Modeling and solving the dynamic patient admission scheduling problem under uncertainty

Sara Ceschia[*], Andrea Schaerf[*]

Department of Electrical, Mechanical and Management Engineering, University of Udine, via delle Scienze 206, I-33100, Udine, Italy

Abstract

Objective. Our goal is to propose and solve a new formulation of the recently-formalized *patient admission scheduling* problem, extending it by including several real-world features, such as the presence of emergency patients, uncertainty in the length of stay, and the possibility of delayed admissions.

Method. We devised a metaheuristic approach that solves both the static (predictive) and the dynamic (daily) versions of this new problem, which is based on simulated annealing and a complex neighborhood structure.

Results. The quality of our metaheuristic approach is compared with an exact method based on integer linear programming. The main outcome is that our method is able to solve large cases (up to 4000 patients) in a reasonable time, whereas the exact method can solve only small/medium-size instances (up to 250 patients). For such datasets, the two methods obtain results at the same level of quality. In addition, the gap between our (dynamic) solver and the static one, which has all information available in advance, is only 4-5%. Finally, we propose (and publish on the web) a large set of new instances, and we discuss the impact of their features in the solution process.

Conclusion. The metaheuristic approach proved to be a valid search method to solve dynamic problems in the healthcare domain.

[*]Corresponding author. Tel.: +39 0432 558272, fax: +39 0432 558251
Email addresses: sara.ceschia@uniud.it (Sara Ceschia), schaerf@uniud.it (Andrea Schaerf)

Keywords: local search, simulated annealing, length of stay uncertainty, patient admission scheduling

1. Introduction

The *patient admission scheduling* (PAS) problem consists in assigning patients to hospital rooms in such a way as to maximize medical treatment effectiveness, management efficiency, and patient comfort. PAS has been formalized by Demeester et al. [1] and further studied by the same research group [2].

In the proposed PAS formulation, each patient has fixed admission and discharge dates and one or more treatments to undergo. Each room is characterized by its equipment and location which may or may not be suitable for a specific patient. In addition, there is a room gender policy that forbids, for regular rooms, the simultaneous presence of male and female patients. Finally, a patient should stay in the same room throughout her/his hospitalization; a change of room (called *transfer*) is undesirable and thus penalized in the objective function.

The problem consists in assigning patients to beds in rooms for each day of their stay in hospital, minimizing the costs and respecting the capacity constraint and the gender policy. The underlying decision problem has been recently proved to be NP complete [3].

In this formulation, it is implicitly assumed that all admission dates are known in advance, and that the problem can be solved just once for the whole planning period. Unfortunately, this *static* version of the problem has little usefulness for most practical cases where patients might arrive at unpredictable times (urgent and emergency patients) [4, 5]. Furthermore, it also frequently happens that the discharge date of the patient is unknown, because it might depend on the progress of her/his gradual recovery [6, 7]. In these cases, the static solution of the problem can provide only a predictive assignment that needs to be subsequently modified several times.

The actual problem that hospitals normally face is the *dynamic* (or *daily*) version, in which the presence of the patient becomes known to the system on her/his registration day, which can also happen just before her/his admission.

According to Ouelhadj and Petrovic [8], our problem belongs to the class of *predictive-reactive scheduling* in which the schedule is periodically adjusted in response to real-time events (*rolling time horizon* policy). Specifically, the dynamic problem is decomposed into a series of static problems which take into account new information collected from the current situation. In our case, for each day, the system schedules all the patients known at that time, i.e. the ones already registered.

In our previous work [9], we have solved the static version of the problem by local search and we have obtained the best known solutions on all the available PAS benchmarks [10] `http://allserv.kahosl.be/~peter/pas/` (Accessed: 11 September 2012). In the same work [9], we deal also with the dynamic version, which however is greatly simplified by assuming that all patients become known to the system a fixed number of days before their admission.

Following Gemmel and Van Dierdonck [5], we believe that many further steps have to be made to design a problem formulation that captures the real dynamic situations that normally happen in hospitals. To this end, in this work we define a new version of PAS (both static and dynamic) that comprises, among others, the registration day for the patients, the uncertainty of the length of stay, and the possibility of delaying the admission of a patient.

Indeed, in practice it is normally possible to delay the entrance of a (non-urgent) patient, if it helps to improve the overall quality of service for the hospital. In addition, the length of stay of the patients is supposed to be fixed in PAS. However, in our model the length of stay is uncertain, and we consider the possibility that a patient might remain *longer* than expected.

As presented by Herroelen and Leus [11], different approaches exist to deal with uncertainty in scheduling: probabilistic distributions, fuzzy models, redundancy-based and proactive techniques. We follow the latter approach, by including an additional cost component in the objective function that accounts for the risk that a room is overcrowded on some specific day, due to the probability of the over-length stay of some patients assigned to the room.

