An ASP-based Solution for Operating Room Scheduling with Beds Management

Carmine Dodaro[1], Giuseppe Galatà[2], Muhammad Kamran Khan[3], Marco Maratea[3], and Ivan Porro[2]

[1] DEMACS, University of Calabria, Rende, Italy,
`dodaro@mat.unical.it`
[2] SurgiQ srl, Italy
E-mail: {name.surname}@surgiq.com
[3] DIBRIS, University of Genova, Genova, Italy
`muhammad.kamrankhan@edu.unige.it,marco.maratea@unige.it`

Abstract. The Operating Room Scheduling (ORS) problem is the task of assigning patients to operating rooms, taking into account different specialties, lengths and priority scores of each planned surgery, operating room session durations, and the availability of beds for the entire length of stay both in the Intensive Care Unit and in the wards. A proper solution to the ORS problem is of utmost importance for the quality of the health-care and the satisfaction of patients in hospital environments. In this paper we present an improved solution to the problem based on Answer Set Programming (ASP) that, differently from a recent one, takes explictly into account beds management. Results of an experimental analysis, conducted on benchmarks with realistic sizes and parameters, show that ASP is a suitable solving methodology for solving also such improved problem version.

1 Introduction

The Operating Room Scheduling (ORS) [1, 8, 25, 26] problem is the task of assigning patients to operating rooms, taking into account different specialties, surgery durations, and the availability of beds for the entire length of stay (LOS) both in the Intensive Care Unit (ICU) and in the wards. Given that patients may have priorities, the solution has to find an accommodation for the patients with highest priorities, and then to the other with lower priorities, if space is still available, at the same time taking into proper account beds availability. Recently, a solution based on Answer Set Programming (ASP) [21, 22, 27, 10, 11] was proposed and proved to be effective for solving ORS problems [15]. Nonetheless, such solution does not take into account beds management. In most modern hospitals, very long surgical waiting list are present and often worsened, if not altogether caused, by inefficiencies in operating room planning, and the availability of beds in the wards and, if necessary, in the Intensive Care Unit (ICU) for each patient for the entire duration of their stay, is a very important factor for such inefficiencies.

In this paper we thus propose an improved solution based on ASP that takes explicitly into account beds management. In such solution, problem specifications related to beds management are modularly added as ASP rules to the previous encoding of the basic version of the problem where beds management was not considered, and then efficient ASP solvers are used to solve the resulting ASP program. We have then generated ORS benchmarks with realistic sizes and parameters inspired by those of small-medium Italian hospitals, and run an experimental analysis on such benchmarks using the ASP solver CLINGO [19]. Benchmarks have been organized in two scenarios: a first scenario is characterized by an abundance of available beds, so that the constraining resource becomes the OR time, while for the second scenario the number of beds is the constrained resource. Overall, results show that ASP is a suitable solving methodology for ORS also when beds management is taken into account, on both scenario, given that our solution is able to utilize efficiently whichever resource is more constrained; moreover, this is obtained in short timings in line with the needs of the application.

To summarize, the main contributions of this paper are the following:

- We provide an ASP encoding for solving the complete ORS problem (Section 4 and 5).
- We run an experimental analysis assessing the good performance of our ASP solution (Section 6).
- We analyze related literature (Section 7), with focus on beds management.

The paper is completed by Section 2, which contains needed preliminaries about ASP, by an informal description of the ORS problem in Section 3, and by conclusions and possible topics for future research in Section 8.

2 Background on ASP

Answer Set Programming (ASP) [11] is a programming paradigm developed in the field of nonmonotonic reasoning and logic programming. In this section we overview the language of ASP. More detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in [11, 13]. Hereafter, we assume the reader is familiar with logic programming conventions.

Syntax. The syntax of ASP is similar to the one of Prolog. Variables are strings starting with uppercase letter and constants are non-negative integers or strings starting with lowercase letters. A *term* is either a variable or a constant. A *standard atom* is an expression $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity n and $t_1, \ldots, t_n$ are terms. An atom $p(t_1, \ldots, t_n)$ is ground if $t_1, \ldots, t_n$ are constants. A *ground set* is a set of pairs of the form $\langle consts : conj \rangle$, where $consts$ is a list of constants and $conj$ is a conjunction of ground standard atoms. A *symbolic set* is a set specified syntactically as $\{Terms_1 : Conj_1; \cdots ; Terms_t : Conj_t\}$, where $t > 0$, and for all $i \in [1, t]$, each $Terms_i$ is a list of terms such that $|Terms_i| =$

$k > 0$, and each $Conj_i$ is a conjunction of standard atoms. A *set term* is either a symbolic set or a ground set. Intuitively, a set term $\{X : a(X, c), p(X); Y : b(Y, m)\}$ stands for the union of two sets: the first one contains the X-values making the conjunction $a(X, c), p(X)$ true, and the second one contains the Y-values making the conjunction $b(Y, m)$ true. An *aggregate function* is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Basically, aggregate functions map multisets of constants to a constant. The most common functions implemented in ASP systems are the following:

- *#count*, number of terms;
- *#sum*, sum of integers.

