Operating Room Scheduling via Answer Set Programming: Improved Encoding and Test on Real Data

Carmine Dodaro$^{1\,[0000-0002-5617-5286]}$, Giuseppe Galatà$^{2\,[0000-0002-1948-4469]}$, Martin Gebser$^{3\,[0000-0002-8010-4752]}$, Marco Maratea$^{1\,[0000-0002-9034-2527]}$, * Cinzia Marte$^{1\,[0000-0003-3920-8186]}$, Marco Mochi$^{4\,[0000-0002-5849-3667]}$, and Marco Scanu^{2}

^{1} University of Calabria, Rende, Italy, {`name.surname@unical.it`}
^{2} SurgiQ srl, Genova, Italy, {`name.surname@surgiq.com`}
^{3} University of Klagenfurt, Klagenfurt, Austria, `martin.gebser@aau.at`
^{4} University of Genoa, Genova, Italy, `marco.mochi@edu.unige.it`

Abstract. The Operating Room Scheduling (ORS) problem deals with the optimization of daily operating room surgery schedules. It is a challenging problem subject to many constraints, like to determine the starting time of different surgeries and allocating the required resources, including the availability of beds in different units. In the last years, Answer Set Programming (ASP) has been successfully employed for addressing and solving the ORS problem. Despite its importance, due to the inherent difficulty of retrieving real data, all the analyses on ORS ASP encodings have been performed on synthetic data so far. In this paper, first we present a new, improved ASP encoding for the ORS problem. Then, we deal with the real case of ASL1 Liguria, an Italian health authority operating through three hospitals, and present adaptations of the ASP encodings to deal with the real-world data. Further, we analyze the resulting encodings on hospital scheduling data by ASL1 Liguria. Results on some scenarios show that the ASP solutions produce satisfying schedules also when applied to such challenging, real data.[5]

Keywords: Answer Set Programming · Scheduling · Healthcare

1 Introduction

Hospitals have long waiting times, surgeries cancellation and even worst resource overload: such problems negatively impact the level of patients' satisfaction and the quality of care provided. Within every hospital, Operating Rooms (ORs) are important units. As indicated in [31], the OR management accounts for approximately 33% of the total hospital budget because it includes high staff (e.g., surgeons, anesthetists, nurses) and material costs. Nowadays, in most modern hospitals, long surgical waiting lists

* Corresponding author.

[5] This paper is an extended and revised version of a contribution presented at the 38th Italian Conference on Computational Logic (CILC 2023), available in the CEUR proceedings [33].

are present because of inefficient planning. Therefore, it is of paramount importance to improve the efficiency of OR management, in order to enhance the survival rate and satisfaction of patients, thereby improving the overall quality of the healthcare system.

To manage the ORs, a solution has to provide the date and the starting time of the surgeries required, considering the availability of ORs and beds, and of other resources requested. The Operating Room Scheduling (ORS) [1, 7, 28, 31] problem is the task of assigning patients to ORs by considering specialties, surgery durations, shift durations, and beds' availability, among others. Further, the solution should prioritize patients based on health urgency. In recent years, a solution based on Answer Set Programming (ASP) [8, 24, 25] was proposed and is used for solving ORS problems [17, 18]: this is because ASP combines an intuitive semantics [10] with the availability of efficient solvers [2, 23]. However, due to the inherent difficulty of retrieving real data, all the analysis on ORS encodings via ASP have been performed on synthetic data so far.

In this paper, first we present a precise, mathematical formulation of the problem. Then, we introduce a new, improved ASP encoding for a basic version of the ORS problem previously employed that, e.g., does not take bed management into account. The improved encoding is more efficient not only for ASP, but also for pseudo-Boolean solvers run on benchmarks obtained by automatic translation from the ASP formulation. Further, we deal with the real case of ASL1 Liguria, an Italian health authority operating through three hospitals in the cities of Sanremo, Imperia, and Bordighera in the Liguria region, for computing weekly operating room surgery schedules. Starting from the improved encoding for the basic version of the problem, we present adaptations to deal with the data of the hospitals. We then define three scenarios: in the first scenario, the goal is to test whether our solution is able to replicate the schedules that a hospital indeed followed, while the other two scenarios evaluate whether "better" schedules could have been determined. Further, we analyze the resulting encodings on the real data: results show that our solutions are able to both replicate and improve the original schedules, for a weekly and a monthly planning horizons, thus indicating that ASP produces satisfying results also when applied to such challenging, real data.

The paper is structured as follows. Section 2 describes the ORS problem in an informal way, and its mathematical formulation is presented in Section 3. Sections 4-6 employ ASP: Section 4 introduces the syntax and semantics of ASP, Section 5 compares two alternative encodings of a basic version of the ORS problem, while Section 6 presents the adaptation to real data. Then, Section 7 provides the results of our experiments on the aforementioned scenarios, and includes the impact of the optimized encoding on the performance of other logic-based solving approaches. Section 8 and Section 9 complete the paper by discussing related work, drawing conclusions and pointing out possible targets for future research.

2 Problem Description

This section provides the description and requirements of the ORS problem dealt with by ASL1 Liguria, Italy, which is a local health authority operating through three hospitals: Bordighera, Sanremo, and Imperia. The elements of a surgical waiting list are called *registrations*. Each registration links a particular surgical procedure, required by

Table 1. Beds available in Imperia.

OR	Day 1	Day 2	Day 3	Day 4	Day 5
Gynecology	12	15	15	15	15
Cardiovascular	7	8	8	8	8
General Surgery	5	6	8	9	10
Urology	8	9	12	12	13

Table 2. Beds available in Sanremo.

OR	Day 1	Day 2	Day 3	Day 4	Day 5
Gynecology	15	15	16	18	18
Orthopedics	7	9	8	12	13
ENT	5	5	5	5	5
General Surgery	6	7	9	10	11

a patient, of a specific duration to a reservation number, a specialty, and a type of hospitalization.

The overall goal of the ORS problem is to assign a maximum number of registrations from a long waiting list to the suitable ORs. It is possible that the surgery of some specialty could not be assigned to some ORs. In this way, we can ensure the most efficient use of OR time, which is a very valuable resource: OR costs are estimated in the range of tens of dollars per minute [34], half of them being fixed costs due even when the OR is not treating patients [30]. Considering that no overlap of patients is allowed in the same OR and that we want to avoid OR overload, the first requirement is to guarantee that the sum of the durations of surgeries assigned to a particular OR does not exceed the operational time of the OR. Considering the three hospitals of ASL1 Liguria, Bordighera has two ORs available from 07:30 to 13:30, while Imperia and Sanremo have five ORs available from 07:30 to 20:00.

Moreover, registrations can be linked to different types of hospitalizations. Specifically, patients could undergo *day surgery* or *ordinary surgery*, with the latter requiring the assignment of a bed to the patient before and/or after the surgery. Bed availability is a vital resource for hospitals and often the actual bottleneck in surgical procedures scheduling. Therefore, an OR schedule must ensure that the number of patients requiring a bed for a particular specialty does not exceed the number of available beds per day. The number of beds for different specialties is based on the beds originally assigned to a specialty via the Master Surgical Schedule (MSS), yet beds already occupied by hospitalized patients or otherwise unavailable may decrease the capacity. The total number of beds available on each day of a week for different specialties in Imperia and Sanremo are presented in Tables 1 and 2, while Bordighera has no own beds available.

Other specific aspects of our problem involve patient priorities and ORs utilization. In fact, registrations are not all equal: they can be related to different medical conditions and can be on the waiting list for different periods of time. These two factors can be combined in a unique concept of *priority*. In our setting, we introduced four different priority categories, namely, p_1, p_2, p_3, and p_4. The first one gathers patients already preplanned by the hospital: it is required that each of these registrations is assigned to an OR. Then, the registrations of the other three categories are assigned depending on the ORs' capacities, prioritizing p_2 over p_3 and p_3 over p_4. Further, there are ORs that are used in a limited way, given they are reserved for emergencies and other purposes.

3 Mathematical Formulation

In this section, we present the mathematical formulation of the ORS problem according to the description presented in Section 2. We start by providing some preliminary concepts essential for the definition of the ORS problem. Subsequently, we proceed by presenting the ORS problem in its basic context, along with the additional consideration of bed assignment.