Foreshortened stays are also possible, but, as customary in the literature, we do not take them into account. The motivation is twofold. Firstly, they have a less significant impact in the process, because they do not lead to conflicts, but only to under-usage of beds. Secondly, they happen less frequently; in fact, the typical distributions of the length of stay of patients are heavily skewed to the right, with a large peak at the start, which then gradually tails off as duration increases (see [12]).

The most significant extension is the notion of delay, which leads to a new search space, in which the admission days of the patients are additional decision variables.

For this problem, we propose a metaheuristic approach based on local search, partly built upon the one used in [9], but using new neighborhood operators.

The main outcome of the paper is that our approach obtains results at the same level as an exact solver based on integer linear programming. In addition, it is able to solve much larger cases in a reasonable computational time.

As a side product of this research, we have defined a large number of new challenging instances, that could be used for future comparisons. They are available at `http://satt.diegm.uniud.it/index.php?page=pasu`, along with their best solutions, their generator, and a solution validator.

2. Dynamic patient admission scheduling problem under uncertainty

In this section we describe the new formulation of the PAS problem, that we call PASU (U for uncertainty). There are several basic notions for the PASU problem:

Day: This is the unit of time and is used to express the length of the planned stay of each patient in the hospital. To be more precise, we should refer rather to the notion of *night*, but it is however customary to use the term day in its general sense of a 24 h period. The set of (consecutive) days considered in the problem is called the *planning horizon*.

Patient: She/he is a person who needs some medical treatment, and consequently must spend a period in the hospital, so that she/he must be placed in a bed in a room.

Patients are split into two groups: *in-patients* that are already present in the hospital at the time of scheduling and *new patients* that have to be admitted.

Each new patient has a planned admission date and a discharge date within the planning horizon. The actual admission might be delayed with respect to the planned one, but not more than a given maximum, which is related to her/his health condition.

In addition, the new patient has a *registration day* in which she/he becomes known to the system. This is equal to the admission day for emergency patients, but can be considerably earlier for elective ones. Only patients registered before the current day are included in the planning.

For in-patients, the registration day has no meaning (it is implicitly equal to 0) and it is obviously not possible to delay the admission because these patients have been already admitted. For in-patients it is necessary to take into account the room in which she/he has spent the previous night. If the in-patient is assigned to a different one, it is necessary to transfer her/him during the day. Transfers are clearly undesirable and should be minimized.

Finally, a patient might have an *overstay risk* depending on her/his health condition, which introduces the possibility that she/he might need to spend one extra night in the hospital.

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

Specialism: Each patient needs one specific specialism for her/his treatment. Departments might be fully qualified for the treatment of a disease, partially qualified, or not qualified at all. The assignment of a patient to a department that is not qualified for the treatment of her/his disease is not feasible; while the assignment to a department partially qualified is possible, but contributes to the cost of the solution.

Room feature: Each room has different features (oxygen, telemetry, ...) necessary for the treatment of particular pathologies. Patients may *need* or simply *desire* specific room features. The assignment of a patient to a room without a needed feature is not feasible, whereas the missing desired features contribute to the objective function.

Room gender policy: Each room has a *gender policy*. There are four different policies, identified by the elements in the set {SG, Fe, Ma, All}. In rooms with policy Fe (resp. Ma) only female (resp. male) patients can be accepted. If the policy is SG the room can be occupied by patients of both genders, but on any day the patients in the room must be all of the same gender. Finally, rooms of policy All can be occupied simultaneously by patients of both genders (e.g., intensive care).

Age policy: Some departments are reserved for 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 PASU problem consists in assigning a room to each patient for a number of consecutive days equal to her/his stay period. The actual admission day can be later (but not earlier) than the planned admission day.

As customary, constraints are split into *hard constraints*, that must be satisfied, and *soft constraints* (or objectives) that can be violated and contribute to the objective function.

Table 1 lists the constraints involved in the PASU problem. The constraint RC is obviously a *hard* one, given that the simultaneous assignment of two patients to the same bed clearly makes the solution unfeasible. The constraint PA is also hard, whereas the constraints DS and RF are both hard and soft. In detail, they are hard for the missing qualification and the missing needed features, but soft for partial qualification and the desired features, respectively.

The remaining constraints, namely RP, RG, Tr, De, and OR are soft constraints.

The De constraint takes into account the discomfort for the patient caused by a delayed admission, and it simply counts the number of days of delay.

The OR constraint sums up, for each room for each day, the difference between the number of patients, including both sure and potentially present ones, and the capacity of the room.

Finally, the Tr constraint counts the number of in-patients that are assigned to a room different from the previously occupied one.

Each soft constraint is associated with a weight w_*, that accounts for its relative importance. In practical cases, the weights are assigned by the final user, based on the specific situation, regulation, and internal policy. For our experimental analysis, we fix them to the “default” values set in Table 1.

