfied. This is mathematically formulated as:

$$\sum_{n \in N} x_{n,s,d,k} \geq \mathscr{R}^{\rho}_{s,d,k}, \quad \forall s \in S, d \in D, k \in K \tag{3}$$

- **H3 Shift type successions:** Shift type assignments of a nurse in two consecutive days must avoid the forbidden shift types successions. The mathematical representation is:

$$\sum_{k \in K} (x_{n,s_1,d-1,k} + x_{n,s_2,d,k}) \leq 1, \\ \forall n \in N, d \in D, (s_1, s_2) \in \mathcal{S} \tag{4}$$

- **H4 Missing required skill:** A shift $s \in S$ of a given skill $k \in K$ must necessarily be fulfilled by a nurse $n \in N$ having that skill $k \in K^n$.

We now describe the soft constraints whose violations (V_{Si}) are accounted in the evaluation function that is presented in Equation 1.

- **S1 Insufficient staffing for optimal coverage:** Number of nurses for each shift, and for each skill per each day should be greater than or equal to the optimal requirement. Each missing nurse will be counted as a violation. Mathematically, the number of violations of this constraint V_{S1} is:

$$\sum_{d \in D} \sum_{s \in S} \sum_{k \in K} max(0, \mathscr{R}^{\nu}_{s,d,k} - \sum_{n \in N} x_{n,s,d,k}) \tag{5}$$

- **S2 Consecutive days off:** Minimum and maximum number of consecutive days off should be respected. Each extra or missing day will be counted as a violation. The evaluation involves also the border data, such as the last worked shift of each nurse. To model the minimum and maximum requirements of consecutive days off, we need to introduce the following additional (artificial) decision variables for each day of the planning horizon and for each nurse:

$$s_{n,d} = \begin{cases} 1 & \text{if nurse } n \text{ is off on day } d; \\ 0 & \text{otherwise.} \end{cases}$$

These variables are the slack variables of Equation 2.

$$s_{n,d} - s_{n,d-1} \leq s_{n,d+\tau} - \sum_{t=1}^{\tau} \xi_{n,d,t} \\ \forall n \in N, d \in D, \tau = 1, \dots, \tau^n_{min} \tag{6}$$

where τ^n_{min} is the minimum number of consecutive days for nurse n. The variables $\xi_{n,d,t}$ are binary and are introduced to capture the interruption of the minimum consecutive days off requirement. The violation of this requirement will be accounted for in the objective function by the term:

$$\sum_{n \in N, d \in D, t=1,\dots,\tau^n_{min}} (\tau^n_{min} - t) \cdot \xi_{n,d,t} \tag{7}$$

The violation of the requirement on the maximum number of days off is calculated by:

$$max\{0; \sum_{t=\tau^n_{max},\dots,\tau} s_{n,d+t} - (\tau^n_{max} + \tau - 1)\} \\ \forall n \in N, d \in D, \tau\tau^n_{max} + 1, \dots, |D| \tag{8}$$

where τ^n_{max} is the maximum number of consecutive days for nurse n.

- **S3 Consecutive assignments:** Similar to S2 but considering the number of consecutive assignments.

- **S4 Consecutive assignments per shift type:** Similar to S2 but considering the number of consecutive assignments per each shift type.

- **S5 Preferences constraint:** Each assignment to an undesired shift is counted as a violation. Mathematically, the number of violations of this constraint V_{S5} is:

$$\sum_{n \in N} \sum_{d \in D} \sum_{s \in S} (U_{n,d,s} \times \sum_{k \in K} x_{n,s,d,k}) \\ + \sum_{n \in N} \sum_{d \in D} (V_{n,d} \times \sum_{s \in S} \sum_{k \in K} x_{n,s,d,k}) \tag{9}$$

Table 1
Weight values of the soft constraints

Soft Constraint	Weight
S1 Insufficient staffing for optimal coverage	30
S2 Consecutive days off	30
S3 Consecutive assignments	30
S4 Consecutive assignments per shift type	15
S5 Preferences	10
S6 Complete weekends	30
S7 Total assignments	20
S8 Total working weekends	30

- **S6 Complete weekends:** Each nurse with a contract c that has $W_c = 1$, must work both weekend days or none, otherwise a violation will be counted. Mathematically, the number of violations of this constraint V_{S6} is:

$$\sum_{n \in N} \sum_{d \in D'} W_{c_n} \times \left| \sum_{s \in S} \sum_{k \in K} x_{n,s,d,k} - \sum_{s \in S} \sum_{k \in K} x_{n,s,d-1,k} \right| \tag{10}$$

  where $D' = \{7, 14, 21, 28\}$ and c_n is the contract of nurse n.

- **S7 Total assignments:** The minimum and the maximum number of assignments over the entire planning horizon should be respected. Each extra or missing assignment will be counted as a violation. Mathematically, the number of violations of this constraint V_{S7} is:

$$\sum_{n \in N} max(0, \sum_{s \in S} \sum_{d \in D} \sum_{k \in K} x_{n,s,d,k} - Q^{\nu}_{c_n}) + \sum_{n \in N} max(0, Q^{\rho}_{c_n} - \sum_{s \in S} \sum_{d \in D} \sum_{k \in K} x_{n,s,d,k}) \tag{11}$$

  where c_n is the contract of nurse n.

- **S8 Total working weekends:** Maximum number of working weekends (if a nurse is assigned to either Saturday or Sunday or both) over the entire planning horizon should be respected. Each extra assignment will be counted as a violation. Mathematically, the number of violations of this constraint V_{S8} is:

$$\sum_{n \in N} max(0, \sum_{d \in D'} max(\sum_{s \in S} \sum_{k \in K} x_{n,s,d,k}, \sum_{s \in S} \sum_{k \in K} x_{n,s,d-1,k}) - B^{\nu}_{c_n}) \tag{12}$$

  where $D' = \{7, 14, 21, 28\}$ and c_n is the contract of nurse n.

4. Methodology

A selection hyper-heuristic is a search methodology that operates on the level of heuristics (or neighbourhood move operators) instead of operating directly on the optimisation search space [11, 12]. A set of search heuristics (low-level heuristics) that might have different search behaviour is defined to search the problem's optimisation space. A selection hyper-heuristic seeks to find an effective way to select and mix the most promising heuristics at each stage of the search optimisation procedure. Different hyper-heuristics have been developed and successfully addressed challenging real-world problems that utilise either one or a sequence of heuristics.

Overview of SSHH The SSHH method employs a hidden Markov model as a selection method aiming to identify effective sequences of heuristics by learning good transitions among the low-level heuristics. To accomplish this, the proposed hyper-heuristic employs a hidden Markov model in which states correspond to low-level heuristics. For each low-level heuristic in the model, a transition matrix is defined to determine the probability of moving from itself to any other low-level heuristic (including itself). Additionally, each low-level heuristic has an associated sequence construction matrix to determine whether to terminate the sequence at this point [41]. The sequence construction matrix stores scores for each of the low-level heuristics in two columns: *continue* and *end*.

Let $H = \{h_0, h_1, \dots, h_{n-1}\}$ represent the set of low-level heuristics. We define two score matrices $Tran_{n\times n}$ (the transition matrix) and $Seq_{n\times 2}$ (the sequence construction matrix). The probability of moving from low-level heuristic h_k to h_l is given by: $Tran_{(k,l)} / \sum_{\forall j} Tran_{(k,j)}$. Initially, all probabilities of moving from one low-level heuristic to any other are initialised uniformly, i.e., $Tran_{(i,j)} = 1$ for all i, j. In order to construct a sequence of heuristics, the matrix Seq is used to compute the status of that sequence: either the sequence will *end* and the low-level heuristics within it will be applied to the current solution in the order in which they appear, or the sequence will *continue*, and the next low-level heuristic will be selected. The sequence construction probabilities for each low-level heuristic are initialised to 0.5, i.e., $Seq_{(i,continue)} = 1, Seq_{(i,end)} = 1$ for $i = 0, 1, \dots, n-1$.