3.1 Preliminaries

Let N be the set of reservation numbers, OR represent the set of operating rooms, and D denote the set of days, where each day consists of two shifts spanning 5 hours, with the hours divided into time slots. Accordingly, let $SH = \{\text{shift}_1, \text{shift}_2\}$ be the set of shifts and $TS = \{t_1, \ldots, t_n\}$ represent the set of time slots. Finally, let SP denote the set of specialities, S represent the set of surgeries, and $P = \{p_1, p_2, p_3, p_4\}$ be the set collecting the four different levels of priorities. We point out that the unit of measurement in terms of time is represented by a time slot. Moreover, let:

- $open : OR \to \mathbb{N}^+$ be the function that associates to each operating room the number of time slots for which it is available;
- $end : SH \to TS$ be the function that associates a shift with its ending time;
- $\Delta : S \to \mathbb{N}^+$ be the function describing the *duration* of each surgery in terms of time slots and let $\Gamma = \{(s, \Delta(s)) \ : \ s \in S\}$ be the set that gathers all surgeries along with their corresponding durations;
- $c : S \to SP$ be the *clustering* function that assigns a surgery to a specific specialty;
- $\tau : SP \times OR \to \{0,1\}$ be the *room assignment* function such that $\tau(x,y) = 1$ if the specialty x is compatible with OR y, 0 otherwise.

Finally, let $COR = \tau^{-1}(1) = \{(x,y) \in SP \times OR \mid \tau(x,y) = 1\}$ be the set collecting all the specialties correctly associated to the ORs.

To link together a reservation number, a priority, and a surgery, we introduce the notion of *registration* via the following definition.

Definition 1 (Registration). *A registration ρ is a function of the form*

$$\rho : N \times P \times S \to \{0,1\}$$

such that $\rho(n,p,s) = 1$ if there exists a reservation n with priority p for the surgery s, 0 *otherwise.*

The next step involves connecting a surgery with its corresponding specialty and identifying the ORs available for its execution. To this end, we define the notion of *assignment*.

Definition 2 (Assignment). *An assignment α is a function of the form*

$$\alpha : \Gamma \times COR \to \{0,1\}$$

such that $\alpha((s,\Delta(s)),(x,y)) = 1$ if there exists a surgery s with duration $\Delta(s)$ such that s belongs to the specialty x, i.e. $c(s) = x$, and x is assigned to the OR y. Otherwise, $\alpha((s,\Delta(s)),(x,y)) = 0$.

In order to consider only suitable tuples, we consider the following sets of elements:

$$R := \rho^{-1}(1) = \{(n,p,s) \in N \times P \times S \mid \rho(n,p,s) = 1\}$$

and

$$A := \alpha^{-1}(1) = \{((s,\Delta(s)),(x,y)) \in \Gamma \times COR \mid \alpha((s,\Delta(s)),(x,y)) = 1\},$$

which collect all the suitable registrations and assignments, respectively. Finally, to join a registration with an assignment related to the surgery, we define the set

$$B := \{(n,p,s,y) \in N \times P \times S \times OR \mid (n,p,s) \in R, \exists x_1,x_2 \ : \ ((s,x_1),(x_2,y)) \in A\}.$$

Therefore, B is the set containing each registration n with priority p for the surgery s allocated to the OR y. Now we can define the notion of *scheduling*, which links together a reservation number, a priority, a surgery, an operating room, a time slot, a shift, and a day.

Definition 3 (Scheduling). *A scheduling σ is a function of the form*

$$\sigma : B \times TS \times SH \times D \to \{0,1\}$$

such that $\sigma((n,p,s,y),u,w,d)) = 1$ if there exists a reservation n with priority p for the surgery s allocated to the OR y in the time slot u of shift w on day d.

The set $\Sigma = \sigma^{-1}(1) = \{\mathbf{t} = ((n,p,s,y),u,w,d) \in B \times TS \times SH \times D \mid \sigma(\mathbf{t}) = 1\}$ collects all the tuples eligible as scheduling. In the rest of the section, with a slight abuse of notation, we may write, for instance, $\sigma((\_,\_,\_,y),u,w,d) = 1$ to denote that there exists a tuple $(n,p,s) \in N \times P \times S$ such that σ assumes value 1.

3.2 ORS Problem

In this section, we define the ORS problem within its basic framework, including day surgery and excluding the specific aspect of bed assignment.

Definition 4 (ORS). *The Operating Room Scheduling (ORS) problem is defined as the problem of finding a set ψ of tuples $\mathbf{t} = ((n,p,s,y),u,w,d) \in \Sigma$ that satisfies the following conditions:*

(c_1) $\forall (n,p,s) \in R$, it holds that $|\{(y,u,w,d) \mid ((n,p,s,y),u,w,d) \in \psi\}| \leq 1$;
(c_2) $\forall (n,p_1,s) \in R$, it holds that $|\{(y,u,w,d) \mid ((n,p_1,s,y),u,w,d) \in \psi\}| = 1$;

(c_3) $\forall \mathbf{t_1} = ((n, \_, \_, y), u, w, d), \mathbf{t_2} = ((n', \_, \_, y'), u, w, d) \in \psi$ *such that* $n \neq n'$*, it holds that* $y \neq y'$*;*

(c_4) $\forall y \in OR, \forall d \in D$*, it holds that* $\sum_{s:((\_,\_,s,y),\_,\_,d)\in\psi} \Delta(s) < open(y)$*;*

(c_5) $\forall \mathbf{t} = ((\_, \_, s, \_), u, w, \_) \in \psi$*, it holds that* $u + \Delta(s) < end(w)$*;*

(c_6) $\forall \mathbf{t_1} = ((\_, \_, s, y), u_1, w, d), \mathbf{t_2} = ((\_, \_, \_, y), u_2, w, d) \in \psi$ *such that* $u_1 < u_2$*, it holds that* $u_1 + \Delta(s) < u_2$*.*

The specified conditions are necessary to enforce the following constraints: (c_1) for each registration there exists at most one scheduling; (c_2) all registrations with priority p_1 must be scheduled; (c_3) double occupation of the same operating room is not allowed; (c_4) the total duration of surgeries assigned to a specific operating room must be within the operational time of the room; (c_5) the duration of surgeries cannot exceed the end of the assigned shift; (c_6) overlapping schedules for the same operating room are prohibited.

3.3 ORS Problem for Ordinary Surgery

In the context of ordinary surgery, the ORS problem entails the assignment of a bed to the patient either before and/or after the surgery. Therefore, a solution to the problem must also consider the allocation of a bed for the patient and the corresponding availability of beds for each specialty and day.

To this aim, we need to define the following functions. Let

- $\beta : SP \times D \to \mathbb{N}_0$ be the function that returns the number of available beds for a specific specialty on a given day;
- $cd : R \to \mathbb{N}_0 \times \mathbb{N}_0$ be the required beds function, with $cd(n, p, s) = (\varepsilon_1, \varepsilon_2)$, where ε_1 (resp., ε_2) represents the number of days preceding (resp., following) the surgery for which the bed is required.

Definition 5 (ORS-OS). *Let* $rb : R \times D \to \{0, 1\}$ *be the function that maps a registration to a bed on a day such that, for each* $((n, p, s, y), u, w, d) \in \psi$*, it holds that* $rb((n, p, s), d') = 1$*, where* $d' \in [d - \varepsilon_1, d + \varepsilon_2]$ *and* $cd((n, p, s)) = (\varepsilon_1, \varepsilon_2)$*.*
The Operating Room Scheduling for Ordinary Surgery (ORS-OS) problem is defined as the problem of finding a set ψ *of tuples* $\mathbf{t} = ((n, p, s, y), u, w, d) \in \Sigma$ *satisfying conditions (*c_1*), …, (*c_6*) such that:*

(c_7) $\forall x \in SP, \forall d \in D$*, it holds that* $|RB| \leq \beta(x, d)$*, where* $RB = \{(n, p, s) \in R \mid c(s) = x, rb((n, p, s), d) = 1\}$*.*

Specifically, (c_7) ensures that every day the number of required beds for a given specialty must be less than or equal to the number of available beds for that specialty.

