J Heuristics (2012) 18:401–434
DOI 10.1007/s10732-011-9192-0

One hyper-heuristic approach to two timetabling problems in health care

Burak Bilgin · Peter Demeester · Mustafa Misir · Wim Vancroonenburg · Greet Vanden Berghe

Received: 7 November 2008 / Accepted: 20 October 2011 / Published online: 9 November 2011

Abstract We present one general high-level hyper-heuristic approach for addressing two timetabling problems in the health care domain: the patient admission scheduling problem and the nurse rostering problem. The complex combinatorial problem of patient admission scheduling has only recently been introduced to the research community. In addition to the instance that was introduced on this occasion, we present a new set of benchmark instances. Nurse rostering, on the other hand, is a well studied operations research problem in health care. Over the last years, a number of problem definitions and their corresponding benchmark instances have been introduced. Recently, a new nurse rostering problem description and datasets were introduced during the first Nurse Rostering Competition. In the present paper, we focus on this nurse rostering problem description.

The main contribution of the paper constitutes the introduction of a general hyper-heuristic approach, which is suitable for addressing two rather different timetabling problems in health care. It is applicable without much effort, provided a set of low-level heuristics is available for each problem. We consider the instances of both health

B. Bilgin · P. Demeester (✉) · M. Misir · W. Vancroonenburg · G. Vanden Berghe
Computer Science, KaHo Sint-Lieven, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
e-mail: Peter.Demeester@kahosl.be

B. Bilgin
e-mail: Burak.Bilgin@kahosl.be

M. Misir
e-mail: Mustafa.Misir@kahosl.be

W. Vancroonenburg
e-mail: Wim.Vancroonenburg@kahosl.be

G. Vanden Berghe
e-mail: Greet.VandenBerghe@kahosl.be

P. Demeester · M. Misir · W. Vancroonenburg · G. Vanden Berghe
Department of Computer Science, K.U. Leuven, Campus Kortrijk, Etienne Sabbelaan 53, 8500 Kortrijk, Belgium

care problems for testing the general applicability of the hyper-heuristic approach. Also, improvements to the previous best results for the patient admission scheduling problem are presented. Solutions to the new nurse rostering instances are presented and compared with results obtained by an integer linear programming approach.

Keywords Hyper-heuristic · Health care · Benchmark instances

1 Introduction

Hyper-heuristics were introduced as high level search and optimisation methods for solving problems in a 'good enough-soon enough-cheap enough' fashion (Burke et al. 2003). Instead of operating directly on a set of candidate solutions, like meta-heuristics, they operate on a set of (meta-)heuristics. This property allows developers to easily deploy various heuristics at hand. Another objective of hyper-heuristics research is to generate synergy between heuristics involved in the search, by making use of their strengths, avoiding their weaknesses, and taking advantage of their combined capabilities (Burke et al. 2003).
Most hyper-heuristic approaches select, at every iteration, a low-level heuristic according to a selection mechanism. Whether a solution generated by that heuristic is accepted depends on the move acceptance criterion. The heuristic selection mechanism, as well as the move acceptance criterion, can be based on (meta-)heuristics. Dowsland et al. (2007), for example, present a hyper-heuristic that uses tabu search as heuristic selection method and simulated annealing as move acceptance criterion. They deploy the approach to determine shipper sizes for storage and transportation. Kendall and Mohamad (2004) introduce a great deluge metaheuristic as acceptance criterion, and apply the resulting hyper-heuristic to the channel assignment problem in cellular communication. Ayob and Kendall (2003) utilise three different Monte Carlo strategies as acceptance criteria in a hyper-heuristic in order to optimise component placement sequencing for multi head machines. Hyper-heuristics have been also successfully applied to several health care problems. Tabu search has been used as a heuristic selection method for tackling timetabling and rostering problems including nurse rostering in Burke et al. (2003). Bilgin et al. (2009) apply a hyper-heuristic for generating rosters for nurses in a Belgian hospital.

In this paper we apply one general high-level hyper-heuristic approach for solving two health care timetabling problems. We report on experiments with different heuristic selection mechanisms and move acceptance criteria. Even though the proposed approach is rather general and widely applicable without much effort, it generates competitive results. Nevertheless, for each and every new problem the problem-specific parts such as the objective function, the low-level heuristics, etc. still need to be designed and implemented.

The first problem is the patient admission scheduling problem, which is gaining increasing attention in health care practice. This can be attributed to the fact that hospitals experience more and more pressure to maximise their bed occupancy and at the same time minimise the duration of each patient stay. These may result in poor conditions for the patients, such as unplanned transfers from one room to another, assignments to rooms that do not match the patients' preference, etc. The goal of the patient admission scheduling problem is on the one hand increasing the patients' comfort, while at the same time assisting the admission scheduler with the execution of his/her task. The problem involves a hospital with several organizational units, e.g. wards, and a number of patients with given arrival dates and expected departure dates. Patient assignments (to a particular bed in a room of a ward) are subject to constraints concerning the medical equipment in the room, the medical skills of the personnel who belong to that ward, the patient's room preference, etc. Although the need for improved efficiency is high, we notice that the problem has not yet attracted the interest of a large group of researchers yet. The patient admission scheduling problem is described in detail in Demeester et al. (2010), in which one problem instance is introduced and solved with a token-ring tabu search approach and some metaheuristic variants. Integer programming has also been applied to the problem instance, but no optimal solution could be obtained after a week of computation. That paper sets a benchmark that enables comparison and evaluation of new algorithms.

Nurse rostering is the second health care problem that is studied in this paper. It addresses the process of assigning nurses to shifts, taking into account coverage, personal and legal constraints. Coverage constraints express the number of nurses needed per shift and per day to satisfy the daily demand of the department. This number depends, amongst other things, on the bed capacity of the department. Typically, the coverage constraint is formulated as a number of nurses with a certain skill type that need to be present on a particular shift of a day. Some legal constraints should be taken into account as well. In Belgium, for example, at least 11 hours of rest are required between two consecutive appearances. This is colloquially translated into the constraint that no nurse should be assigned to more than one shift per day. The coverage and the single shift per day constraint could be modelled as conditions that need to be satisfied in order to obtain a feasible solution. Nurse rostering is, in contrast to the patient admission scheduling problem, a well-studied problem in operations research (see Sect. 3.3).

The main contribution of this paper is the introduction of one high-level hyper-heuristic approach to the patient admission scheduling problem and the nurse rostering problem. By addressing two considerably different problems, we achieve one of the major challenges in hyper-heuristics research, namely raising the level of generality. Moreover, we obtain competitive results for both problems.

The outline of the paper is as follows. We present a detailed problem description of the patient admission scheduling problem in Sect. 2, and compare with other problems described in the literature. Next, we describe the nurse rostering problem in Sect. 3, and review the corresponding literature very briefly. In Sect. 4, the hyper-heuristic approach is introduced. The stress is on the high level selection strategy, the move acceptance criteria, and on the low-level heuristics. We describe the experimental setup for both health care problems and discuss the results in Sect. 5. Regarding the patient admission scheduling problem, the results are compared against previously generated results for the same benchmark problems. For the nurse rostering problem, we compare with the results obtained with an integer linear programming approach. In Sect. 6, we conclude and put forward some promising directions for future research.

2 Patient admission scheduling

2.1 Problem description

The objectives and constraints of the patient admission scheduling problem have been formalised after extensive discussions with decision makers in hospitals and with people who are informed about the hospital occupancy rates that are enforced by the government. The problem that we delineated for this research does not consider intensive care units nor day clinics. Also, we suppose that every patient is attributed an admission and discharge date in advance. In other words, we do not consider patients on waiting lists.

The basic elements of the problem at hand are patients, wards, rooms and timeslots. Associated with a patient are: his/her treatment requirements (in terms of nursing and medical equipment), gender, age category, room preference and the first day and the duration of his/her stay (this is called the length of stay of the patient). Similarly, some characteristics are related to rooms, e.g. existing medical equipment, the number of beds and the ward to which they belong. A ward defines the possible treatments that the rooms are equipped for. A time slot corresponds to a night. The occupation of one bed during one time slot by a patient is called a patient stay unit. The time horizon that we look at equals T, which corresponds to the set of all timeslots. We consider that the patients' stay durations over that period are known in advance and do not change during the stay. The model captures the knowledge of a decision maker at time 0.

The objective is to optimise the overall patient assignment, i.e. satisfy the patients' preferences, while respecting all the hard constraints to the problem. Hard constraints need to be satisfied in any solution that the algorithm comes up with. The hard constraints are:

1. Maximum one patient per bed and time slot;
2. The admission and discharge dates are fixed. In other words, these dates cannot be changed by the algorithm;
3. For each time slot during the length of stay of a patient, s/he must be assigned to a bed.

The quality of a solution is determined by the soft constraints. Soft constraints are applied either to patients, or to rooms, to patients and rooms at the same time. The objective function is the weighted sum of all the violations of the soft constraints. The optimization problem is a minimization problem.

1. Patients in the same room-time slot should have the same gender;
2. The number of room transfers should be minimised;
3. The ward of the patient should satisfy the requirement of his/her pathology;
4. The room of the patient should satisfy the mandatory/preferred requirements of his/her pathology;
5. The room of the patient should satisfy the specialism of his/her pathology;
6. The room preference type of the patient should be satisfied.

2.1.1 Mathematical model

Sets and constants definitions

- Let P be the set of all patients;
- Let B be the set of all beds;
- Let R be the set of all rooms;
- Let T be the set of all timeslots;
- Each patient is either male or female:

$$g_p = \begin{cases} 0 & \text{if patient } p \text{ is female,} \\ 1 & \text{if patient } p \text{ is male;} \end{cases}$$

- Let $f_{r,t}$ be the number of female patients in room r at time slot t, then

$$f_{r,t} = \sum_{p \in P} (1 - g_p).rs_{p,r,t};$$

- Let $m_{r,t}$ be the number of male patients in room r at time slot t, then

$$m_{r,t} = \sum_{p \in P} g_p.rs_{p,r,t};$$

- Set of patient stays:

$$PS_p = \left\{ t \in T \mid t \geq t_{s,p} \wedge t < t_{s,p} + d_p \right\},$$

  where $t_{s,p}$ is the start day of the stay of patient p, and d_p is the duration of the patient's stay;
- Let $rp_{p,r}$ be the total room penalty of the soft constraints 3, 4, 5, and 6 in case the patient p is assigned to room r. These penalties are known in advance, since for every patient assigned to a room, one can calculate the corresponding penalty, based on the soft constraints 3, 4, 5, and 6.

