Modeling and Solving the Dynamic Patient Admission Scheduling Problem under Uncertainty

Sara Ceschia, Andrea Schaerf
DIEGM, University of Udine, via delle Scienze 206, I-33100, Udine, Italy
{ceschia,schaerf}@uniud.it
February 15, 2012

Abstract

We propose a new formulation of the recently-formalized Patient Admission Scheduling (PAS) problem, which extends it by including several real-world features, such as the presence of emergency patients, uncertainty in stay lengths, and the possibility to delay admissions. We propose a metaheuristic approach that solves both the static (predictive) and the dynamic (daily) versions of this new problem, and is based on simulated annealing and a complex neighborhood structure. The quality of our method is compared, on small- and medium-size instances, with an exact method based on integer linear programming. In addition, 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 and quality.

1 Introduction

The Patient Admission Scheduling (PAS) problem consists in assigning patients to hospital rooms in such a way to maximize medical treatment effectiveness, management efficiency, and patients’ comfort. PAS has been for-
malized by Demeester et al. (2010) and further studied by the same research group (Bilgin et al., 2011).

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 equipments and location and it may or 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, patients should possibly not change the room during their stay; a room change (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 problem has been recently proved to be NP complete (Vancroonenburg et al., 2011).

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) (Kusters and Groot, 1996; Gemmel and Van Dierdonck, 1999). Furthermore, it is also frequent that the discharge date of the patients is unknown, because it might depend of the progress of her/his gradual recovery (Eaton and Whitmore, 1977; Forster et al., 2003). 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 the 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 be only shortly before her/his admission date. As in other online optimization problems (Bent and Van Hentenryck, 2004), the scheduling needs to be repeated iteratively at least once a day. For each day, the system schedules all the patients known at that time, i.e. the ones already registered.

In our previous work (Ceschia and Schaerf, 2011), 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 (Demeester, 2009). In the same work (Ceschia and Schaerf, 2011), 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 (Gemmel and Van Dierdonck, 1999), 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 aim, 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 stay length, and the possibility to delay 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 stay length of the patients is supposed to be fixed in PAS. On the contrary, we consider the possibility that a patient stays longer than expected. This extension is complemented by a corresponding cost component in the objective function that accounts for the risk that a room is overcrowded in some specific day, due to the probability of the over-length stay of some patients assigned to the room.

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 solution approach based on local search, partly built upon the one used in (Ceschia and Schaerf, 2011), but using new neighborhood operators.

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, the 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:** 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 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 w.r.t. the planned one, but not more than a given maximum, which is related to her/his health conditions.

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 patients. 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 are 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 conditions, which represents the possibility that she/he needs to spend one extra night in the hospital.

**Room/Department:** A room belongs to an 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 preferences 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. The assignment of a patient to a department that is not qualified for the treatment of her/his disease is not feasible; whereas 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 to treat particular pathologies. Patients may need or simply desire specific room features. Assignment to 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 days equal to her/his stay period, starting in a day not before the planned admission.

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 infeasible. 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, namely RG, Tr, De, and OR are soft constraints.

The De constraint takes into account the discomfort for the patient, 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 ones and potentially present ones due to the overstay risk, and its capacity.
<table>
<thead>
<tr>
<th>Constraint</th>
<th>Type</th>
<th>Default weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room Capacity (RC)</td>
<td>Hard</td>
<td>—</td>
</tr>
<tr>
<td>Room Gender (RG)</td>
<td>Soft</td>
<td>50</td>
</tr>
<tr>
<td>Department Specialism (DS)</td>
<td>Hard/Soft</td>
<td>20</td>
</tr>
<tr>
<td>Room Features (RF)</td>
<td>Hard/Soft</td>
<td>20</td>
</tr>
<tr>
<td>Patient Age (PA)</td>
<td>Hard</td>
<td>—</td>
</tr>
<tr>
<td>Room Preference (RP)</td>
<td>Soft</td>
<td>10</td>
</tr>
<tr>
<td>Transfer (Tr)</td>
<td>Soft</td>
<td>100</td>
</tr>
<tr>
<td>Delay (De)</td>
<td>Soft</td>
<td>2</td>
</tr>
<tr>
<td>Overcrowd Risk (OR)</td>
<td>Soft</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: PASU hard and soft constraints

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 $w_s$, 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 given in Table 1.

