Artificial Intelligence in Medicine 48 (2010) 61–70

Contents lists available at ScienceDirect

Artificial Intelligence in Medicine

journal homepage: www.elsevier.com/locate/aiim

# A hybrid tabu search algorithm for automatically assigning patients to beds

Peter Demeester [a,*], Wouter Souffriau [a], Patrick De Causmaecker [b], Greet Vanden Berghe [a]

[a] KaHo Sint-Lieven, Information Technology, Gebroeders Desmetstraat 1, 9000 Gent, Belgium
[b] Katholieke Universiteit Leuven Campus Kortrijk, Computer Science and Information Technology, Etienne Sabbelaan 53, 8500 Kortrijk, Belgium

ARTICLE INFO

*Article history:*
Received 13 August 2008
Received in revised form 21 August 2009
Accepted 12 September 2009

*Keywords:*
Decision support
Admission scheduling
Metaheuristic

ABSTRACT

*Objective:* We describe a patient admission scheduling algorithm that supports the operational decisions in a hospital. It involves efficiently assigning patients to beds in the appropriate departments, taking into account the medical needs of the patients as well as their preferences, while keeping the number of patients in the different departments balanced.
*Methods:* Due to the combinatorial complexity of the admission scheduling problem, there is a need for an algorithm that intelligently assists the admission scheduler in taking decisions fast. To this end a hybridized tabu search algorithm is developed to tackle the admission scheduling problem. For testing, we use a randomly generated data set. The performance of the algorithm is compared with an integer programming approach.
*Results and conclusion:* The metaheuristic allows flexible modelling and presents feasible solutions even when disrupted by the user at an early stage in the calculation. The integer programming approach is not able to find a solution in 1 h of calculation time.



## 1. Introduction

We introduce a decision support approach to automatically assign patients to hospital beds based on a tabu search algorithm hybridized with a token-ring approach. This algorithm will assist admission schedulers to carry out the hospital management's decisions at the operational level. It enables scheduling patients more efficiently, meaning that they will be assigned to beds in the appropriate departments, satisfying as much as possible the patients' wishes and all the necessary medical constraints. Meanwhile the algorithm will also try to balance the number of patients between the different departments.

Newly admitted patients need a free bed that satisfies both the personal preferences (single, twin room, or a ward) as well as the medical needs of the patient (availability of oxygen or telemetry in the room) located in the department that is specialized in treating the clinical picture. The assignment of patients to beds is often carried out by a central admission office that individually contacts every appropriate department a few days before the effective admission of the patient. Other hospitals organize the admission of patients without a central admission office, leaving admission responsibility with the departments. In the latter case, a lack of overview of the departments may result in under occupancy. Patients may be refused in one department while free suitable beds are available in another department.

Generally speaking patients can be divided in two groups: inpatients and outpatients. Inpatients spend several days or nights in a hospital, whereas the admission of outpatients is expressed in hours. We will concentrate in this paper on inpatients only. Inpatients can further be divided in three groups: emergency, elective and admitted patients. Emergency patients are hard to schedule, since by definition they have no appointment with the physician and arrive at random. Several papers are devoted to the arrival and assignment of emergency patients (see [1] for an overview). Most hospitals provide some slack beds that can be used for emergency patients. Elective inpatients are waiting for an admission date. This means that an admission office can determine when to admit them. Such patients allow the hospital to improve its occupancy rate as they can be assigned to the most appropriate period. In this paper however, we simplify the problem by assuming that the patients' admission dates are known before. The physician who advised the patient to be admitted to the hospital, diagnosed the patient's disease which is associated with a default (average) length-of-stay.

The available literature on admission scheduling mainly concentrates on more abstract levels than what we consider in this paper. Those papers focus on the decisions of the hospital management at the tactical and strategic level, in order to increase

* Corresponding author at: KaHo Sint-Lieven, Vakgroep Informatietechnologie, Gebr. Desmetstraat 1, 9000 Gent, Belgium. Tel.: +32 9 265 86 10; fax: +32 9 225 62 69.
*E-mail addresses:* Peter.Demeester@kahosl.be (P. Demeester), Wouter.Souffriau@kahosl.be (W. Souffriau), Patrick.DeCausmaecker@kuleuven-kortrijk.be (P. De Causmaecker), Greet.VandenBerghe@kahosl.be (G. Vanden Berghe).

0933-3657/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.artmed.2009.09.001

the hospital's efficiency. We will focus in this paper on the operational level, in which the decisions made by the hospital management are executed at the level of the assignment of patients to specific beds in given days. It is our experience, and this was pointed out by Kusters and Groot [2] as well, that a bed admission support application only stands a chance to get accepted if its decisions are supported by human experts and if interactions with the user are allowed. The admission scheduling algorithm that we describe in this article suggests solutions to the manual admission officer that satisfy the decisions made by the hospital management. Since we focus solely on the operational level, we assume that all admission rules and issues have been resolved by the hospital management beforehand. It is aimed at automatising the work of the admission scheduler, who allocates patients to the most appropriate beds/rooms/departments over their entire hospital stay.

In Section 2, we introduce some problem related terminology. A review of the literature is given in Section 3, while in Section 4 the problem is described formally. In Section 5, the mathematical model for the patient admission scheduling problem is formulated. Integer programming is applied to solve the model. However, it does not generate a feasible solution in reasonable time. To this end, we decided to apply a metaheuristic approach to assign patients to beds, which is described in detail in Section 6. In Section 7, we give future research directions and conclude.

## 2. Terminology

We first introduce some problem specific terminology that we will use throughout the rest of the paper.

- *Admitted patients* are patients that are effectively admitted to the hospital and are assigned to a room and a bed.
- *Planned patients* have an admission date and an expected length-of-stay (LOS), which is determined by the government for every pathology. To reduce the expenses in the public health sector, the government demands the hospitals to treat every pathology within this pre-defined amount of time. Planned patients are not admitted to the hospital yet.
- *Transfer*: Moving a patient from one room to another during his/her stay. We distinguish between planned and unplanned transfers. A planned transfer can, for example, concern a move to intensive care after surgery, and a move back to another department after recovery. An unplanned transfer could be caused by bed or room shortage. Unplanned transfers should be avoided as much as possible.
- *Room*: A room is characterized by a number of physical properties, e.g. the availability of oxygen or telemetry. Not every room is appropriate for every patient: different equipment is required for babies and elderly people, for example. A room may contain one or more beds. Hospitals normally do not assign patients of different gender to the same room at the same time.
- *Specialism*: A hospital department is in general highly specialized in treating one kind of pathology (cardiovascular diseases, oncology, dermatology, etc.). We call this the major specialism of the department. Often the same department is also carrying out other treatments as a minor specialism. Some of the rooms are equipped to accommodate and some of the nurses are trained to treat patients with diseases that correspond to the department's minor specialism. Nurses belonging to a department are trained to care for patients with the specific pathology. This is similar to what the authors of [3] call nursing units. Rooms that belong to the department can be especially equipped for the major specialism or one of the minor specialisms of the department.
- *Night*: We will consider a night as the smallest unit of time. Each length-of-stay is expressed in nights. During the night it is highly uncommon to admit planned patients.