At first, a random heuristic h_c is chosen as a starting position. The iterative process of the hyper-heuristic then starts and runs until a time limit is exceeded. It begins by selecting the next heuristic h_x, using the roulette wheel selection strategy, and adding it to the sequence. Next, we determine whether or not the sequence will terminate at this point. The probability of continuing the sequence is given by $Seq_{(x,continue)} / (Seq_{(x,continue)} + Seq_{(x,end)})$ and the probability that the sequence is complete is given by $Seq_{(x,end)} / (Seq_{(x,continue)} + Seq_{(x,end)})$.

Let us assume that the sequence is not complete (i.e. status = *continue*). In this case, we move from h_x to the next low-level heuristic h_y and select the status of Seq using a roulette wheel selection strategy. At this point, we assume that the sequence is complete (i.e. status = *end*). The sequence will now be applied to the current solution to generate a new solution. At this point, if the quality of the new solution is better than the quality of the best solution in hand, then the sequence of low-level heuristics that led to it is awarded accordingly by updating the two score matrices.

In this example, the following scores will be increased by 1: $Tran_{(c,x)}$, $Tran_{(x,y)}$, $Seq_{(x,continue)}$ and $Seq_{(y,end)}$. This will increase the chance of selecting the sequences that generate improved solutions.

Recall that the second component of a traditional selection hyper-heuristic framework is *move acceptance*. The move acceptance criterion is used to decide whether to accept or reject the new candidate solution. If the new candidate solution S_{new} is accepted, it will replace the original solution $S_{candidate}$. The construction of the heuristic sequence is now completed and a new one will begin in the next iteration.

For a more in-depth description of the method with examples, the reader can refer to [41].

Initial solution construction algorithm Recall that a solution method is expected to solve a single stage of the problem (w_m, such that $1 \leq m \leq |\mathscr{W}|$) that corresponds to one week. However, as resources are taken up at any stage, forecasting is needed to cope well with subsequent stages ($\{w_{m+1}, \ldots, w_{|\mathscr{W}|}\}$). Recall also that the daily coverage requirements and the nurse preferences are only available for w_m, which would necessitate the forecasting of these requirements and preferences for the remaining weeks. Our model is building on the assumption that for the remaining stages (w_{m+1} to $w_{|\mathscr{W}|}$), no minimum coverage requirements or nurse preferences are needed to be satisfied, but they may satisfy the same optimal coverage requirements as in stage w_m (e.g. if the optimal number of nurses of a particular skill for a particular type of shift on Monday is 3 at stage w_m, then the same optimal number of nurses specified by the skill and the shift type is expected to be satisfied for each Monday in the remaining stages).

An initial solution to the problem is computed by a local search heuristic that aims to satisfy all the hard constraints. More specifically, the local search heuristic uses a neighbourhood operator that iterates through the days. At each step, it assigns a new shift type with a particular skill from the minimum coverage requirements to the roster for a randomly selected nurse while satisfying the single assignment per day constraint, shift type succession constraint, and required skill constraint. If a feasible assignment of nurses to shifts is not achieved after considering all the available nurses, then all the assignments of the day will be destroyed and the local search method will be re-applied. We consider 1000 trials per day and if feasibility is not obtained, then the whole solution will be destroyed and the solution will be constructed from the first day. The construction of the initial solution is described in Algorithm 1.

The initial solutions are almost always feasible, but they usually have many soft constraint violations. Therefore, it is critical for the hyper-heuristic algorithm to be able to improve the initial solutions. The returned solution for the current stage (w_m) will be fed into the next stage as an initial solution. However, because the minimum coverage requirement will be revealed for the next stage (w_{m+1}), we will need to rectify the solution by assigning those unassigned shifts to available nurses while respecting all the hard constraints. Again, we consider 1000 trials per day and if feasibility is not obtained, we destroy the whole solution and start the construction from the first day as described in Algorithm 1.

Algorithm 1: Construction of initial solution

```
Let $S_{initial}$ represent the initial solution to be constructed;
repeat
    $Reset(S_{initial})$;
    foreach $d \in D$ do
        for $trial \leftarrow 1, 2, \ldots, 1000$ do
            foreach $s \in S$ do
                foreach $k \in K$ do
                    for $r \leftarrow 1, 2, \ldots, \mathscr{R}^{\rho}_{s,d,k}$ do
                        if there is a feasible $n \in N$ then
                            $Assign(n, s, d, k, S_{initial})$;
                        else
                            Destroy assignments in $S_{initial}$ of day $d$;
                            Goto next $trial$;
            if $trial$ successful then
                Break from this loop;
        if $all\ trials\ failed$ then
            Break from this loop;
until $IsFeasible(S_{initial})$;
return $S_{initial}$;
```

Low-level heuristics A key component of the SSHH to produce good quality solutions to the multi-stage nurse rostering problem is the set of effective and diverse low-level heuristics. The developed low-level heuristics span across three well-known and different types of heuristics, namely perturbation, exchange, and ruin and recreate heuristics.

Perturbation heuristics usually perform either small or large changes on a solution such as swapping elements, modifying, adding, or removing solution components. *Exchange heuristics* work by exchanging two, usually large, parts of the solution. Such operations usually induce large steps in the search space and promote global search behaviour. Note that *Exchange heuristics* can be characterised as *Perturbation heuristics* since their search pattern can be similar. Here, we make this distinction to indicate the specific pattern of moving consecutive blocks from one solution to another, followed by a repair operation if required. *Ruin and recreate*, or *destruction-construction* heuristics perform two main operations on a solution: first they destroy a part of the solution and then recreate it. Ruin and recreate heuristics favour exploration of the search space and usually incorporate problem domain knowledge-based heuristics to recreate the destroyed solutions.

To effectively search the space of the multi-stage nurse rostering problem formulation, we develop a set of nine fairly simple low-level domain-specific heuristics:

- Four perturbation heuristics: **LLH0-2**, and **LLH8**,
- Two ruin and recreate heuristics: **LLH3-4**,
- Three exchange heuristics: **LLH5-7**.

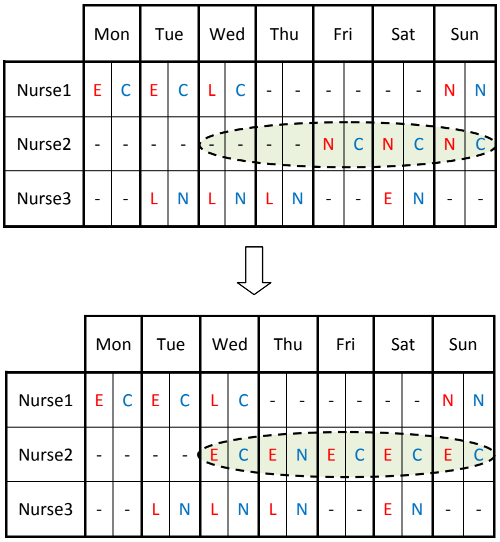

Figure 1: An example to illustrate how the first option of LLH0 works. Shift types (in red): E Early, L Late, N Night; Skills (in blue): N Nurse, C Caretaker. In this example, a block of 5 consecutive days for Nurse2 is selected. The cells have been assigned to the shift type E. Two of the cells (Wed and Thu) were unassigned to any shift and so the heuristic assigns random skills to them