3.4 Maximal Scheduling Solution

In Section 2, we have defined four different priority categories: p_1, p_2, p_3, and p_4. In accordance with constraint (c_2), all surgeries categorized under priority p_1 must be scheduled. Consequently, among the remaining priorities, we adopt a prioritization scheme favoring p_2 over p_3 and p_3 over p_4.

To compare different solutions in terms of the number of schedulings with a specific priority, we provide the following definition.

Definition 6 (Dominating solution). *Let $\Sigma_{\psi}^{p_i} := \{((n,p,s,y),u,w,d) \in \psi \mid p = p_i\}$ be the set collecting all the tuples corresponding to a specific priority p_i that constitute a solution for the problem under consideration. A solution ψ dominates a solution ψ' if $|\Sigma_{\psi'}^{p_2}| < |\Sigma_{\psi}^{p_2}|$, or if $|\Sigma_{\psi'}^{p_2}| = |\Sigma_{\psi}^{p_2}| \Rightarrow |\Sigma_{\psi'}^{p_3}| < |\Sigma_{\psi}^{p_3}|$, or if $|\Sigma_{\psi'}^{p_3}| = |\Sigma_{\psi}^{p_3}| \Rightarrow |\Sigma_{\psi'}^{p_4}| < |\Sigma_{\psi}^{p_4}|$.*

Consequently, we define the notion of a *maximal solution*.

Definition 7 (Maximal scheduling solution). *A scheduling solution is maximal if it is not dominated by any other scheduling solution.*

The optimization variant of the ORS (resp., ORS-OS) problem, denoted as ORS* (resp., ORS-OS*), is to find a maximal scheduling solution.

4 Background on Answer Set Programming

Answer Set Programming (ASP) [8] is a programming paradigm developed in the field of non-monotonic 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 [8, 10]. Hereafter, we assume the reader is familiar with logic programming conventions.

Syntax. The syntax of ASP is inspired by the one of Prolog. Variables are strings starting with an uppercase letter, and constants are 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 an 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:

$$a_1 \mid \dots \mid a_n \text{ :- } b_1, \dots, b_k, not\ b_{k+1}, \dots, not\ b_m.$$

where $a_1, \dots, a_n$ are standard atoms, $b_1, \dots, b_k$ are atoms, $b_{k+1}, \dots, b_m$ are standard atoms, and $n \geq 0, m \geq k \geq 0$. A literal is either a standard atom a or its negation $not\ \ a$. The disjunction $a_1 \mid \dots \mid a_n$ is the *head* of r, while the conjunction $b_1, \dots, b_k, not\ b_{k+1}, \dots, not\ b_m$ 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* [9] ω is of the form:

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

where w and l are the weight and level of ω, respectively. (Intuitively, $[w@l]$ is read as “weight w at level l”, where the weight is the “cost” of violating the condition in the body of ω, 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., $not\ \ell$) 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* [20] 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 [20].

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 $p \mid p'$, where p' is a fresh atom not appearing elsewhere in the program, meaning that the atom p can be chosen as true.

5 ASP Encodings for the ORS* problem

In this section, we present two encodings for the ORS* problem, thus without bed management (with priority level limited to 3 as in previous formulations of the problem), one of which will be the base encoding to further employ in later sections. We start by presenting an encoding used for solving the ORS* problem in the earlier version of this paper [33], and then we introduce a new, improved encoding. The ASP encodings are based on the input language of CLINGO [22]. Finally, through an analysis conducted on synthetic data, we show the benefits of the new version, which thus motivates the adoption of this encoding for the real-world ORS-OS* problem at ASL1 Liguria, to be addressed in the next section.

5.1 ORS* Encoding

In the following, we present the input and output data model as well as the first ASP encoding for the ORS* problem.

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

- Constant `shift_duration` represents the duration of every shift in terms of time slots.
- Constants `totRegsP2` and `totRegsP3` represent the number of registrations of patients with priority p_2 and p_3, respectively.
- Instances of `registration(ID,P,SP,DUR)` represent the registration of the patient identified by an ID (`ID`) with priority level (`P`), the requested specialty (`SP`), and the duration of the surgery (`DUR`).
- Instances of `mss(OR,SP,SHIFT,DAY)` represent which specialty (`SP`) is assigned to an OR (`OR`) in a shift (`SHIFT`) on a day (`DAY`).
- Instances of `time(SHIFT,TS)` represent the available time slots (`TS`) in the shift (`SHIFT`).

The output is an assignment represented by atoms of the form

```
x(ID,P,OR,DAY,SHIFT,TS),
```

where the intuitive meaning is that the patient identified by an ID (`ID`) having a priority (`P`) is assigned to the OR (`OR`) in the shift (`SHIFT`) at the time slot (`TS`) on the day (`DAY`).

```
1 {x(ID,P,OR,DAY,SHIFT,TS): TS+DUR <= shift_duration} :- registration(ID,P,SP,DUR),
      mss(OR,SP,SHIFT,DAY), time(SHIFT,TS).
2 :- #count{P,OR,DAY,SHIFT,TS: x(ID,P,OR,DAY,SHIFT,TS)} > 1, registration(ID,_,_,_).
3 :- #count{ID: x(ID,_,OR,DAY,SHIFT,TS), registration(ID,_,_,DUR), T >= TS, T < TS+DUR} > 1,
      mss(OR,_,SHIFT,DAY), time(SHIFT,T).
4 :- #count{ID: x(ID,1,_,_,_,_)} < totRegsP1.
5 :~ M = #count{ID: x(ID,2,_,_,_,_)}, N = totRegsP2 - M. [N@3]
6 :~ M = #count{ID: x(ID,3,_,_,_,_)}, N = totRegsP3 - M. [N@2]
```

Fig. 1. ASP encoding for the ORS* problem.

Encoding. The related encoding is shown in Figure 1 and described next. To simplify the description, we denote the rule appearing at line *i* of Figure 1 by r_i.

Choice rule r_1 permits assigning an OR, a day, a shift, and a time slot to each registration. Rule r_2 ensures that each registration is assigned at most once. Rule r_3 ensures that, at every time slot, at most one registration is assigned to an OR on the same day and shift. Rule r_4 ensures that every registration with priority p_1 is assigned to some OR on a day, shift, and time slot. The weak constraints r_5 and r_6 minimize the number of unassigned registrations with priority p_2 or p_3, respectively.

5.2 Improved ORS* Encoding

In this subsection, we present a new, optimized version of the encoding for the ORS* problem. This encoding, differently from the previous one, does not try to assign a possible time slot together with the day, shift, and the OR. Instead, since the order of patients has no impact on the usage of the other resources (such as beds considered in the next section), we can just assign an OR, a day, and a shift, and then determine the starting times of the surgeries by taking any order of the patients sharing an OR on the same day and shift.

Data Model. The input data is the same as presented in Subsection 5.1, without the predicate `time` and the constants `totRegsP2` and `totRegsP3`, while the output is an assignment represented by atoms of the form

`x(ID,P,DUR,OR,SHIFT,DAY)`,

where the intuitive meaning is that the patient identified by an ID (`ID`) having a priority (`P`) with a surgery duration (`DUR`) is assigned to the OR (`OR`) in the shift (`SHIFT`) on the day (`DAY`), together with atoms of the form

`end(ID,Q)`,

meaning that the greatest integer for `Q` is the ending time for the surgery of the patient.

Encoding. The related encoding is shown in Figure 2, where we again denote the rule appearing at line *i* of Figure 2 by r_i.