## 3. Literature review

Hospital admission scheduling is well reported on in the research literature. In order to reduce spending in health care, hospitals are compelled to use their resources more efficiently.

Hospital admission scheduling is the process of assigning patients to beds in such a way that the medical concerns and personal wishes are fulfilled as much as possible. This process can be carried out at three different levels: strategic, tactical or operational. According to [4], the operational level applies concrete rules for the admission of patients, which follow the strategic and tactical decisions that are made by the hospital management. A strategic decision could involve maximizing the usage of the resources (MRI scanner, operating theatres, nurses, beds, etc.) in the hospital or minimizing the waiting time of the patients [4,5].

In the first part of this literature review we mention some of the best studied health care examples that are solved using artificial intelligence techniques. Numerous articles can be found that address tactical and strategic levels, but to our knowledge the admission of patients on the operational level is not widely reported on. In the two subsequent sections we discuss the literature on the tactical and strategic levels of admission scheduling and also on hospital capacity planning, since they influence the bed occupancy.

### 3.1. Artificial intelligence in health care

The application of artificial intelligence in health care is widespread. Well-known applications are for example nurse rostering (see [6] for a review), scheduling operating rooms [7], emergency room physicians [8] and outpatient's appointments [9]. These applications range from solving over-simplified exercises of thought to over-constrained real-world problems. Spyropoulos [10] provides an overview of planning and scheduling approaches in the hospital management and therapy planning domain.

Although not directly related to the problem of assigning patients to beds, the problem that is discussed and solved by Hans et al. [11] bears some resemblance with our problem. An algorithm is described which at the same time assigns elective patients to operating rooms, optimizes the operating theatre utilization and minimizes the total overtime of the surgeries. Both constructive and local search heuristics are applied to solve the problem. All the proposed optimization techniques lead to a better utilization of the operating theatres, compared to the schedule that was originally made by the (human) specialists.

Another, somewhat similar problem, is described by Ogulata et al. [12]. It involves generating a weekly work schedule for physiotherapists. Patients are selected from a list and scheduled on a day of the week, according to the priority and the duration of their treatment. They are scheduled in such a manner that the workload of the physiotherapists is equally balanced. Patients' preferences concerning the day of the week for treatment are not taken into consideration. Ogulata et al. solve the problem using the mathematical programming tools GAMS and MPL.

Marinagi et al. [13] describe the patient examination scheduling problem which precedes the problem tackled in this paper. Before a physician can diagnose a patient, several examinations have to be performed, such as X-rays and ultrasound images, blood samples, etc. The goal is to minimize the patient's examination time in the hospital and to maximize the utilization of resources. The problem is solved by a combination of agent technology, a hierarchical planner which supports the decomposition of complex tests into smaller parts and a scheduler. Subject to the different actions that need to be executed, which is the result obtained by the planner, the scheduler tries to assign the actions to appropriate timeslots.

### 3.2. Admission scheduling

Gemmel and Van Dierdonck [5] review the available literature on admission scheduling and conclude that not only the bed utilization should be taken into account, but also information on the availability and use of resources such as operating rooms, nurses, etc. From the reviewed literature they gather two conditions that have to be fulfilled before one can really talk about scheduling admissions, namely

- the scheduler should have access to valid information about the availability of resources;
- the scheduler should have (limited) authority to change the admission dates.

When these conditions are not fulfilled the admission service of the hospital administers patients instead of scheduling them. In the same article, Gemmel and Van Dierdonck survey how 83 Belgian hospitals perceive admission scheduling in practice. It shows a large gap between theory and practice. If the two above conditions are taken into consideration, the survey reveals that only one third of the surveyed Belgian hospitals schedule their admissions. The admission services of the remaining two thirds of the hospitals actually carry out only an administrative function. From the survey it is also clear that most hospitals have no information on the expected length-of-stay or on the expected workload, which is - of course - crucial when scheduling admissions.

In the extensive literature review of Smith-Daniels et al. [1], inpatient admission scheduling is classified as a facility allocation decision, in which it is assumed that the medical services' size already have been defined, and the resources are constrained by the available beds. The maximization of bed occupancy levels depends on three variability sources:

- The admission of emergency patients: in general some slack capacity is available to admit emergency patients.
- The length-of-stay (LOS) of patients: the expected LOS of a patient is an estimate – based on empirical data, or on a probability distribution – that can differ from the actual duration.
- The requirements of the patient service-mix: dependent on the clinical picture of the patient, he/she may require, besides a bed, different other resources, such as telemetry, specific nursing care, surgical rooms, etc. Focusing solely on bed occupancy can have as a consequence that the other resources are over or under used. Ideally, the work load of the nurses should also be taken into consideration when admitting patients.

### 3.3. Hospital capacity planning

Green [3] discusses both the problem of delays in emergency departments, and the problem of deciding on the available number of beds in the hospital. She argues that using bed occupancy levels as a base for determining the number of beds in the hospital is not accurate. Most reported occupancy levels in the literature are taken at midnight, when the hospital usually has a lower occupancy than during the day. This can lead to a discrepancy of 20% between the reported and the real occupancy levels. Another problem that arises is that the occupancy levels are yearly averages, and as such do not take into account periods with higher occupancy (e.g. due to flu epidemics, disasters, etc.) or lower occupancy (due to holiday periods, etc.). The occupancy level is even not uniform over a week: in general there are less patients admitted to the hospital in the weekend than on weekdays. Green [3] suggests to use – instead of occupancy levels – the average delay before a patient can be admitted. One of the solutions that is proposed in the article is to increase the bed flexibility. This means for example that more beds should be equipped with extra facilities.

Harper and Shahani [14] describe a simulation model in which bed occupancy and patients' refusals can be calculated, taking into account different what-if scenarios. Akcali et al. [15] describe a network flow approach that assists in determining the optimal bed capacity in hospitals. It takes into account the hospital budget and the maximum number of days a patient is on the waiting list before being admitted to the hospital. It is however assumed that all the beds in the hospital are identical.

## 4. Problem description and generation of test data

In this section we describe the admission scheduling problem formally and how the test data is generated.

### 4.1. Parameters and variables

We present the notation that is used throughout the paper.