An *aggregate atom* is of the form $f(S) \prec T$, where $f(S)$ is an aggregate function, $\prec \in \{<, \leq, >, \geq, \neq, =\}$ is a comparison operator, and T is a term called guard. An aggregate atom $f(S) \prec T$ is ground if T is a constant and S is a ground set. An *atom* is either a standard atom or an aggregate atom. A *rule* r has the following form:

$$a_1 \ \vee \ \ldots \ \vee \ a_n \ {:\!-} \ b_1, \ldots, b_k, not \ b_{k+1}, \ldots, not \ b_m.$$

where $a_1, \ldots, a_n$ are standard atoms, $b_1, \ldots, b_k$ are atoms, $b_{k+1}, \ldots, b_m$ are standard atoms, and $n, k, m \geq 0$. A literal is either a standard atom a or its negation $not \ \ a$. The disjunction $a_1 \vee \ldots \vee a_n$ is the *head* of r, while the conjunction $b_1, \ldots, b_k, not \ b_{k+1}, \ \ldots, not \ b_m$ is its *body*. Rules with empty body are called *facts*. Rules with empty head are called *constraints*. A variable that appears uniquely in set terms of a rule r is said to be *local* in r, otherwise it is a *global* variable of r. An ASP program is a set of *safe* rules, where a rule r is *safe* if the following conditions hold: *(i)* for each global variable X of r there is a positive standard atom ℓ in the body of r such that X appears in ℓ; and *(ii)* each local variable of r appearing in a symbolic set $\{Terms : Conj\}$ also appears in a positive atom in $Conj$.

A *weak constraint* [12] ω is of the form:

$$:\sim b_1, \ldots, b_k, not \ b_{k+1}, \ldots, not \ b_m. \ [w@l]$$

where w and l are the weight and level of ω, respectively. (Intuitively, $[w@l]$ is read "as weight w at level l", where weight is the "cost" of violating the condition in the body of w, whereas levels can be specified for defining a priority among preference criteria). An ASP program with weak constraints is $\Pi = \langle P, W \rangle$, where P is a program and W is a set of weak constraints.

A standard atom, a literal, a rule, a program or a weak constraint is *ground* if no variables appear in it.

Semantics. Let P be an ASP program. The *Herbrand universe* U_P and the *Herbrand base* B_P of P are defined as usual. The ground instantiation G_P of P is the set of all the ground instances of rules of P that can be obtained by substituting variables with constants from U_P.

An *interpretation* I for P is a subset I of B_P. A ground literal ℓ (resp., *not* ℓ) is true w.r.t. I if $\ell \in I$ (resp., $\ell \notin I$), and false (resp., true) otherwise. An aggregate atom is true w.r.t. I if the evaluation of its aggregate function (i.e., the result of the application of f on the multiset S) with respect to I satisfies the guard; otherwise, it is false.

A ground rule r is *satisfied* by I if at least one atom in the head is true w.r.t. I whenever all conjuncts of the body of r are true w.r.t. I.

A model is an interpretation that satisfies all rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [17] of G_P w.r.t. I is the subset G_P^I of G_P obtained by deleting from G_P the rules in which a body literal is false w.r.t. I. An interpretation I for P is an *answer set* (or stable model) for P if I is a minimal model (under subset inclusion) of G_P^I (i.e., I is a minimal model for G_P^I) [17].

Given a program with weak constraints $\Pi = \langle P, W \rangle$, the semantics of Π extends from the basic case defined above. Thus, let $G_\Pi = \langle G_P, G_W \rangle$ be the instantiation of Π; a constraint $\omega \in G_W$ is violated by an interpretation I if all the literals in ω are true w.r.t. I. An *optimum answer set* for Π is an answer set of G_P that minimizes the sum of the weights of the violated weak constraints in G_W in a prioritized way.

Syntactic shortcuts. In the following, we also use *choice rules* of the form $\{p\}$, where p is an atom. Choice rules can be viewed as a syntactic shortcut for the rule $p \vee p'$, where p' is a fresh new atom not appearing elsewhere in the program, meaning that the atom p can be chosen as true.

3 Problem Description

In this section we provide an informal dsecription of the ORS problem and its requirements.

As we already said in the introduction, most modern hospitals are characterized by a very long surgical waiting list, often worsened, if not altogether caused, by inefficiencies in operating room planning. A very important factor is represented by the availability of beds in the wards and, if necessary, in the Intensive Care Unit for each patient for the entire duration of their stay.

