Local Search and Lower Bounds for the Patient Admission Scheduling Problem

Sara Ceschia, Andrea Schaerf

DIEGM, University of Udine, via delle Scienze 208, I-33100, Udine, Italy

Abstract

We propose a multi-neighborhood local search procedure to solve a healthcare problem, known as the *Patient Admission Scheduling* problem. We design and experiment with different combinations of neighborhoods, showing that they have diverse effectiveness for different sets of weights of the cost components that constitute the objective function. We also compute many lower bounds based on the relaxation of some constraints. The outcome is that our results compare favorably with the previous work on the problem, improving on all available instances, and in some cases are also quite close to the lower bounds. Finally, we propose the application of the technique to the dynamic case, in which admission and discharge dates are not predictable in advance.

Key words: Local search, simulated annealing, lower bounds, healthcare

1. Introduction

Healthcare is surely one of the most important application domain of optimization in general and of metaheuristics in particular. Many papers have been devoted to healthcare, for example to nurse and physician rostering problems (Burke et al., 2004; Rousseau et al., 2002), and more generally to timetabling problems in hospitals (see, e.g., Hans et al., 2008).

The Patient Admission Scheduling (PAS) problem consists in assigning patients to beds in such a way to maximize both medical treatment effectiveness and patients' comfort. PAS has been defined by Demeester et al. (2008), and studied by Bilgin et al. (2008) and Demeester et al. (2010). In addition, Peter Demeester maintains a website (Demeester, 2009) that publishes the available instances, the current best solutions, and also a solution validator to let other researchers double-check their own solutions. The presence of

Email address: {sara.ceschia,schaerf}@uniud.it (Sara Ceschia, Andrea Schaerf)

the validator (a Java `.jar` file, in this case) is very important, as it provides against the risk of misunderstanding the problem formulation and the cost function.

We propose a local search approach to the PAS problem that makes use of different search spaces and neighborhood relations. We also study how to adapt the neighborhood relations for different weights of the components of the cost function. In addition, we propose a relaxation procedure to compute lower bounds (using CPLEX v. 12), which are useful to assess the quality of the solutions more objectively. The outcome of our work is that our results are better than the ones obtained by Bilgin et al. (2008), and in some cases also quite close to the lower bounds.

In the PAS problem, it is assumed that admission and discharge dates are known in advance. However in the actual situations that hospitals have to face these dates may change depending on the evolution of the disease of the patient. Therefore, we also propose a *dynamic* problem, in which admission and discharge dates are not known in advance. We have developed a solver for this case, which is a modification of the one for the *static* case. In practical situations, the static solver is used mainly for simulations and previsional solutions, whereas the dynamic solver is used for the actual day-by-day scheduling.

The paper is organized as follows. In Section 2, we provide the definition of the problem and its mathematical formulation. In Section 3, we discuss related work. In Section 4, we describe our solution technique and, in Section 5, we present the benchmark instances, the lower bounds, and the experimental results. In Section 6 we discuss the dynamic case. Finally, conclusions are drawn in Section 7.

2. Problem Definition

The general problem formulation is provided by Demeester et al. (2008) and Demeester et al. (2010). We report it here in order to make the paper self-contained. We also describe our preprocessing steps that allow us to improve the efficiency significantly. Finally, we provide the mathematical formulation that we have used to obtain the lower bounds.

2.1. Basic Formulation

These are the basic features of the problem:

Day: It is the unit of time and it is used to express the length of the planned stay of each patient in the hospital; the set of (consecutive) days included in the problem is called the *planning horizon*.

Patient: She/he is a person who needs some medical treatments, consequently she/he must spend a period in the hospital and she/he should be placed in a bed in a room. Each patient has a *fixed* admission date and discharge date within the planning horizon.

Bed/Room/Department: A room can be single or can have more beds. The number of beds in a room is called its *capacity* (typically one, two, or four). Patients may (with an extra charge) express preferences for the capacity of the room they will occupy. Each room belongs to an unique department.

Specialism: Each patient needs one or more specific specialisms for her/his treatment. Each department is qualified for the treatment of diseases of various specialisms, but at different level of expertise. In addition, each specific room has a set of specialisms it is particularly suitable for, each with its level of quality. Levels are expressed in integer values from 1 (highest) to 3 (lowest). Most of the patients need one single specialism for their entire treatment. The patients that need more than one specialism are called *multi-spec* patients; for them, it is specified the length of the treatment for each specialism. For example, one patient might need cardiology for the first three days and gerontology for the remaining two.

Room Feature: Each room has different features (oxygen, telemetry, ...) necessary to treat particular pathologies. Every bed in a room has the same equipment. Patients may *need* or simply *desire* specific room features.

Room Gender Policy: Each room has a *gender policy*. The are four different policies, identified by the letters in the set {D, F, M, N}. In rooms with policy F (resp. M) only female (resp. male) patients can be accepted. If the policy is D the room can be occupied by patients of both gender, but on any day the patients in the room must be all of the same gender. Finally, rooms of policy N can be occupied simultaneously by patients of both genders.

Age Policy: Some departments are specialized in patients of a specific age range (e.g., pediatrics or gerontology). For these departments there is a limit on the minimum or the maximum age of the patients admitted.

The PAS problem consists in assigning a bed to each patient on each day of her/his stay period. The assignment is subject to the following constraints and objectives:

Bed Occupancy (BO): There can be at most one person per bed per day.

Room Gender (RG): For each day, the gender of the patients in the same room should obey the gender policy of the room.

Department Specialism (DS): The department should have level 1 for the specialism of the patients hosted in the rooms of the department; lower level are penalized as explained in (Demeester, 2009).

Room Specialism (RS): Similarly to departments, all levels lower than 1 are penalized (see Demeester, 2009).

Room Features (RF): The room should have the features needed or desired by the patients, missing ones are penalized; the penalty is higher for needed ones than for desired ones.

Room Preference (RP): Patients should be assigned to rooms of the preferred capacity or smaller.

Patient Age (PA): Patients should be assigned to department that can accept patients of their age.

Transfer (Tr): Patients should not change the bed during their stay; a bed change is called a *transfer*. All transfers are penalized in equal way.

The constraint BO is obviously a *hard* constraint, given that the simultaneous assignment of two patients to the same bed clearly makes the solution infeasible. In the current formulation, all the other constraints are considered *soft* and are weighted appropriately. The weights of the various components can be assigned by the final user, based on the specific situation, regulation, and internal policy.

Notice that when a patient needs more specialisms, it is quite acceptable that she/he has to be transferred from one room to another one on the day of the change of treatment. Nevertheless, given that departments and rooms may have more than one specialism, it is also possible that a patient can have all of them in the same bed. For this reason, transfers of multi-spec patients are penalized like all the other transfers. However, our solvers are able to deal also with the general situation, in which the penalty of the transfer is not always the same but depends on the treatment of the patient.

2.2. Preprocessing

Based on the definitions given above, we recognize that the problem can be greatly simplified by means of two preprocessing steps which lead to a new problem formulation.

2.2.1. Room Assignment

First of all, it is evident from the formulation that beds belonging to the same room are indistinguishable from each other in terms of features and constraints.

Therefore, we can reformulate the problem as an assignment of patients to *rooms* rather than to *beds*. To this aim, we have to replace BO with the constraint that the number of patients assigned to the same room on each day cannot exceed the capacity of the room.

Given that the output should be delivered in terms of assignments to beds, the room assignment must then be post-processed to be transformed into a bed assignment. In the solution obtained by the post-processor, it is important to avoid to move a patient from one bed to another one in the same room. To this regard, it is easy to prove that if patients are processed on the basis of the order on their admission day, we can always produce an assignment that never moves a patient from one bed to another one in the room.

2.2.2. Patient-Room Penalty Matrix

The second preprocessing step is related to the notions of departments, specialisms, room features, age policy, and preferences. It is evident that all the constraints related to these notions contribute, with their weights, to the penalty of assigning a given patient to a given room.

Therefore, we can "merge" together this information into a single matrix that represents the cost of assigning a patient to a room. This *patient-room penalty matrix* C is computed once for all, when reading the input data, and all the five notions mentioned above (departments, specialisms, room features, age, and preferences) can be removed from the formulation.

The penalty associated to the room gender policy RG can also be *partly* included in the matrix C. More specifically, if the room policy is N there is no penalty, if it is of type F or M, then the penalty of accepting a male patient or a female patient is merged to the matrix C. The only case that is not merged into C, because it depends also on the assignment of the other patients, is the case on policy D, which is the most common one.

2.2.3. Problem Reformulation

According to the first preprocessing step, the constraint BO is removed and replaced by the following one:

Room capacity (RC): The number of patients in a room per day cannot exceed its capacity.