- Patients are denoted $P_i$, with $i = 1, \ldots, P$, with $P$ the total number of patients. There are $F$ female patients and $M$ male patients, with $P = F + M$. Patients have the following properties:
  - an admission date $AD_i$, with $AD_i$ in $1, \ldots, T-1$, and a discharge date $DD_i$, with $DD_i$ in $2, \ldots, T$ and $AD_i < DD_i$;
  - an age $A_i$ and a gender $G_i$;
  - a treatment, which corresponds to a specialism $S_l$; and
  - a room preference $RPref_w$.
- Nights are denoted $N_k$, with $k = 1, \ldots, T$, with $T$ the number of nights in the planning period of the time horizon.
- Departments are denoted as $D_m$, with $m = 1, \ldots, D$, with $D$ the number of departments. Departments can support one or more specialisms $S_l$, with $l = 1, \ldots, S$, with $S$ the total number of specialisms. A department $D_m$ can enforce that assigned patients have a specific age.
- A specialism $S_l$ can enforce that rooms satisfy specific room properties $RP_v$.
- The $j$th room of the hospital is denoted $R_j$, with $j = 1, \ldots, R$, with $R$ the number of rooms in the hospital. A room $R_j$ can have one or more room properties and a gender. According to the specialisms that are supported in the department, rooms can support in some degree different specialisms.
- The $b$ th bed of room $R_j$ is denoted $B_{jb}$, with $b = 1, \ldots, B_j$, with $B_j$ the number of beds in room $R_j$. The capacity (number of occupied beds) of the room $R_j$ at night $N_k$ is denoted $b_{jk}$.
- The transfer of patient $P_i$ from room $R_j$ to another room on timeslot $N_k$ is presented as $T_{ijk}$.

### 4.2. Constraints

We distinguish between hard and soft constraints. Hard constraints are those that have to be satisfied in order to obtain a feasible solution. A feasible solution that satisfies many soft constraints will be considered of better quality than a solution that satisfies fewer of them.

#### 4.2.1. Hard constraints

- **HC1**: During the considered planning period the room $R_j$ needs to be available. A blocked room (due to maintenance, refurbishment, etc.) cannot be used.
- **HC2**: The admission $AD_i$ and discharge date $DD_i$ and consequently the length-of-stay of a planned patient $P_i$ cannot be

changed by the admission office. This can only be adapted by the responsible doctor.
- **HC3**: For each admission of a patient $P_i$ the length-of-stay is contiguous.
- **HC4**: Two patients $P_{i_1}$ and $P_{i_2}$ $(i_1 \neq i_2)$ cannot be assigned to the same bed $B_{jb}$ in the same time slot $N_k$.
- **HC5**: Male/female patients should be assigned to appropriate rooms $R_j$ as described in Section 2.
- **HC6**: Patients $P_i$ should be assigned to departments $D_m$ that are suited for their age. Elderly patients, for example, should not be assigned to a department that is specialized in paediatrics.
- **HC7**: The medical treatment of a patient $P_i$ may require that he/she is assigned to a room $R_j$ with special equipment. These room properties are mandatory for the treatment.
- **HC8**: Some patients $P_i$ have to be assigned to a single room $R_j$ for medical reasons (quarantine).

Note that hard constraint **HC2** is in contradiction with Gemmel and Van Dierdonck's [5] definition of admission scheduling. We opt not to modify the admission dates of patients.

#### 4.2.2. Soft constraints

- **SC1**: The patient's room choice (single, twin, or ward) has to be respected if possible. A patient $P_i$ who asked for a single room, should preferably be assigned – in case of lack of single rooms – to a twin room. If it is not possible to assign a patient to his/her room of choice, the hospital may miss out on revenues. Physicians in some countries (such as Belgium) are allowed to charge a higher honorarium to patients who are nursed in a single room, if and only if these patients requested a single room during the admission process.
- **SC2**: A patient $P_i$ is preferably nursed in a department $D_m$ that has the right equipment and staff to treat the patient's disease.
- **SC3**: A patient $P_i$ is preferably assigned to a room $R_j$ that in some degree corresponds to the specialism that is needed to treat the patient's clinical picture.
- **SC4**: The medical treatment of a patient $P_i$ may require that he/she is assigned to a room $R_j$ with special equipment. These room properties are recommended to treat the patient. Note that this is the soft constraint version of hard constraint **HC7**.
- **SC5**: The number of unplanned transfers should be minimized.

In some hospitals soft constraint **SC1** is treated as a hard constraint. Attributing a higher relative importance to constraint **SC1** than, for example, to constraint **SC5** may lead to a higher number of unplanned transfers. Suppose that an inpatient asked for a single room, which was not available at the time of admission. When a single room becomes available in a department with the right specialism, the admission officer may choose between moving the patient to that free single room or leaving the patient where she/he is. Although the former decision will lead to an unplanned transfer, it will probably please the patient and the physician in attendance more. It is up to the hospital management to balance the relative importance of each of the constraints.

### 4.3. Constraint weights

Not all the constraints of Section 4.2 are equally important. The weights we attribute to the constraints determine their mutual relative importance (Table 1).

In this example we preferred to avoid unplanned transfers above fulfilling the room preferences of the patients. This explains the high value of **SC5**, compared to the low value of **SC1**.

**Table 1**
Weights of the constraints.

| Constraint | Corresponding weight |
|---|---|
| **HC5** | 5.0 |
| **HC6** | 10.0 |
| **HC7** | 5.0 |
| **HC8** | 10.0 |
| **SC1** | 0.8 |
| **SC2** | 1.0 |
| **SC3** | 1.0 |
| **SC4** | 2.0 |
| **SC5** | 11.0 |

### 4.4. Generation of the data set

We opted to generate data for the experiments, since obtaining real-world data was, due to privacy issues, hard. Although the data is generated, it is based on a realistic hospital situation, and on interviews with experienced 'human' patient admission schedulers.

We consider a planning horizon of 2 weeks and a hospital that consists of 6 departments, each having one major specialism and two minor specialisms. The departments are configured in such a way that every specialism is assigned to three different departments, once as a major and twice as a minor. The number of rooms in every department ranges from 20 to 30. They are divided as follows:

- at most five single rooms;
- between five and ten twin rooms;
- and the rest of the rooms in the department contain four beds.

Each room can have 0, 1 or 2 room properties. Per specialism a random number (with a maximum of five) of subspecialisms is generated. With each subspecialism a length-of-stay is associated that is generated randomly based on a normal distribution with mean 5 and variance 3. Each planned patient is randomly associated with a subspecialism (and hence with a corresponding length-of-stay), and his/her room preference is generated. The specialism that is awarded to the patient also implies the requested/required room properties. For every day of the considered planning period 60 new patients are generated. The data set can be found at http://allserv.kahosl.be/~peter/pas/.

Of course, when this application will be used in daily practice, all the necessary patient data, including length-of-stay, assigned specialism, etc. should be provided by the hospital administration.

## 5. Integer programming approach

We introduce a mathematical model for the patient admission scheduling problem. This will be the starting point for solving the problem with integer programming. We start by introducing some extra concepts, that ease the mathematical expression of the problem. After that, we present the decision variables, the objective function and the constraints. Note that the mathematical model only takes into account the most common case in which the gender of the first patient determines the gender of the room.

### 5.1. Extra constants

- $S_{req,i}$ is the set of rooms that have the required equipment to treat patient $P_i$. This limits the rooms to which patients can be assigned based on the age constraint and the mandatory room equipment (**HC6** and **HC7**).

- $P_{ij}$ is the penalty incurred when assigning patient $P_i$ to room $R_j$. The penalty value $P_{ij}$ will be the result of violations of the soft constraints **SC1**, **SC2**, **SC3** and **SC4**.