This means that hospital planners have to balance the need to use the OR time with the maximum efficiency with an often reduced beds availability.

In this paper, the elements of the waiting list are called *registrations*. Each registration links a particular surgical procedure, with a predicted surgery duration and length of stay in the ward and in the ICU, to a patient.

The overall goal of the ORS problem is to assign the maximum number of registrations to the operating rooms (ORs), taking into account the availability of beds in the associated wards and in the ICU. This approach entails that the resource optimized is the one, between the OR time and the beds, that represents the bottleneck in the particular scenario analyzed.

As first requirement of the ORS problem, the assignments must guarantee that the sum of the predicted duration of surgeries assigned to a particular OR session does not exceed the length of the session itself: this is referred in the following as *surgery requirement.* Moreover, registrations are not all equal: they can be related to different medical conditions and can be in the waiting list for different periods of time. These two factors are unified in one concept: *priority.* Registrations are classified according to three different priority categories, namely P_1, P_2 and P_3. The first one gathers either very urgent registrations or the ones that have been in the waiting list for a long period of time; it is required that these registrations are all assigned to an OR. Then, the registrations of the other two categories are assigned to the top of the ORs capacity, prioritizing the P_2 over the P_3 ones (*minimization*).

Regarding the bed management part of the problem, we have to ensure that a registration can be assigned to an OR only if there is a bed available for the patient for the entire LOS. In particular, we have considered the situation where each specialty is related to a ward with a variable number of available beds exclusively dedicated to the patients associated to the specialty. This is referred in the following as *ward bed requirement.* The ICU is a particular type of ward that is accessible to patients from any specialty. However, only a small percentage of patients is expected to need to stay in the ICU. This requirement will be referred as the *ICU bed requirement.* Obviously, during their stay in the ICU, the patient does not occupy a bed in the specialty's ward.

In our model, a patient's LOS has been subdivided in the following phases:

- a LOS in the ward before surgery, in case the admission is programmed a day (or more) before the surgery takes place;
- the LOS after surgery, which can be further subdivided into the ICU LOS and the following ward LOS.

The encoding described in Sections 4 and 5 supports the generation of an optimized schedule of the surgeries either in the case where the bottleneck is represented by the OR time or by the beds availability.

4 ASP Encoding for the basic ORS problem

Starting from the specifications in the previous section, in this section the scheduling problem, limited to the assignments of the registrations to the ORs, is described in the ASP language, in particular following the input language of CLINGO.

4.1 OR scheduling

Data Model. The input data is specified by means of the following atoms:

- Instances of *registration(R,P,SU,LOS,SP,ICU,A)* represent the registrations, characterized by an id (R), a priority score (P), a surgery duration (SU) in

$$
\begin{array}{rll}
\{x(R,P,O,S,D)\} & :\!-\ \mathit{registration}(R,P,\_,\_,SP,\_,\_), \mathit{mss}(O,S,SP,D). & (r_1)\\
 & :\!-\ x(R,P,O,S1,\_), x(R,P,O,S2,\_), S1! = S2. & (r_2)\\
 & :\!-\ x(R,P,O1,\_,\_), x(R,P,O2,\_,\_), O1! = O2. & (r_3)\\
\mathit{surgery}(R,SU,O,S) & :\!-\ x(R,\_,O,S,\_), \mathit{registration}(R,\_,SU,\_,\_,\_,\_). & (r_4)\\
 & :\!-\ x(\_,\_,O,S,\_), \mathit{duration}(N,O,S), & \\
 & \qquad \#\mathit{sum}\{SU,R : \mathit{surgery}(R,SU,O,S)\} > N & (r_5)\\
 & :\!-\ N = \mathit{totRegsP1} - \#\mathit{count}\{R : x(R,1,\_,\_,\_)\},\ N > 0. & (r_6)\\
 & :\sim\ N = \mathit{totRegsP2} - \#\mathit{count}\{R : x(R,2,\_,\_,\_)\}.\ [N@3] & (r_7)\\
 & :\sim\ N = \mathit{totRegsP3} - \#\mathit{count}\{R : x(R,3,\_,\_,\_)\}.\ [N@2] & (r_8)
\end{array}
$$

Fig. 1. ASP encoding of the ORS problem, excluding the bed management

minutes, the overall length of stay both in the ward and the ICU after the surgery (LOS) in days, the id of the specialty (SP) it belongs to, a length of stay in the ICU (ICU) in days, and finally a parameter representing the number of days in advance (A) the patient is admitted to the ward before the surgery. It must be noted that the variables LOS, ICU and A become relevant for the beds management (see Section 5).