Based on the second preprocessing step, the constraints DS, RS, RF, PA, RP, and RG (for all rooms, but those of policy D) are removed and replaced by the following one.

Patient-Room cost (PRC): The penalty of assigning a patient p to a room r is equal to the value of $C_{p,r}$.

In conclusion, the problem includes one hard constraint, namely RC, and three cost components (or soft constraints): RG (case D only), PRC, and Tr.

It is worth noticing that, if we remove the constraint Tr, there is no correlation between the room assigned to a patient on one day with the one assigned to the same patient on a different day. As a consequence, in such a case each single day could be scheduled independently and the problem size reduces greatly.

2.3. Mathematical Modeling

In order to compute the lower bounds, we introduce here the mathematical model of the reformulated problem. We call $\mathcal{P}$ the set of patients, $\mathcal{D}$ the set of days, and $\mathcal{R}$ the set of rooms. We also call $\mathcal{P}_F$ the set of female patients and $\mathcal{P}_M$ the set of male ones (with $\mathcal{P}_F \cup \mathcal{P}_M = \mathcal{P}$). Finally, we call $\mathcal{D}_p$ the set of days in which the patient p is present in the hospital, $\mathcal{R}_{\mathsf{D}}$ the set of rooms that have policy D, and c_r the capacity of room r.

The decision variables are the following:

- $x_{p,r,d}$: 1 if patient p is assigned to room r in day d, 0 otherwise
- $t_{p,r,d}$: 1 if patient p is transferred from room r in day d, 0 otherwise
- $f_{r,d}$, $m_{r,d}$: 1 if there is at least one female (resp. male) patient in room r in day d, 0 otherwise
- $b_{r,d}$: 1 if there are both male and female patients in room r in day d, 0 otherwise

The x variables describe the actual search space of the problem, all other variables, namely t, f, m, and b, are used to express the components of the objective function F:

$$F = F_{\mathsf{PRC}} + F_{\mathsf{RG}} + F_{\mathsf{Tr}} \tag{1}$$

The three components of F refer to PRC, RG, and Tr, and are defined as follows:

$$F_{\mathsf{PRC}} = \sum_{p\in\mathcal{P},r\in\mathcal{R},d\in\mathcal{D}_p} C_{p,r}\cdot x_{p,r,d} \quad (2)$$

$$F_{\mathsf{RG}} = \sum_{r\in\mathcal{R}_{\mathsf{D}},d\in\mathcal{D}} w_{\mathsf{RG}}\cdot b_{r,d} \quad (3)$$

$$F_{\mathsf{Tr}} = \sum_{p\in\mathcal{P},r\in\mathcal{R},d\in\mathcal{D}} w_{\mathsf{Tr}}\cdot t_{p,r,d} \quad (4)$$

The constraints are the following:

$$\sum_{r\in\mathcal{R}} x_{p,r,d} = 1, \qquad \forall\ p\in\mathcal{P}, d\in\mathcal{D}_p \quad (5)$$

$$\sum_{p\in\mathcal{P}} x_{p,r,d} \le c_r, \qquad \forall\ r\in\mathcal{R}, d\in\mathcal{D} \quad (6)$$

$$f_{r,d} \ge x_{p,r,d}, \qquad \forall\ p\in\mathcal{P}_F, r\in\mathcal{R}, d\in\mathcal{D} \quad (7)$$

$$m_{r,d} \ge x_{p,r,d}, \qquad \forall\ p\in\mathcal{P}_M, r\in\mathcal{R}, d\in\mathcal{D} \quad (8)$$

$$b_{r,d} \ge m_{r,d} + f_{r,d} - 1, \qquad \forall\ r\in\mathcal{R}, d\in\mathcal{D} \quad (9)$$

$$t_{p,r,d} \ge x_{p,r,d} - x_{p,r,d+1} \qquad \forall\ p\in\mathcal{P}, r\in\mathcal{R}, d\in\mathcal{D} \quad (10)$$

Constraints (5) ensure that every patient is assigned to a room for all days of his/her. Constraints (6) ensure that the rooms are not overloaded (RO). Constraints (7)—(9) relate the auxiliary variables f, m, and b to x, for taking care of the cost component RG. Similarly, Constraints (10) relate the t variables to x to take into account the costs of the component Tr.

3. Related Work

The general problem of maximizing the bed utilization in hospital has been significantly studied in the literature. The main lines of research regard: controlling patient urgency and demands (*admissions management*), allocation of beds in the wards (*capacity planning*), and patient's placement according to medical specialty (*patient assignment*).

In admission management (see, e.g., Gemmel and Van Dierdonck, 1999), the problem is to plan admissions in hospitals taking into account all resources available, different treatments required by different patients' groups and different policies for *elective*, *urgent* and *emergency* admissions. Elective patients do not have to be treated immediately, their admission to the hospital is usually scheduled or they are put on a waiting list for an appointment; urgent patients need to be admitted in a short time, usually as soon

as there is a bed available; finally emergency patients must be immediately admitted to the hospital and they have the priority over other patients for each treatment they need. In most of the papers, the objective is to improve the scheduling technique in such a way to maximize the use of the resources without considering the level of service offered to patients. In such a view, the waiting lists for elective patients are used as buffer for the variations in the level of urgent and emergency admissions. For a review on waiting list models see the work by Worthington (1987, 1991) and Mullen (1994). In their literature review, Smith-Daniels et al. (1988) conclude that most of the admission scheduling systems consider only bed capacity as objective, and they show that the maximization of the bed occupancy level depends on the number of emergency patients, the length of stay of patients and the typology of treatment required. Recently, Vissers et al. (2007) have considered alternative concepts to deal with the problem of long waiting times for hospital admission, simulating the extreme situations where all the patients have an appointment for admission or every patient is immediately treated (if necessary, using extra-resources). Williams et al. (2009) try to characterize healthcare system performance and to evaluate system efficiency using the waiting time of patients and the usage of healthcare resources as system performance measures.

The level of bed occupancy can be also used for the planning and management of bed capacities. Brandeau et al. (2004) and Harper and Shahani (2002) demonstrate that using a deterministic approach that considers only the bed utilization to plan the capacity can lead to misleading results because the relationship between bed occupancies and refused admission rates is overlooked. In particular, Harper and Shahani (2002) show that a detailed hospital capacity model should incorporate information about the monthly, daily and hourly admission profiles and the distribution of the patients' length of stay.

The problem of allocating beds to patients has been studied by Goldman et al. (1968), who construct a simulation model to evaluate the performance of several bed allocation policies. They state that there are a lot of factors that affect demands for bed as the number of beds per room, treatment protocols, the separation by gender or age, patient preferences and the typology of care unit (intensive, emergency, intermediate). Kao and Tung (1981) construct a demand forecasting system for generating the inputs to a bed allocation model based on marginal analysis. They assume that patient admissions follow a Poisson procedure. Other simulation model for hospital bed planning are proposed by Dumas (1984, 1985) and Vassilacopoulos (1985).

A critical task related to these problems is to determine a timetable that allocates sessions (i.e. timeslots) of the operating rooms to different depart-

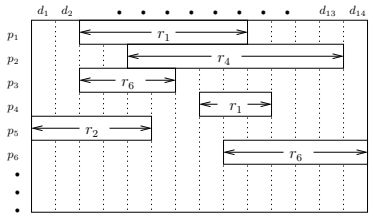

Figure 1: The search space S_0 (d for days, p for patients and r for rooms)

ments to treat elective patients (*Master Surgical Schedule*). For the Master Surgical Schedule Problem, different objectives have been used as obtaining a balanced distribution of operating time among wards (Blake et al., 2002), maximizing the utilization rate (Dexter and Macario, 2002), minimizing both operating room idle time and overtime (Jebali et al., 2006), maximizing throughput, and minimizing average patient waiting time (Tànfani and Testi, 2010). For an overview on operating room planning and scheduling approaches see the work by van Oostrum et al. (2010).

The specific problem that we consider has been defined only recently by Demeester et al. (2008), and extended by Demeester et al. (2010). It has been also studied by Bilgin et al. (2008), who propose the benchmark instances and solve them by using a hyper-heuristic approach.

4. Solution Technique

Our solution technique is based on local search. In order to describe it, we introduce the search space, the cost function, the initial solution, the neighborhood relations, and the metaheuristics.

4.1. Search Space

Differently from Demeester et al. (2008) and Bilgin et al. (2008), our solvers search on the space, called S, of the patient-room assignments rather that on the space of the patient-bed ones. A further difference with respect to the previous work is that we do not consider the space S of all possible assignments, but we restrict ourselves to some limited subsets of it.