### 5.2. Decision variables

- $x_{ijk}$
  - is 1 if patient $P_i$ is assigned to room $R_j$ on night $N_k$;
  - 0 otherwise.
- $t_{ijk}$
  - is 1 if patient $P_i$ is transferred from room $R_j$ to another room at night $N_k$;
  - 0 otherwise.
- $y_{jk}$
  - is 1 if room $R_j$ at night $N_k$ can accommodate only female patients;
  - 0 in case of only males.

### 5.3. Objective function

The objective is to minimize the weighted sum of the total penalty incurred for assigning patients to unpreferred rooms and the number of transfers:

$$Min \sum_{ijk} p_{ij} \cdot x_{ijk} + w \cdot t_{ijk}, \quad (1)$$

with $w$ the weight for the violations of the transfers.

### 5.4. Constraints

Each patient $P_i$ has to be assigned to a room during his/her stay:

$$\sum_{j} x_{ijk} = 1, \forall k = AD_i, \ldots, DD_i, \forall i = 1, \ldots, P \quad (2)$$

At any time, a room should not contain more patients than the available number of beds in the room. Moreover, all patients in a room should have the same gender:

$$\sum_{i=1}^{f} x_{ijk} \leq b_{jk} y_{jk}, \forall j = 1, \ldots, R, \forall k = 1, \ldots, T \quad (3)$$

$$\sum_{i=f+1}^{p} x_{ijk} \leq b_{jk}(1 - y_{jk}), \forall j = 1, \ldots, R, \forall k = 1, \ldots, T \quad (4)$$

Eq. (3) limits the number of female patients per room and per timeslot, while Eq. (4) limits the number of male patients per room and timeslot. If a room accommodates only women, then the right-hand side of Eq. (4) becomes 0; and vice versa.

Decision variable $t_{ijk}$ is 1 if patient $i$ is transferred from room $R_i$ to room $R_j$ at time $k$, 0 otherwise:

$$t_{ijk} \geq x_{ij,k} - x_{ij,k+1}, \forall i = 1, \ldots, p, \forall j = 1, \ldots, R, \forall k = AD_i, \ldots, DD_i - 1 \quad (5)$$

### 5.5. Results

The mathematical model was implemented in ILOG CPLEX. The program was run on a dedicated computer with an Intel quad core processor and 4 GB RAM. However, the integer program did not find a feasible solution within 1 h of calculation. The first feasible solution was found after 13,350 s of calculation. After 1 week of calculation the optimal solution was not obtained.

Since the bed assignment application needs to respond promptly when new patients are admitted, the computation of a solution should take very little time.

## 6. A tabu search algorithm for assigning patients to beds

A good candidate optimization method, which has on condition a feasible solution exists, the property to always provide a solution (good or bad), even after a few moments of calculation is local search. The solution is the result of iteratively changing small parts of the current solution into a new (or candidate) solution by applying neighbourhood moves. The number of moves that are evaluated in an iteration is called the *tournament* factor. From this set the move leading to the best solution is selected and executed, creating a new solution. The essential ingredients in any local search method are

- the representation of a solution,
- the neighbourhoods, which describe the solutions that can be reached from a solution by a single move,
- the cost function, which is a measure for the quality of the solution.

In the next sections we will describe these three items.

Tabu search [16] inherits the above described ingredients from local search, but also adds memory to the search procedure: the properties of the last applied moves are saved in a tabu list, which allows escaping from local optima. The algorithm involves a diversification and intensification strategy. When a better solution is found the tabu list length is decreased, while in the other case the tabu list in increased in order to escape from the local minimum. The tabu search algorithm is extended with a token-ring search as discussed in [17,18]. In a token-ring search, neighbourhoods are switched after a number of non-improving moves. When the maximum number of non-improving moves is reached in the final neighbourhood, the algorithm applies the first neighbourhood again. The algorithm introduces randomness in the search. Small parts of the solution, in this case individual patients are randomly selected and moved. Randomness is furthermore inherently present in the algorithm itself. The quality of a solution may be equal for two possible moves in the same iteration. In this case the tabu search algorithm randomly selects one of the equal moves.

### 6.1. The representation of a solution

We represent a solution as a set of two-dimensional matrices. Each row of a matrix represents a bed in a department. The columns represent the consecutive nights. The number of nights equals the period over which the patient to bed assignment is carried out. The number of individual matrices in a solution equals the number of different departments in the hospital. Fig. 1 represents a rather small hospital that consists of three departments, each possessing several beds.

A patient in this model is represented as an ordered set of smaller objects. Each object correspond to a night stay. We call them 'patient stay parts'. In Fig. 2 patient $P_1$ has a LOS of four nights and is transferred to another bed at the second night. The solution presented in Fig. 2 violates soft constraint **SC5**.

By choosing this representation the hard constraint **HC4** is always fulfilled. The other hard constraints will be included in the cost function.

### 6.2. Neighbourhoods

A neighbourhood defines the solutions that can be obtained from the current one by moving one or more patient stay parts to other positions in the matrix representation. Although we only have two basic moves, by adding restrictions on the direction, several neighbourhoods can be produced.

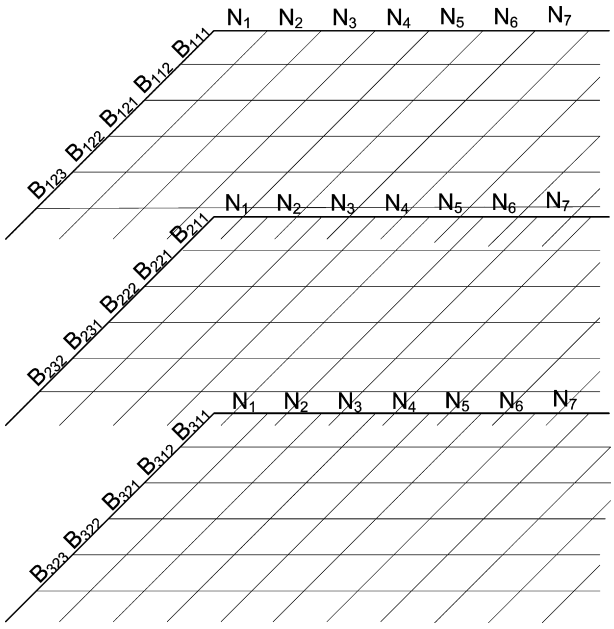


**Fig. 1.** Graphical representation of a solution. Every department is represented by a matrix. The columns denote the nights $N_j$ and the rows denote the beds $B_{ktb}$ that are available in room $R_{kt}$ in department $D_k$.

- The first basic move swaps the content of two matrix elements within the same department. If both matrix elements are non-empty, it means that the corresponding patient stay parts are swapped. If however one of the matrix elements is empty and the other is not, then the non-empty patient stay part is moved to an empty bed. This move will be the basis for the first neighbourhood (see Section 6.2.1).
- In the second basic move, all the patient stay parts of one patient are moved to an empty place either in another department or in the same department. This move is the basis for all the other neighbourhoods (see Sections 6.2.2–6.2.4).