- Instances of *mss(O,S,SP,D)* link each operating room (O) to a session (S) for each specialty and planning day (D) as established by the hospital Master Surgical Schedule (MSS).
- The OR sessions are represented by the instances of the predicate *duration(N,O,S)*, where N is the session duration.

The output is an assignment represented by atoms of the form *x(R,P,O,S,D)*, where the intuitive meaning is that the registration R with priority P is assigned to the OR O during the session S and the day D.

Encoding. The related encoding is shown in Figure 1, and is described in the following. Rule (r_1) guesses an assignment for the registrations to an OR in a given day and session among the ones permitted by the MSS for the particular specialty the registration belongs to.

The same registration should not be assigned more than once, in different OR or sessions. This is assured by the constraints (r_2) and (r_3). Note that in our setting there is no requirement that every registration must actually be assigned.

Surgery requirement. With rules (r_4) and (r_5), we impose that the total length of surgery durations assigned to a session is less than or equal to the session duration.

Minimization. We remind that we want to be sure that every registration having priority 1 is assigned, then we assign as much as possible of the others, giving precedence to registrations having priority 2 over those having priority 3. This is accomplished through constraint (r_6) for priority 1 and the weak constraints (r_7) and (r_8) for priority 2 and 3, respectively, where *totRegsP1*, *totRegsP2* and *totRegsP3* are constants representing the total number of registrations having priority 1, 2 and 3, respectively.

Minimizing the number of unassigned registrations could cause an implicit preference towards the assignments of the registrations with shorter surgery durations. To avoid this effect, one can consider to minimize the idle time, however this is in general slower from a computational point of view and often unnecessary, since the preference towards shorter surgeries is already mitigated by our three-tiered priority schema.

5 ASP Encoding for ORS with Beds Management

This section is devoted to the beds management task of the ORS problem; the ASP rules and data model described here are added to those presented in the previous section.

5.1 OR scheduling with beds

Data Model. In order to deal with the beds management for the wards and the ICU, the data model outlined in Section 4.1 must be supplemented to include data about the availability of beds in each day of the planning and for each ward associated to the specialties and the ICU.

Instances of *beds(SP,AV,D)* represent the number of available beds (AV) for the beds associated to the specialty SP in the day D. The ICU is represented by giving the value 0 to SP.

Encoding. The related encoding is shown in Fig 2, and is described in the following. Rule (r_9) assigns a bed in the ward to each registration assigned to an OR, for the days before the surgery. Rule (r_{10}) assigns a ward bed for the period after the patient was dismissed from the ICU and transferred to the ward. Rule (r_{11}) assigns a bed in the ICU.

Ward bed requirement. Rule (r_{12}) ensures that the number of patients occupying a bed in each ward for each day is never larger than the number of available beds.

ICU bed requirement. Finally, rule (r_{13}) performs a similar check as the one in rule (r_{12}), but for the ICU.

Remark. We note that, given that the MSS is fixed, our problem and encoding can be decomposed by considering each specialty separately in case the beds are not a constrained resource, as will be the case for one of our scenario. We decided not to use this property because (i) this is the description of a practical application that is expected to be extended over time and to correctly work

$$\begin{aligned}
\mathit{stay}(R, D-A..D-1, SP) &\ \text{:-}\ \mathit{registration}(R, \_, \_, LOS, SP, \_, A), \\
&\qquad x(R, \_, \_, \_, D), A > 0. && (r_9) \\
\mathit{stay}(R, D+ICU..D+LOS-1, SP) &\ \text{:-}\ \mathit{registration}(R, \_, \_, LOS, SP, ICU, \_), \\
&\qquad x(R, \_, \_, \_, D), LOS > ICU. && (r_{10}) \\
\mathit{stayICU}(R, D..ICU+D-1) &\ \text{:-}\ \mathit{registration}(R, \_, \_, \_, \_, ICU, \_), \\
&\qquad x(R, \_, \_, \_, D), ICU > 0. && (r_{11}) \\
&\ \text{:-}\ \#\mathit{count}\{R : \mathit{stay}(R, D, SP)\} > AV, \\
&\qquad SP > 0, \mathit{beds}(SP, AV, D). && (r_{12}) \\
&\ \text{:-}\ \#\mathit{count}\{R : \mathit{stayICU}(R, D)\} > AV, \\
&\qquad \mathit{beds}(0, AV, D). && (r_{13})
\end{aligned}$$

Fig. 2. ASP encoding of the bed management portion of the ORS problem

Table 1. Beds availability for each specialty and in each day in scenario A.