Analogously to the original PAS, the PASU problem can be simplified by means of a preprocessing step (see [9]). Specifically, it is evident that all the constraints (hard and soft) related to departments, specialisms, room features, age policy, preferences, and transfer contribute, with their weights, to the penalty of assigning a given patient to a given room. Therefore, we

Table 1: PASU hard and soft constraints.

Constraint	Type	Default weight
Room capacity (RC)	Hard	—
Room gender (RG)	Soft	50
Department specialism (DS)	Hard/Soft	20
Room features (RF)	Hard/Soft	20
Patient age (PA)	Hard	—
Room preference (RP)	Soft	10
Transfer (Tr)	Soft	100
Delay (De)	Soft	2
Overcrowd risk (OR)	Soft	1

can "merge" together this information into two matrices that represent the suitability of a room for a patient (hard constraints: boolean-valued) and the integer-valued cost of the assignment of the patient to the room. Thus, the *patient-room suitability matrix* A and *patient-room penalty matrix* C are computed just once; all the five features mentioned above (departments, specialisms, room features, age, preferences and transfers) can be removed from the formulation.

The penalty associated with the room gender policy RG can also be *partly* included in matrix C. More specifically, if the room is of type Fe or Ma, then the penalty of accepting a male patient or a female patient is merged into 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 SG, which is actually the most common one.

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

Patient-room suitability (PRS): The patient p cannot be assigned to the room r if $A_{p,r} = 0$.

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 two hard constraints, namely RC and PRS, and four cost components (or soft constraints): RG (case SG only), PRC, De, and OR.

3. Test instances

At present unfortunately no real world instance is available, therefore we have decided to design a parametrized generator that can produce realistic data for a large set of different sizes. Although the use of real data is clearly preferable, the generator allows us to have at our disposal an arbitrary number of instances, thus making it impossible to hand-tune algorithms to a particular small set of instances.

The generator receives as parameters the number of departments, rooms, features, patients, and days. It creates a random instance based on predefined distributions concerning various features such as the length of stay, the room capacity, the number of specialisms, and so on.

Our generator does not create single-day cases, but rather long-term scenarios that require the solution of sequences of single cases for each day. The overall solution process starts at day $d = 0$ in which no in-patient is present and only patients registered at day 0 are considered. For each subsequent day d, patients that the solver has scheduled at day $d - 1$ to be admitted on the same day become in-patients (and their room is stored). Patients scheduled for the following days are kept in the problem and can be rescheduled arbitrarily, and new patients registered at day d are added to the problem. The overall cost of the solution is simply the sum of all the costs of the complete solution on the final day.

We have created 9 sets of 50 instances each, using the values shown in Table 2. The datasets correspond to three different sizes in terms of number of patients and planning horizons. When the horizon is doubled, the number of patients is doubled as well so as to maintain approximately the same average bed occupancy.

Instances are generated to be highly dynamic. In fact, using the degree of dynamism, called *reaction time* and proposed by Larsen [13] for the *dynamic vehicle routing problem*, our instances reach the average value of 0.77 (the index takes value 1 if the problem is totally dynamic). In our case, this index corresponds to the level of urgency and is related to the average difference between the registration day and the planned admission day.

The generator is available on the web together with all the instances.

4. Solution techniques

We propose both an *integer linear programming* (ILP) model (Section 4.1) and a local search method (Section 4.2). The results of the two techniques

Table 2: Description of the instances.

Family	Depts	Rooms	Features	Patients	Specialisms	Days
1. Small short	4	8	4	50	3	14
2. Small mid	4	8	4	100	3	28
3. Small long	4	8	4	200	3	56
4. Med short	6	40	5	250	10	14
5. Med mid	6	40	5	500	10	28
6. Med long	6	40	5	1000	10	56
7. Large short	8	160	6	1000	15	14
8. Large mid	8	160	6	2000	15	28
9. Large long	8	160	6	4000	15	56

are compared in terms of solution quality in Section 5.2.

4.1. Integer linear programming

The search space of the PASU problem is very large, because there are also the decision variables about the admission days of patients. As a consequence, we could not obtain an optimal solution for large instances. Therefore, we introduce a variant of the problem in which delays are not permitted. In this case, the problem is greatly simplified, given that it concerns only the assignment of a room to each patient. We add the superscript f (for fixed) to identify this simpler problem.

We first introduce the mathematical model for PASU^f, then we sketch the model of the full problem PASU.