### 6.2.1. Swap-beds neighbourhood

In the first neighbourhood we limit the moves to swapping patient stay parts in the same department. In order to avoid violations of the hard constraints **HC2** and **HC3**, only moves in the same column are allowed (see Fig. 3). This means that patient stay parts can only be moved in the space dimension (to another bed) and not in the time dimension. If a bed is already taken by another patient stay part, the two patient stay parts are swapped.

In this neighbourhood we randomly select a row in which the algorithm searches the best matrix element to move to another row.

### 6.2.2. Move-patient-to-another-department neighbourhood

Since different departments with the same specialism exist, patients do not necessarily need to be nursed in the department supporting the specialism as a major. Due to over occupancy in one

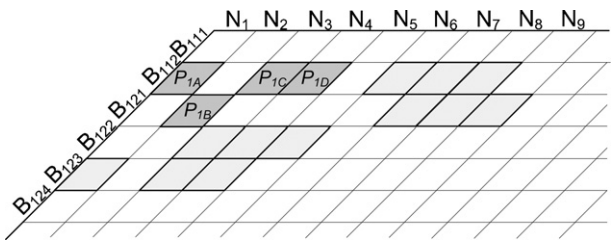


**Fig. 2.** Graphical representation of patient $P_1$'s stay. This patient has a LOS of four nights and he/she is transferred to another room during the second night of stay.

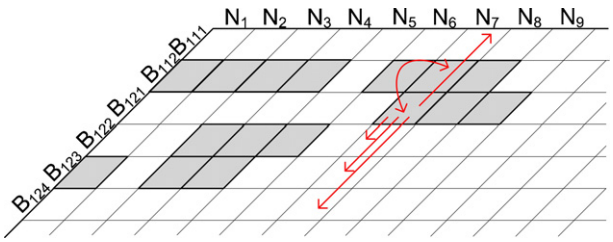


**Fig. 3.** Graphical representation of the swap-beds neighbourhood. The algorithm tries to move the patient stay part at night $N_7$ in bed $B_{121}$ to other beds in the same column.

department, patients can be moved to another department with the same specialism. As the first neighbourhood (see Section 6.2.1) only allows swaps of beds within the same department, we constructed a second neighbourhood in which patients can be moved to another department with the same specialism in case of over occupancy.

This neighbourhood incorporates some load balancing of patients across departments. At the start of every iteration, the algorithm selects the department with the highest number of patient stay parts. From this department a patient is randomly chosen and moved to another department, which has a major or minor specialism that corresponds to the patient's medical requirements. The tabu search algorithm selects the best move from the proposed destination rows for each involved patient stay part. Similarly to the first neighbourhood, the patient stay parts are moved in such a way that constraints **HC2** and **HC3** are never violated.

### 6.2.3. Move-patient-to-same-department neighbourhood

The move-patient-to-same-department neighbourhood is quite analogous to the previous neighbourhood, except that all the patient stay parts of one patient are moved to empty beds in the same department.

### 6.2.4. Move-best-patient-to-another-department neighbourhood

This neighbourhood is similar to the second neighbourhood, except that the algorithm does not select a patient at random. Instead it selects from a list of randomly chosen patients assigned to the same department the best candidate patient to move to another department.

## 6.3. Initial solution

As a start, we generate an initial solution that only takes the hard constraints **HC1**, **HC2**, **HC3**, and **HC4** into account. To assure that constraints **HC2** and **HC3** are satisfied, the patient stay parts of every patient are assigned in such a way that they are contiguous (**HC3**) and begin and end at the appropriate date (**HC2**). Constraint **HC1** is satisfied since we do not assign patient stay parts to blocked beds. The assignment of patients to departments in the initial solution is carried out in a greedy manner:

- The first patient from the patient list is selected.
- According to the patient's medical requirements, a list of departments that support the specialism is generated. The list is ordered as follows:
  - first the departments supporting the specialism as a major;
  - second the departments supporting the specialism as a minor; and
  - last the departments not supporting the specialism.
- When for a specific night, no free room is left in the major specialism department, the initial algorithm searches an empty timeslot in one of the departments that support the specialism as a minor.

- If also these departments are fully occupied for that particular night, the algorithm searches an empty timeslot in another department.
- The above process is repeated until all the patients are assigned or until all the beds for a particular night are occupied and no more patients from the patient list can be admitted.

### 6.4. Cost function

The cost function evaluates a solution and is determined by the weighted sum of the violations of the constraints. Four constraints do not need to be evaluated: the hard constraint **HC4** is always satisfied, due to the representation of a solution. We do not assign patient stay parts to blocked beds (**HC1**). The neighbourhoods are constructed in such a way that hard constraints **HC2** and **HC3** cannot be violated. The constraints that we take into consideration in the cost function are **HC5**, **HC6**, **HC7**, **HC8**, **SC1**, **SC2**, **SC3**, **SC4** and **SC5**.

Due to the fact that some hard constraints can never be violated in this approach, the cost function does not correspond to the objective function of the integer programming formulation (see Section 5.3).

### 6.5. Experiments

We compare the tabu search algorithm hybridized with a token-ring approach with two implementations of variable neighbourhood descent [19]. From the tests we find that the hybridized tabu search produces better results than with a variable neighbourhood descent.

#### 6.5.1. Variable neighbourhood descent variants

We carry out experiments in which we hybridize the tabu search algorithm with

- a token-ring (abbreviated as TR) approach or;
- a variable neighbourhood descent (abbreviated as VND).

The difference between the two methods lies in the point of time when the tabu search algorithm switches to another neighbourhood. In the token-ring search another neighbourhood is chosen each time a number of non-improving moves are exceeded, while the variable neighbourhood descent algorithm switches to another neighbourhood after one non-improving move. In contrast to the token-ring search, the variable neighbourhood descent switches to the first neighbourhood independently of the current neighbourhood, whenever a better solution than the current best solution is found. Intuitively, the first neighbourhood can be considered as a kind of repair neighbourhood that tries to fix some of the moves of the other neighbourhoods.

Since we have to take into account the randomness of the tabu search algorithm (see Section 6) every experiment is carried out ten times. The result of the statistical processing is denoted in box plots, which graphically show the first and third quartiles, the median and the minimum and maximum values of each of the ten runs.

In the first experiment, we investigate two different approaches of the variable neighbourhood descent method. We make a distinction between returning to the first - repair - neighbourhood with or without any information about the department in which the best solution was found. In the top part of Table 2 we present four different settings of the algorithm (Test 1 to Test 4), which are each conducted ten times. The stop criterion of the tabu search algorithm was set to 50,000 iterations. The corresponding box plots of the four algorithms are presented in Fig. 4.

- In Test 1 we add extra memory to the search algorithm by registering the department in which a better solution than the current best solution was found. The registered department is then passed on to the first neighbourhood, and the search continues in this department starting from the best solution. We apply the four neighbourhoods discussed in Section 6.2.
- In Test 2, the algorithm jumps back to the first neighbourhood when finding a better solution than the current best in one of the other neighbourhoods, but we do not register the department in which this solution was found. We apply the four neighbourhoods described earlier.
- In Test 3, we repeat Test 1, except that we only take the first three neighbourhoods into account (Sections 6.2.1, 6.2.2 and 6.2.3).
- The second test is repeated in Test 4, except that we do not use the move-best-patient-to-another-department neighbourhood.