Rule r_1 defines an auxiliary predicate that represents the possible ORs, days, and shifts to which a registration can be assigned. Choice rule r_2 permits assigning at most

```
 1 feasible(ID,P,DUR,OR,SHIFT,DAY) :- registration(ID,P,SP,DUR), mss(OR,SP,SHIFT,DAY),
         DUR <= shift_duration.
 2 {x(ID,P,DUR,OR,SHIFT,DAY) : feasible(ID,P,DUR,OR,SHIFT,DAY)} 1 :- feasible(ID,_,_,_,_,_).
 3 :- #sum{DUR,ID : x(ID,_,DUR,OR,SHIFT,DAY)} > shift_duration, mss(OR,SHIFT,_,DAY).
 4 overlap(ID1,DUR1,ID2,DUR2) :- x(ID1,_,DUR1,OR,SHIFT,DAY), x(ID2,_,DUR2,OR,SHIFT,DAY),
         ID1 < ID2.
 5 {ordering(ID1,ID2,DUR2)} :- overlap(ID1,DUR1,ID2,DUR2).
 6  ordering(ID2,ID1,DUR1)  :- overlap(ID1,DUR1,ID2,DUR2), not ordering(ID1,ID2,DUR2).
 7 end(ID,0..DUR-1) :- feasible(ID,_,DUR,_,_,_).
 8 end(ID,Q) :- ordering(ID1,ID,DUR), end(ID1,Q1), Q = Q1+DUR, Q <= shift_duration.
 9 :- end(ID,shift_duration).
10 :- registration(ID,1,_,_), not x(ID,1,_,_,_,_).
11 :~ registration(ID,P,_,_), not x(ID,P,_,_,_,_), 1 < P. [1@-P,ID]
```

Fig. 2. Improved ASP encoding for the ORS* problem.

one OR, a day, and a shift to each registration. Rule r_3 ensures that, for every OR, day, and shift, the sum of the durations of the assigned surgeries does not exceed the constant `shift_duration`. Rule r_4 defines a predicate indicating registrations sharing the same OR, shift, and day. Rules r_5 and r_6 make use of this predicate to choose an ordering between registrations assigned to an OR on the same day and shift. Rules r_7 and r_8 propagate the ending times for surgeries along the chosen ordering, leading to an interval $[0,\mathtt{Q}]$ to represent the ending time `Q` for a surgery. Rule r_9 ensures that the ordering of operations is non-circular. Moreover, r_{10} ensures that every registration with priority p_1 is assigned, while the weak constraint r_{11} minimizes the number of unassigned registrations with priority p_2 or p_3, respectively.

5.3 Preliminary Comparative Analysis

Here, we present a preliminary comparative analysis of the two encodings given in Figure 1 and Figure 2. In the first paragraph, we describe the benchmarks used for the comparison, while the second one discusses the results. The comparison has been carried out on an Apple M1 CPU @ 3.22 GHz machine with 8 GB of physical RAM and a time limit of 60 seconds per run. The ASP system used was CLINGO 5.6.2, configured with the parameters *--restart-on-model* and *--parallel-mode=6*: these parameters have been found to be effective in a preliminary analysis we performed with several options.

Benchmarks. To compare the two encodings, we used the same synthetic data as previously taken to test the ORS* problem with surgical teams [16], where we disregard such teams here. The data emulate the operations of a typical medium-sized Italian hospital: the setting is composed of 5 different specialties, 10 ORs, and 70 registrations per day by patients with priority p_1, p_2, and p_3. We considered 4 distinct scenarios based on the scheduling duration: 1 day, 2 days, 3 days, and 5 days. Each day consists of two shifts, each spanning 5 hours, with the hours divided into time slots. We tested the encodings based on different values for the length of time slots: 10 minutes, 20 minutes, 30 minutes, and 60 minutes. For every scenario, characterized by a particular number of days and length of the time slots, we randomly generated 10 instances.

Table 3. Comparison of the results obtained by the Base and the Optimized encoding for the ORS* problem on the 5 days scenario. The cells provide the number of patients of priority p_2 (P2, top) and p_3 (P3, bottom) that could not be assigned.

Instance		10 Minutes		20 Minutes		30 Minutes		60 Minutes	
		Base	Optimized	Base	Optimized	Base	Optimized	Base	Optimized
1	P2	13	11	12	11	1	1	1	1
	P3	128	79	136	98	139	98	81	81
2	P2	13	6	5	5	7	6	1	1
	P3	134	114	94	83	85	90	74	77
3	P2	16	13	5	5	11	10	6	6
	P3	140	140	90	81	134	79	74	77
4	P2	1	0	5	5	6	5	9	9
	P3	136	85	142	95	92	84	84	75
5	P2	12	9	4	4	6	5	4	4
	P3	129	129	94	94	130	130	74	75
6	P2	11	9	9	7	5	5	9	9
	P3	102	85	98	81	148	91	135	86
7	P2	2	2	7	6	6	6	14	14
	P3	144	108	141	88	133	86	136	77
8	P2	7	5	3	2	10	10	10	10
	P3	130	130	104	91	139	79	140	72
9	P2	12	9	4	4	9	9	2	2
	P3	130	130	139	88	137	80	80	82
10	P2	13	10	11	10	7	6	6	6
	P3	134	86	126	91	142	142	80	82
Avg. Values	P2	10.0	7.4	6.5	5.9	6.8	6.2	6.2	6.2
	P3	130.7	108.6	116.4	89.0	127.9	95.9	95.8	78.4

Results. Table 3 summarizes the results obtained by the two encodings considering 5 days with different lengths of time slots. Both encodings are unable to reach (proven) optimal solutions on the scenario with 5 days within the time limit. As the table shows, the optimized encoding leaves the same number or fewer patients with the higher priority p_2 unassigned on each tested instance. Moreover, the optimized encoding is able to assign not only more patients with priority p_2, thus providing a better solution, but it almost always assigns more patients with priority p_3 too. These results are also confirmed on the smaller scenarios covering 1, 2, or 3 days. Finally, we note that, while the base encoding does not reach any (proven) optimal solution for the instances, the optimized encoding provides optimal solutions on all instances of the 1 day scenario.

Thus, based on the presented analysis, we decided to use the optimized encoding in Figure 2 as the starting point for an extension to the ORS-OS* problem, as dealt with in the real case of ASL1 Liguria.

6 ASP Encoding for the ORS-OS* Problem

Starting from the improved ORS* encoding in the previous section, here we present our compact and efficient ASP solutions for the ORS-OS* problem, introduced in Section 3, which includes bed management (see Section 3.3) and, moreover, considers real hospital scheduling data. The section is divided into two subsections for presenting solutions to the scenarios we deal with: replicate the hospital's schedule and improve such a schedule, respectively. However, the real data do not include information about surgeries' starting times and shifts, so that we disregard time slots and shifts in the following. The replication encoding in Subsection 6.1 can be understood as a tool for checking whether the available capacities of ORs and beds are respected. This forms the base for two ORS-OS* encodings in Subsection 6.2, aiming to assign a maximal amount of new registrations in addition to patients already scheduled by the hospital. These two encodings vary in whether the original assignments can be moved to another OR and day, or not, and a constraint on the limited use of a specific OR, which has been identified in the original hospital schedule, is considered with one of the ORS-OS* encodings.

6.1 Replicating the Hospital Schedule

Here, we present the input and output data model, and the ASP encoding replicating the original schedule of the hospital.

Data Model. The input data is specified by means of the predicates `registration` and `mss` of Subsection 5.1, slightly adjusted and presented here in the modified version, and the predicates `beds` and `givenSchedule` introduced here for the first time.

- Instances of `registration(ID,P,SP,DUR,D1,D2)` represent the registration of the patient identified by an ID (`ID`) with priority level (`P`), the requested specialty (`SP`), the duration of the surgery (`DUR`), and the number of days in which a bed is required before (`D1`) and after (`D2`) the surgery.
- Instances of `mss(OR,SP,DAY)` represent which specialty (`SP`) is assigned to an OR (`OR`) on a day (`DAY`).
- Instances of `beds(N,SP,DAY)` represent the number (`N`) of available beds for a specialty (`SP`) on the day (`DAY`).
- Instances of `givenSchedule(ID,DAY,OR)` represent the original schedule of the hospital, characterized by the patient identified by an ID (`ID`) scheduled on a day (`DAY`) in an OR (`OR`).

The output is an assignment represented by atoms of the form

```
x(ID,P,DUR,OR,DAY),
```

where the intuitive meaning is that the patient identified by an ID (`ID`) having a priority (`P`) with a surgery duration (`DUR`) is assigned to the OR (`OR`) on the day (`DAY`).

```
4 stay(ID,SP,DD) :- x(ID,P,DUR,OR,DAY), registration(ID,P,SP,DUR,D1,D2), beds(N,SP,DD),
        D1 + D2 > 0, DD = D-D1..D+D2.
5 :- #count{ID: stay(ID,SP,D)} > N, beds(N,SP,D).
6 :- givenSchedule(ID,DAY,OR), not x(ID,_,_,OR,DAY).
```

Fig. 3. ASP encoding for replicating the original hospital schedule.

```
8 :- registration(ID,1,_,_,_,_), not x(ID,1,_,_,_).
9 :~ registration(ID,P,_,_,_,_), not x(ID,P,_,_,_), 1 < P. [1@-P,ID]
```

Fig. 4. ASP rules for dealing with priorities.