Decision variables

- Bed schedule

$$bs_{p,b,t} = \begin{cases} 1 & \text{if patient } p \text{ is assigned to bed } b \text{ on timeslot } t, \\ 0 & \text{otherwise;} \end{cases}$$

- Room schedule

$$rs_{p,r,t} = \begin{cases} 1 & \text{if patient } p \text{ is assigned to room } r \text{ on timeslot } t, \\ 0 & \text{otherwise;} \end{cases}$$

- Transfer

$$tr_{p,t} = \begin{cases} 1 & \text{if } t \in PS_p \wedge t + 1 \in PS_p \wedge \exists r \in R \mid rs_{p,r,t} \neq rs_{p,r,t+1}, \\ 0 & \text{otherwise.} \end{cases}$$

Hard constraints

$$\forall b \in B, \forall t \in T, \quad \sum_{p \in P} bs_{p,b,t} \leq 1. \tag{1}$$

Equation 1 corresponds to the first hard constraint.

$$\forall p \in P, \forall t \in PS_p, \quad \sum_{b \in B} bs_{p,b,t} = 1. \tag{2}$$

Equation 2 corresponds to hard constraints 2 and 3.

Soft constraints

– Gender penalties:

$$GP = \sum_{r \in R} \sum_{t \in T} \min(f_{r,t}, m_{r,t}). \tag{3}$$

Equation 3 corresponds to the first soft constraint.

– The total number of transfers:

$$TR = \sum_{p \in P} \sum_{t \in T} tr_{p,t}. \tag{4}$$

Equation 4 corresponds to the second soft constraint.

– Room penalty:

$$RP = \sum_{p} \sum_{r} \sum_{t} rs_{p,r,t} rp_{p,r}. \tag{5}$$

Equation 5 corresponds to remaining soft constraints.

Objective function

$$MinObj = w_g.GP + w_t.TR + w_r.RP,$$

with w_g the weight of the violation of the gender constraint, w_t the weight of the violation of the transfer constraint, and w_r the weight of the violation of the room constraints.

2.2 Related literature

We refer to Gemmel and Van Dierdonck (1999) for a comprehensive study about the patient admission scheduling problem. It is noted that the problem should not be solved without taking surgery capacity and availability of nurses into account. Smith-Daniels et al. (1988) too identify concerns that should be taken into account when optimizing bed occupancy. Similar problems to patient admission scheduling are addressed with mathematical programming (Ogulata and Erol 2003), local search (Hans et al. 2008) and a multi agent system (Marinagi et al. 2000).

Vermeulen et al. (2009) describe a CT-scan scheduling problem, which will—when solved to optimality—contribute to efficient patient scheduling. Central diagnostic resources such as CT-scans are often bottlenecks in a hospital environment and the goal is to minimise the patients' waiting time and to maximise the resources' utilization. Patients are scheduled according to their arrival order. For every patient a random free time slot is selected that fits into the patient's time window. The goal in the rehabilitation patient scheduling paper of Chien et al. (2008) is similar: reduce the patients' waiting times and increase the equipment usage. Since some of the rehabilitation therapies require a specific order of execution, this problem can be seen as a hybrid shop scheduling problem in which the medical resources correspond to the machines and the patients to the jobs. Chien et al. solve the problem with a genetic algorithm.

Vermeulen et al. (2007) apply agent technology is applied to solve the patient appointment exchanging problem. Patients are represented by agents that try to decrease the patients' waiting time. Initially, patients are assigned to a schedule that consists of several small appointments which are attributed a time slot. Each agent will try to exchange appointments with other agents in order to obtain a better schedule. An agent accepts an exchange only if it does not worsen its schedule. Vermeulen et al. (2008) describe the on line scheduling of outpatients that have a combination of appointments in different wards. The goal is to optimise the appointments of the outpatients and the wards' efficiency. This is a complex problem since it needs a lot of collaboration between the wards of the hospital. Similar to Vermeulen et al. (2007), agent technology is applied to model the wards and the outpatients. The agents communicate with each other to find the best time slot for each party involved.

The objective of the problems presented in the papers above is to assign patients to timeslots in which they can be examined or diagnosed. The smallest timeslots are 15 minutes. In this paper; timeslots correspond to an overnight stay. However, the major difference between the problem described in this paper and the other problems in the literature, is that we are also confronted with the problem of consecutiveness. This means that patients who have been admitted to the hospital for multiple nights, should preferably be assigned to the same bed during their stay. This constraint adds considerable complexity to the problem.

Hutzschenreuter et al. (2008) describe an agent-based patient admission scheduling application for a highly decentralised problem. In order to analyse and evaluate different policies concerning bed scheduling, what-if scenario's are applied. Patients are automatically assigned to beds with a brute-force optimiser. The time needed to assign patients to a small number of beds in a planning horizon of one year was 12.8 hours. No further details are given about the brute-force algorithm. From all the problems that we came across in the literature, it bears the closest resemblance to the one that we address in this paper.

3 Nurse rostering

3.1 Problem description

The specific nurse rostering problem that is addressed by the same hyper-heuristic approach that is applied to the patient admission scheduling problem, was taken from the Nurse Rostering Competition (Haspeslagh et al. 2010). We are aware of some assumptions in the nurse rostering benchmarks that do not fully correspond to the real world situation. For example, the problems do not take any previous assignments into account. They also limit the number of assignments per nurse and per day to one shift. Anyway, for investigating the performance of the hyper-heuristic, they are perfectly suitable.

Nurse rostering is the process of assigning nurses to shifts taking into account the coverage, personal and legal constraints. A roster consists of several days that are further divided into shifts. Every shift of a day corresponds to coverage requirements that specify the number of nurses that should be working at that time. The coverage constraints are hard constraints to the problem. The only other hard constraint states that nurses can be assigned at most one shift per day. Each nurse has exactly one contract, which provides the following information:

- the maximum and minimum number of assignments per planning period and per nurse;
- the maximum and minimum number of consecutive working days;
- the maximum and minimum number of consecutive free days;
- the maximum number of consecutive working weekends;
- whether a nurse needs two free days after being assigned to a night shift;
- whether a nurse has to work all days of the weekend (complete weekends);
- whether a nurse has to work the same shift types during a working weekend;
- which combination of shift types are unwanted, for example, do not assign a nurse to the late shift of the first day, the early shift of the second day and the late shift of the following day;
- the shifts and days on/off requests of the nurses. These are requests of the nurses (not) to be assigned on certain shifts or days.

The elements of these contracts correspond to the soft constraints of the problem. For all data instances of the Nurse Rostering Competition, the planning period is 4 weeks, and the number of shifts per day ranges from 4 to 5.

For the complete problem description of the Nurse Rostering Competition instances, we refer to Haspeslagh et al. (2010).

3.2 Mathematical model

The mathematical model for this specific nurse rostering problem is based on the model in Burke et al. (2010). For the interested reader, we have added the complete mathematical model for this specific problem in the Appendix.

3.3 Related literature

Nurse rostering research has gained a lot of academic attention, in contrast to patient admission scheduling. We therefore restrict the overview to a small set of contributions. Burke et al. (2004) cover a large part of the literature on nurse rostering until 2003. They present an extensive analysis of problem characteristics and solution approaches. De Causmaecker and Vanden Berghe (2011) introduce a classification of nurse rostering problems that enables comparison of descriptions. Brucker et al.

(2010) were the first to present a variety of nurse rostering benchmark instances, together with a solution validator. The Nurse Rostering Competion based its problem description on these benchmark instances.

4 Solution methods

4.1 Overview

A local search neighbourhood corresponds to the set of all candidate solutions that can be reached from one solution by carrying out a specific move. A local search algorithm searches for a good quality solution by traversing the neighbourhoods of a single candidate solution. Tabu search, for example, adds some memory to the local search by applying a finite tabu list. Elements of every accepted modification are added at the start of the tabu list replacing the oldest item in the tabu list. If the quality of the candidate solution is better than the previous best solution, the modification is executed. If the modification does not lead to a better solution, the best move is accepted if that is not prohibited by the tabu list. This mechanism helps escaping from local optima and forces the algorithm to explore new regions of the solution space. In Demeester et al. (2010) the patient admission scheduling problem is tackled with a tabu search algorithm hybridised with a token-ring approach. This mechanism deploys several neighbourhoods, and carries out the search by switching between them in a circular fashion.

Hyper-heuristics are solution methods that deploy a set of heuristics. This way they are distinguished from the metaheuristics approaches that operate directly on the solution space. Several classes of hyper-heuristics have been developed: 'heuristics to choose heuristics' or 'heuristics to generate heuristics' (Burke et al. 2010). In this work, we consider hyper-heuristics that choose heuristics. This mechanism iteratively selects a heuristic and applies it to a single candidate solution. After the selection step, the resulting candidate solution is either accepted or rejected. The hyper-heuristic framework that is used in this article is based on the one described in Özcan et al. (2008). It consists of two main parts: the *heuristic selection mechanism* and the *move acceptance criterion*.

– The heuristic selection mechanism selects every iteration one heuristic from the set of low-level heuristics.
– The move acceptance criterion decides every iteration which move is accepted. Several well-known metaheuristic methods can be applied as a move acceptance criterion.

The hyper-heuristic framework is the same for both health care problems. The followed approach for both problems only differs in the solution representation, the construction of the initial solution, and the low-level heuristics. These parts contain information specific to each of the problems.

4.2 Modeling approach

For both health care problems, we describe in the following sections the solution representation and the objective function.

4.2.1 Patient admission scheduling

Since the modeling approach of the patient admission scheduling problem is described elsewhere (Demeester et al. 2010), not all the details need to be included in this paper. A solution is represented as a set of matrices, each representing a ward of the hospital. The rows of an individual matrix correspond to the available beds in the ward, while the columns correspond to the timeslots. In the solution representation, a patient assignment is represented as a contiguous set of patient stay parts. There are as many patient stay parts as the length of stay of the corresponding patient. By choosing this particular representation, violations of the first hard constraint (not more than one patient per bed) are automatically excluded.

The initial solution is constructed such that all patients are assigned to beds as long as there are free beds available. This automatically leads to satisfaction of the last hard constraint. The second hard constraint will always be fulfilled by only allowing low-level heuristics that do not violate this constraint (see Sect. 4.3.1).