We start by introducing some terminology. We call:

- $\mathcal{P}$: the set of patients
- $\mathcal{P}_F$, $\mathcal{P}_M$ the sets of female and male patients (with $\mathcal{P}_F \cup \mathcal{P}_M = \mathcal{P}$)
- $\mathcal{P}_H$ the set of in-patients and r_p is the room occupied by in-patient $p \in \mathcal{P}_H$
- $\mathcal{D}$: the set of days
- $\mathcal{R}$: the set of rooms and c_r is the capacity of room $r \in \mathcal{R}$
- $\mathcal{R}_{\mathsf{SG}}$: the subset of rooms that have policy SG

In addition, we call:

- $\mathcal{D}_p$: the set of days in which a patient $p \in \mathcal{P}$ is present in the hospital
- $\mathcal{P}_d$: the set of patients present in day d (i.e., set of patients p such that $d \in \mathcal{D}_p$)

The main decision variables are the following:

- $x_{p,r}$: 1 if patient p is assigned to room r, 0 otherwise

The constraints on the x variables are:

$$\sum_{r \in \mathcal{R}} x_{p,r} = 1, \qquad \forall\, p \in \mathcal{P} \tag{1}$$

$$\sum_{p \in \mathcal{P}_d} x_{p,r} \leq c_r, \qquad \forall\, d \in \mathcal{D}, r \in \mathcal{R} \tag{2}$$

$$x_{p,r} \leq A_{p,r} \qquad \forall\, p \in P, r \in \mathcal{R} \tag{3}$$

Constraints (1) ensure that every patient is assigned to exactly one room. Constraints (2) ensure that the rooms are not overloaded (RC). Constraints (3) provide against unfeasible assignments due to patient-room unsuitability (PRS).

The x variables describe the actual search space of the problem. We need, however, several other variable sets, so as to express the components of the objective function F. We first introduce the variables for the management of the RG component:

- $f_{r,d}$, $m_{r,d}$: 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

These new variables are related to the x and to each other by the following constraints:

$$f_{r,d} \geq x_{p,r}, \qquad \forall\, p \in \mathcal{P}_F, r \in \mathcal{R}, d \in \mathcal{D}_p \tag{4}$$

$$m_{r,d} \geq x_{p,r}, \qquad \forall\, p \in \mathcal{P}_M, r \in \mathcal{R}, d \in \mathcal{D}_p \tag{5}$$

$$b_{r,d} \geq m_{r,d} + f_{r,d} - 1, \qquad \forall\, r \in \mathcal{R}, d \in \mathcal{D} \tag{6}$$

$$\tag{7}$$

Constraints (4–5) relate the auxiliary variables f and m to x, stating that whenever a female (resp. male) patient is in the room, then all the f (resp. m) variables corresponding to the days $d \in \mathcal{D}_p$ must be set to 1. Constraints (6) relate b to m and f, by implying that if m and f are both 1, then b must be 1.

We now move to the auxiliary variables for the modeling of the OR component:

- $y_{r,d}$: 1 if room r risks being overcrowded in day d, 0 otherwise

In order to define the constraints relating y variables to x, we need the following additional definitions. We call P_d^+ the set of patients potentially present in day d, that is the patients present in day d plus those present in day $d-1$ with the risk of overstay.

The constraints relating y to x are then the following ones:

$$\sum_{p \in \mathcal{P}_d^+} (1 - x_{p,r}) \geq (|P_d^+| - c_r) \cdot (1 - y_{r,d}) \qquad \forall d \in \mathcal{D}, r \in \mathcal{R} \tag{8}$$

Intuitively, when $y_{r,d} = 1$ the variables $x_{p,r}$ can take any value. Conversely, when $y_{r,d} = 0$ then at least $|P_d^+| - c_r$ of the x involved must take the value 0 (implying that at most c_r can take the value 1).

The objective function is computed as follows (the De component is equal to 0 for the PASU^f problem).

$$F = F_{\mathsf{PRC}} + F_{\mathsf{RG}} + F_{\mathsf{OR}} \tag{9}$$

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

$$F_{\mathsf{PRC}} = \sum_{p \in \mathcal{P}, r \in \mathcal{R}} C_{p,r} \cdot x_{p,r} \cdot |\mathcal{D}_p| \tag{10}$$

$$F_{\mathsf{RG}} = \sum_{r \in \mathcal{R}_{\mathsf{SG}}, d \in \mathcal{D}} w_{\mathsf{RG}} \cdot b_{r,d} \tag{11}$$

$$F_{\mathsf{OR}} = \sum_{r \in \mathcal{R}, d \in \mathcal{D}} w_{\mathsf{OR}} \cdot y_{r,d} \tag{12}$$

Eq. (10) accounts for the cost of each patient-room assignment. Eq. (11) accounts for the number of rooms occupied by both male and female patients. Finally, Eq. (12) estimates the overcrowd risk.

This concludes the model for PASU^f. For the sake of brevity, we do not include the much more complex model for the full problem. Intuitively, it includes a three-dimensional matrix of decision variables z, such that $z_{p,r,d} = 0$ if and only if patient p is in room r in day d. Several additional variables and constraints are necessary so as to force z to assume feasible values. For example, it is necessary to ensure that the 1's in the z matrix are consecutive, and their number is equal to the length of stay of the patient.