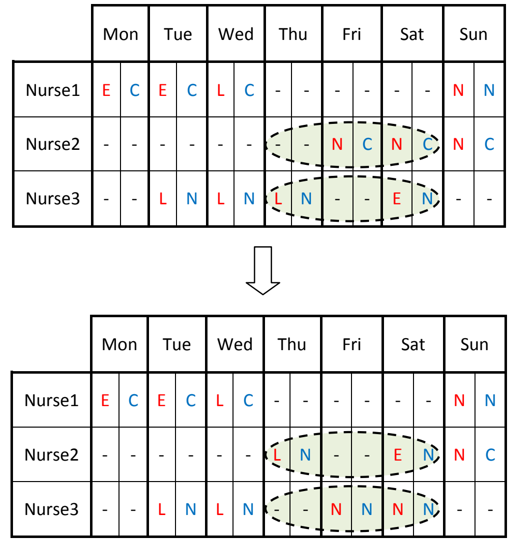

Figure 2: An example to illustrate how the first option of LLH1 works. Shift types (in red): E Early, L Late, N Night; Skills (in blue): N Nurse, C Caretaker. In this example, shifts of three consecutive days for Nurse2 and Nurse3 are vertically swapped. Skill C is not feasible for Nurse3, hence the skills on Fri and Sat are changed randomly to feasible skills using the rectify method

The low-level heuristics modify a solution by taking into account shifts, shift types, and skills within specific blocks of time. A brief description of the low-level heuristics follows.

- **LLH0:** Selects a block of adjacent P days for a randomly selected nurse and then assigns (or reassigns) shifts to the selected block. The parameter P (length of block) can take any value between 1 and the planning horizon at random. Recall that a shift is a combination of a shift type and a skill. One of the following five options is randomly selected and applied:
  - **Option 1:** Select a single type of shift randomly and assign it to the selected block. The skills remain without any change, but if one of the cells was not assigned to any shift then we randomly select a new feasible skill and assign it to that cell. An example of this option is given in Figure 1.
  - **Option 2:** Select a single shift type randomly and assign it to the selected block. The feasible skills will be randomly selected for each shift.
  - **Option 3:** Select a feasible single skill randomly and assign it to the selected block. The shift types remain without any change, but if one of the cells was not assigned to any shift then we randomly select a new shift type and assign it to that cell.
  - **Option 4:** Select a feasible single skill randomly and assign it to the selected block. The shift types will be randomly selected for each shift.
  - **Option 5:** Delete the shifts of the selected block.
- **LLH1:** Selects two nurses randomly and a block of adjacent P days and swaps vertically. The parameter P (length of block) can take any value between 1 and the planning horizon at random. Because a given skill might not be feasible for a given nurse, a rectify method is applied to change the unfeasible assignments of skills to another randomly selected feasible skills. One of the following three options is randomly selected and applied:
  - **Option 1:** Swap both shift types and skills. Figure 2 provides an example of this option.
  - **Option 2:** Swap shift types only.
  - **Option 3:** Swap skills only.
- **LLH2:** Follows the same procedure as **LLH1**, apart from the fact that the heuristic selects two blocks of adjacent P days of a randomly selected nurse and swaps horizontally. One of the following three options is randomly selected and applied:
  - **Option 1:** Swap both shift types and skills. Figure 3 provides an example of this option.
  - **Option 2:** Swap shift types only.
  - **Option 3:** Swap skills only.
- **LLH3:** This LLH defines a ruin and recreate operator for one nurse. It works by un-assigning several shifts in one randomly selected nurse and then rebuilding them at random by adding, replacing, or deleting shift types and/or skills. The number of modified shifts P can take

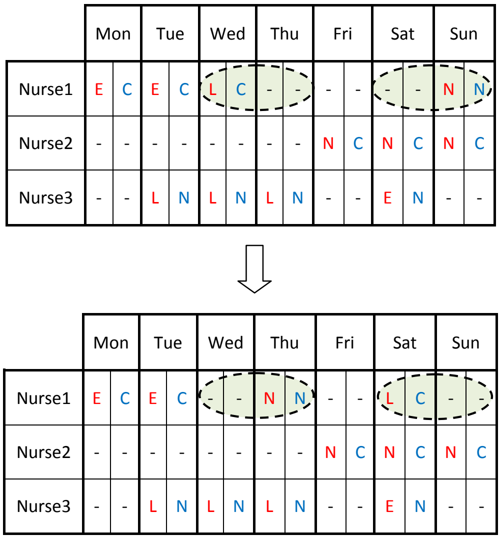

Figure 3: An example to illustrate how the first option of LLH2 works. Shift types (in red): E Early, L Late, N Night; Skills (in blue): N Nurse, C Caretaker. In this example, two shifts each of two consecutive days for Nurse1 are horizontally swapped

any value between 1 and the planning horizon at random. Algorithm 2 demonstrates the exact procedure that is followed for this operation.

Algorithm 2: Ruin and recreate for one nurse

Let $LLH0(n, O, P_{LLH0})$ represent applying option O of LLH0 to nurse n and P_{LLH0} is the parameter of LLH0;
Let $Rand(a, b)$ return uniform random number in $[a, b]$;
Select random $n \in N$;
for $i \leftarrow 1, 2, \dots, P$ **do**
    $op \leftarrow Rand(1, 3)$;
    if $op = 1$ **then**
        $LLH0(n, 1, Rand(1, 7))$;
    else if $op = 2$ **then**
        $LLH0(n, 3, 1)$;
    else if $op = 3$ **then**
        $LLH0(n, 5, Rand(1, 7))$;

- **LLH4:** This LLH defines a ruin and recreate operator for many nurses. It is clearly described in Algorithm 3.
- **LLH5:** This LLH defines an exchange operator for two blocks of adjacent P days for two randomly selected nurses. For each cell, the heuristic either swaps the shift types, skills, or both. The parameter P of this heuristic can take any value between 1 and the planning horizon at random. Figure 4 illustrates a characteristic example that clearly demonstrates the operation.
- **LLH6:** This heuristic is the same as **LLH5**, apart from the fact that the operator selects two *vertical* blocks of adjacent P days for two randomly selected nurses; and for each cell, the exchange probability is 50%. The parameter P of this heuristic can take any value between 1 and the planning horizon at random.

Algorithm 3: Ruin and recreate for several nurses

Let $LLH0(n, O, P_{LLH0})$ represent applying option O of LLH0 to nurse n and P_{LLH0} is the parameter of LLH0;
Let $Rand(a, b)$ return uniform random number in $[a, b]$;
for $i \leftarrow 1, 2, \dots, P$ **do**
    Select random $n \in N$;
    $op \leftarrow Rand(1, 3)$;
    if $op = 1$ **then**
        $LLH0(n, 1, Rand(1, 7))$;
    else if $op = 2$ **then**
        $LLH0(n, 3, 1)$;
    else if $op = 3$ **then**
        $LLH0(n, 5, Rand(1, 7))$;

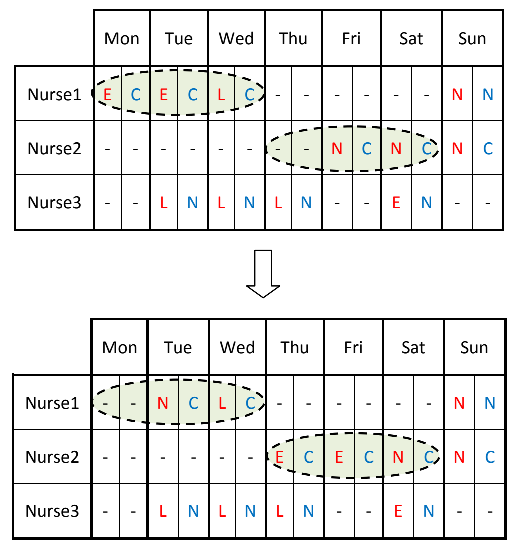

Figure 4: An example to illustrate how LLH5 works. Shift types (in red): E Early, L Late, N Night; Skills (in blue): N Nurse, C Caretaker. In this example, an exchange operator is applied to shifts of three consecutive days for Nurse1 and Nurse2. The shift types and skills of the first cell are swapped. Only the shift types are swapped for the second cell. Only the skills are swapped for the third cell

- **LLH7:** This heuristic is the same as **LLH6**, apart from the fact that the parameter P of this heuristic (length of block) is a multiple of 7 (week days). For example, if we are currently solving the problem at stage 5 of 8 stages, then the parameter can take one of the following four values: 7, 14, 21 and 28 at random.
- **LLH8:** This heuristic essentially is a swap operator that works similar to **LLH1**, apart from the fact that the parameter P of this heuristic is a multiple of 7. Once again, one of the following three options is randomly selected and applied:
  - **Option 1:** Swap both shift types and skills verti-

cally.