4.2.2 Nurse rostering

A solution for the nurse rostering problem is modeled as a 0–1 matrix, in which the columns represent the shifts arranged per day, and the rows represent the nurses. A nurse is assigned to a shift on a particular day if the value of the corresponding matrix element is 1. Although the initial solution is randomly constructed, the initialization algorithm makes sure that the solution is feasible. This means that the coverage is met and that no nurse works more than one shift in the same day. Feasibility is maintained during the subsequent search by only considering assignment moves within the same column. No assignment can be removed without making a new one within the column. This means that the coverage constraint and the single shift per day constraint will always be satisfied.

In both problem cases the evaluation of the soft constraints will be included in the objective function, which is the weighted sum of their violations.

4.3 Low-level heuristics

The heuristics that are called by the hyper-heuristic are inspired by the mechanism of tournament selection in genetic algorithms. At each iteration, the selected heuristic creates a number of moves and returns the move that results in the best value of the objective function. The number of moves considered per iteration is a parameter of the search algorithm that we call the tournament size.

4.3.1 Patient admission scheduling

The heuristics only consider moves that do not change the admission and the discharge dates of the patients:

1. Swap two patients' assignments in the same ward. The ward and the patients are selected randomly.
2. Swap two patients' assignments in different wards. The wards and the patients are selected randomly.

3. Transfer all the assignments of a patient to (an) empty bed(s) in another ward. The source and destination wards, the patient, and the destination bed(s) are selected randomly.
4. Transfer all the assignments of a patient to (an) empty bed(s) in the same ward. The ward, the patient, and the destination bed(s) are selected randomly.

The source and destination beds in the first and the second heuristic can either be empty or assigned to a patient.

4.3.2 Nurse rostering

We present a sample of the low-level heuristics in the hyper-heuristic approach: in all the low-level heuristics, the first step is to randomly select two nurses n_1, n_2, ($n_1 \neq n_2$, and $n_1, n_2 \in N$, with N the set of nurses)

– randomly select a subset $d \subset D$, with D the set of all days:

$$\mathit{swap}(\mathit{roster}(n_1, d), \mathit{roster}(n_2, d)),$$

– randomly select a subset $w \subset W$, with W the set of all weekends:

$$\mathit{swap}(\mathit{roster}(n_1, w), \mathit{roster}(n_2, w)),$$

– randomly select a subset $v \subset V$, with V the set of all work days (or $V = D \backslash W$):

$$\mathit{swap}(\mathit{roster}(n_1, v), \mathit{roster}(n_2, v)).$$

In fact, we have a dozen low-level heuristics, which are all based on these three abstract heuristics. For example, we use a heuristic that selects the even or odd weekends (or work weeks) of two randomly chosen nurse rosters. Another low-level heuristic swaps one, two, three, four or five (non-)contiguous days of two randomly chosen nurses.

4.4 Hyper-heuristic framework

As mentioned in Sect. 4.1, the hyper-heuristic framework that is employed in this paper consists of two main parts: a mechanism to select one of the low-level heuristics during the search, and a criterion to accept moves.

4.4.1 Heuristic selection method

The heuristic selection methods are *simple random*, *choice function*, and a *dynamic heuristic set strategy with simple random*.

– Simple random randomly selects a heuristic from a list of heuristics at each iteration.
– Choice function (Cowling et al. 2001) considers a number of criteria: the performance of each heuristic and each pair of heuristics when called consecutively, and the elapsed time since the last call of a heuristic. The choice function consists of a linear combination of the following components (see, Soubeiga 2003, for more details):

- $f_1(h_j)$, which is the current performance of every heuristic h_j. This is expressed as $f_1(h_j) = \sum_n \alpha^{n-1}(\frac{I_n(h_j)}{T_n(h_j)})$, where $I_n(h_j)$ corresponds to the change in the evaluation function of the heuristic h_j, and $T_n(h_j)$ corresponds to the amount of CPU time taken in heuristic h_j, and where $\alpha \in [0, 1]$.
- $f_2(h_k, h_j)$, which is the joint performance of couples of heuristics (h_k, h_j). This is expressed as $f_2(h_k, h_j) = \sum_n \beta^{n-1}(\frac{I_n(h_k, h_j)}{T_n(h_k, h_j)})$, where $I_n(h_k, h_j)$ is the change in the evaluation function since the last time heuristic h_j was called immediately after heuristic h_k, and where $T_n(h_k, h_j)$ is the amount of CPU time taken since the last time heuristic h_j was called immediately after heuristic h_k, with $\beta \in [0, 1]$.
- $f_3(h_j)$ which is the elapsed time since the last execution of heuristic h_j.

  The resulting choice function can then be expressed as $f(h_j) = \alpha f_1(h_j) + \beta f_2(h_k, h_j) + \delta f_3(h_j)$.

- The motivation behind the dynamic heuristic set strategy with simple random (Misir et al. 2010), is to determine the best heuristic subsets for the different phases of a search. Each phase refers to a predefined number of iterations in which the performance, based on a performance metric, of the available n heuristics is measured. The lesser performing heuristics are excluded for a number of phases. The number of phases for the exclusion process is called the tabu duration.

  In this study, we employ the following performance metric:

$$p_i = M_1(i)w_1 + M_2(i)w_2 + M_3(i)w_3 \tag{6}$$

- $M_1(i)$ denotes the total number of new best solutions found by heuristic i,
- $M_2(i)$ denotes the improvement of the objective function per execution time during the current phase by heuristic i,
- $M_3(i)$ denotes the total improvement per execution time by heuristic i.

w_1, w_2 and w_3 are the weights for each sub performance metric and their values satisfy $w_1 \gg w_2 \gg w_3$. This way, some kind of priority between these three sub performance metrics is provided.

The tabu duration value and phase length are determined based on the size of the heuristic set. The tabu duration is $d = \sqrt{2n}$ and the phase length is $pl = d * 1000$ iterations. At the end of each phase, each heuristic gets a quality index (QI) value, which can range between 1 and n. The best performing heuristic gets n' which is the number of heuristics in the current heuristic subset. The worst one gets 1 for QI. The excluded heuristics have $QI = 1$. Then, the average of the QIs is calculated:

$$avg = \left\lfloor \Big(\sum_i^n QI_i\Big)/n \right\rfloor. \tag{7}$$

The heuristics with $QI < avg$ are excluded from the heuristic set during d phases.

Algorithm 1 Pseudo code of the *simulated annealing* acceptance criterion

C_i = objective function value at ith iteration
$\delta = C_{i+1} - C_i$
T = total execution time
R = remaining execution time
P_i = random variable between [0,1[at ith iteration
if $\delta \leq 0$ **then**
  Accept
else
  if $P_i < \exp(-\delta * T/R)$ **then**
    Accept
  else
    Reject
  end if
end if

4.4.2 *Move acceptance criteria*

We have experimented with four acceptance criteria. The acceptance criteria are *only improving*, *improving and equal*, *simulated annealing* (Kirkpatrick et al. 1983), and *great deluge* (Dueck 1993).

- Only the moves leading to solutions that are better than the current solution are accepted by the *only improving* acceptance criterion.
- The *improving and equal* acceptance criterion works similarly, except that it also accepts moves that result in solutions that are as good as the current solution.
- The *simulated annealing* algorithm accepts all improving and equal moves. The moves that result in solutions that are not at least as good as the current solution are carried out with a probability, which is decreased throughout the execution. This acceptance criterion is presented in Algorithm 1.
- The *great deluge* acceptance criterion maintains a deluge level throughout the execution. The initial value of the deluge level is set equal to the cost value. Throughout the search, this level is reduced at each iteration. The criterion accepts all improving and equal moves and the moves that result in a cost value below the deluge level. This acceptance criterion is explained in Algorithm 2.

5 Experiments

All experiments were performed on Intel Core2Duo (3 GHz) PC's running Windows XP Professional SP3, with a Java 1.6 JRE configured to run in server mode with a heap size of 128 MB.

We experiment with the four different acceptance criteria: *simulated annealing* (SA), *great deluge* (GD), *improving and equal* (IE), and *only improving* (OI), and with the three heuristic selection mechanisms: *simple random* (SR), *choice function*

Algorithm 2 Pseudo code of the *great deluge* acceptance criterion

C_i = Cost value of candidate solution at ith iteration
T = total execution time
R = remaining execution time
C_0 = initial cost
$D = C_0 * R/T$
if $C_{i+1} \leq C_i$ **then**
  Accept
else
  if $C_{i+1} < D$ **then**
    Accept
  else
    Reject
  end if
end if

Table 1 The characteristics of the benchmark instances

Property/Benchmark Instances	0	1	2	3	4	5	6
Number of departments	6	4	6	5	6	4	4
Number of rooms	150	98	151	131	155	102	104
Number of beds	447	286	465	395	471	325	313
Number of female patients	339	315	359	365	361	279	349
Number of male patients	321	337	396	343	385	308	336
Maximum occupancy	0.69	0.77	0.79	0.82	0.75	0.73	0.91
Average occupancy	0.50	0.60	0.60	0.57	0.54	0.49	0.64

(CH), and the *dynamic heuristic set strategy with simple random* (DHS). Three different tournament sizes are explored: 4, 16, and 64. As a result, 36 hyper-heuristic variations are tested on each patient admission scheduling and nurse rostering benchmark instance.

5.1 Patient admission scheduling problem

The experiments are carried out on one existing and six new benchmark instances. The existing dataset (`testdata 0`) was first introduced in Demeester et al. (2010). All instances are automatically generated based on several interviews with people responsible for scheduling patients in Belgian hospitals. Obtaining real world data about patients turned out to be difficult, due to privacy issues. Although automatically generated, it was confirmed by practitioners that the datasets are realistic. Each benchmark instance has a planning horizon of two weeks. The properties of the benchmark instances are given in Table 1. For the new instances, the length of stay of every patient (expressed in nights) is presented in Figs. 2(a), 2(b), 2(c), 3(a), 3(b), 3(c). The occupancy rate varies over the planning horizon and it is higher during

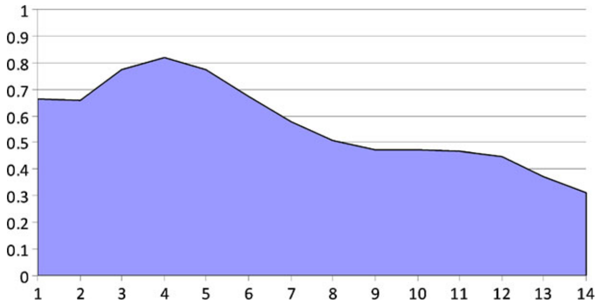


Fig. 1 Example of the occupancy rate over the planning horizon (benchmark instance 3)

the week than in the weekend. During the first week, it is set higher than in the second week. That resembles the real world situation in which patients' medical treatments are not always scheduled two weeks in advance. In Fig. 1, the occupancy rate of benchmark instance 3 is depicted over the planning horizon, as an example. Table 2 presents the distribution of the room types. There are more quadruple rooms than double rooms, and more double rooms than single rooms. This is in correspondence with the situation in Belgian hospitals. The benchmark instances and the weights of the objective function can be found on the Patient Admission Scheduling website (Demeester et al. 2008).