Encoding. The related encoding includes rules similar to r_1, r_2, and r_3 from Figure 2, with the predicates `registration`, `mss`, and `x` adjusted as described above. Regarding the additional rules shown in Figure 3, the predicate defined by r_4 indicates the days before and after a patient's surgery on which a bed is required for the respective specialty. Rule r_5 ensures that the number of patients requiring a bed for a particular specialty does not exceed the number of available beds per day. Moreover, r_6 ensures that the newly generated schedule coincides with the original hospital schedule.

6.2 Improving the Original Hospital Schedule

We now turn to ASP encodings that aim at improving the original hospital schedule. The idea is to handle both the assignments of patients originally scheduled by the hospital as well as additional, new registrations. Our first encoding, referred to by OPT1, assigns the patients of the original schedule without requiring that the original OR and day are maintained, while the second encoding, denoted by OPT2, keeps the original patients as scheduled by the hospital. Moreover, OPT2 incorporates a constraint derived from the real data.

Data Model. The input data for the encodings OPT1 and OPT2 is the same as presented in Subsection 6.1, yet the predicate `givenSchedule` is not used by OPT1.

Encodings. The ASP encoding OPT1 modifies the replication encoding in Figure 3 by dropping r_6, so that the original assignments of patients can be changed, while r_8 and r_9 in Figure 4 are added. Similar to the previous encoding in Figure 2, the new rules ensure that patients with priority p_1 get assigned, while the number of assignments for patients with priority p_2, p_3, or p_4 is maximized in decreasing order of significance. The requirement that originally scheduled patients get assigned can thus be expressed by categorizing them as priority p_1.

The second encoding OPT2 keeps the original hospital schedule unchanged by including all rules from Figure 3 and Figure 4, i.e., r_6 is not dropped. Moreover, the rule r_{10} in Figure 5 is added to ensure that the specific OR "OR A" is assigned to at most one patient, as "OR A" was reserved for emergencies and used in this limited way in the original data.

```
:- #count{ID: x(ID,_,_,"OR A",_)} > 1.
```

Fig. 5. ASP rule that encodes a constraint from the real data.

7 Experimental Results

In this section, we report the results of an empirical analysis conducted using the defined ASP encodings, on the scenarios previously defined. For all the settings we used the original data.

The experimental setting is the same as Section 5.3, including the time limit set to 60 seconds. Encodings and benchmarks employed in this section (as well as those of Section 5) can be found at: `https://github.com/MarcoMochi/JLC2023ASL1`.

7.1 Benchmarks and Tested Scenarios

Data Description. To test our solution for scheduling surgeries, we utilized data from the hospitals of ASL1 in the Liguria region, Italy. The hospitals in ASL1 serve a population of around $213,000$ people. For our analysis, we used data from a weekly schedule of surgeries across the three hospitals, as well as data from other weeks, including a list of available beds and ORs for all hospitals.

We collected and prepared the data for testing by working with five different xls files, each file represents a different type of data, in particular:

– The operating list of the considered week of surgeries, from 04/03/2019 to 10/03/2019, which provided information on the required surgery, the operating room, and the specialty originally scheduled.
– The historical list of surgeries scheduled in 2019, which includes information on the required surgery, the starting and ending time of the surgery, and the date of the surgery.
– The list of ORs in each hospital and their opening hours.
– The list of patients hospitalized the week before the considered week of the scheduling, along with their admission and discharge times.
– The list of beds in each specialty at each hospital.

Tested scenarios. Having presented the data, now we will present the different scenarios we utilized to test the encodings. In the first scenario, the solution has to provide a schedule for the patients of the considered week and the number of available resources, beds and ORs, replicating the original schedule. This scenario was meant to confirm the consistency of the schedule produced by our encoding with the schedule of the patients as done by the hospital. For the two remaining scenarios, OPT1 and OPT2, we wanted to test our solution by scheduling the patients scheduled by the hospital plus other patients. This enabled us to assess how the ASP solutions could have improved patient's assignment and optimized resource allocation. In particular, in OPT1, we considered both the original patients assigned by the hospital and additional new patients, without imposing constraints requiring the ASP solution to replicate the hospital's schedule,

Table 4. Percentage usage of ORs in Bordighera. A "-" means that the OR is not available on that day.

OR	Day 1	Day 2	Day 3	Day 4	Day 5	Average
OR A	-	-	-	8.2%	-	8.2
OR B	59.1%	69.7%	70.0%	74.2%	80.0%	70.6%

Table 5. Percentage usage of ORs in Imperia. A "-" means that the OR is not available on that day.

OR	Day 1	Day 2	Day 3	Day 4	Day 5	Average
OR A	57.9 %	85.9 %	50.4 %	86.3 %	39.6 %	64.0 %
OR B	44.5 %	48.0 %	45.0 %	41.6 %	60.1 %	47.8 %
OR C	24.9 %	24.7 %	32.3 %	38.0 %	32.0 %	30.4 %
OR E	25.3 %	34.0 %	36.3 %	25.2 %	28.4 %	29.8 %
OR Ophthalmology	38.5 %	38.4 %	-	-	-	38.5 %

while, in OPT2, the solution had to schedule the original patients in the same way done by the hospital. In the hospital of Bordighera, we had to make a slight change between OPT1 and OPT2. Indeed, the hospital scheduled just 1 patient in one OR, without using it for other patients. Thus, we decided to discard this OR in OPT1, while we maintained it just for that patient in OPT2 (this is linked to rule r_{10} in Figure 5). Both for OPT1 and OPT2, a selection of new patients was needed. To select these additional patients and distinguish them from the original ones, we made use of the concept of priority. In particular, originally scheduled patients were assigned priority 1 (we remind that patients with priority 1 are forced to be assigned, freely in OPT1 while following the original schedule in OPT2). Then, we randomly selected a number of patients assigned by the hospital in the following weeks, assigning priority 2 to patients assigned in the next week, priority 3 to patients assigned after 2 weeks, and priority 4 to patients assigned at least 3 weeks later. The number of additional patients the solution will try to schedule is linked to the original number of scheduled patients. Each scenario was tested with 10 different instances composed of different samples of additional patients. In particular, for all the hospitals, the solution tried to schedule a number of patients equivalent to the 250% of the original one. We decided to increase the number of patients to 250% because, after a preliminary analysis, we found that even when this value was increased, the number of patients assigned did not increase. Conversely, using a lower value resulted in solutions where all the patients were assigned. OPT1 and OPT2 enabled us to assess the potential impact of our solution in terms of reducing patient waiting lists and optimizing resource allocation.

7.2 Results for Scenario 1

Scenario 1 consists of recreating the same schedule done by the hospital. The solution is obtained in less than 0.5 seconds for all the hospitals, indicating the correctness of our solution. In Tables 4, 5, and 6 it is possible to see the original percentage usage of

Table 6. Percentage usage of ORs in Sanremo. A "-" means that the OR is not available on that day.

OR	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	AVERAGE
OR 1	59.1 %	51.9 %	25.7 %	41.9 %	74.0 %	50.5 %
OR 2	-	21.6 %	63.2 %	-	-	42.4 %
OR 3	24.7 %	34.0 %	-	-	24.8 %	27.8 %
OR 4	-	35.3 %	-	86.3 %	-	60.8 %
OR C	12.9 %	-	-	-	14.4 %	13.7 %

Table 7. Number of assigned patients in Bordighera grouped by their priority level.

P1	P2	P3	P4
28/28	14/29	0/28	0/13
28/28	15/29	0/28	0/13
28/28	13/29	0/28	0/13
28/28	14/29	0/28	0/13
28/28	14/29	1/28	0/13
28/28	14/29	0/28	0/13
28/28	13/29	2/28	1/13
28/28	14/29	1/28	0/13
28/28	14/29	0/28	0/13
28/28	13/29	0/28	0/13

the ORs obtained by the three hospitals of Bordighera, Imperia, and Sanremo, respectively. These results represent the benchmarks to compare to when evaluating OPT1 and OPT2.