Both problems are modeled as ILP problems, and they can be directly implemented in any general-purpose ILP solver. In our case, they have been coded using CPLEX 12.

4.2. Local search

Here we describe the features of our local search approach. We start with the search space, the initial solution, and the cost function. Subsequently, we discuss the neighborhood relations, and finally we introduce the metaheuristic technique used, namely simulated annealing.

4.2.1. Search space, initial solution, and cost function

A state in the search space is represented by two integer-valued vectors of size $|\mathcal{P}|$. The first one represents the room assigned to each patient, and the second one is the delay of the patient.

From the search space we remove states that violate PRS constraints, whereas the RC constraints can be violated and are taken care of in the cost function.

The initial solution is constructed in such a way that there are no transfers and no delays. That is, in-patients are assigned to their previous room and all delays are set to 0. The room of the new patients is selected at random, but still satisfying PRS constraints.

The cost function f includes the cost components of the PASU problem: RG (room gender), PRC (patient-room cost), OR (overbooking risk), and De (delay discomfort). In addition, it takes into account, with an appropriate high weight W, the constraint RC (room capacity), which accounts for the violations of the capacity constraints (also called *distance to feasibility*).

4.2.2. Neighborhood structure

The neighborhood used is the composition of three basic moves. The first two work on the room assignment, and they are the change of the room assigned to a patient (called `CR`, for Change Room) and the swap of the assigned room between two patients (`SP`, for Swap Patients).

The third neighborhood is used to explore the larger search space in which admissions of patients can be delayed. It is called `S` (for Shift) and it shifts the admission of a patient forward or backward *by one day*, keeping the room unchanged.

The complete neighborhood considered is thus $\mathtt{CR} \oplus \mathtt{SP} \oplus \mathtt{S}$, where the symbol $\oplus$ stands for neighborhood union according to the terminology used in [14]. From this neighborhood we obviously exclude moves that lead outside the search space. For example, `SP` moves that assign a patient to a room that does not satisfy the PRS constraints, or `S` moves that shift the admittance over the maximum admissible delay, are not permitted.

4.2.3. Simulated annealing

Simulated annealing (SA) was proposed by Kirkpatrick et al. [15] and Černý [16] and extensively studied by many authors, including Aarts and Korst [17] and van Laarhoven and Aarts [18] among others.

There are many different variants of SA. We describe here the one used in our solver. The process starts by creating a random initial state s_0, and setting it as the current state. The main procedure consists of a loop that randomly generates, at each iteration, a neighbor of the current state. Given a move m, we denote with Δf the difference in the cost function between the new state and the current one, i.e., $\Delta f = f(s \odot m) - f(s)$ (where the operator $\odot$ denotes the execution of a move in a given state). If $\Delta f \leq 0$ the new state is accepted and becomes the current one. In addition, if the value of the cost function of the current state is lower than that of the best state so far found, then the best state is updated to the current one. Conversely, if $\Delta f > 0$ the new state is accepted with probability $e^{-\Delta f/T}$, where T is a parameter, called the *temperature*.

The temperature T is initially set to a high value T_0. After a fixed number of iterations N, the temperature is decreased by the *cooling rate* α, so that at each cooling step n we set $T_n = \alpha \times T_{n-1}$. The procedure stops when the temperature reaches a *low-temperature level* $T_{\min}$, that is when no state that increases the cost function is accepted anymore.

The control parameters of the procedure are the cooling rate α, the number of neighbors sampled at each temperature N, and the starting and final temperatures T_0 and T_{min}.

Given that we use a composite neighborhood, the random selection mechanism prescribed by SA needs to be further detailed. In our case, it is performed by selecting first the neighborhood used, and then the specific move within the neighborhood. Regarding the selection of the neighborhood, the probability is not uniform (i.e. 1/3 each), but set to three given values $p_{\texttt{CR}}$, $p_{\texttt{SP}}$, and $p_{\texttt{S}}$ (with $p_{\texttt{S}} = 1 - p_{\texttt{CR}} - p_{\texttt{SP}}$). The values of $p_{\texttt{CR}}$ and $p_{\texttt{SP}}$ are subject to tuning, along with the other SA parameters.

In addition, the SA procedure prescribes that the move acceptance is based on Δf, therefore the value of W is crucial for the performance of our solver. In fact, if W is too high the initial temperature needs to be set to a very high value, which would result in a waste of time for the search. On the other hand, if W is too small it is possible that the solver could follow trajectories that "prefer" unfeasible states to feasible ones, if they have a lower objective cost. In conclusion, W also needs to be set experimentally, as discussed in Section 5.