In total, 36 different solution methods are tested for each of the patient admission scheduling benchmark instances. Each solution method is run 10 times per instance. The termination criterion is the computation time, which is set to 3000 seconds for each run. In order to have an objective means of comparison with other researchers, we opt for applying the same benchmark method as provided by the organizers of the Second International Timetabling Competition (ITC).[1] It measures the speed of the competitor's computer by solving a combinatorial optimisation problem (i.e. the travelling tournament problem) and measuring the time it takes to achieve a certain solution. Based on this local measurement, the benchmark program determines the maximum allowed computation time for solving the ITC data sets. We multiply the resulting computation time by 10, which results in the 3000 seconds computation time mentioned above. The benchmark program enables other researchers to conduct experiments in exactly the same circumstances.

5.2 Nurse Rostering Competition

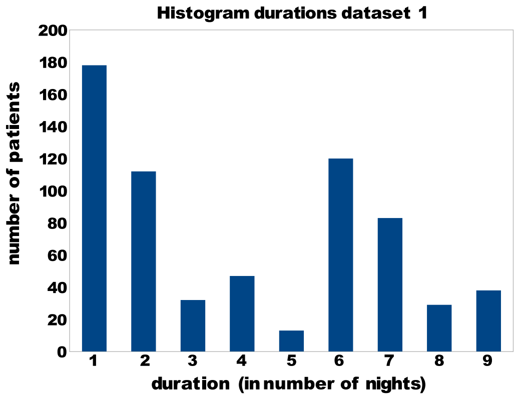
We restrict the experiments to two types of data instances from the Nurse Rostering Competition, namely sprint and medium. The names refer to the available computation time for the particular data instance types. A benchmark tool[2] was again provided by the organisers of the Nurse Rostering Competition.

– The simplest type of data instances (sprint) consists of 10 nurses. For these instances the computation time is on an average desktop PC about 10 seconds.

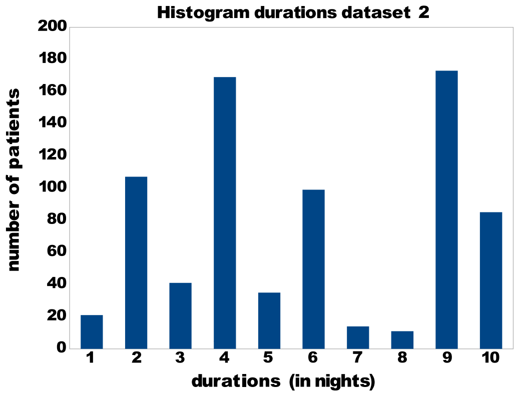
[1] http://www.cs.qub.ac.uk/itc2007/index_files/benchmarking.htm.

[2] http://www.kuleuven-kortrijk.be/nrpcompetition/benchmarking.

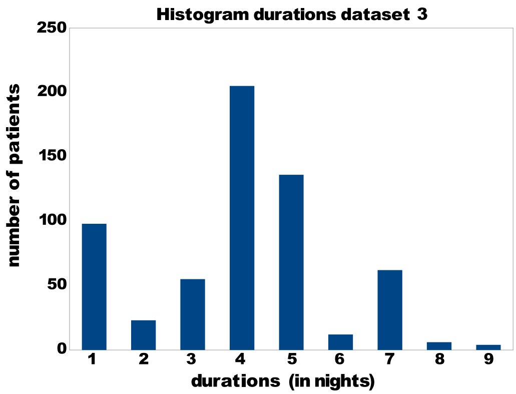
Fig. 2 Histogram of length of stays for every instance



(a) Histogram of the durations in dataset 1



(b) Histogram of the durations in dataset 2

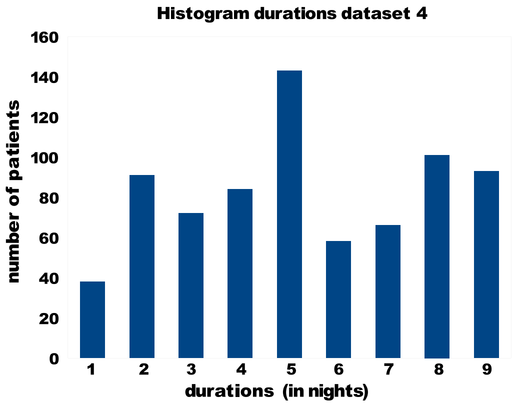


(c) Histogram of the durations in dataset 3

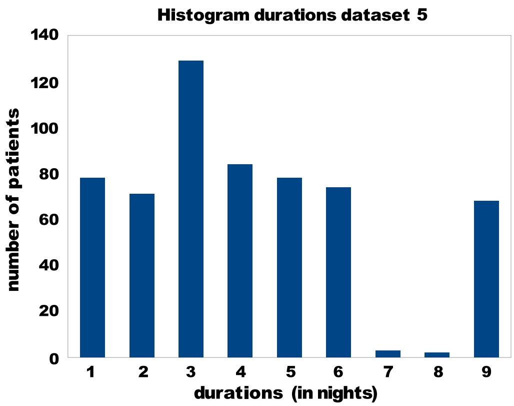
– The second type of data instances (medium) consists of around 30 nurses. The computation time is on an average desktop PC about 10 minutes.

Due to the inherent randomness of the approach, each solution method is run 10 times for every benchmark instance.

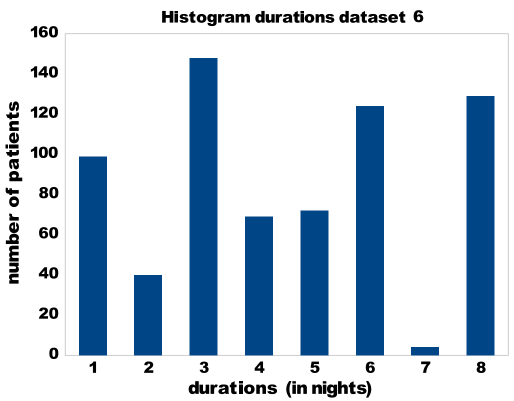
Fig. 3 Histogram of length of stays for every instance



(a) Histogram of the durations in dataset 4



(b) Histogram of the durations in dataset 5



(c) Histogram of the durations in dataset 6

Table 2 Distribution of single, double and quadruple rooms

	% of single rooms	% of double rooms	% of quadruple rooms
testdata 0	12%	32%	55%
testdata 1	16%	29%	54%
testdata 2	12%	26%	60%
testdata 3	13%	30%	57%
testdata 4	12%	30%	58%
testdata 5	8%	27%	64%
testdata 6	10%	34%	56%

5.3 Results

5.3.1 Patient admission scheduling

The minimum, average, and standard deviation of the objective function over 10 runs for each benchmark instance are presented in Tables 3, 4, and 5. The results of the experiments have been statistically processed for detecting significant differences with the Mann-Whitney Ranked Sum test. The best solution methods for each benchmark instance are indicated in bold. We call the set of solutions that perform significantly better than the others the best performing group. The hyper-heuristics in Tables 3, 4, and 5 are first ranked according to the number of times their results are in the best performing group. If there is a tie, the hyper-heuristics are further ordered according to the best total average value over all instances.

For the patient admission scheduling problem, the results show that the DHS heuristic selection mechanism combined with the great deluge move acceptance criterion and tournament factor 4 is the best performing hyper-heuristic. There is, however, no statistical evidence that this combination (DHS-GD-4) performs significantly better than SR-GD-16, CF-GD-4, DHS-GD-64, and SR-GD-4. From the results, it is clear that the hyper-heuristics based on the great deluge move acceptance criterion are the best performers. The 'only improving' move acceptance criterion results in the worst performing hyper-heuristics. There is no statistical evidence to conclude that one of the heuristic selection criteria outperforms the others. Actually, this shows that the performance of the hyper-heuristic depends largely on the performance of the move acceptance criterion.

Analyzing the best solution obtained for benchmark instance 1 (with a cost equal to 672.80) in more detail, reveals that all hard constraints are satisfied, and that only the room type preference soft constraint is violated. This is a consequence of the fact that the demand of single and double rooms exceeds the supply.

The tabu search method that was previously applied to `testdata 0` is outperformed by the current hyper-heuristic. Even the worst performing hyper-heuristic variant (DHS-OI-64) generates better solutions for `testdata 0` than the token-ring tabu search algorithm presented in Demeester et al. (2010).