7.3 Results for OPT1

Tables 7, 8, and 9 show the results obtained in this setting in terms of the number of patients assigned on all the ten instances in the hospitals of Bordighera, Imperia, and Sanremo, respectively. As can be seen from the tables, the solution is able to assign a consistent number of additional patients of priority levels P2, P3, and P4 for all the hospitals. Indeed, patients P1 are the patients originally scheduled, while all the other patients represent the additional ones. Moreover, even if not all the patients with priority 2 are assigned, some patients with lower priorities are. This is due to the fact that a bottleneck of the hospitals taken into account is beds availability. Thus, once all the beds are occupied, the solution is able to schedule some patients with lower priorities that do not require a bed before and/or after the surgery. To corroborate the explanation above, the percentages of beds usage in the different specialties, in Sanremo, are presented in Table 10. From the table it can be seen that the beds are used almost at full capacity throughout the week; thus, for the solution is not possible to assign additional patients requiring a bed. In the hospital of Imperia, beyond the beds, even the ORs are used almost at full capacity with the additional patients. In particular, in Figure 6, it can be seen a comparison between the obtained percentage usage of the ORs with the ASP

Table 8. Number of assigned patients in Imperia grouped by their priority level.

P1	P2	P3	P4
143/143	112/120	109/130	63/108
143/143	112/120	109/130	59/108
143/143	112/120	109/130	59/108
143/143	112/120	109/130	55/108
143/143	112/120	109/130	61/108
143/143	112/120	109/130	65/108
143/143	112/120	109/130	60/108
143/143	112/120	109/130	49/108
143/143	112/120	109/130	63/108
143/143	112/120	109/130	62/108

Table 9. Number of assigned patients in Sanremo grouped by their priority level.

P1	P2	P3	P4
43/43	12/28	7/26	5/54
43/43	12/28	7/26	6/54
43/43	12/28	7/26	3/54
43/43	12/28	7/26	4/54
43/43	12/28	7/26	5/54
43/43	12/28	7/26	5/54
43/43	12/28	7/26	9/54
43/43	12/28	7/26	7/54
43/43	12/28	7/26	5/54
43/43	12/28	7/26	1/54

solution in OPT1 and the usage obtained by the ASL1. Without following the previous assignments of ASL1, the ASP solution is able to schedule three times the number of patients originally scheduled.

7.4 Results for OPT2

After having tested, with the scenario OPT1, a solution assigning all the patients with no constraints, we now check the performance in the more constrained scenario OPT2, where the original schedule and a constraint entailed by real data are considered. The results obtained in this scenario are summarized for each hospital in Figure 7. From the figure, it can be noted that even in this more constrained scenario we are able to assign a consistent number of patients of priority levels P2, P3, and P4 for all the hospitals. Compared to OPT1, results are overall similar, and mixed if we consider individual hospitals.

In particular, in Bordighera, the two scenarios assign the same number of patients. However, we remind that in this hospital one patient is assigned to an OR by the additional constraint for OPT2, which is discarded in OPT1. Therefore, even using fewer resources, the solution of OPT1 is still able to match the performance of OPT2.

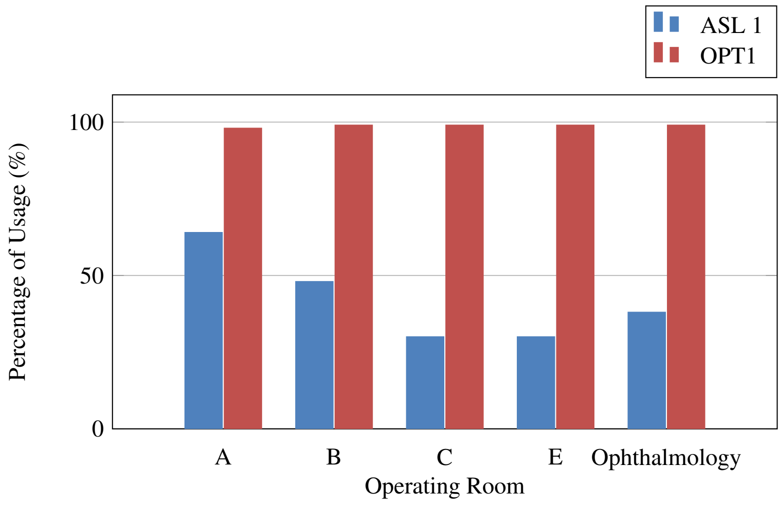

Fig. 6. Comparison of the ORs usage in Imperia between the ASP solution of OPT1 and the ASL1 schedule.

Table 10. Percentage of usage of beds in Sanremo.

Gynecology	Orthopedics	ENT	General Surgery
90%	100%	95%	100%
90%	100%	95%	99%
90%	100%	94%	97%
90%	100%	82%	99%
90%	100%	97%	99%
90%	100%	94%	97%
82%	100%	97%	97%
90%	100%	90%	97%
90%	100%	100%	97%
82%	100%	85%	100%

In Imperia, the total number of patients assigned by OPT2 is actually higher than the one assigned in OPT1 but, as can be seen in Table 11, the schedule provided by OPT1 is of higher quality, since it is able to assign more patients with higher priority. Finally, in Sanremo, the schedule done by OPT1 outperforms that of OPT2 by assigning more patients while not decreasing the quality of the solution.

7.5 Results on Monthly Data

After having presented the results obtained using the data corresponding to one week of surgeries and evaluated the viability of our solution, we decided to further test our solution using one month of data. This test has been done to ensure that our solution was able to outperform the schedule of the hospital when considering a longer planning

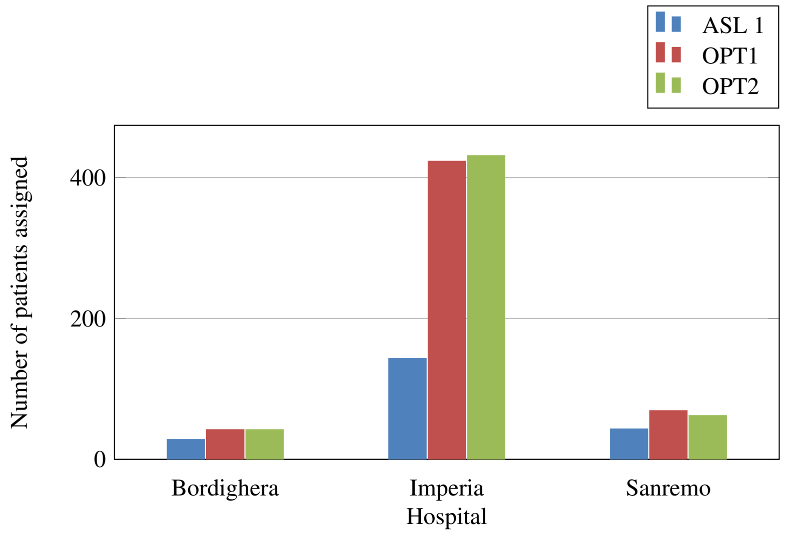

Fig. 7. Comparison on the number of patients scheduled by ASL1, and by OPT1 and OPT2 solutions.

Table 11. Mean percentage of assigned patients with different priorities in Imperia for OPT1 and OPT2.

SETTING	P1	P2	P3	P4
OPT1	100%	93%	83%	41%
OPT2	100%	90%	83%	63%

horizon. This is because, given our weekly solution schedules a number of patients that lead to having the resources completely used, it could be the case that these results are obtained at the price of sacrificing the performance in the following weeks (but we will show that this is not the case). In particular, we decided to use the data of the Imperia hospital, the biggest one of the ASL1 health authority. We decided to start the analysis from the same week used in Section 7, since it is the only week with the information regarding the beds' occupation. Moreover, we collected the new registrations from the following weeks as new input and derived the availability of the beds from the schedule of the week before the considered one. Following this schema, we started with 10 different instances having the original values for the usage of the beds. Subsequently, we used the result of each instance to derive the beds' availability, and the patients still not assigned in the subsequent week, and used them as new inputs for the instances of the next week. As a result, for every week, we have 10 different instances whose resource availability depends on the instances of the previous week.