- **Option 2:** Swap shift types only vertically.
- **Option 3:** Swap skills only vertically.

Parameter settings The sequence selection method is generally parameter-free apart from the threshold for the move acceptance method. Specifically, a generated solution is accepted by the move acceptance method if either its quality is better than or equal to the quality of the candidate solution, or its quality is better than or equal to the quality of the best solution in hand plus a threshold value. The quality of the solution is calculated using the evaluation function provided in Equation 1. The value of the threshold value is set to 30. This parameter value is chosen after performing a set of fine-tuning experiments. Additionally, to avoid stagnation of the search process, the solution will be partially restarted (by applying three randomly selected low-level heuristics) if there is no improvement to the best-recorded solution in hand for a large number of iterations ($|N| \times 250000$).

5. Computational Results

In this section, we summarise the computational results of SSHH on a set of instances provided by the organisers of the INRC-II competition. We also compare and contrast the computational performances of the SSHH with the competing algorithms.

The organisers of the benchmark provided a testbed composed of 20 datasets, each with 3 initial history files and 10-week data files. The same week data file can also be used multiple times in the same instance. Table 2 summarises the main characteristics of these datasets.

The INRC-II competition was run in two phases: a qualification round and a final phase that revealed the winner of the competition. For the qualification phase, 14 datasets, each composed of two instances, were released. The competitors submitted their executable files and the solutions of the 28 instances. To facilitate fair comparisons across different computational environments, a benchmarking software tool provided by the organisers, available at the competition website, was used by each competitor to estimate the computational time-budget per stage and per dataset that was available to spend based on their computational environment (machine to be used for the execution of their algorithm).

Qualification phase of the INRC-II In the qualification phase, the organisers compared all the approaches on the same computational machine and using the same time limit. For each instance solved, a rank of the competing approaches was defined according to the objective function value. The average rank across all the instances solved was used to qualify the teams for the second phase of the INRC-II challenge. Out of 15 submitted solvers, only 7 teams were admitted to the final phase. Figure 5 displays the average rank for all the competing approaches in the qualification phase. SSHH ranked in the third place.

Table 2
Characteristics of the 20 datasets (14 were used for the qualification phase and 6 for the final phase): $|N|$ is the number of nurses, $|K|$ is the number of available skills, $|S|$ is the number of shift types, $|\mathscr{W}|$ is the number of weeks, $|C|$ is the number of available contracts, $|\mathcal{S}|$ is the number of forbidden shift types successions

Dataset	$\|N\|$	$\|K\|$	$\|S\|$	$\|\mathscr{W}\|$	$\|C\|$	$\|\mathcal{S}\|$
D1	30	4	4	4	3	6
D2	30	4	4	8	3	6
D3	35	4	4	4	3	5
D4	35	4	4	8	3	5
D5	40	4	4	4	3	5
D6	40	4	4	8	3	5
D7	50	4	4	4	3	5
D8	50	4	4	8	3	5
D9	60	4	4	4	4	6
D10	60	4	4	8	3	5
D11	70	4	4	4	3	5
D12	70	4	4	8	3	5
D13	80	4	4	4	4	5
D14	80	4	4	8	4	5
D15	100	4	4	4	4	5
D16	100	4	4	8	4	5
D17	110	4	4	4	4	5
D18	110	4	4	8	4	5
D19	120	4	4	4	3	5
D20	120	4	4	8	3	5

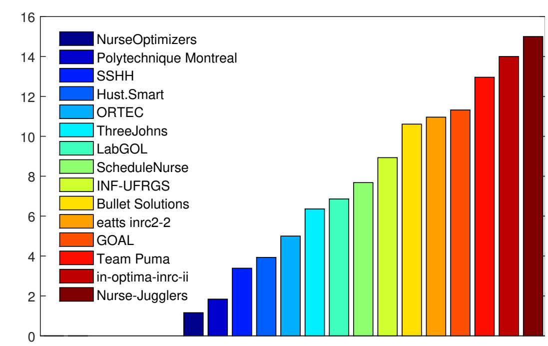

Figure 5: Ranking results of the qualification phase of INRC-II

Final phase of the INRC-II In the final phase, the finalists' solvers were compared on a set of 60 hidden instances taken from six hidden datasets (D3, D4, D11, D12, D17, and D18 as shown in Table 2). To facilitate fair comparisons, 10 independent execution runs with different random seeds for each instance were conducted, resulting in a total of 600 independent execution runs per competitor. The final objective value obtained from each execution run in each instance were ranked and then averaged to determine the winner of the INRC-II.

Table 3 summarises the performance results for all competitors in the final phase of the challenge on all problem instances of the six hidden datasets. More specifically, for each dataset (D $\in \{$D1, D4, D11, D12, D17, D18$\}$), each problem instance (I $\in \{$I1, I2, …, I10$\}$), and each competing algorithm (SSHH and the remaining six teams, Team2–Team7), the average (μ_f) of the best objective value are reported, which were obtained over 10 independent execution runs. For each instance, the number of successful execution runs (%S) is also reported, i.e., the number of times an algorithm was able to compute a feasible solution within the predefined time-budget. In addition, for each problem instance, we conduct a pairwise Wilcoxon-signed rank test [42] between the proposed algorithm and each considered competitor (for $\alpha = 0.05$), to assess whether there exist significant differences between their performance values over the 10 independent runs. The status of the pairwise test is reported next to the μ_f performance value of each team, where marks +/=/– indicate a statistical better, equal and worse performance of the proposed SSHH against the corresponding team. At the bottom of the table, we report the number of times the SSHH algorithm exhibits significantly better (+), equal (=), or worse (–) performance gains against each competitor, over all problem instances. Finally, the number of infeasible solutions over all runs is reported as well as the final average ranking of each algorithm as it was calculated by the organisers of the competition. Note that the final average ranking of each algorithm is the average ranking of their rankings for each problem instance in each independent run.

Team7, the winner of the challenge, is a problem-specific hybrid algorithm between an exact and a heuristic approach. Although it is able to find feasible solutions in several cases, there are some datasets where it struggles to perform well such as in the D4 set, for instances I1, I2, I3, and I5 and in two problem instances of the D17 set (i.e., I3, and I7). In the D4 dataset, in 3 out of 10 instances Team7's algorithm is able to find solutions in less than 20% of the execution runs (for instances I1, I3, and I5), while it cannot find any feasible solution for the I1 instance. Team3 follows the results of the winner with a very close performance across the majority of the problem instances and datasets. However, it is not able to compute a feasible solution for instance I6 of the D4 dataset.

Among the competitors, SSHH is the first reliable approach that exhibits very good performance gains in terms of average objective values without producing any infeasible solution on the considered problem instances. SSHH ranked third overall with a score of 2.84 and is the approach with the lowest rank among the competitors that produces only feasible solutions across all problem instances and datasets.