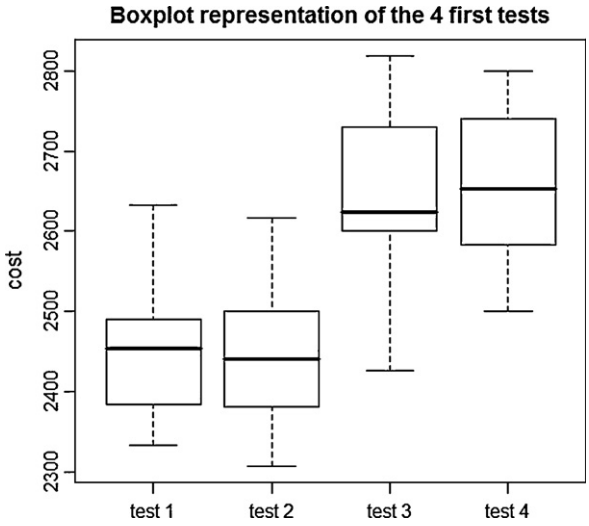


**Fig. 4.** Box plot of the result of Test 1 up to Test 4. The maximum number of iterations is 50,000.

Since the last neighbourhood is, compared to the other neighbourhoods, a rather large neighbourhood to explore, we investigated whether it is favourable to exclude this neighbourhood or not. The first four tests in Table 2 demonstrate that for the same number of iterations the execution times for the third and last test are shorter than for the first two tests. We conduct a Wilcoxon Rank Sum test, in order to prove that the difference between applying an extra neighbourhood or not is statistically significant. According to the Wilcoxon test, there is a statistically significant difference between the first two tests and the last two (with 95% confidence), in the sense that better solutions are obtained when adding the last neighbourhood.

We conducted experiments with two variable neighbourhood descent variants (with and without recollection of the previous department) and we found that the difference between both approaches is not statistically significant (with 95% confidence).

#### 6.5.2. Variable neighbourhood descent versus token-ring

In the next experiments we check whether there is a statistically significant difference between the variable neighbourhood descent and the token-ring approach. The stop criterion in all of the considered experiments is 100,000 iterations and we apply all four neighbourhoods.

In Test 5 and Test 6 (see Table 2), we repeat the experiments of Test 1 and Test 2, with a stop criterion that equals 100,000 iterations. It is obvious from Tests 1, 2, 5 and 6 that the tabu search algorithm produces better results if it can search for a longer period of time.

In contrast to the variable neighbourhood descent approach, the token-ring search has two parameters to be set: the maximum

**Table 2**
The cost of the best solution; the number of iterations needed to find the best solution; and the elapsed time to obtain the best solution are presented for 12 different tests. Each test is executed ten times. The best values for each test are presented in bold.

| Test 1: VND | | | Test 2: VND | | | Test 3: VND | | | Test 4: VND | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| cost | iter. | time | cost | iter. | time | cost | iter. | time | cost | iter. | time |
| 2355.2 | 49,377 | 3234 | 2600.2 | 49,810 | 3183 | 2664.6 | 49,957 | 1477 | 2764.4 | **46,687** | **1305** |
| 2423.8 | 49,131 | 3169 | 2381.8 | 48,893 | 3223 | 2818.2 | 49,246 | 1447 | 2799.2 | 48,128 | 1343 |
| 2484.8 | **48,720** | 3226 | **2308** | 49,193 | 3175 | 2628.8 | 48,343 | **1419** | 2655 | 49,430 | 1385 |
| 2423.2 | 49,900 | 3328 | 2389.8 | 49,106 | 3119 | 2601.2 | 49,238 | 1458 | 2650.8 | 48,292 | 1366 |
| **2332.8** | 49,720 | 3267 | 2472.2 | 49,073 | 3176 | 2532 | 49,955 | 1466 | 2511.2 | 49,118 | 1385 |
| 2384.2 | 49,730 | 3175 | 2616.6 | 49,486 | 3174 | 2782.8 | **48,090** | 1432 | **2500.4** | 49,956 | 1391 |
| 2632.2 | 49,860 | 3171 | 2500.2 | **48,553** | **3083** | 2618.8 | 49,622 | 1462 | 2740.6 | 48,574 | 1361 |
| 2490 | 48,840 | **3078** | 2425.2 | 49,927 | 3151 | **2426.2** | 49,531 | 1464 | 2583.4 | 49,986 | 1389 |
| 2514.2 | 49,832 | 3139 | 2457.2 | 49,806 | 3124 | 2603.8 | 49,679 | 1462 | 2673.2 | 47,844 | 1325 |
| 2486.2 | 49,216 | 3210 | 2346.8 | 49,905 | 3318 | 2730.4 | 49,405 | 1461 | 2620.6 | 49,047 | 1364 |
| Test 5: VND | | | Test 6: VND | | | Test 7: TR(100-200) | | | Test 8: TR(300–1000) | | |
| cost | iter. | time | cost | iter. | time | cost | iter. | time | cost | iter. | time |
| 2232.6 | 96,746 | 5908 | 2064.8 | **95,211** | 5937 | 1792 | 99,361 | 4844 | 1672.2 | 79,923 | 3841 |
| 2329.2 | 98,463 | 6031 | **1917.6** | 99,911 | 6011 | 1832.4 | **90,586** | **4249** | 2013 | **36,412** | **1865** |
| 2311 | 97,200 | **5778** | 2340.8 | 97,656 | 6038 | 1805.8 | 99,151 | 4759 | 1729.6 | 86,997 | 3604 |
| 2131 | 99,089 | 5913 | 2156.6 | 96,389 | **5694** | 2035 | 91,207 | 4370 | 1495.4 | 96,159 | 4004 |
| 2376.8 | 97,513 | 6102 | 2199.4 | 99,824 | 6090 | 1661.8 | 91,266 | 4393 | 1680.4 | 77,239 | 3821 |
| **2054.4** | **96,700** | 6202 | 2028.8 | 98,540 | 6076 | 1678.8 | 99,873 | 4937 | 1680.4 | 87,349 | 3867 |
| 2127.4 | 99,206 | 6403 | 2252.6 | 99,578 | 6017 | 1737.2 | 98,414 | 4782 | 1707.8 | 92,963 | 4755 |
| 2209.6 | 96,917 | 5856 | 2200 | 99,680 | 6069 | 1901.2 | 92,760 | 4576 | **1465.6** | 93,594 | 4515 |
| 2060.6 | 99,403 | 6179 | 2105.8 | 97,748 | 5849 | **1616.6** | 98,657 | 4997 | 1600.2 | 53,695 | 2335 |
| 2118.2 | 98,160 | 6169 | 2127 | 96,645 | 5917 | 1861.6 | 97,860 | 4612 | 1700.6 | 99,917 | 4413 |
| Test 9: TR+VND | | | Test 10: TR(300–1500) | | | Test 11: TR(120–180) | | | Test 12: TR(200–800) | | |
| cost | iter. | time | cost | iter. | time | cost | iter. | time | cost | iter. | time |
| 1906.2 | 97,680 | 4460 | 1472.8 | 91,463 | 5108 | 1701 | 96,726 | 4796 | **1607.6** | 91,197 | 4409 |
| **1806.4** | 98,624 | 4425 | 1687 | 89,927 | 4742 | **1687.6** | 91,316 | 4512 | 1813.4 | 97,210 | 3699 |
| 2030.2 | 99,028 | 4351 | 1633.4 | 93,201 | 4376 | 1740.2 | 98,833 | 4758 | 1760.2 | 92,586 | 4278 |
| 2137.6 | 99,989 | 4410 | **1446.4** | 97,193 | 4973 | 1787.2 | 96,963 | 4806 | 1793.6 | 72,221 | 3536 |
| 1983 | 96,021 | 4326 | 1460.2 | 86,285 | 4193 | 2039.8 | 99,189 | 4744 | 1728.2 | 97,709 | 3526 |
| 1911 | 99,585 | 4510 | 1578.2 | 96,073 | 4766 | 1883.6 | **90,767** | **4354** | 2105.8 | 94,513 | 4418 |
| 1837.6 | 96,596 | 4454 | 1514.4 | 96,794 | 4696 | 1978.2 | 99,709 | 4717 | 1694.4 | 89,477 | 4224 |
| 2044.2 | **95,179** | 4276 | 1558.6 | 96,073 | 4413 | 1821.2 | 99,478 | 4857 | 1738.6 | 98,581 | 4890 |
| 1838.8 | 99,531 | 4381 | 1686.6 | **57,068** | **2470** | 2037.2 | 92,228 | 4394 | 1718.2 | 99,888 | 4642 |
| 1836.2 | 96,271 | **4245** | 1519.4 | 77,205 | 4341 | 1745.4 | 97,022 | 4780 | 1679.2 | **69,467** | **3293** |