The first space that we investigate, called S_0, considers only states in which a patient is assigned to the same room for the full stay. In all states in S_0 the cost of transfers is always 0, given that all solutions that contain transfers are excluded by construction. Obviously, it is possible that S_0 contains none of the optimal solutions.

For the above reason, we also consider the search space S_1, which is larger than S_0, but still smaller than S ($S_0 \subseteq S_1 \subseteq S$). In a state $s \in S_1$

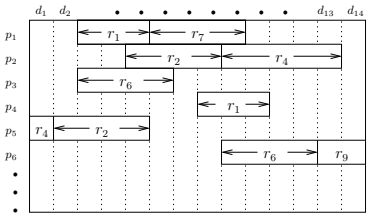

Figure 2: The search space S_1 (d for days, p for patients and r for rooms)

we consider the possibility to transfer a patient *at most once* during her/his stay. Therefore, in a state $s \in S_1$ we have *transferred* patients, that are moved on one day during their stay, and *unmoved* ones, that stay all the time in the same room. For transferred patients, we call *admission room* and *discharge room* the room of the first part and of the second part of the stay, respectively. A fragment of a state in S_0 and S_1 is shown in Figures 1 and 2, respectively.

The intuition behind the use of S_1 is that, in practice, it is quite unlikely that we have to transfer a patient twice, and in addition it is reasonable that the same quality of a solution with two transfers on a single patient can be obtained by a similar one with one transfer for two different patients.

Notice that for patients with more than one treatment, it is rather intuitive to transfer them on the day of the change of treatment. However, for generality we do not limit the search space to this case, but each patient can be transferred on any day (or not transferred at all).

Summarizing, we have two distinct search spaces S_0 and S_1, that will be searched in different solvers and/or in consecutive phases of the solution process. For these spaces we will also use different neighborhood relations.

4.2. Cost Function and Initial Solution

The cost function obviously includes the cost components RG, PRC, and Tr. It also takes into account, with an appropriate high value, the constraint RC, given that assignments in which a room is occupied by more patients than its capacity are included.

The initial solution is generated in a random way: Assign to each patient a random room among all of them, independently of the capacity. However, a unique random room is assigned to each patient for the full stay length, so that in the initial state all patients are unmoved. This way the initial solution belongs to both S_0 and S_1.

4.3. Neighborhood Relations

We present here four neighborhood relations, that will be used in combination. We first illustrate the two neighborhood relations for S_0:

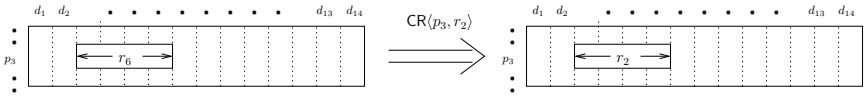

Figure 3: A CR move

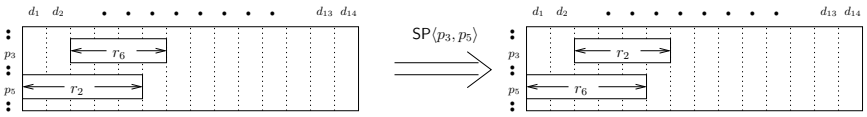

Figure 4: A SP move

Change Room (CR): A patient is moved from one room to another one for her/his full stay. A CR move is identified by the pair $\langle p, r\rangle$, where p is the selected patient and r the new room.

Swap Patients (SP): The rooms of two patients are swapped for their full stay. A SP move is identified by the pair $\langle p_1, p_2\rangle$, where p_1 and p_2 are two patients. The stay period of the two patients must overlap (for at least one day).

It is evident that a SP move can be obtained as a sequence of two CR moves. However, it is important to have SP moves in the neighborhood, because it is frequent that the intermediate state in which only one CR move is executed has a higher cost, due to the copresence in the same room of the two patients. This way, the possibility of performing swaps, resulting from two consecutive CR moves, is severely inhibited.

The reason why we limit swaps to patients with an overlapping period is that, in the opposite case, the effects of the two CR moves composing the SP move would be independent. In such a case, the intermediate state would never have a cost higher than the initial state and the final one of the SP move. Therefore, an SP move of this type is useless for the overall search process. Examples of CR and SP moves are shown in Figures 3 and 4.

Moving to the space S_1, we consider two other neighborhoods. The first one is an extension of CR that also introduces and removes transfers.

Partial Change Room (PCR):

- For transferred patients: The admission room or the discharge room of a patient is changed to a new one. If the new room is the same as in the other segment of the stay, the transfer is canceled.
- For unmoved patients: First it is decided if the move has to introduce a transfer or not:

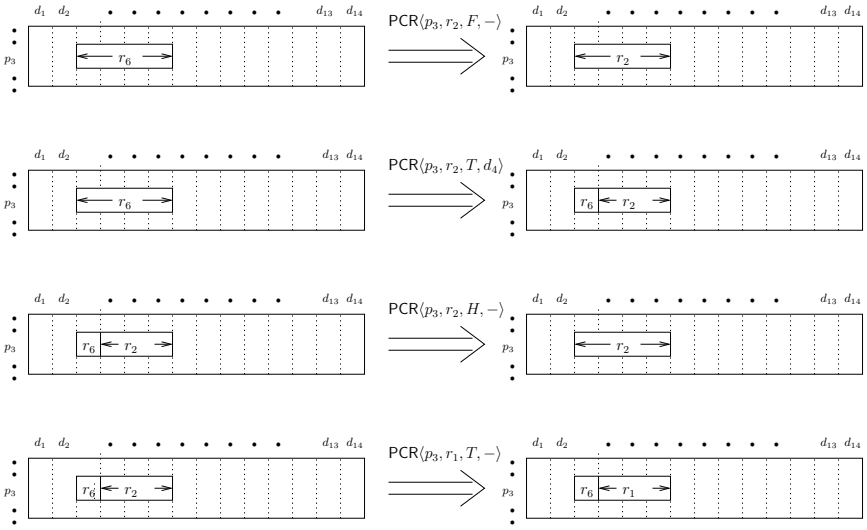

Figure 5: Some PCR moves

- If no transfer is to be introduced, then the move behaves like CR.
- Otherwise, a transfer day, a new room, and the segment (admission or discharge) are selected; the move then consists in inserting a transfer on the selected day and changing the admission or discharge room to the new one.

A PRC move is identified by the quadruple $\langle p, r, s, d \rangle$ where p is the patient, r the new room, s the segment, and d the new transfer day. Examples of PCR moves are shown in Figure 5. The attribute s can assume one of the three values H, T, F, which represent the first segment of the stay (H for head), the second segment (T for tail), or the complete stay (F for full). The attribute d represents the day of the transfer, and it is only meaningful when both the following conditions hold: the patient is unmoved and $s \neq F$ (see second case of Figure 5).

Our next neighborhood is an extension of SP to deal with transferred patients.

Partial Swap Patients (PSP): The rooms of the segments of two patients are swapped. For each single patient, if she/he is transferred, the segment can be either the first or the second; if the patient is not transferred, the segment is the full stay. The transfer day is never modified. Like for SP, the segments of the two patients must overlap.

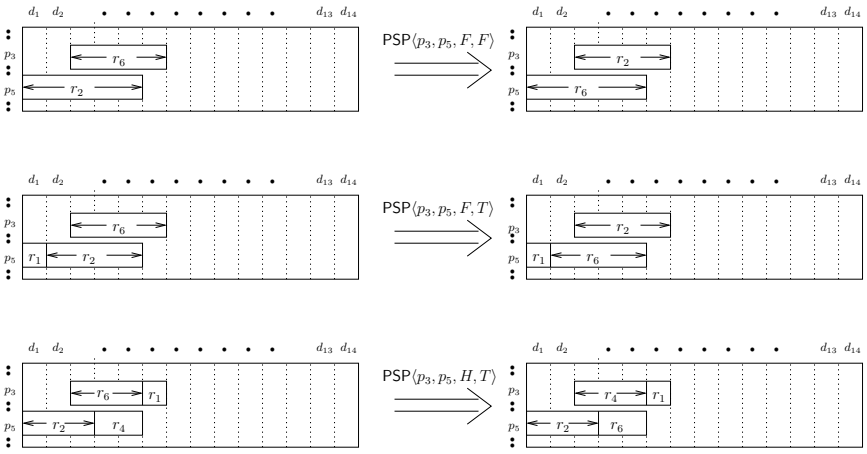

Figure 6: A PSP move

A PSP move is identified by the quadruple $\langle p_1, p_2, s_1, s_2 \rangle$ where p_1 and p_2 are the patients, and s_1 and s_2 are the segments (H, T, or F). The segment of a patient is F if the patient is unmoved, and H or T if she/he is transferred.