Figure 8 shows the total cumulative number of patients assigned in each week by our solution in all the 10 instances, compared to the result of the hospital. From the results, we can state not only that the solution is able to schedule more patients in a week,

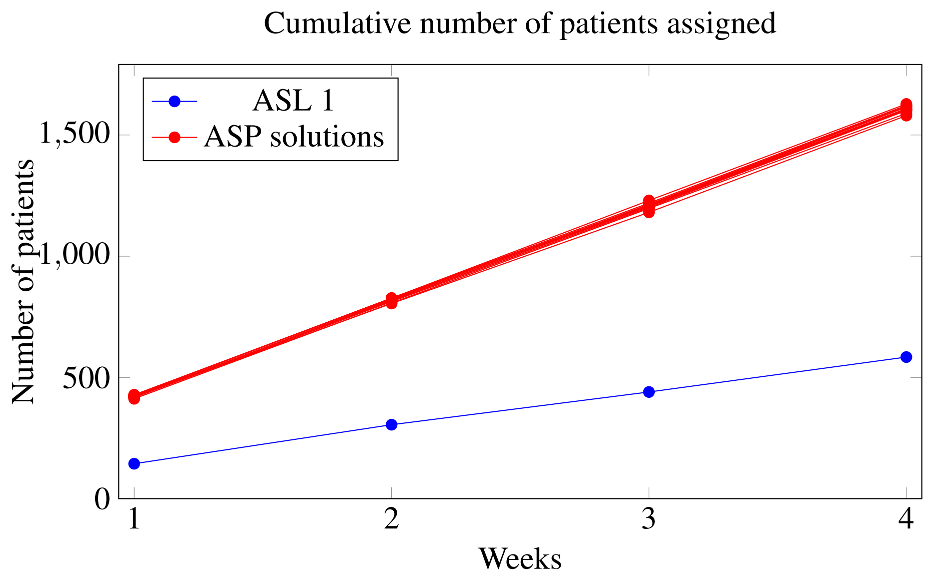

Fig. 8. Cumulative number of patients assigned by the hospital (blue line) in 4 weeks compared to the number of patients assigned by the ASP-based solution (red line) in all the 10 instances of the Imperia hospital.

compared to the hospital, but that our solution is able to assign more patients week after week. Moreover, by inspecting the single instances, in all the 10 instances (dealing with different registrations, thus different requirements in terms of surgery duration and beds requirement) the quality of the solution did not decrease, meaning that our solution is not assigning too many patients in a week at the cost of the quality of the schedules in the subsequent weeks.

7.6 Comparison to Alternative Logic-based Formalisms

In the following, we present an empirical comparison of the original solution presented in [33] and our new optimized ASP-based solution to the scenario OPT1 presented in Section 7 on an alternative logic-based formalism, obtained by applying automatic translations of ASP instances. With this analysis, we want to ensure that the new solution presented in Section 5.1 leads to better results than the original solution even when using different logic-based formalisms. In more detail, we used the ASP solver WASP [3], with the option `--pre=wbo`, which converts ground ASP instances into pseudo-Boolean instances in the wbo format [32].

Then, we considered CLINGO, with option `restart-on-model` (CLINGO-ROM), and the state-of-the-art industrial tool GUROBI [27], which are able to process instances in the wbo format.

The experiment was executed on the 10 instances of the Imperia hospital, which we remind is the biggest hospital of the ASL1 health authority, with a timeout of 60

Table 12. Comparison of the original and the optimized ASP solution using CLINGO with the option `restart-on-model` (CLINGO-ROM in the table) and the alternative logic-based solution GUROBI on wbo instances. The value in each cell represents the time in seconds required to reach an optimal solution if the solver was able to find it in less than 60 seconds, or a percentage value representing the gap to the optimal solution.

Instance	Original Solution		Optimized Solution	
	CLINGO-ROM	GUROBI	CLINGO-ROM	GUROBI
1	0.01%	0.0001%	0.001%	2s
2	0.01%	0.0002%	0.001%	3s
3	0.01%	36s	0.001%	2s
4	0.01%	13s	0.001%	2s
5	0.01%	18s	0.001%	5s
6	0.01%	13s	0.001%	2s
7	0.01%	0.0001%	0.001%	2s
8	0.01%	0.0002%	0.001%	2s
9	0.01%	0.0003%	0.001%	3s
10	0.01%	0.0001%	0.001%	2s

seconds as in Section 7. Results are reported in Table 12, where for each solver and instance we report the required time, in seconds, to reach an optimal solution or, if an optimal solution is not found within the limit, the percentage gap between the sub-optimal solution found and the optimal one. The results obtained show that the new solution allows to reach an optimal solution with GUROBI. In particular, employing the new solution GUROBI is able to obtain an optimal solution in all the instances in a few seconds while, starting from the original encoding, the solver is able to reach the optimal solution in 4 instances. Concerning the performances of CLINGO-ROM, even if the solver is not able to reach optimal solutions, the obtained solutions improved and have a cost that is very near to the optimal one. Indeed, the average cost has a gap to the optimal cost that is smaller than 0.01%.

8 Related Work

This paper is an extended and revised version of our CILC 2023 contribution [33]. The main additions of the current work are: (*i*) a precise, mathematical formulation of the ORS problem (Section 3); (*ii*) a new, improved ASP encoding for a basic version of the problem (Section 5.2), which performs better than the previous approach (Section 5.3) and is then employed as a base version for the other solutions presented in Section 6, dealing with real data and constraints such as bed availability and ORs usage; and (*iii*) an extended experimental analysis, which now includes an evaluation on larger planning horizons (Section 7.5) as well as of the impact of the new optimized encoding on the performance of other logic-based solving approaches (Section 7.6).

In the following, we first review works using different techniques to solve the ORS problem, with a focus on the works using real data. We then turn to works employing ASP in the healthcare domain, explicitly stating what are the few that use real data.

Solutions to the Operating Room Scheduling problem. Şeyda and Tamer [26] present a comprehensive overview of different approaches to the ORS problem. Two works proposing solutions to the ORS problem and testing them with real data are the ones by Aringhieri et al. [7] and Landa et al. [29]. In the former, the problem of scheduling surgical interventions over a one-week planning horizon is analyzed, considering several departments that share a fixed number of ORs and post-operative beds. The proposed two-phase method aims at minimizing patient waiting times and maximizing hospital resource utilization. The second work deals with two interconnected sub-problems: first patients are assigned to specific dates in a given planning horizon, while the second sub-problem concerns the assignment to ORs and the sequencing of operations in each OR. To solve the overall problem, a hybrid two-phase optimization algorithm exploits neighborhood search techniques combined with Monte Carlo simulation. Moreover, Hamid et al. [28] incorporate the Decision-Making Styles (DMS) of surgical teams to deal with constraints such as the disposition of material and resources, patient priorities and availability, as well as skills and competencies of the surgical team. Two metaheuristics to find Pareto-optimal solutions, namely, a non-dominated sorting genetic algorithm and multi-objective particle swarm optimization, are developed and evaluated on the data from a hospital in Iran. Zhang, Dridi, and El Moudni [35] address the problem of scheduling ORs with different needs for both elective and non-elective patients. A time-dependent policy is applied to determine patient priorities based on urgency levels and waiting times. This problem is formulated as a stochastic shortest-path Markov Decision Process (MDP) with blind alleys and solved using the asynchronous value iteration method. Results of a numerical experiment on synthetic data show that, compared to the regular MDP model, the proposed time-dependent policy is more efficient in reducing patient waiting times without leading to an excessive increase of the ORs usage.

Solving healthcare domain problems with ASP. ASP has been successfully used for solving hard combinatorial and scheduling problems in several application areas. In the healthcare domain (see, e.g., the recent survey by Alviano et al. [4]), ASP has already been applied for solving the ORS problem: this includes the basic problem formulation [17, 18] as well as extensions to bed management [15, 21]. Regarding other problems, the first application of ASP was the *Nurse Scheduling Problem* [5, 19, 6], where the goal is to create a schedule for nurses working in hospital units. Further applications include the *Chemotherapy Treatment Scheduling* problem [14], in which patients are assigned a chair or a bed for their treatments, and the *Rehabilitation Scheduling Problem* [12], assigning patients to operators in rehabilitation sessions. The latter two works are the only ones that have so far employed real data, representing the S. Martino Hospital, Genova, and the ICS Maugeri.[6] Recent works deal with the *Pre-Operative Assessment Clinic* problem [13], which concerns the scheduling of patients before they actually arrive to the hospital, and the *Non-transmissible Chronic Diseases Agenda* problem of assigning visit dates for multiple recurrent examinations to chronic patients [11].