Table 3 First set of results for the patient admission scheduling problem. Results that are significantly better are marked in bold. The solution methods are ordered from left to right, top to bottom, according to the best total average value over all instances

	DHS-GD-4			SR-GD-16			CF-GD-4			DHS-GD-64		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	**820,00**	**833,66**	**13,41**	**822,40**	**837,18**	**10,64**	**831,40**	**841,84**	**11,79**	**821,60**	**836,62**	**8,72**
testdata 1	**674,80**	**685,68**	**7,43**	**673,00**	**684,56**	**10,62**	**673,60**	**688,04**	**13,95**	**672,80**	**686,10**	**6,38**
testdata 2	**1185,20**	**1202,64**	**15,85**	**1193,40**	**1205,00**	**10,50**	**1174,20**	**1200,26**	**14,23**	**1185,20**	**1202,78**	**14,53**
testdata 3	**803,80**	**822,62**	**22,53**	**803,20**	**813,36**	**7,24**	**800,80**	**815,72**	**11,43**	**804,80**	**819,08**	**7,00**
testdata 4	**1228,20**	**1251,00**	**12,28**	**1242,60**	**1262,44**	**14,23**	**1219,60**	**1262,22**	**22,40**	**1230,40**	**1261,12**	**20,21**
testdata 5	**639,20**	**642,88**	**4,77**	**637,60**	**643,52**	**4,07**	**636,80**	**642,00**	**4,07**	**640,80**	**644,88**	**2,43**
testdata 6	**825,80**	**836,00**	**8,87**	**824,80**	**836,24**	**8,35**	**823,40**	**834,22**	**9,30**	**834,40**	**840,58**	**6,00**
	SR-GD-4			CF-GD-16			SR-GD-64			DHS-GD-16		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	**827,20**	**844,36**	**12,19**	**827,60**	**840,88**	**11,04**	**825,60**	**840,44**	**13,61**	**819,20**	**842,48**	**17,77**
testdata 1	**674,40**	**687,42**	**10,46**	**679,20**	**686,12**	**3,38**	**676,00**	**689,50**	**11,44**	**676,00**	**686,64**	**9,36**
testdata 2	**1179,20**	**1196,92**	**14,43**	**1179,60**	**1199,08**	**12,28**	**1186,40**	**1204,26**	**13,90**	**1180,00**	**1204,94**	**18,00**
testdata 3	**797,80**	**823,42**	**15,68**	**806,00**	**820,46**	**9,94**	**807,20**	**816,04**	**6,89**	808,00	820,66	8,39
testdata 4	**1238,80**	**1264,56**	**20,43**	1244,60	1262,54	13,46	**1227,80**	**1254,12**	**16,17**	**1233,20**	**1256,68**	**9,62**
testdata 5	**638,40**	**643,28**	**3,78**	**634,40**	**642,48**	**4,32**	**637,60**	**642,56**	**3,84**	**637,60**	**643,28**	**3,61**
testdata 6	**825,60**	**838,08**	**10,08**	**822,80**	**834,76**	**9,15**	832,60	844,44	9,11	**818,60**	**838,66**	**12,98**

Table 3 (*Continued*)

	CF-GD-64			CF-IE-16			SR-SA-64			DHS-IE-4		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	**819,40**	**845,98**	**21,25**	**821,60**	**841,22**	**14,00**	**827,20**	**843,26**	**14,17**	**816,80**	**837,46**	**12,95**
testdata 1	684,00	692,30	7,94	**680,00**	**695,88**	**14,07**	687,20	710,98	15,79	680,00	709,00	18,13
testdata 2	**1177,00**	**1204,20**	**16,34**	1196,40	1220,30	16,97	1205,20	1225,56	19,00	1203,20	1232,90	14,57
testdata 3	**802,20**	**822,42**	**13,12**	825,60	843,10	12,66	819,60	848,38	18,75	827,00	841,48	11,52
testdata 4	1245,20	1263,90	14,56	1253,60	1304,56	29,12	1272,60	1313,70	31,48	1278,00	1320,54	21,70
testdata 5	636,80	646,56	5,01	**639,20**	**645,76**	**4,36**	642,40	649,76	3,95	**640,00**	**646,16**	**4,76**
testdata 6	830,00	848,06	9,96	825,20	861,08	18,24	**827,60**	**844,94**	**17,70**	834,00	856,60	16,20
	SR-IE-64			CF-IE-4			SR-SA-16			SR-SA-4		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	**820,80**	**851,34**	**26,67**	**827,60**	**849,82**	**25,73**	**824,20**	**841,36**	**13,14**	**819,20**	**842,00**	**18,70**
testdata 1	685,60	705,22	11,46	686,40	706,48	16,81	682,40	704,12	13,88	692,00	705,58	10,75
testdata 2	1210,00	1230,60	12,83	1212,40	1234,50	12,73	1199,20	1219,82	12,91	1205,60	1219,08	10,45
testdata 3	802,60	845,22	18,98	822,00	845,56	19,17	799,60	835,26	20,75	818,60	837,94	10,01
testdata 4	1294,40	1313,70	23,01	1273,60	1317,46	23,50	1280,20	1300,90	22,30	1260,40	1311,64	33,84
testdata 5	**638,40**	**645,28**	**3,83**	**638,40**	**645,52**	**4,66**	636,80	647,52	5,99	643,20	649,28	4,65
testdata 6	839,20	859,98	22,84	834,60	853,26	10,20	834,00	855,28	16,97	833,20	851,40	12,24

Table 4 Second set of results for the patient admission scheduling problem. Results that are significantly better are marked in bold. The solution methods are ordered from left to right, top to bottom, according to the best total average value over all instances

	SR-IE-16			DHS-SA-16			CF-SA-16			CF-IE-64		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	**816,00**	**836,18**	**16,29**	**822,40**	**840,90**	**13,42**	**819,20**	**838,64**	**16,11**	**815,20**	**845,02**	**22,33**
testdata 1	684,40	707,24	15,70	680,00	709,24	26,65	695,20	706,24	8,40	676,80	713,32	23,29
testdata 2	1188,60	1220,32	18,95	1191,60	1213,34	17,56	1189,40	1221,84	17,58	1207,00	1222,60	16,48
testdata 3	816,20	838,14	16,49	823,60	847,28	15,12	806,60	841,90	19,94	813,60	839,66	12,03
testdata 4	1266,80	1309,24	29,57	1274,60	1310,40	22,09	1280,20	1317,28	29,99	1281,00	1319,86	28,67
testdata 5	642,40	647,92	5,50	640,80	648,08	5,08	642,40	647,92	3,85	646,40	649,84	1,92
testdata 6	832,40	860,44	21,33	835,80	855,56	14,11	828,40	855,22	21,19	840,20	852,20	12,93
	SR-IE-4			DHS-IE-16			DHS-SA-64			CF-SA-64		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	**817,60**	**843,32**	**17,14**	**818,40**	**841,98**	**18,41**	**832,20**	**842,44**	**7,09**	**828,00**	**841,34**	**12,87**
testdata 1	679,60	708,26	18,09	691,60	710,04	14,11	688,40	718,14	20,57	691,60	713,76	16,64
testdata 2	1193,80	1223,56	13,72	1192,60	1230,56	25,74	1216,00	1235,42	17,75	1222,20	1237,62	13,82
testdata 3	824,60	847,48	15,95	809,40	838,40	16,51	817,60	841,94	18,49	829,20	848,32	14,29
testdata 4	1273,40	1320,14	40,41	1290,80	1315,70	24,22	1302,60	1328,32	21,21	1276,40	1324,08	36,30
testdata 5	640,80	646,72	3,72	642,40	652,08	7,13	644,80	650,88	5,92	644,00	649,84	3,27
testdata 6	827,80	856,90	19,31	840,40	863,86	18,07	839,20	851,48	7,12	845,00	857,16	6,14

Table 4 (*Continued*)

	DHS-IE-64			CF-SA-4			DHS-SA-4			CF-OI-16		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	**820,00**	**834,16**	**10,13**	827,20	847,26	20,17	831,80	854,12	23,41	1181,00	1224,46	34,80
testdata 1	831,40	841,84	11,79	676,00	703,64	16,12	692,40	716,66	17,67	902,40	946,20	27,19
testdata 2	1202,80	1222,22	19,30	1201,80	1222,46	17,02	1205,40	1227,90	13,47	1540,60	1604,18	53,17
testdata 3	802,40	828,88	12,98	816,80	835,82	15,15	801,80	837,94	24,80	1147,40	1197,70	32,81
testdata 4	1271,00	1320,14	32,93	1282,20	1329,08	32,63	1283,20	1312,54	23,74	1629,60	1704,42	55,61
testdata 5	644,80	648,80	2,82	638,40	646,72	5,27	640,80	649,70	8,74	742,80	766,10	18,44
testdata 6	837,20	854,78	11,55	834,60	853,88	15,21	842,40	859,94	17,59	1062,00	1122,02	43,88
	CF-OI-4			SR-OI-16			SR-OI-4			DHS-OI-4		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	1152,80	1228,62	47,29	1150,00	1230,86	55,74	1165,80	1213,64	39,11	1163,60	1217,96	48,33
testdata 1	872,20	949,72	36,44	920,80	974,92	41,08	867,00	956,92	52,20	884,40	959,38	51,25
testdata 2	1554,60	1602,90	37,75	1565,00	1623,08	33,88	1543,80	1612,50	32,52	1584,80	1622,82	39,90
testdata 3	1148,60	1209,36	32,01	1154,80	1216,36	47,82	1150,00	1223,58	51,89	1126,00	1185,10	35,00
testdata 4	1544,40	1710,90	69,08	1649,20	1715,04	53,38	1636,60	1727,08	58,82	1633,80	1749,80	67,58
testdata 5	753,20	767,14	12,44	726,60	766,98	27,32	739,60	770,74	20,07	749,40	771,44	16,87
testdata 6	1075,20	1124,72	46,19	1052,20	1106,58	29,68	1068,00	1130,26	43,31	1073,60	1137,34	43,89

Table 5 Last set of results for the patient admission scheduling problem. Results that are significantly better are marked in bold. The solution methods are ordered from left to right, top to bottom, according to the best total average value over all instances

	SR-OI-64			DHS-OI-16		
	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	1174,80	1223,76	30,10	1135,40	1220,98	49,04
testdata 1	937,00	987,12	35,26	888,20	982,72	60,02
testdata 2	1583,60	1613,54	24,00	1519,80	1606,82	53,26
testdata 3	1154,00	1208,54	41,85	1099,00	1204,88	56,08
testdata 4	1595,40	1716,48	53,72	1642,80	1744,38	75,77
testdata 5	739,20	768,16	19,18	746,40	768,56	17,70
testdata 6	1090,80	1129,90	39,77	1108,20	1138,94	20,33

	CF-OI-64			DHS-OI-64		
	min.	avg.	st. dev.	min.	avg.	st. dev.
testdata 0	1140,20	1243,52	64,21	1179,00	1219,30	22,12
testdata 1	892,00	951,16	36,27	946,60	992,80	29,26
testdata 2	1532,20	1632,74	64,24	1550,80	1632,28	56,85
testdata 3	1159,60	1198,52	31,33	1143,40	1209,32	41,64
testdata 4	1660,40	1751,72	53,03	1664,80	1733,82	50,43
testdata 5	743,20	771,44	19,10	754,00	788,04	21,43
testdata 6	1063,20	1122,46	38,08	1068,80	1118,62	33,50

Table 6 ILP solutions for the nurse rostering problem dataset. Values indicated in bold are optimal values

Medium					Medium_hint			Medium_late				
1	2	3	4	5	1	2	3	1	2	3	4	5
240	**240**	**236**	**237**	**303**	84	119	14750	179	59	32	48	147

Sprint										Sprint_hint		
1	2	3	4	5	6	7	8	9	10	1	2	3
56	**58**	**51**	**59**	**58**	**54**	**56**	**56**	**55**	**52**	**74**	**43**	63

Sprint_late									
1	2	3	4	5	6	7	8	9	10
39	43	54	124	45	**42**	**42**	27	28	**43**

5.3.2 Nurse rostering

We have tackled the nurse rostering datasets first with an ILP solver (CPLEX). The computations were interrupted as soon as the available time was consumed. In some occasions, that was before the optimal solution was found. The results obtained with the ILP solver are presented in Table 6. Values indicated in bold are optimal solutions.