Notice that when a PSP move is applied to states in S_0, the attributes s_1 and s_2 have always the value F, and a move PSP behaves exactly like a SP move. Therefore, SP and PSP can actually be seen as a single neighborhood applied in different states. This is not the case for PCR; in fact, a PCR move applied to a state in S_0 can lead outside S_0 itself. Examples of PSP moves are shown in Figure 6.

4.4. Metaheuristics

We have initially experimented with two metaheuristics, namely Simulated Annealing (SA) and Tabu Search (TS). In preliminary experiments, better results have been found by SA, and we therefore report here only the analysis and the tuning for it.

The origin of Simulated Annealing lies in the physical annealing process (Kirkpatrick et al., 1983; Černý, 1985). In the literature, many variants of SA have been proposed (see, e.g., Aarts and Lenstra, 1997; Hoos and Stützle, 2005). The version used here, which is shown in Figure 7 as a flowchart, is the one with probabilistic acceptance $e^{-\Delta F/T}$ and geometric cooling $T_{i+i} = \beta \cdot T_i$.

The procedure shown in Figure 7 has four parameters: start temperature T_0, stop temperature T_{min}, cooling rate β, and number of neighbors sampled at each temperature N.

The search for adequate parameter values has been the subject of many practical and theoretical studies over the years (see, e.g., van Laarhoven and Aarts, 1987; Johnson et al., 1989, 1991).

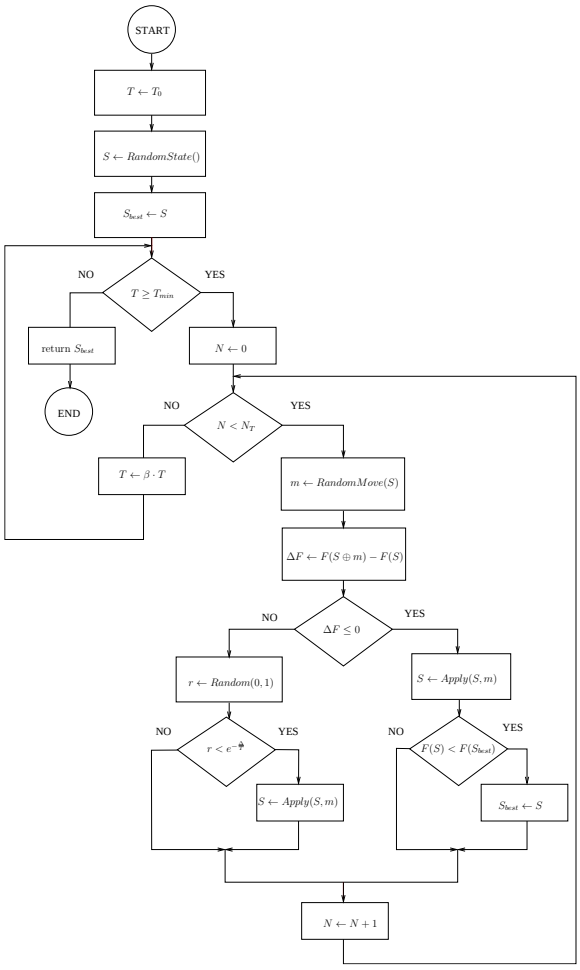

Figure 7: Flowchart of the SA procedure

Firstly, the initial temperature T_0 can be set based on trial runs. In this case, an initial probability of acceptance is defined and T_0 is set in proportion to the maximal difference in cost between any two neighboring solutions, in order to find approximately the fraction of accepted moves during the initial stage.

The number of neighbors N sampled at each temperature can be fixed depending on the size of the neighborhoods or the time granted, or it can vary depending on the current temperature T.

Concerning the way to update the temperature, i.e. the cooling schedule, we can basically distinguish between static and dynamic schedules: In the

first case β is fixed, whereas in the latter one it is adaptively changed during execution of the algorithm. Interested readers can refer to Triki et al. (2005) and Thomson and Dowsland (1995) for further information.

Finally, the final value of the temperature T_{min} that determines the stop of the search process can be fixed at some small value, which may be related to T_0 or to the maximum cost difference between two neighboring solutions (Aarts and Lenstra, 1997). Other stopping criteria can be used to state if the system is *frozen*, based on the improvement of the cost function in a given interval of iterations.

Our choice is to select the four parameters based on a tuning phase, which consists in an extensive experimentation with many different configurations as illustrated in Section 5.

For making a proper tuning of the parameters, we have to consider the problem that different configurations would result in different running times. Nevertheless, we decide to compare them in a fair setting, giving to all of them the same amount of computational time. To this aim, we let the three parameters T_0, T_{min}, and β vary and we compute N in such a way to have exactly the same number of total iterations. In detail, calling I the fixed total number of iterations, we compute N from the following formula

$$N = I / \log_\beta (T_0/T_{min})$$

Moving to the solver composition, the first one that we consider, called M_0, is SA using as neighborhood the union of CR and SP; using the terminology of Di Gaspero and Schaerf (2006) this would be defined as $SA(\texttt{CR} \oplus \texttt{SP})$. It is clear that M_0 explores only the space S_0.

The second one, called M_1, explores the space S_1 by using the neighborhood $\texttt{PCR} \oplus \texttt{PSP}$, and it is defined as $SA(\texttt{PCR} \oplus \texttt{PSP})$

Our third solver is a two-stage metaheuristic, that we call M_0+M_1. First it runs $SA(\texttt{CR} \oplus \texttt{SP})$, in order to get to a good solution quickly, and subsequently it runs $SA(\texttt{PCR} \oplus \texttt{PSP})$, starting from the best solution found by $SA(\texttt{CR} \oplus \texttt{SP})$.

5. Experimental Analysis

In this section, we first introduce the benchmark instances and the experimental settings, then we compute the lower bounds, and we show our experimental results.

I	B	R	D	P		MP	TP	PRC	BO	SL
1	286	98	14	652	(693)	0	2390	32.16	59.69	3.66
2	465	151	14	755	(778)	0	3905	36.74	59.98	5.17
3	395	131	14	708	(757)	0	3156	35.96	57.07	4.46
4	471	155	14	746	(782)	0	3576	38.39	54.23	4.79
5	325	102	14	587	(631)	0	2244	31.23	49.32	3.82
6	313	104	14	685	(726)	0	2821	29.53	64.38	4.12
7	472	162	14	519	(770)	0	2215	102.87	33.52	4.27
8	441	148	21	895	(895)	0	4066	103.72	43.90	4.54
9	310	105	28	1400	(1400)	0	6864	113.25	79.08	4.90
10	308	104	56	1575	(1575)	0	8237	80.57	47.76	5.23
11	318	107	91	2514	(2514)	0	13270	91.29	45.86	5.28
12	310	105	84	2750	(2750)	0	14285	94.00	54.86	5.19
13	368	125	28	907	(907)	202	5348	112.62	51.90	5.90

Table 1: Description of the instances: I: Instance, B: Beds, R: Rooms, P: Patients, MP: Multi-spec Patients, TP: Total Presence, PRC: average Patient/Room Cost, BO: average percentage Bed Occupancy, SL: average Stay Length

5.1. Instances and Settings

We experiment on the 13 instances[1] available from Demeester (2009). Their most important features are shown in Table 1. For the number of patients P we report the ones that are actually included in the problem. The one in parenthesis is the number reported in the instance files, but it also includes patients that stay for zero days (same admission and discharge date) or that have an admission day subsequent to the end of the planning horizon. The column TP reports the total daily presences obtained by summing up the stay length of all patients. The column PRC is the average value of the patient-room cost matrix C. The column BO is the average percentage occupancy of the beds. Finally, SL is the average length of patients' stay.

Notice that only instance 13 has multi-spec patients. More specifically, all the 202 multi-spec patients of instance 13 require two specialisms, and thus no patient that requires three or more of them is present in this benchmark dataset.

Looking at the table, we see that there is a big gap in the column PRC between instances 1–6 and 7–13. This is related to the structure of the instances: instances 1-6 describe situations in which the rooms are more homogeneous, with a few different room features and similar departments. Instances 7–13 come from heterogeneous situations, with many features and

[1]We exclude instance 0 used in (Demeester et al., 2010) because it has a different format, which is not accepted by the validator

departments, which create a variability of costs for the patients.