5. Experimental analysis

For the ILP solver we use the default CPLEX configuration, which usually leads to good results. We decided to grant 60 seconds for the run on a single day of the horizon. The total running time of an instance is thus at most 840s, 1680s, or 3360s depending on the length of the planning horizon (14, 28, or 56 days).

The SA solver was tuned using a tool for automated tuning. The time granted is still 60 seconds for each day. This is not enforced using a software timeout, but selecting the parameter configurations so that the total number of iterations is fixed.

The software is written in C++ language, it uses the framework EasyLocal++ [19], and it is compiled using the GNU C/C++ compiler, v. 4.4.3, under Ubuntu Linux. All experiments were run on an Intel Core i7 @1.6 GHz PC (64 bit).

5.1. Parameter tuning for SA

Metaheuristics, such as SA, have a number of configurable parameters, and the performance depends significantly on their particular setting. We tune our SA solver using an automatic tool, namely the *iterated racing procedure* (I/F-Race) recently introduced by Birattari *et al* [20], which allows us to explore the space of parameters effectively on a large set of instances, and to verify the robustness of our methods. In addition, I/F-Race uses principled tests to assess if a configuration is statistically superior to another one with a predefined confidence (typically 95%).

In summary, SA has four parameters to tune: start temperature T_0, stop temperature T_{min}, cooling rate α, and number of neighbors sampled at each temperature N. In addition, the values of the probabilities $p_{\texttt{CR}}$ and $p_{\texttt{SP}}$, and the weight W are subject to tuning, too.

We decided to tune $\delta = T_0/T_{min}$ instead of T_{min}, because it turned out to provide a better distribution of the configurations than using T_{min} directly. In order to grant a similar amount of time to the same instance for each parameter configuration, we fixed the total number of SA iterations to $I = 10^8$, and we computed the number of neighbors sampled for each temperature from the other parameters, as $N = -I/\log_\alpha(\delta)$. Preliminary experiments showed that in this setting α is not significant, we therefore set α to the fixed value 0.999.

We set a tuning budget for I/F-Race of 1000 experiments. We used the following domains for the parameter settings: $T_0 \in [10^{1.5}, 10^{4.5}]$, $\delta \in [10^{2.5}, 10^6]$ and $W \in [10^2, 10^3]$, $p_{\texttt{CR}} \in [0.25, 0.6]$, $p_{\texttt{SP}} \in [0.25, 0.6]$. We used the full set of all 450 instances as training set, so that it is not specifically tuned for a particular size or structure.

The outcome of the I/F-Race procedure has been that the best configuration is $T_0 = 10^2$, $\delta = 10^{2.5}$, $W = 256$, $p_{\texttt{SP}} = 0.28$ and $p_{\texttt{SP}} = 0.57$.

5.2. Computational results

The results obtained for the PASU problem by SA (in the configuration found above) are shown in Table 3, along with the best results found by the ILP solver. SA was run 10 times on each instance, collecting both the average and the best result for each instance. Each entry reports the averages upon the 50 instances of each family.

The best value of each single instance, together with the corresponding solution, is published on our web site. For the ILP solver, the $*$ symbol marks the fact that for all instances the solver obtained the optimal solution.

In some cases, the solver was not able to solve to feasibility all the instances of the family within the time granted, therefore we report in parenthesis the number of instances solved. The average and best values are com-

Table 3: Results of the ILP and SA solvers on the PASU problem.

Family	ILP_{PASU}		SA_{PASU}			Both
	Avg	Solved	Avg	Solved	Best	Solved
Small short	2,768.82*	(50)	**2,761.07**	(50)	2,723.02	(50)
Small mid	**6,140.80**	(50)	6,165.21	(50)	6,058.18	(50)
Small long	12,228.58	(40)	**12,011.47**	(49)	11,849.18	(40)
Med short	**12,549.10**	(50)	12,659.97	(50)	12,416.62	(50)
Med mid	–	(0)	27,197.44	(47)	26,767.77	(0)
Med long	–	(0)	63,422.70	(49)	62,665.73	(0)
Large short	–	(0)	41,087.20	(48)	40,342.19	(0)
Large mid	–	(0)	111,126.13	(49)	109,024.41	(0)
Large long	–	(0)	227,083.12	(48)	223,578.79	(0)

puted considering only the instances solved by both solvers (whose number is reported in the last column). If the ILP solver was not able to solve any instance of the family, we report the results of the SA solver without comparison.

Looking at Table 3, it is evident that the SA solver finds a feasible solution for most cases (97%), whereas the ILP solver is able to solve only 42% of the instances. In terms of solution quality, comparing the average results of the two solvers on the three *small* families and the *med short* one there is no strong evidence in favor of one solution technique.