Analogously to the original PAS, the PASU problem can be simplified by means of a preprocessing step (see (Ceschia and Schaerf, 2011)). 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 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 to the room gender policy RG can also be partly included in the 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 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 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 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 removing the opportunity 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 stay length, 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 in 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 (2001) 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 solution (Section 4.2). The results of the two techniques 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 the 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

<table>
<thead>
<tr>
<th>Family</th>
<th>Depts</th>
<th>Rooms</th>
<th>Features</th>
<th>Patients</th>
<th>Specialisms</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Small Short</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>50</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>2. Small Mid</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>100</td>
<td>3</td>
<td>28</td>
</tr>
<tr>
<td>3. Small Long</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>200</td>
<td>3</td>
<td>56</td>
</tr>
<tr>
<td>4. Med Short</td>
<td>6</td>
<td>40</td>
<td>5</td>
<td>250</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>5. Med Mid</td>
<td>6</td>
<td>40</td>
<td>5</td>
<td>500</td>
<td>10</td>
<td>28</td>
</tr>
<tr>
<td>6. Med Long</td>
<td>6</td>
<td>40</td>
<td>5</td>
<td>1000</td>
<td>10</td>
<td>56</td>
</tr>
<tr>
<td>7. Large Short</td>
<td>8</td>
<td>160</td>
<td>6</td>
<td>1000</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>8. Large Mid</td>
<td>8</td>
<td>160</td>
<td>6</td>
<td>2000</td>
<td>15</td>
<td>28</td>
</tr>
<tr>
<td>9. Large Long</td>
<td>8</td>
<td>160</td>
<td>6</td>
<td>4000</td>
<td>15</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 2: Description of the instances
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 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 \( r \in \mathcal{R} \)
- \( \mathcal{R}_{SG} \): 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:

\[
\sum_{r \in \mathcal{R}} x_{p,r} = 1, \quad \forall \ p \in \mathcal{P} \tag{1}
\]

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

\[
x_{p,r} \leq A_{p,r} \quad \forall p \in \mathcal{P}, r \in \mathcal{R} \tag{3}
\]

Constraints (1) ensure that every patient is assigned to exactly one room. Constraints (2) ensure that the rooms are not overloaded (RO). Constraints (3) provide against infeasible 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 RP 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:

\[
\begin{align*}
    f_{r,d} & \geq x_{p,r}, & \forall p \in P_{F}, r \in R, d \in D \\
    m_{r,d} & \geq x_{p,r}, & \forall p \in P_{M}, r \in R, d \in D \\
    b_{r,d} & \geq m_{r,d} + f_{r,d} - 1, & \forall r \in R, d \in D
\end{align*}
\]

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 \( d \in D \) 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 to be 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 \( d \), which are the patients present in day \( d \) plus those present in day \( d - 1 \) with the risk of overstay.

In addition, we call \( |Z| \) the cardinality of a set \( Z \), and we call \( \overline{z} \) the complement of variable \( z \).

The constraints relating \( y \) to \( x \) are then the following ones:
\[
\sum_{p \in P_d^+} x_{p,r} \geq (|P_d^+| - c_r) \cdot (1 - y_{r,d}) \quad \forall d \in D, r \in R
\] (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).

\[
F = F_{\text{PRC}} + F_{\text{RG}} + F_{\text{OR}}
\] (9)

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

\[
F_{\text{PRC}} = \sum_{p \in P, r \in R} C_{p,r} \cdot x_{p,r} \cdot |D_p|
\] (10)

\[
F_{\text{RG}} = \sum_{r \in R, d \in D} w_{RG} \cdot b_{r,d}
\] (11)

\[
F_{\text{OR}} = \sum_{r \in R, d \in D} w_{OR} \cdot y_{r,d}
\] (12)

Equation (10) accounts for the cost of each patient-room assignment. Equation (11) accounts for the number of rooms occupied by both male and female patients. Finally, Equation (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 stay length of the patient.