Regarding the remaining competing algorithms, Team4, Team5, and Team2 obtained average rankings of 3.75, 5.35, and 6.13 while they did not produce any infeasible solutions. In general, their performance gains were significantly worse than the top three. On the other hand, Team6 had an average ranking of 6.32 and produced six infeasible solutions all of them in the D11 dataset I1, I3, I4, and I8–I10, resulting in the worst performance among the finalists of the competition.

To have a general overview of how the algorithms perform per dataset, we normalised the objective function values per problem instance. As such, Figure 6 illustrates boxplot graphs to compare the distributions of the normalised objective values of each considered algorithm per dataset, where the diamond mark denotes the mean value of the underline performance values' sample. The boxplot graphs

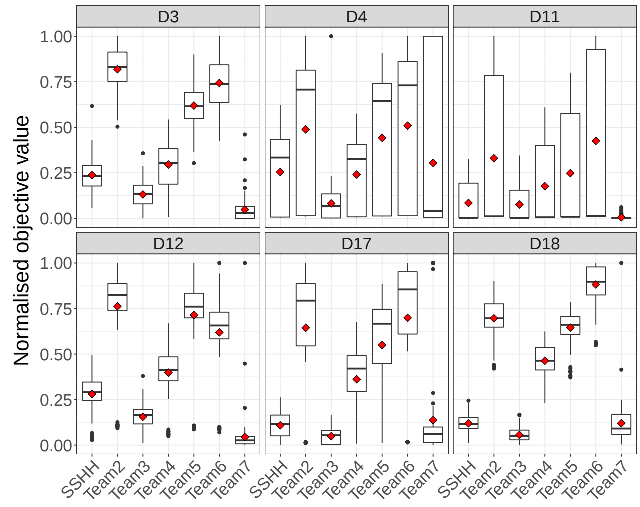

Figure 6: Normalised objective value per problem instance, summarised over datasets

clearly depict the aforementioned described performance of all algorithms. It is worth noting that the algorithm of Team7 in almost all datasets (apart from D3 and D11) exhibits outliers (i.e., generation of infeasible solutions), while its poor performance in the D4 case is captured by the size of the boxplot, which essentially lies in the whole range of the normalised objective values.

It is worth noting that the performance of the SSHH algorithm closely follows the first two ranked approaches, while its performance gains are more evident as the problems become more challenging, i.e., as the number of the nurses increases up to 110 nurses (as in datasets D17 and D18).

An analysis of SSHH SSHH incorporates a set of low-level heuristics and creates sequences of low-level heuristics using transition and sequence construction matrices. As such, an analysis of the most frequently used sequence of low-level heuristics might reveal insights about the effectiveness of such a combination on specific problem classes (defined here by the different datasets and their corresponding problem instances).

Figures 7 and 8 illustrate the average probabilities of the transition and sequence construction matrices over 10 independent trials of each low-level heuristic while solving the first stage of the nurse rostering problem for two selected instances from the D3 and D17 datasets. A simulator was then developed to mimic the way in which the algorithm works, using the final HMM probability matrices as input for the selected two problem instances. Table 4 shows the sets of likely sequences that have been generated using the simulator. Although, it is argued that the proposed SSHH algorithm is a general approach that can be applied to any problem instance and therefore the best-discovered sequences of heuristics for a given instance it may not be the best for another problem instance, yet the method seems to favour the same set of sequences while solving the two instances from the different datasets. LLH0 (whether alone or when combined with any other heuristic) is the most successful heuris-

Table 3
The performance comparison of the SSHH approach against the other competitors in the final phase of the challenge. For each competitor, problem instance, and datasets, the table demonstrates the average (μ_f) of the objective value over 10 independent runs as well as the success rate, i.e., the percentage of instances for which a feasible solution has been computed (%S). Team2 is LabGOL, Team3 is Polytechnique Montreal, Team4 is Hust.Smart, Team5 is ORTEC, Team6 is ThreeJohns, Team7 is NurseOptimizers