Specialty	Monday	Tuesday	Wednesday	Thursday	Friday
0 (ICU)	40	40	40	40	40
1	80	80	80	80	80
2	58	58	58	58	58
3	65	65	65	65	65
4	57	57	57	57	57
5	40	40	40	40	40

even if the problem becomes non-decomposable, e.g. a (simple but significant) extension in which a room is shared among specialties brings to a problem which is not anymore decomposable in all cases, and (*ii*) it is not applicable to all of our scenarios. Additionaly, even not considering this property at the level of encoding, the experimental analysis that we will present is already satisfying for our use case.

6 Experimental Results

In this section we report about the results of an empirical analysis of the ORS problem. Data have been randomly generated but having parameters and sizes inspired by real data. Both experiments were run on a Intel Core i7-7500U CPU @ 2.70GHz with 7.6 GB of physical RAM. The ASP system used was CLINGO [18], version 5.5.2.

6.1 ORS benchmarks

The final encoding employed in our analysis is composed by the ASP rules $(r_1), \ldots, (r_{13})$. The test cases we have assembled are based on the requirements

Table 2. Beds availability for each specialty and in each day in scenario B.

Specialty	Monday	Tuesday	Wednesday	Thursday	Friday
0 (ICU)	4	4	5	5	6
1	20	30	40	45	50
2	10	15	23	30	35
3	10	14	21	30	35
4	8	10	14	16	18
5	10	14	20	23	25

Table 3. Parameters for the random generation of the scheduler input.

Specialty	Reg.	ORs	Surgery Duration (min) mean (std)	LOS (d) mean (std)	ICU %	ICU LOS (d) mean (std)	LOS (d) before surgery
1	80	3	124 (59.52)	7.91 (2)	10	1 (1)	1
2	70	2	99 (17.82)	9.81 (2)	10	1 (1)	1
3	70	2	134 (25.46)	11.06 (3)	10	1 (1)	1
4	60	1	95 (19.95)	6.36 (1)	10	1 (1)	0
5	70	2	105 (30.45)	2.48 (1)	10	1 (1)	0
Total	350	10					

of a typical small-medium size Italian hospital, with five surgical specialties to be managed over the widely used 5-days planning period. Two different scenarios were assembled. The first one (scenario A) is characterized by an abundance of available beds, so that the constraining resource becomes the OR time. For the second one (scenario B), we severely reduced the number of beds, in order to test the encoding in a situation with plenty of OR time but few available beds. Each scenario was tested 10 times with different randomly generated inputs. The characteristics of the tests are the following:

- 2 different benchmarks, comprising a planning period of 5 working days, and different numbers of available beds, as reported in Table 1 and Table 2 for scenario A and B, respectively;
- 10 ORs, unevenly distributed among the specialties;
- 5 hours long morning and afternoon sessions for each OR, summing up to a total of 500 hours of ORs available time for the 2 benchmarks;
- 350 generated registrations, from which the scheduler will draw the assignments. In this way, we simulate the common situation where a hospital manager takes an ordered, w.r.t. priorities, waiting list and tries to assign as many elements as possible to each OR.

The surgery durations have been generated assuming a normal distribution, while the priorities have been generated from a uneven distribution of three possible values (with weights respectively of 0.20, 0.40 and 0.40 for registrations having priority 1, 2 and 3, respectively). The lengths of stay (total LOS after surgery and ICU LOS) have been generated using a truncated normal distribution, in order to avoid values less than 1. In particular for the ICU, only a small percentage of patients have been generated with a predicted LOS while the large

Table 4. Scheduling results for the scenario A benchmark.

Assigned Registrations				OR time Efficiency	Bed Occupation Efficiency
Priority 1	Priority 2	Priority 3	Total		
62 / 62	132 / 150	72 / 138	266 / 350	96.6%	52.0%
72 / 72	128 / 145	64 / 133	264 / 350	95.6%	51.0%
71 / 71	132 / 132	69 / 147	272 / 350	96.7%	53.0%
66 / 66	138 / 142	57 / 142	261 / 350	96.2%	50.7%
79 / 79	119 / 130	67 / 141	265 / 350	96.0%	51.9%
67 / 67	131 / 131	66 / 152	264 / 350	96.6%	53.8%
66 / 66	121 / 132	69 / 152	256 / 350	96.0%	49.8%
69 / 69	130 / 135	68 / 146	267 / 350	96.8%	51.6%
60 / 60	139 / 153	59 / 137	258 / 350	96.0%	50.8%
68 / 68	138 / 142	57 / 139	263 / 350	95.2%	51.3%

Table 5. Scheduling results for the scenario B benchmark.