length of the dynamic varying tabu list and the number of non-improving moves before the algorithm switches to another neighbourhood. We use primes as values for the tabu list length. Before the search starts, a list of primes smaller than the parameter value is generated. If in one iteration a better solution is found, the previous prime in the ordered list becomes the new tabu list length. The lower limit of the tabu list length is 7. If, however, an iteration generates no better solution, the next prime in the prime list becomes the new tabu list length. In the next tests we experiment with different parameter settings of the token-ring search (Fig. 5).

- In Test 7 (see Table 2) the maximum tabu list length is 100 (actually this means the largest prime smaller than 100), while the number of non-improving moves is set to 200. Applying a Wilcoxon Rank Sum test to Test 5 and Test 7 reveals that the difference between the two tests is statistically significant (95% confidence), which means in this case that the token-ring search of Test 7 finds better results than the variable neighbourhood descent of Test 5.
- The parameter settings in Test 8 for the maximum tabu list length are 300, while the algorithm switches neighbourhoods after 1000 non-improving moves. According to the applied Wilcoxon test, Test 8 does not produce significantly better solutions than Test 7 (95% confidence).
- In Test 9 we combine both the token-ring search and the variable neighbourhood descent approach. The maximum tabu list length is 120 and the allowed number of non-improving moves is 10. When switching back to the first neighbourhood the algorithm remembers the department that was explored last.
- The next three tests (Test 10, Test 11, Test 12) are analogous to Test 8. The parameters for the maximum tabu list length are set to 300, 120 and 200 while the maximum number of non-improving moves are set to 1500, 180 and 800 respectively.

For Test 5 up to Test 12, we also statistically processed the elapsed times and present the results in box plots in Fig. 6. Wilcoxon tests indicate that all token-ring search algorithms produce significantly better solutions in a smaller amount of time than the variable neighbourhood descent.

### 6.5.3. General remarks

We end this section with a general note. The execution times seem rather long, since we start from a situation in which no patients are admitted or assigned to beds. In reality, several beds will be occupied and are – as long as these admitted patients are assigned to the right beds in the right departments – preferably not available for the planned patients. This means that only a percentage of the beds will be available for the planned patients, which reduces the search space considerably.

The data set considers the ideal case in which all patients leave the hospital at the foreseen dates. Of course, the discharge dates of the patients during their stay may change. The idea is to fix the