		SSHH		Team2		Team3		Team4		Team5		Team6		Team7	
		μ_f	%S	μ_f	%S	μ_f	%S	μ_f	%S	μ_f	%S	μ_f	%S	μ_f	%S
D1	I1	1731.5	100	2105.5^{+}	100	1695.0$^{=}$	100	1756.5$^{=}$	100	2016.0^{+}	100	2059.5^{+}	100	1630.0^{-}	100
D1	I2	2005	100	2342.0^{+}	100	1873.0^{-}	100	2021.5$^{=}$	100	2166.5^{+}	100	2295.0^{+}	100	1831.5^{-}	100
D1	I3	1928.5	100	2219.0^{+}	100	1867.0^{-}	100	1834.5^{-}	100	2176.0^{+}	100	2158.5^{+}	100	1755.0^{-}	100
D1	I4	1666	100	2069.0^{+}	100	1617.0^{-}	100	1723.5^{+}	100	1963.5^{+}	100	1935.0^{+}	100	1586.0^{-}	100
D1	I5	1695.5	100	2065.0^{+}	100	1569.5^{-}	100	1737.0$^{=}$	100	1939.5^{+}	100	2072.5^{+}	100	1545.0^{-}	100
D1	I6	1631	100	1992.5^{+}	100	1544.5^{-}	100	1644.5$^{=}$	100	1818.0^{+}	100	1954.5^{+}	100	1510.0^{-}	100
D1	I7	1407.5	100	1845.0^{+}	100	1352.5$^{=}$	100	1370.5$^{=}$	100	1644.0^{+}	100	1680.0^{+}	100	1367.5$^{=}$	100
D1	I8	1845.5	100	2206.5^{+}	100	1780.0^{-}	100	1947.5^{+}	100	2113.0^{+}	100	2236.0^{+}	100	1708.0^{-}	100
D1	I9	1804	100	2304.5^{+}	100	1774.5$^{=}$	100	1970.5^{+}	100	2101.5^{+}	100	2309.5^{+}	100	1695.0^{-}	100
D1	I10	1804	100	2225.5^{+}	100	1739.0^{-}	100	1927.5^{+}	100	2083.5^{+}	100	2179.0^{+}	100	1652.0^{-}	100
D4	I1	3619.5	100	4247.0^{+}	100	3142.5^{-}	100	3628.0$^{=}$	100	4171.0^{+}	100	4303.0^{+}	100	99999.0^{+}	0
D4	I2	3485.5	100	4085.5^{+}	100	2947.5^{-}	100	3653.5$^{=}$	100	4045.5^{+}	100	4151.0^{+}	100	12669.0$^{=}$	90
D4	I3	3432.5	100	4093.5^{+}	100	2987.5^{-}	100	3378.5$^{=}$	100	4019.0^{+}	100	4078.0^{+}	100	80576.0^{+}	20
D4	I4	3472.5	100	4112.0^{+}	100	2928.0^{-}	100	3325.0^{-}	100	3969.0^{+}	100	4105.5^{+}	100	2849.5^{-}	100
D4	I5	3419	100	4129.0^{+}	100	3072.0^{-}	100	3548.5$^{=}$	100	4011.0^{+}	100	4296.5^{+}	100	2842.0^{-}	100
D4	I6	3499	100	4205.0^{+}	100	12718.0$^{=}$	90	3672.0^{+}	100	4098.0^{+}	100	4285.0^{+}	100	90304.0^{+}	10
D4	I7	3699.5	100	4317.5^{+}	100	3179.0^{-}	100	3632.5$^{=}$	100	4143.5^{+}	100	4423.5^{+}	100	3028.5^{-}	100
D4	I8	3582	100	4105.5^{+}	100	3024.0^{-}	100	3603.0$^{=}$	100	4096.5^{+}	100	4132.0^{+}	100	2863.0^{-}	100
D4	I9	3659	100	4350.5^{+}	100	3205.5^{-}	100	3533.5^{-}	100	4228.0^{+}	100	4287.5^{+}	100	3083.5^{-}	100
D4	I10	3508	100	4155.0^{+}	100	2962.0^{-}	100	3488.0$^{=}$	100	3967.5^{+}	100	4278.0^{+}	100	2928.0^{-}	100
D11	I1	2905	100	3694.5^{+}	100	2892.0$^{=}$	100	3151.0^{+}	100	3531.5^{+}	100	13475.0^{+}	90	2723.0^{-}	100
D11	I2	2616.5	100	3306.5^{+}	100	2605.5$^{=}$	100	2889.0^{+}	100	3088.0^{+}	100	3417.5^{+}	100	2446.0^{-}	100
D11	I3	2609.5	100	3394.0^{+}	100	2671.5$^{=}$	100	2948.0^{+}	100	3242.0^{+}	100	13064.0^{+}	90	2557.5^{-}	100
D11	I4	2688.5	100	3559.5^{+}	100	2662.5$^{=}$	100	3016.0^{+}	100	3336.0^{+}	100	13360.0^{+}	90	2477.0^{-}	100
D11	I5	2590	100	3198.5^{+}	100	2536.5$^{=}$	100	2864.0^{+}	100	3055.0^{+}	100	3337.5^{+}	100	2323.0^{-}	100
D11	I6	2875.5	100	3553.0^{+}	100	2918.0$^{=}$	100	3134.5^{+}	100	3325.5^{+}	100	3596.0^{+}	100	2728.0^{-}	100
D11	I7	2824	100	3535.5^{+}	100	2740.5^{-}	100	3012.0^{+}	100	3217.5^{+}	100	3649.5^{+}	100	2533.0^{-}	100
D11	I8	2836	100	3410.5^{+}	100	2764.0^{-}	100	3141.5^{+}	100	3329.5^{+}	100	13307.0^{+}	90	2635.0^{-}	100
D11	I9	2762	100	3398.5^{+}	100	2729.0$^{=}$	100	3005.5^{+}	100	3262.5^{+}	100	13124.0^{+}	90	2544.5^{-}	100
D11	I10	2835.5	100	3542.0^{+}	100	2775.5$^{=}$	100	3046.0^{+}	100	3268.0^{+}	100	13187.0^{+}	90	2652.0^{-}	100
D12	I1	5954.5	100	7106.0^{+}	100	5640.5^{-}	100	6222.0^{+}	100	7049.5^{+}	100	6820.0^{+}	100	5164.0^{-}	100
D12	I2	6023.5	100	7258.0^{+}	100	5750.5^{-}	100	6602.0^{+}	100	7117.5^{+}	100	7039.0^{+}	100	5478.5^{-}	100
D12	I3	6157	100	6976.0^{+}	100	5769.5^{-}	100	6236.5$^{=}$	100	7182.0^{+}	100	6756.5^{+}	100	5549.0^{-}	100
D12	I4	5594	100	6977.5^{+}	100	5516.0$^{=}$	100	6018.5^{+}	100	6662.0^{+}	100	6615.5^{+}	100	5167.0^{-}	100
D12	I5	5997.5	100	7237.5^{+}	100	5705.5^{-}	100	6259.0^{+}	100	7033.5^{+}	100	6716.0^{+}	100	5581.5^{-}	100
D12	I6	6048	100	7290.0^{+}	100	5724.0^{-}	100	6315.0^{+}	100	6985.0^{+}	100	6790.5^{+}	100	5359.5^{-}	100
D12	I7	6203	100	7244.0^{+}	100	5859.5^{-}	100	6317.5^{+}	100	7155.5^{+}	100	6862.5^{+}	100	5531.5^{-}	100
D12	I8	5858	100	6885.0^{+}	100	5428.5^{-}	100	6255.0^{+}	100	6864.5^{+}	100	6706.0^{+}	100	5240.0^{-}	100
D12	I9	6054.5	100	7207.5^{+}	100	5672.5^{-}	100	6492.5^{+}	100	7054.5^{+}	100	6890.5^{+}	100	7138.5$^{=}$	100
D12	I10	5886.5	100	7126.0^{+}	100	5688.0^{-}	100	6044.5^{+}	100	6957.0^{+}	100	6711.5^{+}	100	5374.0^{-}	100
D17	I1	2823.5	100	3861.0^{+}	100	2765.5^{-}	100	3539.0^{+}	100	3817.0^{+}	100	4051.5^{+}	100	2725.0^{-}	100
D17	I2	3125.5	100	4229.0^{+}	100	3020.0^{-}	100	3663.0^{+}	100	4074.0^{+}	100	4529.5^{+}	100	3065.0^{-}	100
D17	I3	3044.5	100	4149.5^{+}	100	3027.5$^{=}$	100	3769.0^{+}	100	4030.0^{+}	100	4429.0^{+}	100	12690.0$^{=}$	90
D17	I4	3058.5	100	4215.5^{+}	100	2923.5^{-}	100	3569.5^{+}	100	3976.0^{+}	100	4342.5^{+}	100	3210.0$^{=}$	100
D17	I5	3532.5	100	4628.0^{+}	100	3438.5$^{=}$	100	4092.0^{+}	100	4427.0^{+}	100	4700.0^{+}	100	3425.0^{-}	100
D17	I6	3043	100	4152.0^{+}	100	2957.0^{-}	100	3661.0^{+}	100	3958.5^{+}	100	4210.5^{+}	100	2855.5^{-}	100
D17	I7	3290	100	4381.5^{+}	100	3113.0^{-}	100	3903.5^{+}	100	4198.5^{+}	100	4559.5^{+}	100	51625.0$^{=}$	50
D17	I8	3189	100	4394.0^{+}	100	3032.5^{-}	100	3637.5^{+}	100	4018.5^{+}	100	4369.5^{+}	100	3095.0^{-}	100
D17	I9	3654.5	100	4744.5^{+}	100	3497.0^{-}	100	4025.0^{+}	100	4542.5^{+}	100	4708.0^{+}	100	3502.5^{-}	100
D17	I10	3419.5	100	4382.5^{+}	100	3233.5^{-}	100	3769.0^{+}	100	4214.0^{+}	100	4600.5^{+}	100	3540.0$^{=}$	100
D18	I1	5485	100	7364.0^{+}	100	5326.0^{-}	100	6596.0^{+}	100	7359.5^{+}	100	7939.5^{+}	100	5243.0^{-}	100
D18	I2	5121.5	100	7010.0^{+}	100	4930.5^{-}	100	6172.5^{+}	100	6679.0^{+}	100	7475.5^{+}	100	4982.5^{-}	100
D18	I3	5237.5	100	6869.0^{+}	100	4874.0^{-}	100	6227.0^{+}	100	6795.0^{+}	100	7640.5^{+}	100	4939.0^{-}	100
D18	I4	5224.5	100	7144.0^{+}	100	5071.0^{-}	100	6251.5^{+}	100	6996.5^{+}	100	7837.5^{+}	100	5234.0$^{=}$	100
D18	I5	4800	100	6824.0^{+}	100	4624.5^{-}	100	6146.5^{+}	100	6627.0^{+}	100	7229.0^{+}	100	5021.5^{+}	100
D18	I6	5226	100	7048.5^{+}	100	5040.5^{-}	100	6469.0^{+}	100	6996.5^{+}	100	7728.0^{+}	100	5510.5^{+}	100
D18	I7	5329	100	7157.5^{+}	100	5259.5$^{=}$	100	6514.0^{+}	100	6923.5^{+}	100	7721.5^{+}	100	5259.5$^{=}$	100
D18	I8	4869.5	100	6801.0^{+}	100	4576.0^{-}	100	6115.5^{+}	100	6581.5^{+}	100	7235.0^{+}	100	4842.5$^{=}$	100
D18	I9	5358	100	7311.0^{+}	100	5039.5^{-}	100	6222.5^{+}	100	7163.5^{+}	100	8222.0^{+}	100	5274.0$^{=}$	100
D18	I10	4752	100	6708.0^{+}	100	4519.5^{-}	100	5809.0^{+}	100	6454.0^{+}	100	7384.5^{+}	100	5292.0$^{=}$	100
Stat. Sig. # (+/=/−)				(60/0/0)		(0/16/44)		(45/12/3)		(60/0/0)		(60/0/0)		(5/12/43)	
# Infeasible Solutions			**0**		0		1		0		0		6		34
Average Ranking			**2.84**		6.13		1.86		3.75		5.35		6.32		1.76