We applied the same hyper-heuristic approach as for patient admission scheduling to these datasets. The eight best performing hyper-heuristics[3] are presented in Tables 7 and 8. In contrast to the patient admission scheduling problem, there is no hyper-heuristic that is in the best performing group for all data instances. The hyper-heuristic that is ranked first (CF-GD-4) is in the best performing group for 30 out of 36 instances. The predominance of the great deluge based hyper-heuristics is not as considerable as in the patient admission scheduling experiments. Again, the hyper-heuristics based on the 'only improving' move acceptance criterion generate the poorest results. There is again no statical evidence that one of heuristic selection mechanisms outperforms the others.

The experiments confirm the good performance of the hyper-heuristic approach, when comparing to the solutions obtained by the ILP method. No other results have been generated so far for these instances.

Bilgin et al. (2007) concluded, after numerous experiments on benchmarks with different combinations of move acceptance criteria and heuristic selection methods, that not any combination of acceptance criteria and selection methods dominates any other combination on all benchmarks. In their experiments, improving equal resulted in the best average performance, while choice function was on average slightly better than the other heuristic selection mechanisms. Our experiments show that the great deluge move acceptance criterion with tournament size 4 results in the best average performance. However, the heuristic selection mechanism is different on both occasions.

Although simple random is the easiest heuristic selection mechanism, it does not perform worse than the more 'intelligent' selection mechanisms. It is therefore always a good idea to compare, as a first test, a newly developed heuristic selection method with simple random.

6 Conclusion

We have introduced a general hyper-heuristic approach for solving two operations research problems in health care. The main challenges in hyper-heuristic research, namely generalising the applicability and generating synergy between existing low-level heuristics, have been pursued. Regarding the patient admission scheduling problem, we also provide new benchmark instances and a validation tool that enables comparing algorithm performance. Other researchers are invited to address the problem instances and we will keep track of the best solutions on the Patient Admission Scheduling website (Demeester et al. 2008). Within the problem settings of this paper, it is not possible to largely increase the bed occupancy rate in the hospital. That is due to the assumption of a fixed patient list and fixed patient stays over the considered time horizon. Assigning patients as much as possible to the room type of their choice increases their satisfaction during their stay. As a consequence, it may also increase the income of the hospital, since a higher fee can be charged for treating

[3]The results for all 36 hyper-heuristic variants can be found at http://allserv.kahosl.be/~peter/pas/JOH/results-nrc.xls.

Table 7 First set of results for the nurse rostering problem. Results that are significantly better are marked in bold. The solution methods are ordered from left to right, top to bottom, according to the best total average value over all instances

	CF-GD-4			SR-GD-4			SR-SA-4			DHS-SA-4		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
medium01	242,00	242,20	0,42	242,00	242,40	0,52	242,00	242,90	0,74	242,00	243,00	0,67
medium02	240,00	241,00	0,47	**240,00**	**240,80**	**0,42**	240,00	241,60	0,84	241,00	241,20	0,42
medium03	**238,00**	**238,50**	**0,53**	238,00	238,60	0,52	**237,00**	**238,60**	**0,84**	**237,00**	**238,50**	**0,71**
medium04	**238,00**	**238,40**	**0,52**	**238,00**	**238,50**	**0,53**	**238,00**	**238,70**	**0,67**	**238,00**	**238,70**	**0,67**
medium05	**303,00**	**303,30**	**0,48**	303,00	303,50	0,53	303,00	304,20	0,63	303,00	304,10	1,20
medium_hint01	**40,00**	**42,50**	**2,12**	41,00	43,90	2,47	**40,00**	**41,80**	**1,03**	**39,00**	**42,22**	**2,17**
medium_hint02	**93,00**	**97,70**	**2,95**	**94,00**	**99,90**	**5,24**	98,00	106,80	10,09	89,00	105,00	8,47
medium_hint03	**142,00**	**153,80**	**7,00**	**149,00**	**156,70**	**5,58**	**143,00**	**151,50**	**6,64**	**141,00**	**156,33**	**11,05**
medium_late01	**178,00**	**181,30**	**2,71**	**171,00**	**182,00**	**5,31**	**171,00**	**179,50**	**5,97**	**173,00**	**179,50**	**4,06**
medium_late02	**23,00**	**25,90**	**1,73**	24,00	27,60	1,96	**23,00**	**25,00**	**1,05**	**23,00**	**25,80**	**1,81**
medium_late03	32,00	36,70	2,21	**32,00**	**35,20**	**2,04**	**32,00**	**34,40**	**1,35**	**32,00**	**34,20**	**1,40**
medium_late04	**38,00**	**40,50**	**1,72**	39,00	40,90	1,79	**38,00**	**39,70**	**0,95**	**37,00**	**39,00**	**1,33**
medium_late05	**138,00**	**147,70**	**5,60**	**137,00**	**146,70**	**5,74**	**142,00**	**148,90**	**5,02**	**139,00**	**152,10**	**9,92**
sprint01	56,00	56,60	0,52	56,00	56,70	0,67	56,00	56,70	0,48	56,00	57,00	0,94
sprint02	**58,00**	**58,70**	**0,48**	**58,00**	**58,50**	**0,53**	**58,00**	**58,90**	**0,57**	**58,00**	**58,70**	**0,67**
sprint03	**51,00**	**51,80**	**0,92**	**51,00**	**52,40**	**0,97**	**51,00**	**52,30**	**0,82**	52,00	52,30	0,48
sprint04	**59,00**	**60,10**	**0,99**	**59,00**	**59,40**	**0,52**	60,00	60,60	0,52	59,00	60,80	0,92
sprint05	**58,00**	**58,10**	**0,32**	**58,00**	**58,10**	**0,32**	**58,00**	**58,30**	**0,48**	**58,00**	**58,20**	**0,42**

Table 7 (*Continued*)

	CF-GD-4			SR-GD-4			SR-SA-4			DHS-SA-4		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
sprint06	**54,00**	**54,30**	**0,48**	**54,00**	**54,50**	**0,53**	54,00	54,80	0,42	**54,00**	**54,40**	**0,52**
sprint07	**56,00**	**56,60**	**0,70**	56,00	56,60	0,52	56,00	57,20	0,63	56,00	56,80	0,79
sprint08	**56,00**	**56,60**	**0,52**	**56,00**	**56,30**	**0,48**	56,00	56,80	0,42	57,00	57,00	0,00
sprint09	**55,00**	**55,80**	**0,63**	**55,00**	**55,80**	**1,03**	55,00	56,10	1,10	55,00	56,00	0,67
sprint10	**52,00**	**52,70**	**0,67**	**52,00**	**52,60**	**0,70**	**52,00**	**52,80**	**0,63**	**52,00**	**52,90**	**0,57**
sprint_hint01	**81,00**	**88,40**	**6,80**	**81,00**	**89,50**	**6,08**	**77,00**	**87,90**	**7,46**	**79,00**	**86,50**	**3,92**
sprint_hint02	**45,00**	**52,60**	**6,02**	**46,00**	**52,10**	**3,38**	**45,00**	**51,40**	**5,13**	**47,00**	**50,30**	**2,21**
sprint_hint03	**57,00**	**68,80**	**8,52**	63,00	68,80	4,73	**59,00**	**63,70**	**3,53**	**60,00**	**66,10**	**5,40**
sprint_late01	**40,00**	**42,10**	**1,66**	**40,00**	**41,60**	**1,07**	**39,00**	**41,10**	**1,60**	**40,00**	**41,50**	**1,18**
sprint_late02	45,00	47,40	1,43	**43,00**	**45,40**	**1,71**	**45,00**	**45,60**	**0,84**	**43,00**	**45,00**	**1,50**
sprint_late03	**50,00**	**51,60**	**1,17**	**49,00**	**51,20**	**1,75**	**50,00**	**51,70**	**1,49**	**50,00**	**52,30**	**1,25**
sprint_late04	**86,00**	**94,60**	**7,92**	**84,00**	**92,40**	**6,60**	**87,00**	**92,20**	**5,05**	**82,00**	**91,20**	**4,42**
sprint_late05	**45,00**	**47,50**	**1,58**	46,00	48,20	1,40	**46,00**	**47,50**	**1,27**	**45,00**	**47,00**	**1,56**
sprint_late06	**43,00**	**43,20**	**0,42**	43,00	43,40	0,52	**42,00**	**43,10**	**0,57**	**43,00**	**43,10**	**0,32**
sprint_late07	**45,00**	**51,60**	**3,81**	**46,00**	**52,70**	**3,86**	**45,00**	**51,30**	**6,38**	46,00	53,20	4,26
sprint_late08	**17,00**	**21,00**	**5,14**	**17,00**	**19,60**	**3,50**	17,00	21,70	4,22	17,00	22,44	3,84
sprint_late09	17,00	22,20	3,26	**17,00**	**21,70**	**4,74**	17,00	22,00	4,03	21,00	24,40	4,14
sprint_late10	**48,00**	**52,50**	**5,42**	**48,00**	**53,90**	**4,48**	**48,00**	**52,90**	**4,43**	**45,00**	**53,00**	**5,08**

Table 8 Second set of results for the nurse rostering problem. Results that are significantly better are marked in bold. The solution methods are ordered from left to right, top to bottom, according to the best total average value over all instances