On the other hand, it is interesting to notice that for all families the cost of best solutions of SA_{PASU} is inferior to the one of the ILP_{PASU} and, in particular for the *small short* family, SA achieves results that are even better than the ones obtained by ILP with all optimal solutions. This can happen because the problem is solved day by day. Therefore, although each partial (daily) solution is proven to be optimal for the current day, there is no guarantee that the final solution is optimal too.

We can thus conclude that SA behaves very well, achieving results that are very close to (or even better than) the ones of the ILP solver, and it is able to scale to large instances.

5.3. Impact of delays

In Section 4.1, we have already introduced the PASU^f problem in which delay of the admission of a patient is not permitted. For this problem, the set of decision variables is considerably reduced and therefore the search space

Table 4: Comparison between the results of the ILP solver on the problem without delays (PASU^f) and SA solver on the problem with delays (PASU).

Family	$\mathrm{ILP}_{\mathrm{PASU}^f}$		$\mathrm{SA}_{\mathrm{PASU}}$			both	Δ
	Avg	Solved	Avg	Solved	Best	Solved	
Small short	2,925.48*	(50)	2,761.07	(50)	2,723.02	(50)	-5.62%
Small mid	6,820.76*	(50)	6,165.21	(50)	6,058.18	(50)	-9.61%
Small long	14,131.80*	(50)	12,383.17	(49)	12,190.86	(49)	-12.37%
Med short	12,835.60*	(50)	12,659.97	(50)	12,416.62	(50)	-1.37%
Med mid	28,405.64	(50)	27,197.44	(47)	26,767.77	(47)	-4.25%
Med long	65,951.27	(50)	63,422.70	(49)	62,665.73	(49)	-3.83%
Large short	38,810.81	(50)	41,087.20	(48)	40,342.19	(48)	5.87%
Large mid	74,342.00	(1)	71,341.38	(49)	70,217.00	(1)	-4.04%
Large long	–	(0)	227,083.12	(48)	223,578.79	(0)	–

is also much smaller. Consequently, for the PASU^f problem, it is possible to obtain optimal solutions using the ILP for instances larger than those solved for the PASU problem, as shown in Table 3.

In addition, the definition of this delay-free variant gives us the opportunity to analyze the relevance of using delays as a means for improving the overall quality of the solution.

In Table 4 we present the results of the ILP solver on the problem without delays (PASU^f) compared with the SA solver on the problem with delays (PASU). In this case, the ILP solver is able to solve 78% of the instances, so a more extensive comparison between the two solvers is now possible. To make the comparison easier, we also give the percentage gap between the cost values (evaluated as $\Delta = 100 \cdot (avg_{\mathrm{SA}_{\mathrm{PASU}}} - avg_{\mathrm{ILP}_{\mathrm{PASU}^f}})/avg_{\mathrm{ILP}_{\mathrm{PASU}^f}})$.

The outcome is that delays lead to an average saving of 4.4% of the value of the cost function, in particular their use becomes more effective as the planning horizon increases. Indeed, for short term instances exploring a larger search space can become disadvantageous, leading to worse results (see the results on the *large short* family in Table 4).

Obviously, these results are influenced to an extent by the default values of the weights of the problem. Using different values could lead to completely different results. In particular, if the value of w_{De} is relatively high, it is probably better to give up the possibility of having delays completely and explore more effectively the smaller search space composed of the solutions without delays.

Table 5: Competitive analysis of the performance of the dynamic SA solver compared with the ILP static one.

Family	$\text{ILP}_{\text{static}-\text{PASU}}$		SA_{PASU}			Both	Δ
	Avg	Solved	Avg	Solved	Best	Solved	
Small short	2,623.06	(50)	2,761.07	(50)	2,723.02	(50)	5.25%
Small mid	5,906.80	(50)	6,165.21	(50)	6,058.18	(50)	4.37%

5.4. Performance evaluation of the dynamic solver

We now examine the performance of SA on the dynamic problem (SA_{PASU}) in comparison with the ILP on the static one ($\text{ILP}_{\text{static}-\text{PASU}}$), where it is assumed that all patients are registered on the first day of the planning horizon. In the static case, the problem can be solved just once for the whole planning period.

This is obviously undoable in practice because it assumes the availability of information that will be revealed only in the future. It is anyway interesting to compute to what extent the absence of this knowledge influences the solution quality, by computing the gap between an *off-line* and *on-line* solver for this set of instances.

Table 5 compares the static ILP solver and SA_{PASU}. Unfortunately, the exact solver found (in 24 h for each instance) optimal solutions only for *small short* and *small mid* instances; thus, the comparison is limited to these two families. Surprisingly enough, the gap between the predictive and the real-time solutions is relatively small (4–5%), although all instances are characterized by a high degree of dynamism.

6. Discussion, conclusions and future work

In this paper, we reconsidered the *patient admission scheduling problem*, and we proposed a new problem formulation, namely the *dynamic patient admission scheduling problem under uncertainty.* The new version captures the real dynamic situations that normally happen in hospitals by including emergency patients, uncertain length of stay, and delays of admissions.