Table 4
The top fifteen constructed sequences of low-level heuristics while solving the first stage of D3-I1 and D17-I1 instances using the developed simulator

D3-I1		D17-I1	
Sequence	Count	Sequence	Count
LLH0	507215	LLH0	430487
LLH3	119856	LLH4	183879
LLH1	70442	LLH3	169359
LLH8	57824	LLH8	56044
LLH4	56641	LLH1	38556
LLH6	30429	LLH2	17576
LLH7	25220	LLH6	11567
LLH2	21799	LLH5	10853
LLH5	15428	LLH7	8681
LLH0-LLH0	4925	LLH0-LLH0	2853
LLH7-LLH0	1851	LLH7-LLH0	2836
LLH2-LLH0	1729	LLH2-LLH0	2418
LLH8-LLH0	1516	LLH8-LLH0	1819
LLH2-LLH3	1475	LLH3-LLH3	1385
LLH8-LLH3	1462	LLH7-LLH4	1221

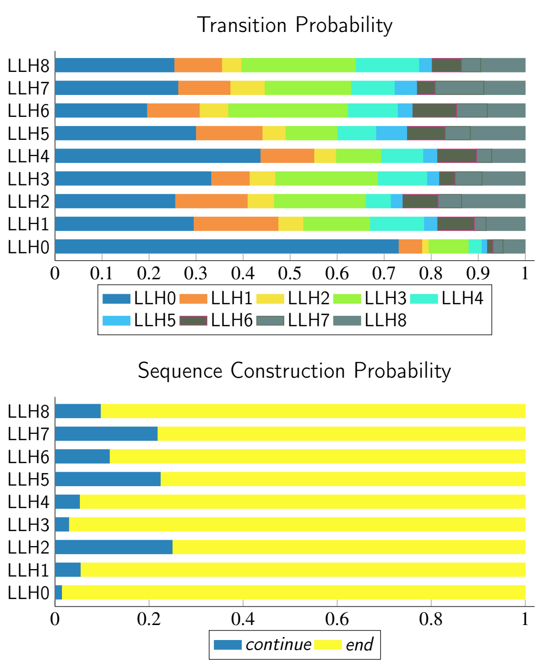

Figure 7: Average probabilities of the transition and sequence construction matrices from 10 trials while solving the first stage of D3-I1

tic to contribute to improving the best-recorded solution in hand while solving both considered instances. The proposed SSHH does not take the size of sequences as a parameter, but rather it learns the optimum size during the optimisation (in an online manner). Table 4 shows that single heuristics are dominantly used, although it identified sequences of size 2 as 'potentially' useful to the search.

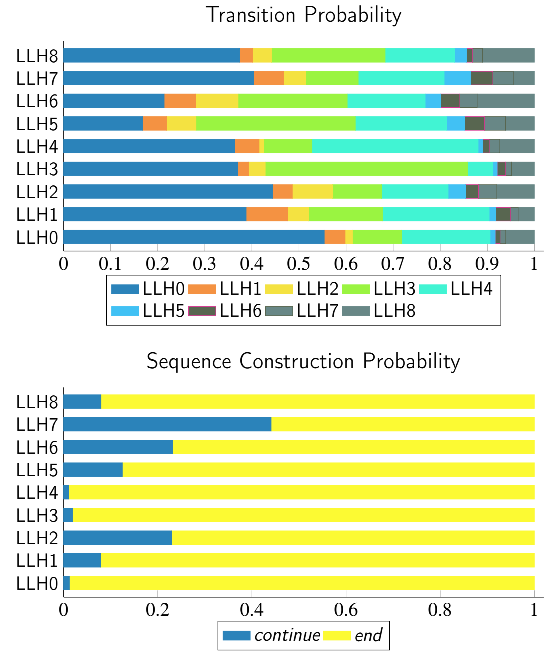

Figure 8: Average probabilities of the transition and sequence construction matrices from 10 trials while solving the first stage of D17-I1

6. Conclusion

The level of generality that a hyper-heuristic can achieve has always been of interest to hyper-heuristic researchers. To address the multi-stage nurse rostering problem posed by INRC-II, we develop and exploit a sequence-based selection hyper-heuristic (SSHH) that essentially adopts the main algorithmic structure of the original SSHH algorithm [41, 43] and crucially modifies two key components to address the multi-stage nurse rostering formulation, namely the construction of the initial solution and the low-level heuristics. Due to the characteristics of the problem formulation, a dedicated algorithm for building feasible initial solutions and a series of low-level heuristics with different characteristics are developed. The method aims to control the application of sequences of heuristics as opposed to a simple selection of a single heuristic. The proposed method is the best-ranked method to achieve feasibility across all problems and also the first ranked among general-purpose hyper/metaheuristic approaches.

References

[1] A. Ernst, H. Jiang, M. Krishnamoorthy, D. Sier, Staff scheduling and rostering: A review of applications, methods and models, European Journal of Operational Research 153 (2004) 3–27.

[2] J. V. d. Bergh, J. Beliën, P. D. Bruecker, E. Demeulemeester, L. D. Boeck, Personnel scheduling: A literature review, European Journal of Operational Research 226 (3) (2013) 367–385.

[3] S. Haspeslagh, P. De Causmaecker, A. Schaerf, M. Stolevik, The first international nurse rostering competition 2010, Annals of Operations Research 218 (1) (2014) 221âĂŞ236.

[4] E. K. Burke, T. Curtois, New approaches to nurse rostering benchmark instances, European Journal of Operational Research 237 (1) (2014) 71–81.

[5] F. Della Croce, F. Salassa, A variable neighborhood search based matheuristic for nurse rostering problems, Annals of Operations Research 218 (1) (2014) 185–199.

[6] I. P. Solos, I. X. Tassopoulos, G. N. Beligiannis, A generic two-phase stochastic variable neighborhood approach for effectively solving the nurse rostering problem, Algorithms 6 (2) (2013) 278–308.

[7] S. Ceschia, N. T. T. Dang, P. De Causmaecker, S. Haspeslagh, A. Schaerf, Second International Nurse Rostering Competition (INRC-II) – Problem Description and Rules –, arXiv:1501.04177 [cs] ArXiv: 1501.04177.

[8] S. Ceschia, N. Dang, P. De Causmaecker, S. Haspeslagh, A. Schaerf, The Second International Nurse Rostering Competition, Annals of Operations Research 274 (1) (2019) 171–186, ISSN 1572-9338.

[9] C. A. Glass, The nurse rostering problem: A critical appraisal of the problem structure, European Journal of Operational Research 202 (2) (2010) 379–389.

[10] P. I. Cowling, G. Kendall, E. Soubeiga, A Hyperheuristic Approach to Scheduling a Sales Summit, in: Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling III, PATAT'00, Springer-Verlag, London, UK, UK, 176–190, 2001.

[11] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: a survey of the state of the art, Journal of the Operational Research Society 64 (12) (2013) 1695–1724.