	CF-SA-4			DHS-GD-4			SR-SA-16			SR-IE-4		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
medium01	242,00	243,40	0,70	**240,00**	**241,30**	**0,67**	242,00	243,00	0,67	241,00	242,30	0,95
medium02	241,00	241,80	0,63	**240,00**	**240,80**	**0,63**	241,00	241,60	0,52	**240,00**	**240,50**	**0,53**
medium03	238,00	239,00	0,47	**236,00**	**238,10**	**0,99**	**237,00**	**238,60**	**0,84**	**237,00**	**238,20**	**0,63**
medium04	**238,00**	**239,10**	**0,99**	**238,00**	**238,10**	**0,32**	**238,00**	**239,10**	**0,99**	**238,00**	**238,70**	**0,82**
medium05	303,00	304,30	0,67	**303,00**	**303,20**	**0,42**	304,00	304,90	0,57	**303,00**	**303,40**	**0,70**
medium_hint01	**39,00**	**41,40**	**2,01**	42,00	44,50	2,37	**38,00**	**43,30**	**3,16**	41,00	48,50	3,72
medium_hint02	**91,00**	**101,20**	**8,48**	**93,00**	**101,40**	**6,93**	99,00	113,30	9,01	116,00	128,80	7,07
medium_hint03	**144,00**	**152,60**	**7,89**	**143,00**	**156,60**	**7,62**	**144,00**	**170,40**	**39,97**	152,00	163,50	7,75
medium_late01	**173,00**	**178,80**	**3,26**	**177,00**	**181,70**	**7,59**	175,00	188,30	6,77	182,00	197,10	6,67
medium_late02	**23,00**	**25,30**	**2,26**	26,00	28,10	2,18	**22,00**	**26,50**	**2,55**	27,00	34,90	4,41
medium_late03	**33,00**	**34,50**	**1,35**	**32,00**	**35,70**	**2,41**	**32,00**	**35,40**	**2,59**	34,00	38,40	3,41
medium_late04	**37,00**	**39,40**	**1,51**	38,00	41,10	2,02	**38,00**	**39,70**	**1,42**	41,00	44,00	2,16
medium_late05	**140,00**	**150,20**	**6,23**	**141,00**	**149,80**	**4,54**	151,00	159,60	6,11	155,00	171,90	11,82
sprint01	56,00	56,60	0,70	56,00	57,10	0,99	56,00	56,90	0,57	56,00	56,50	0,71
sprint02	**58,00**	**58,80**	**0,92**	**58,00**	**58,70**	**0,67**	**58,00**	**58,50**	**0,53**	**58,00**	**58,50**	**0,53**
sprint03	**51,00**	**52,30**	**0,82**	51,00	52,50	0,71	**51,00**	**52,10**	**0,99**	**51,00**	**51,60**	**0,84**
sprint04	59,00	60,30	0,82	59,00	60,40	0,84	59,00	60,60	1,07	**59,00**	**60,10**	**0,88**
sprint05	**58,00**	**58,20**	**0,42**	58,00	58,00	0,00	58,00	58,00	0,00	58,00	58,00	0,00

Table 8 (*Continued*)

	CF-SA-4			DHS-GD-4			SR-SA-16			SR-IE-4		
	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
sprint06	**54,00**	**54,50**	**0,53**	**54,00**	**54,40**	**0,52**	**54,00**	**54,40**	**0,52**	**54,00**	**54,30**	**0,48**
sprint07	56,00	56,70	0,67	56,00	57,20	1,03	56,00	57,00	0,82	**56,00**	**56,30**	**0,48**
sprint08	56,00	56,80	0,63	56,00	56,80	0,42	**56,00**	**56,30**	**0,48**	**56,00**	**56,20**	**0,42**
sprint09	**55,00**	**55,90**	**0,88**	**55,00**	**55,70**	**0,95**	**55,00**	**55,70**	**0,48**	**55,00**	**55,30**	**0,48**
sprint10	52,00	53,40	0,70	**52,00**	**52,90**	**0,74**	**52,00**	**52,90**	**0,74**	**52,00**	**52,10**	**0,32**
sprint_hint01	**75,00**	**87,90**	**6,49**	**81,00**	**88,90**	**5,04**	**80,00**	**88,50**	**5,48**	**78,00**	**89,30**	**5,31**
sprint_hint02	48,00	53,70	3,13	47,00	54,80	4,59	49,00	54,40	4,48	49,00	54,50	6,65
sprint_hint03	62,00	69,10	5,45	62,00	71,00	6,62	61,00	69,50	7,09	59,00	69,70	6,82
sprint_late01	**41,00**	**41,40**	**0,70**	**40,00**	**42,50**	**1,43**	**38,00**	**41,70**	**1,95**	**40,00**	**42,30**	**2,36**
sprint_late02	**44,00**	**45,60**	**1,07**	**42,00**	**44,90**	**1,60**	**44,00**	**45,50**	**1,08**	**44,00**	**45,60**	**1,51**
sprint_late03	**48,00**	**51,60**	**1,84**	**50,00**	**52,40**	**1,65**	**49,00**	**51,20**	**1,69**	**49,00**	**52,10**	**2,08**
sprint_late04	**77,00**	**90,40**	**7,65**	**88,00**	**94,30**	**4,64**	**85,00**	**94,80**	**8,05**	**83,00**	**91,00**	**5,23**
sprint_late05	**45,00**	**47,10**	**1,45**	**45,00**	**47,10**	**0,99**	**45,00**	**46,90**	**1,20**	**45,00**	**46,70**	**1,34**
sprint_late06	43,00	43,40	0,52	43,00	43,50	0,71	**42,00**	**43,20**	**0,79**	**42,00**	**42,80**	**0,63**
sprint_late07	**46,00**	**49,70**	**2,31**	47,00	55,40	7,55	48,00	53,50	4,20	49,00	58,10	5,17
sprint_late08	17,00	22,70	4,95	17,00	20,10	2,13	17,00	23,60	4,33	**17,00**	**21,30**	**5,62**
sprint_late09	17,00	21,50	2,72	**17,00**	**20,10**	**4,09**	17,00	24,50	4,93	17,00	20,40	2,32
sprint_late10	**47,00**	**53,40**	**4,09**	50,00	56,11	5,75	**48,00**	**54,30**	**4,50**	**49,00**	**54,50**	**6,65**

patients in a single room. The potential of the presented approach is to serve as an instrument for optimizing patient assignments each time new information becomes available (e.g. adjustment of the patient stay duration, additional patients on the list, canceled treatments, etc.) in a real world setting. It serves as an aid for the human admission scheduler to help him/her quickly select a room for each patient. Although not discussed in the paper, provisions have been made for locking assignments of patients after admission.

Directions for future research on patient admission scheduling include (1) expanding the model for intensive care and day clinic units, adding extra real world constraints such as age compatibility, quarantine, and workload balance of the health care professionals, (2) developing new stochastic benchmark instances taking into account patients on waiting lists, (3) rescheduling in case of small changes to the data, and (4) tackling both problems as one integrated problem with the same hyper-heuristic. The latter is based on the following reflection. The number of nurses needed per shift (the coverage) depends on the occupancy of the beds. The more patients that are assigned to beds, the more nurses will be needed. Both problems are strongly intertwined.

The hyper-heuristic approach to the patient admission scheduling problem significantly outperforms a previously published tabu search algorithm.

Concerning the nurse rostering problem, the same hyper-heuristic approach generates results that are close to the solutions obtained by an ILP approach, within the same amount of computation time. The only preconditions to applying the hyper-heuristics are the availability of a solution representation, a constructor for the initial solution and a set of low-level heuristics. In a short software development cycle, competitive results for the nurse rostering problem could be obtained. This paper shows that it is possible to generate solutions in a 'good enough-soon enough-cheap enough' manner with this hyper-heuristic.

We advocate applying existing local search neighbourhoods to serve as low-level heuristics within a hyper-heuristic framework for general application. The implementation effort is limited compared to the experienced performance increase for the patient admission scheduling problem. As we have shown, the effect is not limited to one single problem, and one can expect that this conclusion will also hold for other problems.

Acknowledgements Research partly funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT).

Appendix: Mathematical model

The ILP model consists of the objective function to be minimised, the hard and soft constraints, the decision variables and the auxiliary variables. Let N be the set of all nurses, D the set of all days in the schedule period, and S the set of all shift types.

The Decision Variables

$$x_{n,d,s} = \begin{cases} 1 & \text{if nurse } n \text{ is assigned on day } d \text{ and shift type } s, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

The Auxiliary Variables Let $|S|$ be the number of shift types.

$$-|S|p_{n,d} + \sum_j x_{n,d,j} \leq 0, \tag{9}$$

$$-p_{n,d} + \sum_j x_{n,d,j} \geq 0. \tag{10}$$

Equations 9 and 10 imply Eq. 11.

$$p_{n,d} = \begin{cases} 1 & \text{if nurse } n \text{ is assigned on day } d, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Hard Constraints

– Coverage constraint. Let $c_{d,s}$ be the number of assignments required on day d and shift type s.

$$\forall d \in D, \forall s \in S, \quad \sum_n x_{n,d,s} = c_{d,s}. \tag{12}$$

– Single assignment per day.

$$\forall n \in N, \forall d \in D \quad \sum_s x_{n,d,s} \leq 1. \tag{13}$$

Soft Constraints

– Maximum days worked. Let M be the constraint parameter, the maximum number of days a nurse is allowed to work.
  v_n denotes the number of violations to this constraint for nurse n.

$$\forall n \in N, \quad v_n \geq \sum_d \sum_s x_{n,d,s} - M. \tag{14}$$

  The total number of violations to this constraint is calculated with Eq. 15.

$$TMax = \sum_n v_n. \tag{15}$$

– Minimum days worked. Let M be the constraint parameter, the minimum number of days a nurse is allowed to work.
  v_n denotes the number of violations to this constraint for nurse n.

$$\forall n \in N, \quad -v_n + \sum_d \sum_s x_{n,d,s} \geq M. \tag{16}$$

  The total number of violations to this constraint is calculated with Eq. 17.

$$TMin = \sum_n v_n. \tag{17}$$

– Unwanted Patterns
Let $N = \{n \mid \text{days where the pattern entry is "}None\text{"}\}$.
Let $A = \{a \mid \text{days where the pattern entry is "}Any\text{"}\}$.
Let $S = \{s \mid \text{shift types where a pattern entry is defined}\}$.
Let $D_s = \{d \mid \text{days where a pattern entry is the shift type with the index } s\}$.

$$v + \sum_{n \in N} p_{e,n} - \sum_{a \in A} p_{e,a} - \sum_{s \in S} \sum_{d \in D_s} x_{e,d,s} \geq 1 - |A| - \sum_{s \in S} |D_s|. \tag{18}$$

Equation 18 implies Eq. 19

$$v = \begin{cases} 1 & \text{if the assignment sequence satisfies the pattern,} \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

– Complete identical weekends. Let W be the set of all Saturdays in the schedule period.
$v_{n,d,s,1}$ and $v_{n,d,s,2}$ denote the number of violations to this constraint for nurse n, the weekend that starts with day d, and shift type s.