Assigned Registrations				OR time Efficiency	Bed Occupation Efficiency
Priority 1	Priority 2	Priority 3	Total		
62 / 62	106 / 150	13 / 138	181 / 350	66.3%	92.7%
72 / 72	77 / 145	43 / 133	192 / 350	67.5%	94.2%
71 / 71	80 / 132	38 / 147	189 / 350	68.2%	96.1%
66 / 66	81 / 142	41 / 142	188 / 350	71.4%	93.4%
79 / 79	90 / 130	20 / 141	189 / 350	69.0%	94.1%
67 / 67	95 / 131	25 / 152	187 / 350	66.5%	93.9%
66 / 66	92 / 132	30 / 152	188 / 350	71.8%	94.1%
69 / 69	84 / 135	36 / 146	189 / 350	68.7%	92.7%
60 / 60	91 / 153	34 / 137	185 / 350	69.7%	94.1%
68 / 68	82 / 142	35 / 139	185 / 350	69.3%	95.1%

majority do not need to pass through the ICU and their value for the ICU LOS is fixed to 0. Finally, since the LOS after surgery includes both the LOS in the wards and in the ICU, the value generated for the ICU LOS must be less than or equal to the total LOS after surgery. The parameters of the test have been summed up in Table 3. In particular, for each specialty (1 to 5), we reported the number of registrations generated, the number of ORs assigned to the specialty, the mean duration of surgeries with its standard deviation, the mean LOS after the surgery with its standard deviation, the percentage of patients that need to stay in the ICU, the mean LOS in the ICU with its standard deviation and, finally, the LOS before the surgery (i.e. the number of days, constant for each specialty, the patient is admitted before the planned surgery is executed).

6.2 Results

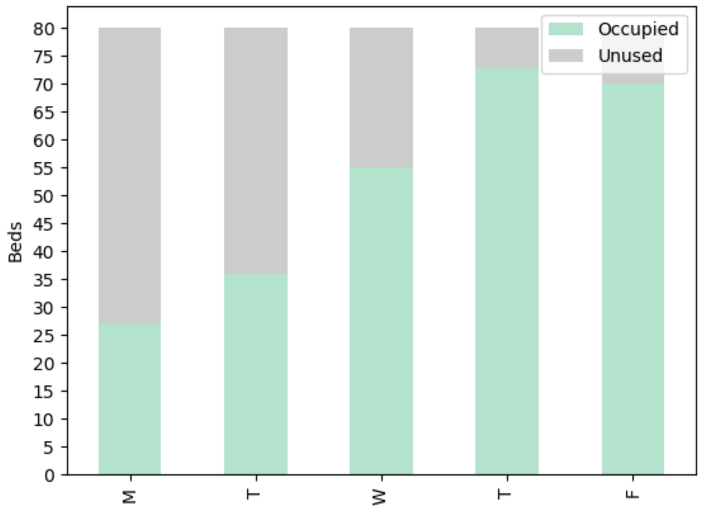
Results of the experiments are reported for scenario A in Table 4 and for scenario B in Table 5, respectively. A time limit of 60 seconds was given in view of a practical use of the program, each scenario was run 10 times with different input registrations. For each of the 10 runs executed, the tables report in the first three columns the number of the assigned registrations out of the generated ones for each priority, and in the remaining two columns a measure of the total time occupied by the assigned registrations as a percentage of the total OR time available (indicated as OR time Efficiency in the Table) and the ratio between the beds occupied after the planning to the available ones before the planning (labeled as Bed Occupation Efficiency in the tables). As a general observation, these results show that our solution is able to utilize efficiently whichever resource is more constrained: scenario A runs manage to reach a very high efficiency, over 95%, in the use of OR time, while scenario B achieves an efficiency of bed occupation between 92% and 95%. Note that to better be able to confront the results, for each run the bed configurations of the two scenarios were applied to the same generated registrations. Taking into consideration a practical use of this solution, the user would be able to individuate and quantify the resources that are more constraining and take the appropriate actions. This means that the solution can also be used to test and evaluate "what if" scenarios.

Finally, in Figure 3 we (partially) present the results achieved on one instance (i.e., the first instance of Table 4 and Table 5) with 350 registrations for 5 days. Each bar represents the total number of available beds for specialty 1, as reported in Table 1 for the plot at the top and Table 2 for the bottom one, for each day of the week, from Monday through Friday. The colored part of the bars indicates the amount of occupied beds while the gray part the beds left unoccupied by our planning.

7 Related Work

In this section we review related literature, organized into two paragraphs. The first paragraph is devoted to outlining different techniques for solving the ORS problem, with focus on the inclusion of beds management, while in the second paragraph we report about other scheduling problems where ASP has been employed.