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

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: $\text{RG}$ (room gender), $\text{PRC}$ (patient-room cost), $\text{OR}$ (overbooking risk), and $\text{De}$ (delay discomfort). In addition, it takes into account, with an appropriate high weight $W$, the constraint $\text{RC}$ (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 $\text{CR}$, for Change Room) and the swap of the assigned room between two patients ($\text{SP}$, 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 $\text{S}$ (for Shift) and it shifts forward or backward the admission of a patient by one day, keeping the room unchanged.

The complete neighborhood considered is thus $\text{CR} \oplus \text{SP} \oplus \text{S}$, where the symbol $\oplus$ stands for neighborhood union according to the terminology used in (Di Gaspero and Schaerf, 2006). From this neighborhood we obviously exclude moves that lead outside the search space. For example, $\text{SP}$ 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. (Kirkpatrick
et al., 1983) and Černý (Černý, 1985) and extensively studied by many au-
thors, including Aarts and Korst (1989) and van Laarhoven and Aarts (1987)
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 $s_0$. The
main procedure consists of a loop that randomly generates at each iteration
a neighbor of the current solution. Given a move $m$, we denote with $\Delta f$ the
difference in the cost function between the new solution 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 solution is accepted and becomes
the current one. In addition, if the value of the cost function of the current
solution is lower than the one of the best solution so far found, then the
best solution is updated to the current one. Conversely, if $\Delta f > 0$ the new
solution 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 $\alpha$, 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 $T_{\text{min}}$, that is when no solution
that increases the cost function is accepted anymore.

The control parameters of the procedure are the cooling rate $\alpha$, the num-
ber of neighbors sampled at each temperature $N$, and the starting and final
temperatures $T_0$ and $T_{\text{min}}$.

Given that we use a composite neighborhood, the random selection mech-
anism prescribed by SA needs to be further detailed. In our case, it is per-
formed 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 $p_{\text{CR}},$
$p_{\text{SP}}$, and $p_{\text{S}}$ (with $p_{\text{S}} = 1 - p_{\text{CR}} - p_{\text{SP}}$). The values of $p_{\text{CR}}$ and $p_{\text{SP}}$ are subject
of tuning, along with the other SA parameters.

In addition, the SA procedure prescribes that the move acceptance is
based on $\Delta f$, therefore the value of $W$ is crucial for the performances 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 follows trajectories that “prefer” infeasible solutions to feasible ones, if they have lower objective cost. In conclusion, $W$ also needs to be set experimentally, as discussed in Section 5.

5 Experimental analysis

For the IP 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).

The SA solver has been 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 Easy-Local++ (Di Gaspero and Schaerf, 2003), and it is compiled using the GNU C/C++ compiler, v. 4.4.3, under Ubuntu Linux. All experiments have been 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 (Birattari et al., 2010), which allows us to explore effectively the space of parameters 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 $\alpha$, and number of neighbors sampled at each
temperature \(N\). In addition, the values of the probabilities \(p_{CR}\) and \(p_{SP}\), and the weight \(W\) are subject of tuning, too.

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

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

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

### 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 IP 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 in our web site. For the IP solver, the * symbol marks the fact that for all instances it has obtained the optimal solution within the time granted.

In some cases, the solver has not been able to solve to feasibility all the instances of the family in the granted timeout, therefore we report in parenthesis the number of instances solved. The average and best values are computed considering only the instances solved by both solvers (whose number is reported in the last column). If the IP 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 (88%), whereas the IP solver is able to solve only the 38% of
### Table 3: Results of the IP and SA solvers on the PASU problem.