The default values for the weights, defined by Demeester et al. (2008) and based on realistic hospital situations, are the following: $w_{\mathsf{RG}} = 5$, $w_{\mathsf{DS}} = w_{\mathsf{RS}} = 1$, $w_{\mathsf{RFn}} = 5$, $w_{\mathsf{RFd}} = 2$, $w_{\mathsf{RP}} = 0.8$, $w_{\mathsf{PA}} = 10$, $w_{\mathsf{Tr}} = 11$, where w_{RFn} and w_{RFd} are the weights for needed and desired room features, respectively.

Technically, in our solver we multiply all weights by 10 in order to have all integer values, and then use the faster integer arithmetic of the CPU. However, in the following tables we divide back the results by 10, so as to have the same scale used in previous work.

Besides the default values, we investigate also other combinations of weights for the cost components. Given that w_{Tr} is the most crucial one, we decide to experiment with many different values of it, while keeping all the others at the same level.

The total number of iterations is set to $I = 6.7 \cdot 10^8$ which results in a running time of about 400s on our PC, an Intel QuadCore PC (64 bit). However, we prefer to set the number of iterations, rather than using a real timeout because, as advocated by Johnson (2002), the use of the timeout makes the experiments less reproducible.

The software is written in C++ language, it uses the framework EasyLocal++ (Di Gaspero and Schaerf, 2003), and it is compiled using the GNU C/C++ compiler, v. 4.1.2, under Debian Linux.
All our results have been validated with the Java program available on Demeester (2009).

5.2. Lower Bounds

Lower bounds are obtained relaxing the integrality constraint for the variables of the model presented in Section 2.3 and solving it using CPLEX (IBM ILOG, 2009). The full model, even with the integer variables replaced with real-valued ones, however turned out to large to be solved to optimality in reasonable time (24 hours) for some instances. Therefore, we resort also to some weaker, but faster, lower bounds. We produce a set of 4 lower bounds of increasing quality and increasing running times:

LB_{PRC}**:** remove the terms F_{RG} and F_{Tr} (and the corresponding variables f, m, b, and t); the problem can be solved separately for each day and it reduces to a simple assignment problem in which the costs are given by the C matrix. The assignment problem is solved in very short time, and the optimal solution has only integer values for the variables.

$LB_{\mathsf{PRC+RG}}$**:** remove the term F_{Tr} but keep F_{RG}; the problem can still be solved separately for each day, but the solution is no more integral. Running time increases but remains acceptable.

$LB_{\mathsf{PRC+RG+Tr_7}}$**:** keep both terms F_{Tr} and F_{RG}, but split the solution in sub-instances of 7 days each. This way, only the costs associated to the transfer in the first day of a week are neglected.

$LB_{\mathsf{PRC+RG+Tr}}$**:** keep all constraints, removing only the integrality constraints.

Table 2 shows, for the default weights, the lower bounds with their running times (in seconds), along with the cost of the *best solution*. The best solution is the one with the lowest value obtained by our solver throughout the whole experimentation. This value is the current best known value. The symbol — means that no solution has been found within 24 hours. Obviously, intermediate quality bounds (with intermediate running times) can be obtained splitting the instance in sequence of days of length different from 7.

We can see that for instances 1-6, the fast method LB_{PRC} is already quite tight, showing that the unavoidable cost related to the patient-room penalty matrix C is indeed the most significant component for these cases.

For instances 7-12, even using the most time-consuming LB, the gap is rather high. This is related to the difference in the values of the C matrix highlighted in the comments of Table 1.

A solution that either ignores or relaxes transfers can get all the low C values. On the contrary, a solution to the complete problem is forced to select rooms with higher costs.

Instance 13 exhibits a peculiar behavior: the last one ($LB_{\mathsf{PRC+RG+Tr}}$) is rather tight, whereas the others are not. This is clearly related to the multi-spec patients.

	LB_{PRC}		$LB_{\mathsf{PRC+RG}}$		$LB_{\mathsf{PRC+RG+Tr_7}}$		$LB_{\mathsf{PRC+RG+Tr}}$		best
I	cost	time	cost	time	cost	time	cost	time	cost
1	636.0	0.5	637.6	6.0	640.0	1227.3	645.0	13957.8	655.6
2	1104.0	1.5	1104.0	55.4	1108.8	11352.8	1111.5	47465.5	1137.2
3	719.6	1.1	722.8	18.5	742.0	4879.7	747.0	23574.2	773.6
4	1074.2	1.3	1074.2	24.8	1136.8	9644.1	1141.1	71291.7	1172.2
5	618.4	0.5	618.4	7.6	619.2	1941.0	620.8	34718.9	625.6
6	769.6	0.9	769.6	15.0	777.6	4079.8	787.0	14473.9	798.0
7	682.2	1.1	682.2	1.3	698.0	201.6	707.6	1363.1	1193.0
8	2627.2	2.3	2637.4	3.0	2748.7	1164.8	2882.1	46287.4	4149.8
9	10085.2	4.1	10089.6	4.6	10630.9	7088.7	—	—	21501.8
10	6590.2	3.2	6655.7	3.6	6917.9	625.5	7169.3	57402.3	8036.2
11	7795.6	4.8	7830.9	6.1	8158.3	1443.9	—	—	11811.8
12	12504.4	5.7	12575.2	8.9	13113.5	6584.7	—	—	23344.2
13	3462.8	2.4	3476.8	3.1	4211.4	619.8	8742.4	16369.9	9340.8

Table 2: Lower bounds

5.3. Experiments with the Default Weights

Our first set of experiments is performed with the default values of the weights, which allow us to compare with the best known results available.

In the default setting, for all instances without multi-spec patients (1–12) we observed experimentally that all best solutions have no transfers. Therefore it is not a surprise that the best solutions have been obtained by M_0. Therefore, we discuss here our experimental analysis for M_0.

For the tuning of M_0, we first select the parameters to be evaluated. To this regard, we decide to use T_0, β, and $\delta = T_0/T_{min}$, which turned out to provide a better selection of the configurations than using T_{min} directly. Given that we use two different types of moves, namely `CR` and `SP`, we add an additional parameter, called sr (for swap rate) which is the probability of drawing a move of type `SP`.

Preliminary experiments show that β is not significant. This is not surprising, because in our setting N is computed from the other parameters, and therefore β only determines the entity of the single step in the temperature and not the actual slope of the cooling trajectory, which is determined by δ. We therefore set β to a fixed value 0.9999.

For the remaining three parameters, we have to select the configurations to be tested. Instead of using a classical full factorial design, which would result in a very large set of configurations, we resort to the *Nearly Orthogonal Latin Hypercubes (NOLH)* proposed by Cioppa and Lucas (2007), that allow us to fill the space using much less configurations. In detail, we set the intervals of the parameters to the following values (based on preliminary runs): $T_0 \in [10^2, 10^4]$, $\delta \in [10^2, 10^4]$, and $sr \in [0.1, 0.7]$.

To generate the actual configurations we use the NOLH spreadsheet made available by Sanchez (2005), using the design with 33 points.

For the comparison of the 33 configurations we resort to F-Race (Birattari, 2005). F-Race is a sequential testing procedure that uses the Friedman two-way analysis of variance by ranks to decide upon the elimination of inferior candidates. At each stage a new instance is selected, all left configurations run on it and weaker configurations are discarded if enough statistical evidence has arisen against them. We use the canonical value 0.05 as significance level in the tests. The transformation of results in ranks prescribed in F-Race guarantees that in the statistical test procedure the aggregation of results over the instances is not negatively influenced by the differences in the cost function values of the instances.

The outcome of the F-Race procedure is that the best configuration is the following: $T_0 = 114.81$, $\delta = 134.89$ ($T_{min} = 0.85$), and $sr = 0.38$. The results for this configuration are presented in Section 5.4 (Table 4) in comparison with previous work.

solver	best	avg	(dev)	#Tr	p-value
M_0	9492.4	9572.64	(47.79)	0	0.005
M_1	9426.8	9541.88	(51.80)	9.92	—
M_0+M_1	9498.8	9600.88	(39.01)	4.01	$9 \cdot 10^{-8}$

Table 3: Results on instance 13 and t-test values

For instance 13, the performances of M_0, M_1, and M_0+M_1 are much more comparable. Table 3 shows the results of the best configuration for all three solvers for 50 runs each. Here, given that we have one single instance, there is no reason to resort to ranks and we could use a more accurate parametric test.

We then apply the *Student's t-test*, provided that the underlying distributions can be assumed to be normal and independent (Venables and Ripley, 2002). If the calculated p-value is below the threshold chosen for statistical significance (typically $p < 0.05$), then the *null hypothesis*, which states that the two groups do not differ, is rejected in favor of an alternative hypothesis, which states that an algorithm is superior to another on the proposed instances. The dash in the column p-value identifies the solver with the minimum average cost. For the others, this column shows the p-value of the comparison between the solver of the current row and the best one.