Besides the extensions listed in Section 2, we have also produced a simplification of the problem model. In PAS, a patient may need more than one treatment, so that she/he is assigned for the first part of her/his stay to a given treatment and for the second part to a different one. This feature has been removed because of its limited interest compared to the complexity of its management. As a consequence of this choice, we have also removed the possibility of having *planned transfers* in the solution. Planned transfers are solutions in which a patient is assigned to different rooms during her/his stay. In fact, they proved to be quite useless for patients with a single treatment. In the current formulation, transfers can happen if the room assigned on a given day is different from the one already occupied on the previous one.

We proposed a local search solution method based on simulated annealing, which works on a combination of neighborhoods. The SA results were compared with an integer linear programming solver on a large set of instances. The comparison is possible only for the small-size instances as, for the large one, the ILP solver was not able to find a feasible solution in a reasonable time. The outcome is that the SA solver obtains very competitive results. This quality is further confirmed by the comparison with the results for the static problem, which are only slightly superior.

For the future, we plan to further refine the management of delays. For example, delays should be penalized differently depending on the closeness of the event or on whether the patient has already been notified or not about the perspective admission date.

In addition, we plan to explore techniques that make some use of the statistical distribution on arrival, so as to make plans that are robust with respect to different scenarios of future arrivals. Nevertheless, given the relatively small distance between the current solutions and the static ones, we can conjecture that only a quite limited improvement will be found.

Finally, the problem considered includes very diverse objectives that range from medical effectiveness, to hospital policies, to patient comfort and/or risk management. It is clear that finding a good balance between all these components is extremely difficult. For this reason, we believe that a multi-objective view of the problem and a deep analysis on the relative influence and correlation between objectives would be very useful to understand the behavior of the problem in practical cases.

Acknowledgements

We thank Wim Vancroonenburg for helpful comments on the paper. We would like also to thank Michael Tooke for reviewing the English in this article.

[1] Demeester, P., Souffriau, W., De Causmaecker, P., Vanden Berghe,

G.. A hybrid tabu search algorithm for automatically assigning patients to beds. Artificial Intelligence in Medicine 2010;48(1):61–70.

[2] Bilgin, B., Demeester, P., Misir, M., Vancroonenburg, W., Vanden Berghe, G.. One hyper-heuristic approach to two timetabling problems in health care. Journal of Heuristics 2011; Online first `http://dx.doi.org/10.1007/s10732-011-9192-0`.

[3] Vancroonenburg, W., Goossens, D., Spieksma, F.. On the complexity of the patient assignment problem; 2011. Available from `http://allserv.kahosl.be/~wimvc/pas-complexity-techreport.pdf`.

[4] Kusters, R., Groot, P.M.. Modelling resource availability in general hospitals design and implementation of a decision support model. European Journal of Operational Research 1996;88(3):428–445.

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

[6] Eaton, W., Whitmore, G.A.. Length of stay as a stochastic process: A general approach and application to hospitalization for schizophrenia. The Journal of Mathematical Sociology 1977;5(2):273–292.

[7] Forster, A.J., Stiell, I., Wells, G., Lee, A.J., van Walraven, C.. The effect of hospital occupancy on emergency department length of stay and patient disposition. Academic Emergency Medicine : official Journal of the Society for Academic Emergency Medicine 2003;10(2):127–33.

[8] Ouelhadj, D., Petrovic, S.. Survey of dynamic scheduling in manufacturing systems. Journal of Scheduling 2008;12(4):417–431.

[9] Ceschia, S., Schaerf, A.. Local search and lower bounds for the patient admission scheduling problem. Computers & Operations Research 2011;38(10):1452–1463.

[10] Demeester, P.. Patient admission scheduling website. `http://allserv.kahosl.be/~peter/pas/`; 2009.

[11] Herroelen, W., Leus, R.. Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research 2005;165(2):289–306.

[12] Weissman, C.. Analyzing intensive care unit length of stay data: Problems and possible solutions. Critical Care Medicine 1997;25(9):1594–1600.

[13] Larsen, A.. The dynamic vehicle routing problem. Ph.D. thesis; Technical University of Denmark (DTU); 2001.

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

[15] Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.. Optimization by simulated annealing. Science 1983;220:671–680.

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

[17] Aarts, E.H.L., Korst, J.. Simulated Annealing and Boltzmann Machines. New York: John Wiley & Sons; 1989.

[18] van Laarhoven, P.J.M., Aarts, E.H.L.. Simulated annealing: theory and applications. D. Reidel Publishing Company, Kluwer Academic Publishers Norwell, MA, USA; 1987.

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

[20] Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.. F-Race and iterated F-race: An overview. Berlin: Springer; 2010.