Solving ORS problems. Aringhieri et al. [8] addressed the joint OR planning (MSS) and scheduling problem, described as the allocation of OR time blocks to specialties together with the subsets of patients to be scheduled within each time block over a one week planning horizon. They developed a 0-1 linear programming formulation of the problem and used a two-level meta-heuristic to solve it. Its effectiveness was demonstrated through numerical experiments carried out on a set of instances based on real data and resulted, for benchmarks of 80-100 assigned registrations, in a 95-98% average OR utilization rate, for a number of ORs ranging from 4 to 8. The execution times were around 30-40 seconds. In [25], the same authors introduced a hybrid two-phase optimization algorithm which exploits neighborhood search techniques combined with Monte Carlo simulation, in order to solve the joint advance and allocation scheduling problem,



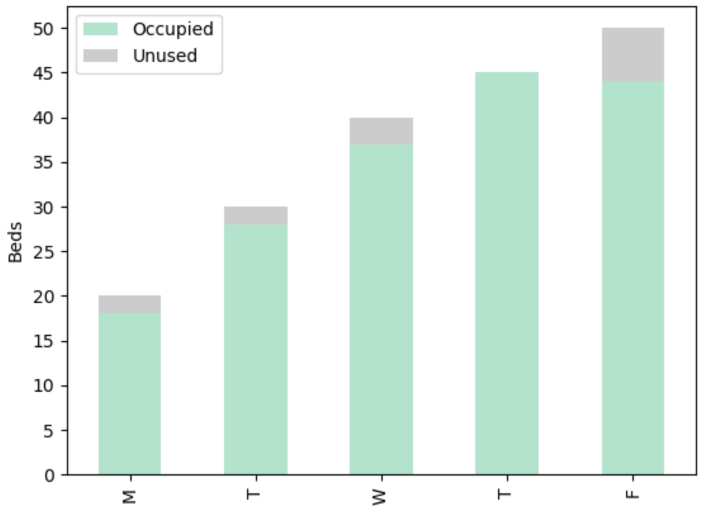


Fig. 3. Example of bed occupation of the ward corresponding to specialty 1 for 5 days scheduling. The plot on the top corresponds to the first instance of scenario A, while the one on the bottom to the first instance of scenario B.

taking into account the inherent uncertainty of surgery durations. In both the previous works, the authors solve the beds management problem limited to week-end beds, while assuming that each specialty has its own post-surgery beds from Monday to Friday with no availability restriction. In [9], some of the previous authors face the beds management problem for all the days of the week, with the aim to level the post-surgery ward bed occupancies during the days, using a Variable Neighbourhood Search approach.

Other relevant approaches are: Abedini et al. [1], that developed a bin packing model with a multi-step approach and a priority-type-duration rule; Molina-Pariente et al. [26], that tackled the problem of assigning an intervention date and an operating room to a set of surgeries on the waiting list, minimizing access time for patients with diverse clinical priority values; and Zhang et al. [29], that addressed the problem of OR planning with different demands from both elective patients and non-elective ones, with priorities in accordance with urgency levels and waiting times. However, beds management is not considered in this last three mentioned approaches.

ASP in scheduling problems. We already mentioned in the introduction that ASP has been already successfully used for solving hard combinatorial and application problems in several research areas. Concerning scheduling problems other than ORS, ASP encodings were proposed for the following problems: *Incremental Scheduling Problem* [14, 20], where the goal is to assign jobs to devices such that their executions do not overlap one another; *Team Building Problem* [28], where the goal is to allocate the available personnel of a seaport for serving the incoming ships; and *Nurse Scheduling Problem* [2, 16, 3], where the goal is to create a scheduling for nurses working in hospital units. Other relevant problems are *Interdependent Scheduling Games* [5], which requires interdependent services among players, that control only a limited number of services and schedule independently, the *Conference Paper Assignment Problem* [7], which deals with the problem of assigning reviewers in the PC to submitted conference papers, and the *Stable Roommates Problem* [6], which is a modified version of the well-known Stable Marriage Problem.

8 Conclusions

In this paper we have employed ASP for solving to the ORS problem with beds management, given ASP has already proved to be a viable tool for solving scheduling problems due to the readability of the encoding, and availability of efficient solvers. Specifications of the problem are modularly expressed as rules in the ASP encoding, and ASP solver CLINGO has been used. We finally presented the results of an experimental analysis on ORS benchmarks with realistic sizes and parameters on two scenario, that reveal that our solution is able to utilize efficiently whichever resource is more constrained, being either the OR time or the beds. Moreover, for the planning length of 5 days usually used in small-medium Italian hospitals, this is obtained in short timings in line with

the needs of the application. Future work includes the design and analysis of a re-scheduling solution, in case the off-line solution proposed in this paper can not be fully implemented for circumstances such as canceled registrations, and the evaluation of heuristics and optimization techniques (see, e.g., [23, 24, 4]) for further improving the effectiveness of our solution.
All materials presented in this work, including benchmarks, encodings and results, can be found at: `http://www.star.dist.unige.it/~marco/RuleMLRR19/material.zip`.