The outcome is that M_1 is the best solver, with a high statistical confidence (the p-value is clearly below the standard threshold 0.05), but M_0 is superior to M_0+M_1. The table also reports, in the column #Tr, the average number of transfers in the solution. It is interesting to notice that, as expected, M_1 prefers solutions with transfers, but their number is rather limited (< 10) in comparison to the number of multi-spec patients (202).

5.4. Comparison with Best Known Results

Table 4 shows the comparison of the best configuration (of M_0 for instances 1–12 and of M_1 for instance 13) with the ones obtained by Bilgin et al. (2008), which solve only instances 1–6. The timeout used by Bilgin et al. is 3000 seconds, whereas we grant our solver about 400 seconds[2]. The table also reports the percentage gap between the average and the best lower bounds of Table 2.

It is clear from Table 4 that we obtain better results than Bilgin et al. for all instances and in a shorter computational time.

[2]The two PCs are of comparable speed as witnessed by the fact that a benchmark program (the one of the Second Timetabling Competition, see McCollum et al. (2010)) runs exactly in the same time (300s) on our and Bilgin et al.'s PC.

	Our results				Bilgin et al.		
I	Best	Avg.	(Dev)	ΔLB	Avg.	(Dev)	ΔLB
1	659.2	665.61	(33.2)	3.20%	830.36	(18.8)	28.74%
2	1143.6	1150.96	(53.8)	3.55%	1382.28	(14.2)	24.36%
3	776.6	786.67	(57.1)	5.31%	923.16	(20.7)	23.58%
4	1176.0	1190.58	(86.4)	4.34%	1608.68	(29.2)	40.98%
5	625.6	631.87	(24.6)	1.78%	661.52	(4.0)	6.56%
6	801.2	811.18	(62.2)	3.07%	955.04	(20.1)	21.35%
7	1199.0	1216.30	(83.4)	71.89%	–	–	–
8	4158.6	4191.70	(197.6)	45.44%	–	–	–
9	21942.0	22052.83	(905.0)	107.44%	–	–	–
10	8146.6	8261.20	(609.8)	15.23%	–	–	–
11	12016.8	12105.71	(530.4)	48.39%	–	–	–
12	23758.4	23968.80	(788.6)	82.78%	–	–	–
13	9426.8	9541.88	(51.8)	9.14%	–	–	–

Table 4: Comparison with previous work

5.5. Experiments with Reduced Transfer Cost

To understand the relative effect of the constraint Tr, we test M_1 on all instances for several different values of w_{Tr}. All other weights are kept unchanged.

Table 5 shows the result of M_1 for values of w_{Tr} going from 11 to 0. It also shows the average number of transfers in the solution (column $\#\mathsf{Tr}$). The horizontal line represents the threshold below which there is 1% improvement upon the results of M_0 reported in the last row, which can be seen as the level for which the use of transfers becomes profitable for the hospital. This threshold depends on the specific instance, but it is included between $w_{\mathsf{Tr}} = 2.7$ and $w_{\mathsf{Tr}} = 0.1$.

Notice that transfers are used also for values above the threshold, but without actual improvements of the overall cost.

Notice also that, due to the stochasticity of the solver, in some cases the average cost increases when decreasing w_{Tr}. The general trend however remains clear.

5.6. Experiments with Higher Occupancy

As shown in Table 1 the average bed occupancy of these instances is rather low (about 50–60%) compared to real-world situations, in which it easily goes close to 100%. However, this is only the average occupancy, but the actual value varies significantly from day to day, reaching also high peaks in some days.

I	1		2		3		4		5		6		7	
w_{Tr}	Cost	#Tr	Cost	#Tr	Cost	#Tr	Cost	#Tr	Cost	#Tr	Cost	#Tr	Cost	#Tr
11	667.4	0.0	1150.2	0.0	789.9	0.0	1192.7	0.0	631.5	0.0	810.8	0.0	1220.5	0.0
5.5	667.8	0.0	1153.9	0.0	790.6	0.0	1189.6	1.8	629.8	0.0	809.7	0.0	1223.0	0.2
2.7	664.7	0.3	1151.0	0.1	790.1	0.6	1184.9	7.3	633.2	0.1	809.7	0.3	1217.4	10.4
1.3	665.5	5.1	1149.9	8.3	783.6	15.8	1167.1	21.3	633.6	1.2	805.3	7.9	1195.0	30.0
0.6	660.8	13.1	1141.6	28.7	766.4	30.1	1144.0	41.9	630.1	9.7	799.0	19.9	1166.6	54.2
0.3	660.4	32.5	1138.6	65.6	764.8	61.8	1136.9	81.6	632.8	43.0	801.2	50.3	1143.0	65.2
0.1	660.0	114.3	1138.0	212.1	763.9	190.1	1139.2	211.1	632.8	176.3	800.4	171.4	1129.5	94.6
0	647.7	214.3	1116.2	354.6	744.0	313.6	1117.5	347.9	621.6	299.3	780.4	301.5	1117.2	141.3
M_0	665.61	0.0	1150.96	0.0	786.67	0.0	1190.58	0.0	631.87	0.0	811.18	0.0	1216.30	0.0

I	8		9		10		11		12		13	
w_{Tr}	Cost	#Tr	Cost	#Tr	Cost	#Tr	Cost	#Tr	Cost	#Tr	Cost	#Tr
11	4191.4	0.7	22072.5	66.2	8263.6	0.6	12186.0	1.4	24126.5	12.8	9570.6	9.6
5.5	4170.4	10.3	21652.6	149.8	8264.1	16.5	12194.2	42.4	24158.3	99.9	9396.1	48.4
2.7	4130.5	38.8	21196.8	231.2	8190.9	74.4	12023.9	164.5	23794.2	270.7	9168.4	116.8
1.3	4066.6	78.7	20798.3	297.4	8017.3	173.3	11662.4	352.1	23305.7	446.6	8914.6	182.5
0.6	3994.1	118.8	20535.9	343.0	7858.8	259.4	11363.1	484.7	22920.8	590.2	8746.4	238.0
0.3	3943.1	142.5	20393.4	356.7	7735.3	303.5	11204.4	556.6	22693.9	643.2	8666.7	262.0
0.1	3913.8	182.7	20370.8	370.5	7674.8	361.5	11028.7	631.8	22497.0	712.3	8598.3	289.2
0	3888.0	236.9	20233.6	397.7	7655.1	414.8	10977.1	700.9	22419.3	787.9	8548.5	323.3
M_0	4191.70	0.0	22052.83	0.0	8261.20	0.0	12105.71	0.0	23968.80	0.0	9541.88	0.0

Table 5: Results for experiments with M_1 on reduced transfer costs

11		12	
Occupancy	No. Beds	Occupancy	No. Beds
45%	318	55%	283
50%	286	60%	283
55%	264	65%	262
60%	244	70%	241
65%	224	75%	224
70%	210	80%	213
73%	200	81%	210

Table 6: Feature of instances 11 and 12 for increasing levels of occupancy

In order to test our solver on situations with very high occupancy and also to highlight the impact of the occupancy on the quality of the solutions, we have created and tested extra instances with higher occupation. In detail, we have replicated instances 11 and 12 (the largest ones) into a set of modified instances of increasing occupancy, up to 100% in the peak days. This is obtained by removing some of the beds from the original endowment. Table 6 reports the different occupancy rates along with the corresponding number of beds for the new instances. The average occupancy ranges from the one of the original case, reported in the first row, to the highest possible (100% in the peak days).

Figure 8 shows the total cost obtained for 10 runs along with the lower

bounds $LB_{\mathsf{PRC+RG+Tr_7}}$. The outcome is that the total cost grows almost linearly with the occupancy and the gap between lower bounds and results is not significantly worsen. This suggests that the solver is able to solve harder instances without loss of performances.