<table>
<thead>
<tr>
<th>Family</th>
<th>$\text{IP}_{\text{PASU}}$ avg solved</th>
<th>$\text{SA}_{\text{PASU}}$ avg solved</th>
<th>best</th>
<th>both solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Short</td>
<td>2,768.82 (50)</td>
<td>2,761.07 (50)</td>
<td>2,723.02 (50)</td>
<td></td>
</tr>
<tr>
<td>Small Mid</td>
<td>6,140.80 (50)</td>
<td>6,165.21 (50)</td>
<td>6,058.18 (50)</td>
<td></td>
</tr>
<tr>
<td>Small Long</td>
<td>12,228.58 (40)</td>
<td>12,011.47 (49)</td>
<td>11,849.18 (40)</td>
<td></td>
</tr>
<tr>
<td>Med Short</td>
<td>12,549.10 (50)</td>
<td>12,659.97 (50)</td>
<td>12,416.62 (50)</td>
<td></td>
</tr>
<tr>
<td>Med Mid</td>
<td>– (0)</td>
<td>27,197.44 (47)</td>
<td>26,767.77 (0)</td>
<td></td>
</tr>
<tr>
<td>Med Long</td>
<td>– (0)</td>
<td>63,422.70 (49)</td>
<td>62,665.73 (0)</td>
<td></td>
</tr>
<tr>
<td>Large Short</td>
<td>– (0)</td>
<td>41,087.20 (48)</td>
<td>40,342.19 (0)</td>
<td></td>
</tr>
<tr>
<td>Large Mid</td>
<td>– (0)</td>
<td>111,126.13 (49)</td>
<td>109,024.41 (0)</td>
<td></td>
</tr>
<tr>
<td>Large Long</td>
<td>– (0)</td>
<td>227,083.12 (48)</td>
<td>223,578.79 (0)</td>
<td></td>
</tr>
</tbody>
</table>

On the contrary, it is interesting to notice that for all families the cost of best solutions of $\text{SA}_{\text{PASU}}$ is inferior to the one of the $\text{IP}_{\text{PASU}}$ and, in particular for the Small Short family, SA achieves results that are even better than the ones obtained by IP 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 not any guarantee that the final solution is optimal too.

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

#### 5.3 Impact of delays

As shown on Table 3, the IP solver cannot obtain a solution for most of the instances on the PASU problem. On Section 4.1, we have already introduced the $\text{PASU}^f$ problem in which a delay of the admission of a patient is not permitted; this considerably reduces the set of decision variables and therefore the search space, allowing to obtain optimal solutions even for large instances.

In addition, the definition of this new problem variant gives us the opportunity to analyze the relevance of delays for the overall quality of the
solution.

In Table 5.3 we present the results of the IP solver on the problem without delays (PASUF) compared with the SA solver on the problem with delays (PASU). By reducing the search space, the IP solver is able to solve 70% of the instances, therefore a more extensive comparison between the two solver is now possible. To make the comparison easier, we also give the percentage gaps between the cost values, evaluated as $\Delta = 100 \cdot \frac{\text{avg}_{\text{SA\ PASU}} - \text{avg}_{\text{IP\ PASUf}}}{\text{avg}_{\text{IP\ PASUf}}}$. The outcome is that delays lead to an average saving of 4.4% of value of the cost function, in particular their use becomes more efficacious as the planning horizon increases. Indeed, for short term instances exploring a larger search space can become disadvantageous, leading to worst results (see the results on the Large Short family in Table 5.3).

Obviously, these results are pretty much influenced 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 $w_{\text{De}}$ is relatively high, it is probably better to give up completely the possibility of having delays and explore more effectively the smaller search space composed by the solutions without delays.
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 IP on the static one ($IP_{static-PASU}$), 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, because it is only necessary to make out the schedule on the first day. 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 which extend 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 IP solver and $SA_{PASU}$. Unfortunately, the exact solver found (in 24 hours 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.

<table>
<thead>
<tr>
<th>Family</th>
<th>$IP_{static-PASU}$</th>
<th>$SA_{PASU}$</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg solved</td>
<td>avg solved</td>
<td>best solved</td>
</tr>
<tr>
<td>Small Short</td>
<td>2,623.06 (50)</td>
<td>2,761.07 (50)</td>
<td>2,723.02 (50)</td>
</tr>
<tr>
<td>Small Mid</td>
<td>5,906.80 (50)</td>
<td>6,165.21 (50)</td>
<td>6,058.18 (50)</td>
</tr>
</tbody>
</table>

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

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 comprising emergency patients, uncertain stay length, and delays of admissions.

Besides the extensions listed in Section 2, we have also made a simplification of the problem model. In PAS, a patient may need more than one treatment, so that she/he is planned to be for the first part of her/his stay...
under a given treatment and the second part under 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 to have 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 in a given day is different from the one already occupied in 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 a 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 IP solver was not able to find a feasible solution in 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, in some cases it would be advisable to forbidding to retract or modify a decision about a delayed admission on the scheduling of the following days, for example when the patient has been already notified about the delay.

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 improvements could 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.
References