References

1. Abedini, A., Ye, H., Li, W.: Operating Room Planning under Surgery Type and Priority Constraints. Procedia Manufacturing **5**, 15–25 (2016)
2. Alviano, M., Dodaro, C., Maratea, M.: An advanced answer set programming encoding for nurse scheduling. In: AI*IA. LNCS, vol. 10640, pp. 468–482. Springer (2017)
3. Alviano, M., Dodaro, C., Maratea, M.: Nurse (re)scheduling via answer set programming. Intelligenza Artificiale **12**(2), 109–124 (2018)
4. Alviano, M., Dodaro, C., Marques-Silva, J., Ricca, F.: Optimum stable model search: Algorithms and implementation. J. Log. Comput. . https://doi.org/10.1093/logcom/exv061, `http://dx.doi.org/10.1093/logcom/exv061`, in press
5. Amendola, G.: Preliminary results on modeling interdependent scheduling games via answer set programming. In: RiCeRcA@AI*IA. CEUR Workshop Proceedings, vol. 2272. CEUR-WS.org (2018)
6. Amendola, G.: Solving the stable roommates problem using incoherent answer set programs. In: RiCeRcA@AI*IA. CEUR Workshop Proceedings, vol. 2272. CEUR-WS.org (2018)
7. Amendola, G., Dodaro, C., Leone, N., Ricca, F.: On the application of answer set programming to the conference paper assignment problem. In: AI*IA. Lecture Notes in Computer Science, vol. 10037, pp. 164–178. Springer (2016)
8. Aringhieri, R., Landa, P., Soriano, P., Tànfani, E., Testi, A.: A two level metaheuristic for the operating room scheduling and assignment problem. Computers & Operations Research **54**, 21–34 (2015)
9. Aringhieri, R., Landa, P., Tànfani, E.: Assigning surgery cases to operating rooms: A vns approach for leveling ward beds occupancies. Electronic Notes in Discrete Mathematics **47**, 173 – 180 (2015). https://doi.org/https://doi.org/10.1016/j.endm.2014.11.023, `http://www.sciencedirect.com/science/article/pii/S1571065314000651`, the 3rd International Conference on Variable Neighborhood Search (VNS'14)
10. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003). https://doi.org/10.1017/CBO9780511543357
11. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011)
12. Buccafurri, F., Leone, N., Rullo, P.: Enhancing Disjunctive Datalog by Constraints. IEEE Transactions on Knowledge and Data Engineering **12**(5), 845–860 (2000)
13. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T.: ASP-Core-2 Input Language Format (2013), `https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.01c.pdf`

14. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the Fifth Answer Set Programming Competition. Artificial Intelligence **231**, 151–181 (2016)
15. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: Operating room scheduling via answer set programming. In: AI*IA. LNCS, vol. 11298, pp. 445–459. Springer (2018)
16. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: LPNMR. LNCS, vol. 10377, pp. 301–307. Springer (2017)
17. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1), 278–298 (2011)
18. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: ICLP (Technical Communications). OASICS, vol. 52, pp. 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
19. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. **187**, 52–89 (2012)
20. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. Journal of Artificial Intelligence Research **60**, 41–95 (2017)
21. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the Fifth International Conference and Symposium , Seattle, Washington, August 15-19, 1988 (2 Volumes). pp. 1070–1080. MIT Press (1988)
22. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Comput. **9**(3/4), 365–386 (1991)
23. Giunchiglia, E., Maratea, M., Tacchella, A.: Dependent and independent variables in propositional satisfiability. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA. Lecture Notes in Computer Science, vol. 2424, pp. 296–307. Springer (2002)
24. Giunchiglia, E., Maratea, M., Tacchella, A.: (In)Effectiveness of look-ahead techniques in a modern SAT solver. In: Rossi, F. (ed.) CP. Lecture Notes in Computer Science, vol. 2833, pp. 842–846. Springer (2003)
25. Landa, P., Aringhieri, R., Soriano, P., Tànfani, E., Testi, A.: A hybrid optimization algorithm for surgeries scheduling. Operations Research for Health Care **8**, 103–114 (2016)
26. Molina-Pariente, J.M., Hans, E.W., Framinan, J.M., Gomez-Cia, T.: New heuristics for planning operating rooms. Computers & Industrial Engineering **90**, 429–443 (2015)
27. Niemelä, I.: Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. Ann. Math. Artif. Intell. **25**(3-4), 241–273 (1999)
28. Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N.: Team-building with answer set programming in the Gioia-Tauro seaport. Theory and Practice of Logic Programming **12**(3), 361–381 (2012)
29. Zhang, J., Dridi, M., El Moudni, A.: A stochastic shortest-path MDP model with dead ends for operating rooms planning. In: ICAC. pp. 1–6. IEEE (2017)