[6] `https://www.ospedalesanmartino.it/` and `https://www.icsmaugeri.it/`

9 Conclusion

We have presented mathematical formulations for a basic and an extended version of the ORS problem. Our new encoding for the basic version turns out to be more efficient not only for ASP, but also for other logic-based solving approaches run on benchmarks obtained by automatic translation from the ASP formulation. This new version has been employed as a starting point for the adaptions tested on real data from the Italian health authority ASL1 Liguria. Results on some scenarios show that the ASP solutions produce satisfying schedules also when applied to such challenging, real data. Future work includes the development of a web application for the easy use of our solution, where a prototype [18] designed for the basic ORS problem can be taken as a starting point.

Acknowledgements. We would like to thank Muhammad Kamran Khan for running a preliminary version of the comparative analysis presented in Section 5.3. We are also grateful to the anonymous reviewers of CILC 2023 and this extended version for their constructive and highly valuable comments. This work was partially supported by Italian Ministry of Research (MUR) (i) under PNRR project FAIR "Future AI Research", CUP H23C22000860006, (ii) under PNRR project Tech4You "Technologies for climate change adaptation and quality of life improvement", CUP H23C22000370006, and (iii) under PRIN project PINPOINT, CUP H23C22000280006; and by Italian Ministry of Enterprises and Made in Italy (MIMIT) under project EI-TWIN "Concezione, progettazione e sviluppo di soluzioni di Energy digital TWIN" Prog n. F/310168/05/X56 - CUP: B29J24000680005 - COR: 22338845. The research of two of the authors of the paper, Giuseppe Galatà and Marco Scanu, is partially funded by the "POR FESR Liguria 2014-2020" public grant scheme. Carmine Dodaro is member of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

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., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F.: Evaluation of disjunctive programs in WASP. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) LPNMR. LNCS, vol. 11481, pp. 241–255. Springer (2019)
3. Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F.: Evaluation of disjunctive programs in WASP. In: LPNMR 2019. LNCS, vol. 11481, pp. 241–255. Springer (2019)
4. Alviano, M., Bertolucci, R., Cardellini, M., Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Mochi, M., Morozan, V., Porro, I., Schouten, M.: Answer set programming in healthcare: Extended overview. In: IPS and RCRA 2020. CEUR Workshop Proceedings, vol. 2745. CEUR-WS.org (2020), `http://ceur-ws.org/Vol-2745/paper7.pdf`
5. 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)
6. Alviano, M., Dodaro, C., Maratea, M.: Nurse (re)scheduling via answer set programming. Intelligenza Artificiale **12**(2), 109–124 (2018)
7. 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)

8. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011)
9. Buccafurri, F., Leone, N., Rullo, P.: Enhancing Disjunctive Datalog by Constraints. IEEE Transactions on Knowledge and Data Engineering **12**(5), 845–860 (2000)
10. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., Schaub, T.: ASP-Core-2 input language format. Theory and Practice of Logic Programming **20**(2), 294–309 (2020)
11. Cappanera, P., Gavanelli, M., Nonato, M., Roma, M.: Logic-based Benders decomposition in answer set programming for chronic outpatients scheduling. Theory and Practice of Logic Programming **23**(4), 848–864 (2023). https://doi.org/10.1017/S147106842300025X, `https://doi.org/10.1017/s147106842300025x`
12. Cardellini, M., Nardi, P.D., Dodaro, C., Galatà, G., Giardini, A., Maratea, M., Porro, I.: A two-phase ASP encoding for solving rehabilitation scheduling. In: Moschoyiannis, S., Peñaloza, R., Vanthienen, J., Soylu, A., Roman, D. (eds.) Proceedings of the 5th International Joint Conference on Rules and Reasoning (RuleML+RR 2021). Lecture Notes in Computer Science, vol. 12851, pp. 111–125. Springer (2021)
13. Caruso, S., Galatà, G., Maratea, M., Mochi, M., Porro, I.: Scheduling pre-operative assessment clinic with answer set programming. Journal of Logic and Computation **34**(3), 465–493 (2024)
14. Dodaro, C., Galatà, G., Grioni, A., Maratea, M., Mochi, M., Porro, I.: An ASP-based solution to the chemotherapy treatment scheduling problem. Theory and Practice of Logic Programming **21**(6), 835–851 (2021)
15. Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Porro, I.: An ASP-based solution for operating room scheduling with beds management. In: Fodor, P., Montali, M., Calvanese, D., Roman, D. (eds.) Proceedings of the Third International Joint Conference on Rules and Reasoning (RuleML+RR 2019). Lecture Notes in Computer Science, vol. 11784, pp. 67–81. Springer (2019)
16. Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Porro, I.: Solving operating room scheduling problems with surgical teams via answer set programming. In: Baldoni, M., Bandini, S. (eds.) AIxIA 2020 - Advances in Artificial Intelligence - Revised Selected Papers of the 19th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2020). Lecture Notes in Computer Science, vol. 12414, pp. 204–220. Springer (2020)
17. 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)
18. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: An ASP-based framework for operating room scheduling. Intelligenza Artificiale **13**(1), 63–77 (2019)
19. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: LPNMR. LNCS, vol. 10377, pp. 301–307. Springer (2017)
20. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1), 278–298 (2011)
21. Galatà, G., Maratea, M., Mochi, M., Morozan, V., Porro, I.: An asp-based solution to the operating room scheduling with care units. In: Benedictis, R.D., Maratea, M., Micheli, A., Scala, E., Serina, I., Vallati, M., Umbrico, A. (eds.) Proceedings of the 9th Italian workshop on Planning and Scheduling (IPS'21) and the 28th International Workshop on "Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion" (RCRA'21) co-located with AIxIA 2021. CEUR Workshop Proceedings, vol. 3065. CEUR-WS.org (2021), `http://ceur-ws.org/Vol-3065/paper8\_183.pdf`
22. 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)

23. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence **187**, 52–89 (2012)
24. 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)
25. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Comput. **9**(3/4), 365–386 (1991)
26. Şeyda Gür, Eren, T.: Application of operational research techniques in operating room scheduling problems: Literature overview. Journal of Healthcare Engineering **2018** (2018)
27. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2021), `https://www.gurobi.com`
28. Hamid, M., Nasiri, M.M., Werner, F., Sheikhahmadi, F., Zhalechian, M.: Operating room scheduling by considering the decision-making styles of surgical team members: A comprehensive approach. Computers & Operation Research **108**, 166–181 (2019)
29. 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)
30. Macario, A.: What does one minute of operating room time cost? Journal of Clinical Anesthesia **22**(4), 233–236 (2010). https://doi.org/https://doi.org/10.1016/j.jclinane.2010.02.003, `https://www.sciencedirect.com/science/article/pii/S0952818010000917`
31. Meskens, N., Duvivier, D., Hanset, A.: Multi-objective operating room scheduling considering desiderata of the surgical team. Decis. Support Syst. **55**(2), 650–659 (2013). https://doi.org/10.1016/j.dss.2012.10.019, `https://doi.org/10.1016/j.dss.2012.10.019`
32. Olivier Roussel and Vasco Manquinho: Input/Output Format and Solver Requirements for the Competitions of Pseudo-Boolean Solvers (2012)
33. Scanu, M., Mochi, M., Dodaro, C., Galatà, G., Maratea, M.: Operating room scheduling via answer set programming: The case of ASL1 Liguria. In: Dovier, A., Formisano, A. (eds.) Proc. of the 38th Italian Conference on Computational Logic (CILC 2023). CEUR Workshop Proceedings, vol. 3428. CEUR-WS.org (2023)
34. Smith, T., Evans, J., Moriel, K., Tihista, M., Bacak, C., Dunn, J., Rajani, R., Childs, B.: Cost of or time is $46.04 per minute. Journal of Orthopaedic Business **2**, 10–13 (10 2022). https://doi.org/10.55576/job.v2i4.23
35. 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)