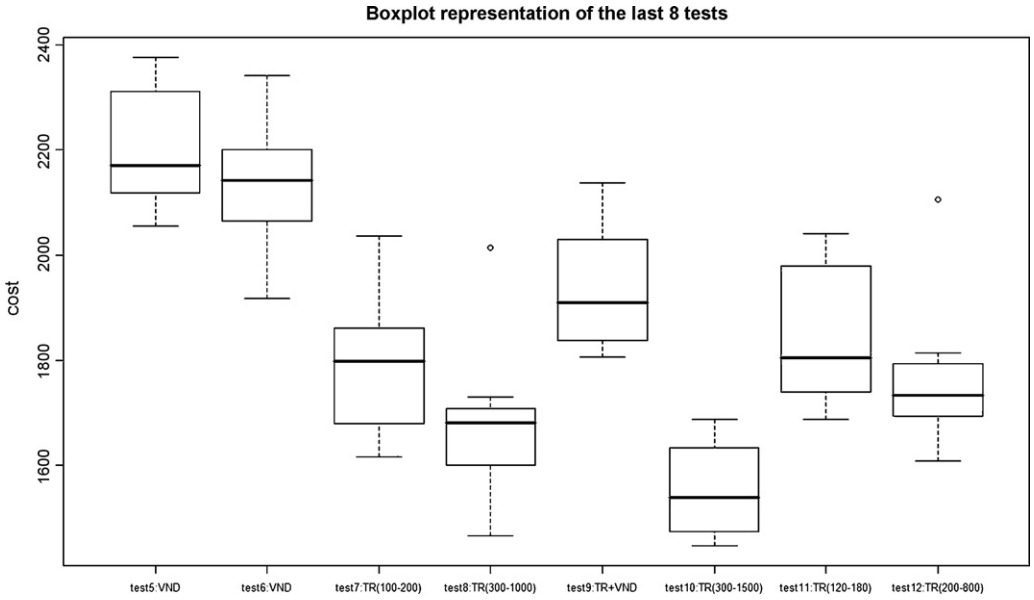


**Fig. 5.** Box plot of the results of Test 5 up to Test 12. Maximum number of iterations is 100,000.

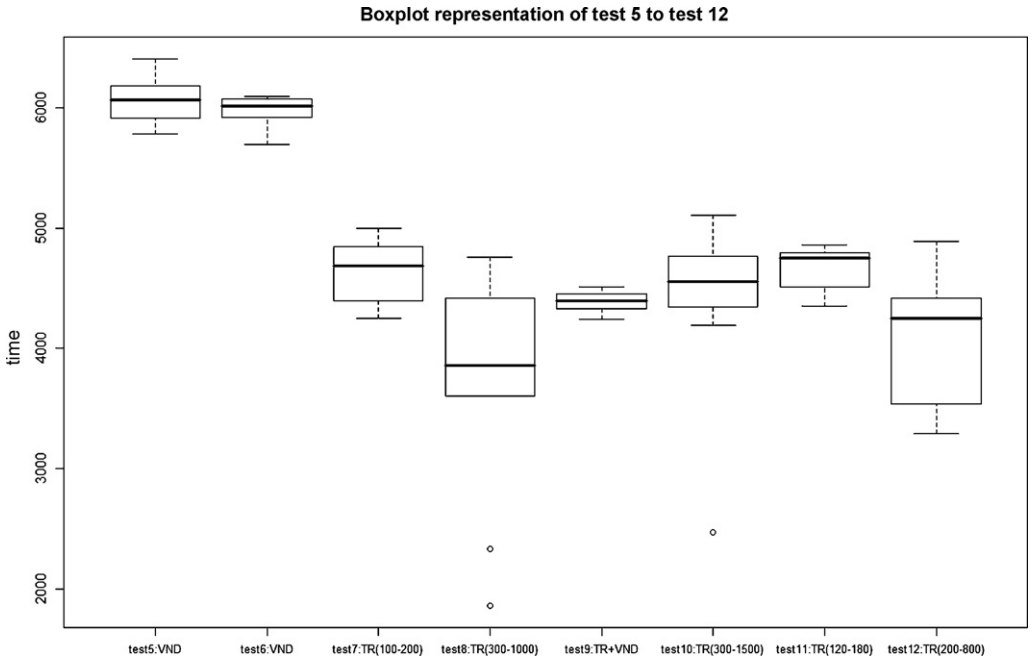


**Fig. 6.** Elapsed time box plots of Test 5 up to Test 12.

current assignment of all patients that are in the hospital and to manually adapt the discharge dates.

## 7. Conclusions and future work

In this paper we present a hospital admission scheduling problem, which to our knowledge has not been reported on in the artificial intelligence literature. We introduce a tabu search algorithm hybridized with both a token-ring and a variable neighbourhood descent approach to tackle the problem. Actually, the problem was first solved with integer programming. However, it took a state-of-the art commercial MIP solver more than 3 h to find a feasible solution.

Apart from the computational complexity exhibited by the integer programming attempt to solve the problem when compared to the tabu search, we feel that the latter approach in this case can profit from another essential difference. In general metaheuristic and local search methods combine significant modelling power with superior flexibility in implementing domain dependent experience in the algorithm. In the present example this is apparent in the implementation of the various neighbourhoods. Our results demonstrate that the resulting transparency

leads to fast algorithms producing satisfactory solutions. As our statistical analysis shows, the behaviour is sufficiently stable for use in an actual application.

The tabu search algorithm is meant to ease the admission schedulers task, since it automatically assigns patients to beds taking into account their wishes and medical nursing needs.

Also the number of patients is balanced across the different departments. As a consequence of the latter, the workload and resource utilization will be spread more evenly over the entire hospital.

The presented algorithmic support will especially improve the hospital performance when considering patients' waiting lists. Issues will arise with respect to the priority of patients and/or treatments, acceptable waiting times for non-urgent treatments, etc. Extending the problem definition with waiting lists will be one of our first future research concerns. This implies that the patient assignments can also be shifted in the time dimension (next to the current moves in the space dimension). Besides the existing neighbourhoods, which only allow moves in the space dimension, additional 'time dimension' neighbourhoods should be included. Other research directions involve addressing emergency admissions, considering the intensive care department, the combination of patient assignment and operating theatre scheduling. An additional point of attention considers evenly spreading the nursing capacity across the entire hospital. Some patients demand more attention than others and that requires more detailed modelling. Initial test results of applying the presented approach to the patient assignment problem are promising and this research area is likely to become very challenging in the near future.

## References

[1] Smith-Daniels VL, Schweikhart SB, Smith-Daniels DE. Capacity management in health care services: review and future research directions. Decision Sci 1988;19(Fall (4)):889–919.
[2] Kusters RJ, Groot PMA. Modelling resource availability in general hospitals. design and implementation of a decision support model. Eur J Operat Res 1996;88:428–45.
[3] Green LV. Operations research and health care: a handbook of methods and applications, vol. 70 of international series in operations research & management science, chapter capacity planning and management in hospitals. Boston, MA: Kluwer Academic Publishers Group; 2004. p. 15–42.
[4] Vissers JMH, Adan IJBF, Dellaert NP. Developing a platform for comparison of hospital admission systems: an illustration. Eur J Operat Res 2007;180: 1290–301.
[5] Gemmel P, Van Dierdonck R. Admission scheduling in acute care hospitals: does the practice fit with the theory? Int J Operat Product Manage 1999;19(9): 863–78.
[6] Burke EK, De Causmaecker P, Vanden Berghe G, Van Landeghem H. The state of the art of nurse rostering. J Schedul 2004;7(November/December (6)):441–99.
[7] Beliën J, Demeulemeester E. Building cyclic master surgery schedules with leveled resulting bed occupancy. Eur J Operat Res 2007;176:1185–204.
[8] Carter MW, Lapierre SD. Scheduling emergency room physicians. Health Care Manage Sci 2001;4:347–60.
[9] Kaandorp GC, Koole G. Optimal outpatient appointment scheduling. Health Care Manage Sci 2007;10:217–29.
[10] Spyropoulos CD. Ai planning and scheduling in the medical hospital environment. Artif Intell Med 2000;20(October (2)):101–11.
[11] Hans E, Wullink G, van Houdenhoven M, Kazemier G. Robust surgery loading. Eur J Operat Res 2008;185:1038–50.
[12] Ogulata SN, Koyuncu M, Karakas E. Personnel and patient scheduling in the high demanded hospital services: a case study in the physiotherapy service. J Med Syst 2008;32(June (3)):221–8.
[13] Marinagi CC, Spyropoulos CD, Papatheodorou C, Kokkotos S. Continual planning and scheduling for managing patient tests in hospital laboratories. Artif Intell Med 2000;20(October (2)):139–54.
[14] Harper PR, Shahani AK. Modelling for the planning and management of bed capacities in hospitals. J Operat Res Soc 2002;53:11–8.
[15] Akcali E, Côté MJ, Lin C. A network flow approach to optimizing hospital bed capacity decisions. Health Care Manage Sci 2006;9:391–404.
[16] Glover F, Laguna M. Tabu search. Boston/Dordrecht/London: Kluwer Academic Publishers; 1997.
[17] Di Gaspero L, Schaerf A. Multi-neighbourhood local search with application to course timetabling. In: Burke EK, De Causmaecker P, editors. Selected revised papers of the fourth international conference on practice and theory of automated timetabling, vol. 2740 of LNCS. Berlin/Heidelberg, Gent, Belgium: Springer–Verlag; 2003. p. 262–75.
[18] Di Gaspero L, Schaerf A. Neighborhood portfolio approach for local search applied to timetabling problems. J Math Model Algorithms 2006;5(1):65–89.
[19] Burke EK, Kendall G, editors. Search methodologies: introductory tutorials in optimization and decision support techniques. New York: Springer; 2005.