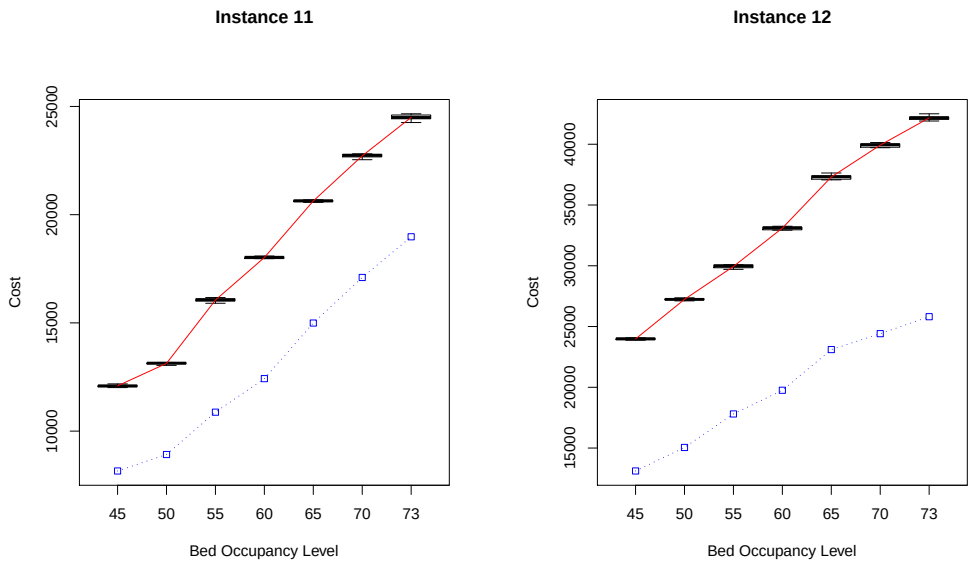

Figure 8: Results on instances 11 and 12 with higher bed occupancy (lower bounds are marked with a dotted line and square points)

6. Application to the Dynamic Case

In this work, we have assumed that all patient admissions are planned in advance (elective patients). However, in many practical cases, this is not the case and patients arrival dates are unpredictable (urgent and emergency patients). In such cases, the problem becomes dynamic and it needs to be solved in real-time, considering new admissions and currently assigned patients. Furthermore, it is also frequent that the discharge date of the patients is unknown as well, because it might depend of the progress of the gradual recovery of the patient.

Our solution technique can be adapted to the dynamic case with some modifications. First, we have to add two notions:

Current day: the current day d_{now} represents the day in which the scheduling takes place. The assignments for all days before d_{now} cannot be changed, because this is assumed to have been already happened.

Registration day: the *registration day* rd_p for each patient $p \in \mathcal{P}$ represents the day in which the admission of the patient becomes known to the system. When running the solver for a given value of d_{now}, only patients already registered ($rd_p \leq d_{now}$) are taken into account, the others are left unscheduled.

For the solution of a dynamic case, the scheduling is repeated, starting with $d_{now} = 0$ and by increasing d_{now} by one at every step. For each value of $d_{now} \geq 1$, the solution of the previous run is stored in the solution variables, and all the assignments for the day $d_{now} - 1$ are set as preassigned. Furthermore, new patients registered in day d_{now} are added to the problem.

The process continues up to the point in which all patients are eventually registered and thus scheduled. The assignment generated by the last invocation provides the final cost of the solution.

In order to test the dynamic version of the solver we would need to have a set of dynamic cases, which unfortunately, up to our knowledge, are not available. Dynamic cases could be generated using simulation techniques with suitable probability distributions (see, e.g., Vissers et al., 2007). However, such techniques are out of the scope of this work.

Nevertheless, we could still experiment on the available instances, by adding the missing information in a reasonable, but arbitrary, way. To this aim, we set for all patients the registration day a fixed number of days n before the admission day. We call n the *forecast level*. From each of the 13 instances of the static case, we thus obtain a set of dynamic instances, one for each value of the forecast level n.

In order to take into account the fact that also the discharge date is uncertain, we modify our solver in such a way that, given a value of d_{now}, the discharge date of the patients is at most equal to $d_{now} + n$. In other word, it is not known whether the patients will remain in the hospital for the days from $d_{now} + n$ onward.

This procedure gives us the possibility to obtain a picture of the behavior of the dynamic solver, by solving the instances for all possible values of n, ranging from 0 to $|\mathcal{D}| - 1$ and compare their results. The value $n = 0$ corresponds to the case in which the behavior of all patients is totally unpredictable; this happens when we have urgent and emergency admissions and situations in which the clinical conditions of the patients can change unexpectedly (e.g., for infections). On the other hand, the value $n = |\mathcal{D}| -$

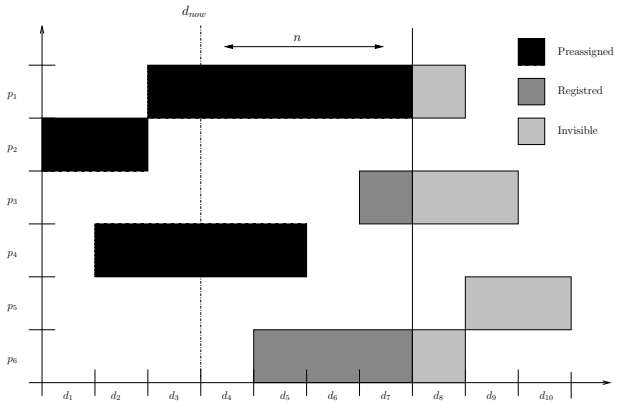

Figure 9: Dynamic case with unknown admission and discharge date

1 corresponds to the fully static case, in which the patients have elective admissions and known discharge dates.

For each value of n, the solver starts with $d_{now} = 0$ and stops when $d_{now} = |\mathcal{D}| - n - 1$. At this point all patients are registered, and from this point on the situation remains stable, thus there is no need to run the solver any further. The number of runs, thus depends on n. For each runs, the number of iterations is a share of the total granted ones.

An example for a give value of d_{now} and n is shown in Figure 9, which highlights a situations where patients have different status: preassigned, registered and invisible. Preassigned patients can not be moved, registered ones are those recently added that have to be scheduled, and invisible patients are those still unknown, thus left unscheduled.

Figure 10 shows the results for 10 runs of M_0 on four selected instances (the others exhibit similar behavior). As expected, for $n = 0$ and $n = 1$ there is a significant deterioration of the quality of the solution. For values $n \geq 2$, the loss of quality is reduced sensibly. This means that a forecast of 2 days is enough to produce a scheduling of reasonable quality.

It is interesting to observe also that for instances 10 and 11, that have a much longer planning horizon, there is a local minimum in the cost around the value $n = 6$. This is quite unexpected, because the intuition is that the longer is the forecast the better is the solution. However, given that the solver is not exact, we experience the fact that solving iteratively smaller instances turns out to be more effective that solving larger ones in one time. Notice however that the global minimum is for $n = |\mathcal{D}| - 1$, showing that the effect of this decomposition is eventually surpassed by the effect of having full information on future admissions and discharges.

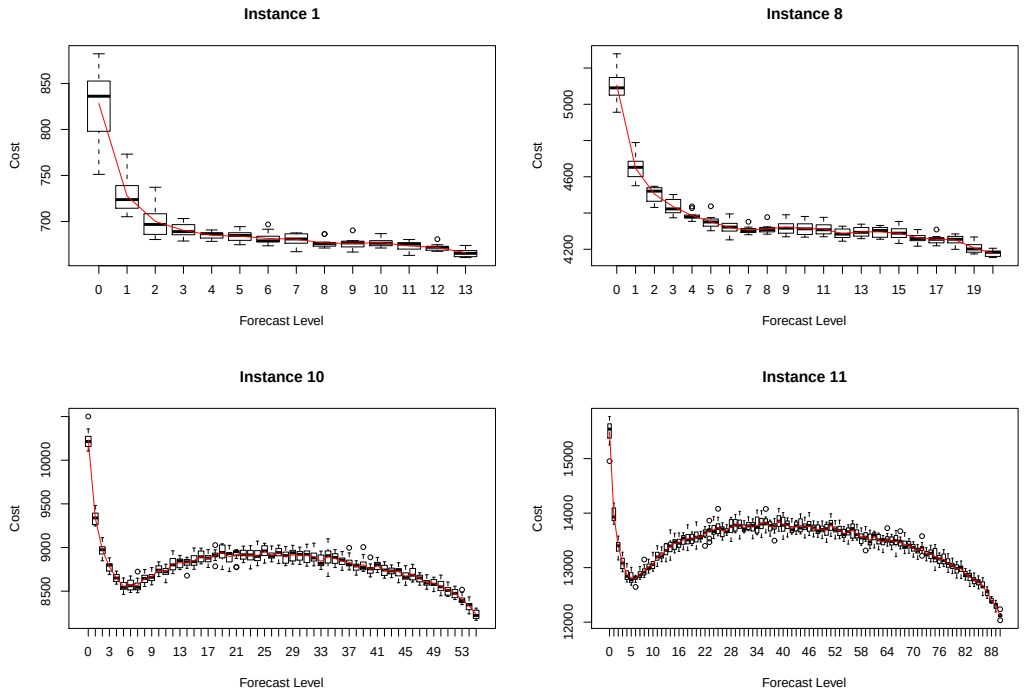

Figure 10: Performance on instances 1, 8, 10, and 11 for the dynamic case

7. Conclusions

We have proposed two local search methods M_0 and M_1 for the PAS problem. Both solvers outperform previous work on the problem, which is based on hyper-heuristics, in terms of quality and computational time. We believe that this difference in performance and speed can be attributed to both the preprocessing steps and, most importantly, the new neighborhood combinations.