$$\forall n \in N, \forall d \in W, \forall s \in S, \quad v_{n,d,s,1} - x_{n,d,s} + x_{n,d+1,s} \geq 0, \tag{20}$$

$$\forall n \in N, \forall d \in W, \forall s \in S, \quad v_{n,d,s,2} + x_{n,d,s} - x_{n,d+1,s} \geq 0. \tag{21}$$

The total number of violations to this constraint is calculated with Eq. 22.

$$CIW = \sum_{n} \sum_{d} \sum_{s} v_{n,d,s,1} + v_{n,d,s,2}. \tag{22}$$

– The succession of shift types. Let S' be the set of shift type pairs (s_k, s_l) such that s_l is not allowed to be assigned the day after s_k is assigned.
Let $D' = \{d \mid d \geq 1 \wedge d < |D|\}$.
$v_{n,d,(s_k,s_l)}$ denotes the number of violations to this constraint for nurse n, day d, and shift type succession (s_k, s_l).

$$\forall n \in N, \forall d \in D', \forall (s_k, s_l) \in S', \quad -v_{n,d,(s_k,s_l)} + x_{n,d,s_k} + x_{n,d+1,s_l} \leq 1. \tag{23}$$

The total number of violations to this constraint is calculated with Eq. 24.

$$SNA = \sum_{n} \sum_{d} \sum_{(s_k,s_l)} v_{n,d,(s_k,s_l)}. \tag{24}$$

– Maximum consecutive working days. Let M be the constraint parameter, the maximum number of consecutive working days for a nurse.
Let $D' = \{d \mid d \geq 1 \wedge d \leq |D| - M\}$.
$v_{n,d}$ denotes the number of violations to this constraint for nurse n, and the day series that starts at day d.

$$\forall n \in N, \forall d \in D', \quad -v_{n,d} + \sum_{k=0}^{M} p_{n,d+k} \leq M. \tag{25}$$

The total number of violations to this constraint is calculated with Eq. 26.

$$MaxCon = \sum_{n}\sum_{d} v_{n,d}. \tag{26}$$

– Minimum consecutive working days. Let $|M|$ be the constraint parameter, the minimum number of consecutive working days for a nurse.
Let $D' = \{d \mid d \geq 1 \wedge d \leq |D| - M\}$.
Let $L = \{l|l \geq 2 \wedge l \leq M\}$.
$v_{n,d,l}$ denotes the number of violations to this constraint for nurse n, and the period between days d and $d + l$. $v_{n,0,l}$ denotes the number of violations to this constraint for nurse n, and the period between schedule period start and day l.

$$\forall n \in N, \forall d \in D', \forall l \in L, \quad -\frac{v_{n,d,l}}{M - l + 1} + p_{n,d} - p_{n,d+1} + p_{n,d+l} \geq 0, \tag{27}$$

$$\forall n \in N, \forall l \in L, \quad -\frac{v_{n,0,l}}{M - l + 1} - p_{n,1} + p_{n,l} \geq 0. \tag{28}$$

The total number of violations to this constraint is calculated with Eq. 29.

$$MinCon = \sum_{n}\sum_{l} v_{n,0,l} + \sum_{n}\sum_{d}\sum_{l} v_{n,d,l}. \tag{29}$$

– Maximum consecutive free days. Let $|M|$ be the constraint parameter, the maximum number of consecutive free days for a nurse.
Let $D' = \{d|d \geq 1 \wedge d \leq |D| - M\}$.
$v_{n,d}$ denotes the number of violations to this constraint for nurse n, and the day series that starts at day d.

$$\forall n \in N, \forall d \in D', \quad v_{n,d} + \sum_{j=0}^{M}\sum_{s} x_{n,d+j,s} \geq 1. \tag{30}$$

The total number of violations to this constraint is calculated with Eq. 31.

$$MaxFree = \sum_{n}\sum_{d} v_{n,d}. \tag{31}$$

– Minimum consecutive free days.
Let $|M|$ be the constraint parameter, the minimum number of consecutive free days for a nurse.
Let $D = \{d|d \geq 1 \wedge d \leq |D| - m\}$.
Let $K = \{k|k \geq 2 \wedge k \leq M\}$.
$v_{n,d,k}$ denotes the number of violations to this constraint for nurse n, and the period between days d and $d + k$. $v_{n,0,k}$ denotes the number of violations to this constraint for nurse n, and the period between schedule period start and day k.

$$\forall n \in N, \forall d \in D, \forall k \in K, \quad v_{n,d,k} - p_{n,d} + p_{n,d+1} - p_{n,d+k} \geq -1, \tag{32}$$

$$\forall n \in N, \forall k \in K, \quad v_{n,0,k} + p_{n,1} - p_{n,k} \geq 0. \tag{33}$$

The total number of violations to this constraint is calculated with Eq. 34.

$$MinFree = \sum_{n}\sum_{k} v_{n,0,k} + \sum_{n}\sum_{d}\sum_{k} v_{n,d,k}. \tag{34}$$

The Objective Function The objective of the optimization is to minimise F, which equals to the number of all violations of the soft constraints (Eq. 35).

$$F = TMax + TMin + CIW + SNA + MaxCon + MinCon + MaxFree + MinFree + SAD. \tag{35}$$

References

Ayob, M., Kendall, G.: A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In: The International Conference on Intelligent Technologies, InTech'03 (2003)

Bilgin, B., Özcan, E., Korkmaz, E.E.: An experimental study on hyper-heuristics and exam timetabling. In: Burke, E.K., Rudová, H. (eds.) Revised Selected Papers of 6th International Conference on Practice and Theory of Automated Timetabling VI (Patat 2006). LNCS, vol. 3867, pp. 394–412. Springer, Berlin (2007)

Bilgin, B., De Causmaecker, P., Vanden Berghe, G.: A hyperheuristic approach to belgian nurse rostering problems. In: Proceedings of the 4th Multidisciplinary International Conference on Scheduling: Theory and Applications, pp. 693–695 (2009)

Brucker, P., Burke, E.K., Curtois, T., Qu, R., Vanden Berghe, G.: A shift sequence based approach for nurse scheduling and a new benchmark dataset. J. Heuristics **16**(4), 559–573 (2010)

Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern search technology. In: Handbook of Metaheuristics, pp. 457–474. Springer, New York (2003)

Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. J. Heuristics **9**(6), 451–470 (2003)

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

Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A classification of hyper-heuristic approaches. In: Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 146, pp. 449–468. Springer, Berlin (2010)

Burke, E.K., Curtois, T., Qu, R., Vanden Berghe, G.: A scatter search approach to the nurse rostering problem. J. Oper. Res. Soc. **61**, 1667–1679 (2010)

Chien, C.-F., Tseng, F.-P., Chen, C.-H.: An evolutionary approach to rehabilitation patient scheduling: A case study. Eur. J. Oper. Res. **189**, 1234–1253 (2008)

Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E.K., Erben, W. (eds.) Proceedings of the 3rd International Conference on Practice and Theory of Automated Timetabling. LNCS, vol. 2079, pp. 176–190. Springer, Berlin (2001)

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

Demeester, P., Bilgin, B., Vanden Berghe, G.: Patient admission scheduling benchmark instances. http://allserv.kahosl.be/~peter/pas/index.html (2008)

Demeester, P., Souffriau, W., De Causmaecker, P., Vanden Berghe, G.: A hybrid tabu search algorithm for automatically assigning patients to beds. Artif. Intell. Med. **48**(1), 61–70 (2010)

Dowsland, K.A., Soubeiga, E., Burke, E.K.: A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. Eur. J. Oper. Res. **179**(3), 759–774 (2007)

Dueck, G.: New optimisation heuristics. the great deluge algorithm and record-to-record travel. J. Comput. Phys. **104**, 86–92 (1993)

Gemmel, P., Van Dierdonck, R.: Admission scheduling in acute care hospitals: does the practice fit with the theory? Int. J. Oper. Prod. Manag. **19**(9), 863–878 (1999)

Hans, E.W., Wullink, G., van Houdenhoven, M., Kazemier, G.: Robust surgery loading. Eur. J. Oper. Res. **185**, 1038–1050 (2008)

Haspeslagh, S., De Causmaecker, P., Stølevik, M., Schaerf, A.: First international nurse rostering competition 2010. Technical report, K.U. Leuven, CODeS (May 2010)

Hutzschenreuter, A.K., Bosman, P.A.N., Blonk-Altena, I., van Aarle, J., La Poutré, J.A.: Agent-based patient admission scheduling in hospitals. In: Mueller, J.P., Padgham, L., Parkes, D.C., Parsons, S. (eds.) Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems—AAMAS-2008, pp. 45–54. ACM, New York (2008)

Kendall, G., Mohamad, M.: Channel assignment in cellular communication using a great deluge hyperheuristic. In: Proc. of the 2004 IEEE International Conference on Network (ICON2004) (2004)

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Sci. New Ser. **220**(4598), 671–680 (1983)

Marinagi, C.C., Spyropoulos, C.D., Papatheodorou, C., Kokkotos, S.: Continual planning and scheduling for managing patient tests in hospital laboratories. Artif. Intell. Med. **20**, 139–154 (2000)

Misir, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G.: Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC'10), Barcelona, Spain, July 18–23 (2010)

Ogulata, S.N., Erol, R.: A hierarchical multiple criteria mathematical programming approach for scheduling general surgery operations in large hospitals. J. Med. Syst. **27**(3), 259–270 (2003)

Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intell. Data Anal. **12**(1), 3–23 (2008)

Smith-Daniels, V.L., Schweikhart, S.B., Smith-Daniels, D.E.: Capacity management in health care services: Review and future research directions. Decis. Sci. **19**(4), 889–919 (1988)

Soubeiga, E.: Development and application of hyperheuristics to personnel scheduling. Ph.D. thesis, University of Nottingham (2003)

Vermeulen, I.B., Bohte, S.M., Somefun, D.J.A., La Poutré, J.A.: Multi-agent Pareto appointment exchanging in hospital patient scheduling. Serv. Oriented Comput. Appl. **1**(3), 185–196 (2007)

Vermeulen, I.B., Bohte, S.M., Elkhuizen, S.G., Bakker, P.J.M., La Poutré, J.A.: Decentralized online scheduling of combination-appointments in hospitals. In: Proc. of the International Conference on Automated Planning and Scheduling. ACM, New York (2008)

Vermeulen, I.B., Bohte, S.M., Elkhuizen, S.G., Lameris, H., Bakker, P.J.M., La Poutré, J.A.: Adaptive resource allocation for efficient patient scheduling. Artif. Intell. Med. **46**, 67–80 (2009)