[12] J. H. Drake, A. Kheiri, E. Özcan, E. K. Burke, Recent advances in selection hyper-heuristics, European Journal of Operational Research 285 (2) (2020) 405–428.

[13] A. Kheiri, E. Keedwell, A hidden Markov model approach to the problem of heuristic selection in hyper-heuristics with a case study in high school timetabling problems, Evolutionary Computation 25 (3) (2017) 473–501.

[14] A. Kheiri, A. G. Dragomir, D. Mueller, J. Gromicho, C. Jagtenberg, J. J. van Hoorn, Tackling a VRP challenge to redistribute scarce equipment within time windows using metaheuristic algorithms, EURO Journal on Transportation and Logistics 8 (5) (2019) 561–595.

[15] A. Kheiri, L. Ahmed, B. Boyacı, J. Gromicho, C. Mumford, E. Özcan, A. S. Dirikoç, Exact and hyper-heuristic solutions for the distribution-installation problem from the VeRoLog 2019 challenge, Networks 76 (2) (2020) 294–319.

[16] A. Kheiri, Heuristic Sequence Selection for Inventory Routing Problem, Transportation Science 54 (2) (2020) 302–312.

[17] D. Wilson, S. Rodrigues, C. Segura, I. Loshchilov, F. Hutter, G. L. Buenfil, A. Kheiri, E. Keedwell, M. Ocampo-Pineda, E. Özcan, S. I. V. Pena, B. Goldman, S. B. Rionda, A. Hernandez-Aguirre, K. Veeramachaneni, S. Cussat-Blanc, Evolutionary computation for wind farm layout optimization, Renewable Energy 126 (2018) 681–691.

[18] L. Ahmed, C. Mumford, A. Kheiri, Solving urban transit route design problem using selection hyper-heuristics, European Journal of Operational Research 274 (2) (2019) 545–559.

[19] E. K. Burke, P. De Causmaecker, G. Vanden Berghe, H. Van Landeghem, The state of the art of nurse rostering, Journal of Scheduling 7 (6) (2004) 441–499.

[20] P. De Causmaecker, G. Vanden Berghe, A categorisation of nurse rostering problems, Journal of Scheduling 14 (2011) 3âĂŞ16.

[21] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey, in: P. L. Hammer, E. L. Johnson, B. H. Korte (Eds.), Annals of Discrete Mathematics, vol. 5 of *Discrete Optimization II*, Elsevier, 287–326, 1979.

[22] H. Santos, T. Toffolo, R. Gomes, S. Ribas, Integer programming techniques for the nurse rostering problem, Annals of Operations Research 239 (2016) 225–251.

[23] F. He, R. Qu, A constraint programming based column generation approach to nurse rostering problems, Computers & Operations Research 39 (2012) 3331–3343.

[24] S. Asta, E. Özcan, T. Curtois, A tensor based hyper-heuristic for nurse rostering, Knowledge-Based Systems 98 (2016) 185–199, ISSN 0950-7051.

[25] A. Gretsista, E. K. Burke, An Iterated Local Search Framework with Adaptive Operator Selection for Nurse Rostering, in: R. Battiti, D. E. Kvasov, Y. D. Sergeyev (Eds.), Learning and Intelligent Optimization, Lecture Notes in Computer Science, Springer International Publishing, ISBN 978-3-319-69404-7, 93–108, 2017.

[26] Z. Zheng, X. Liu, X. Gong, A simple randomized variable neighbourhood search for nurse rostering, Computers & Industrial Engineering 110 (2017) 165–174.

[27] F. Knust, L. Xie, Simulated annealing approach to nurse rostering benchmark and real-world instances, Annals of Operations Research 272 (1) (2019) 187–216, ISSN 1572-9338.

[28] E. Rahimian, K. Akartunal, J. Levine, A hybrid integer and constraint programming approach to solve nurse rostering problems, Computers & Operations Research 82 (2017) 83–94.

[29] E. Rahimian, K. Akartunal, J. Levine, A hybrid Integer Programming and Variable Neighbourhood Search algorithm to solve Nurse Rostering Problems, European Journal of Operational Research 258 (2017) 411–423.

[30] Z. Lü, J.-K. Hao, Adaptive neighborhood search for nurse rostering, European Journal of Operational Research 218 (3) (2012) 865–876, ISSN 0377-2217.

[31] C. Rae, N. Pillay, Investigation into an evolutionary algorithm hyperheuristic for the nurse rostering problem, in: Proceedings of the 10th International Conference on the Practice and Theory of Automated, PATAT'14, 527–532, 2014.

[32] S. Adriaensen, T. Brys, A. Nowé, Fair-share ILS: A Simple State-of-the-art Iterated Local Search Hyperheuristic, in: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, GECCO '14, ACM, 1303–1310, 2014.

[33] M. Mustafa, K. Verbeeck, P. De Causmaecker, G. Vanden Berghe, An Intelligent Hyper-Heuristic Framework for CHeSC 2011, in: Y. Hamadi, M. Schoenauer (Eds.), Learning and Intelligent Optimization, LNCS, Springer, ISBN 978-3-642-34412-1 978-3-642-34413-8, 461–466, 00030, 2012.

[34] M. Römer, T. Mellouli, A direct MILP Approach Based on State-Expanded Network Flows and Anticipation for Multi Stage Nurse Rostering under Uncertainty, in: PATAT'16 Proceedings of the 11th International Conference on Practice and Theory of Automated Timetabling, PATAT'16, 549–551, 2016.

[35] H. Jin, G. Post, A. Schaerf, E. Vanden Veen, ORTEC's contribution to the Second International Nurse Rostering Competition, in: PATAT'16 Proceedings of the 11th International Conference on Practice and Theory of Automated Timetabling, PATAT'16, 499–501, 2016.

[36] A. Legrain, J. Omer, S. Rosat, An Online Stochastic Algorithm for a Dynamic Nurse Scheduling Problem, Submitted for Publication (2018) 1–27.

[37] F. Mischek, M. Nysret, Integer Programming and Heuristic Approaches for a Multi-Stage Nurse Rostering Problem, in: PATAT'16 Proceedings of the 11th International Conference on Practice and Theory of Automated Timetabling, PATAT'16, 245–262, 2016.

[38] N. Thi Thanh Dang, S. Ceschia, A. Schaerf, P. De Causmaecker, S. Haspeslagh, Solving the Multi-Stage Nurse Rostering Problem, in: PATAT'16 Proceedings of the 11th International Conference on Practice and Theory of Automated Timetabling, PATAT'16, 473–475, 2016.

[39] S. Ceschia, R. Guido, A. Schaerf, Solving the static INRC-II nurse rostering problem by simulated annealing based on large neighborhoods, In print Annals of Operations Research (2020) 1–19.

[40] P. De Causmaecker, G. Vanden Berghe, Relaxation of Coverage Constraints in Hospital Personnel Rostering, in: E. K. Burke, P. De Causmaecker (Eds.), Practice and Theory of Automated Timetabling IV, Springer Berlin Heidelberg, Berlin, Heidelberg, 129–147, 2003.

[41] A. Kheiri, E. Keedwell, A sequence-based selection hyper-heuristic utilising a hidden Markov model, in: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, GECCO '15, ACM, New York, NY, USA, 417–424, 2015.

[42] M. Hollander, D. A. Wolfe, E. Chicken, Nonparametric Statistical Methods, Wiley, 3 edition edn., ISBN 9780470387375, 2013.

[43] A. Kheiri, E. Keedwell, M. J. Gibson, D. Savic, Sequence analysis-based hyper-heuristics for water distribution network optimisation, Procedia Engineering 119 (2015) 1269–1277, computing and Control for the Water Industry (CCWI2015) Sharing the best practice in water management.