We have also shown that the relative performance of M_0 and M_1 depends, unsurprisingly, on the value of the weight of a crucial feature, namely the patient transfer.

In addition, we have shown that our results are in some cases relatively close to the lower bound provided by the relaxation of the integrality constraints.

We have also tested our solver on instance with higher occupancy, showing that it is able to solve these cases too.

Finally, we have studied the behavior of the solver on the case in which patients admission and discharge dates are not known in advance, but become known during the planning period itself. For this dynamic case, we have shown the relationship between the solution quality and the forecast level.

For the future, we plan to explore new combinations of neighborhoods and metaheuristics to obtain better results, to experiment with new (possibly more challenging) instances, and to explore further the dynamic case.

We also plan to explore new problem formulations that could be useful in other healthcare settings. For example, it it interesting to consider in the model the possibility to delay the admission of a patient in order to improve the overall objective function.

Acknowledgements

We thank Peter Demeester and Greet Vanden Berghe for setting up the PAS website, for assisting us in the use of the validation, and for many fruitful discussions about the PAS problem.

We also thank Luca Di Gaspero for helping us in the use of the R package for the statistical analysis and Ruggero Bellio for introducing the NOLH design to us.

References

Aarts, E., Lenstra, J. K., 1997. Local Search in Combinatorial Optimization. John Wiley & Sons, Chichester.

Bilgin, B., Demeester, P., Vanden Berghe, G., 2008. A hyperheuristic approach to the patient admission scheduling problem. Tech. rep., KaHo Sint-Lieven, Gent.

Birattari, M., April 2005. The RACE package.
URL `http://cran.r-project.org/web/packages/race/`

Blake, J. T., Dexter, F., Donald, J., 2002. Operating room managers' use of integer programming for assigning block time to surgical groups: A case study. Anesth. Analg. 94 (1), 143–148.

Brandeau, M. L., Sainfort, F., Pierskalla, W. P., 2004. Operations research and health care: a handbook of methods and applications. Springer.

Burke, E. K., De Causmaeker, P., Vanden Berghe, G., Van Landeghem, H., 2004. The state of the art of nurse rostering. Journal of Scheduling 7, 441–499.

Cioppa, T. M., Lucas, T. W., 2007. Efficient nearly orthogonal and space-filling latin hypercubes. Technometrics 49 (1), 45–55.

Demeester, P., 2009. Patient admission scheduling website. `http://allserv.kahosl.be/~peter/pas/`, accessed 24 September 2013.

Demeester, P., De Causmaecker, P., Vanden Berghe, G., 2008. Applying a local search algorithm to automatically assign patients to beds. In: Proceedings of the 22nd Conference on Quantitative Methods for Decision Making (Orbel 22). pp. 35–36.

Demeester, P., Souffriau, W., De Causmaecker, P., Vanden Berghe, G., January 2010. A hybrid tabu search algorithm for automatically assigning patients to beds. Artificial Intelligence in Medicine 48 (1), 61–70.

Dexter, F., Macario, A., 2002. Changing allocations of operating room time from a system based on historical utilization to one where the aim is to schedule as many surgical cases as possible. Anesthesia Analgesia 94 (5), 1272–1279.

Di Gaspero, L., Schaerf, A., 2003. EasyLocal++: An object-oriented framework for flexible design of local search algorithms. Software—Practice and Experience 33 (8), 733–765.

Di Gaspero, L., Schaerf, A., 2006. Neighborhood portfolio approach for local search applied to timetabling problems. Journal of Mathematical Modeling and Algorithms 5 (1), 65–89.

Dumas, M. B., August 1984. Simulation modeling for hospital bed planning. Simulation 43 (2), 69–78.

Dumas, M. B., April 1985. Hospital bed utilization: an implemented simulation approach to adjusting and maintaining appropriate levels. Health services research 20 (1), 43–61.

Gemmel, P., Van Dierdonck, R., 1999. Admission scheduling in acute care hospitals: does the practice fit with the theory? International Journal of Operations & Production Management 19 (9), 863–878.

Goldman, J., Knappenberger, H. A., Eller, J. C., January 1968. Evaluating bed allocation policy with computer simulation. Health services research 3 (2), 119–29.

Hans, E. W., Wullink, G., van Houdenhoven, M., Kazemier, G., 2008. Robust surgery loading. European Journal of Operational Research 185 (3), 1038–1050.

Harper, P. R., Shahani, A. K., 2002. Modelling for the planning and management of bed capacities in hospitals. The Journal of the Operational Research Society 53 (1), 11–18.

Hoos, H. H., Stützle, T., 2005. Stochastic Local Search – Foundations and Applications. Morgan Kaufmann Publishers, San Francisco, CA (USA).

IBM ILOG, 2009. CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, v. 12.1.

Jebali, A., Hadjalouane, A., Ladet, P., January 2006. Operating rooms scheduling. International Journal of Production Economics 99 (1-2), 52–62.

Johnson, D. S., 2002. A theoretician's guide to the experimental analysis of algorithms. In: Goldwasser, M. H., Johnson, D. S., McGeoch, C. C. (Eds.), Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges. American Mathematical Society, pp. 215–250.
URL http://www.research.att.com/~dsj/papers.html

Johnson, D. S., Aragon, C. R., McGeoch, L. A., Schevon, C., 1989. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. Operations Research 37 (6), 865–892.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., Schevon, C., 1991. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. Operations Research 39 (3), 378–406.

Kao, E. P. C., Tung, G. G., 1981. Bed allocation in a public health care delivery system. Management Science 27 (5), 507–520.

Kirkpatrick, S., Gelatt, D., Vecchi, M., 1983. Optimization by simulated annealing. Science 220, 671–680.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Di Gaspero, L., Qu, R., Burke, E. K., 2010. Setting the research agenda in automated timetabling: The second international timetabling competition. INFORMS Journal on Computing 22 (1), 120–130.

Mullen, P. M., May 1994. Waiting lists in the post-review NHS. Health services management research : an official journal of the Association of University Programs in Health Administration / HSMC, AUPHA 7 (2), 131–45.

Rousseau, L.-M., Gendreau, M., Pesant, G., 2002. A general approach to the physician rostering problems. Annals of Operations Research 115, 193–205.

Sanchez, S. M., 2005. NOLH designs spreadsheet. `http://diana.cs.nps.navy.mil/SeedLab/`, visited on May 13, 2011. Last updated on April 7, 2006.
URL `http://diana.cs.nps.navy.mil/SeedLab/`

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

Tànfani, E., Testi, A., May 2010. A pre-assignment heuristic algorithm for the master surgical schedule problem (mssp). Annals of Operations Research 178 (1), 105–119.

Thomson, J., Dowsland, K., 1995. General cooling schedules for simulated annealing-based timetabling system. In: Proc. of the 1st Int. Conf. on the Practice and Theory of Automated Timetabling (ICPTAT-95). pp. 345–363.

Triki, E., Collette, Y., Siarry, P., Oct. 2005. A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. European Journal of Operational Research 166 (1), 77–92.

van Laarhoven, P. J. M., Aarts, E. H. L., 1987. Simulated annealing: theory and applications. Kluwer Academic Publishers.

van Oostrum, J. M., Bredenhoff, E., Hans, E. W., August 2010. Suitability and managerial implications of a master surgical scheduling approach. Annals of Operations Research 178 (1), 91–104.

Vassilacopoulos, G., November 1985. A simulation model for bed allocation to hospital inpatient departments. Simulation 45 (5), 233–241.

Černý, V., 1985. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications 45 (l), 41–51.

Venables, W. N., Ripley, B. D., 2002. Modern applied statistics with S, 4th Edition. Statistics and Computing. Springer.

Vissers, J. M., Adan, I. J., Dellaert, N. P., Aug. 2007. Developing a platform for comparison of hospital admission systems: An illustration. European Journal of Operational Research 180 (3), 1290–1301.

Williams, P., Tai, G., Lei, Y., June 2009. Simulation based analysis of patient arrival to health care systems and evaluation of an operations improvement scheme. Annals of Operations Research 178 (1), 263–279.

Worthington, D., 1991. Hospital waiting list management models. The Journal of the Operational Research Society 42 (10), 833–843.

Worthington, D. J., 1987. Queueing models for hospital waiting lists. The Journal of the Operational Research Society 38 (5), 413–422.