Operating Room Scheduling via Answer Set Programming: the Case of ASL1 Liguria

Marco Scanu[1], Marco Mochi[2], Carmine Dodaro[3], Giuseppe Galatà[1] and Marco Maratea[3,*]

[1]SurgiQ srl, Genova, Italy
[2]University of Genoa, Genova, Italy
[3]University of Calabria, Rende, Italy

Abstract

The Operating Room Scheduling (ORS) problem deals with the optimization of daily operating room surgery schedule. It is a challenging problem given it is subject to many constraints, like to determine the start 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. However, due to the inherent difficulty of retrieving real data, all the analysis on ORS encodings have been performed on synthetic data so far. In this paper, instead, we deal with the real case of ASL1 Liguria, and present adaptations of the ASP encodings to include real requirements of such case. Further, we analyze the resulting encodings on these real data. Results on some scenarios show that the ASP solutions produce satisfying results also when tested on real, challenging data.

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 an important unit. As indicated in [1], the OR management account for approximately 33% of the total hospital budget because it includes high staff costs (e.g. surgeons, anesthetists, nurses) and material costs. Nowadays, in most modern hospitals, long surgical waiting lists 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.

CILC'23: 38th Italian Conference on Computational Logic, June 21–23, 2023, Udine, Italy
*Corresponding author.
marco.scanu@surgiq.com (M. Scanu); marco.mochi@edu.unige.it (M. Mochi); carmine.dodaro@unical.it (C. Dodaro); giuseppe.galata@surgiq.com (G. Galatà); marco.maratea@unical.it (M. Maratea)
https://www.marcomochi.me/ (M. Mochi); https://www.cdodaro.eu/ (C. Dodaro); http://www.star.dist.unige.it/~marco/ (M. Maratea)
0000-0002-5849-3667 (M. Mochi); 0000-0002-5617-5286 (C. Dodaro); 0000-0002-1948-4469 (G. Galatà); 0000-0002-9034-2527 (M. Maratea)

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 the other resources requested. More in details, the Operating Room Scheduling (ORS) [2, 3, 4, 1] 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) [5, 6, 7] was proposed and is used for solving such problem [8, 9], that followed other similar scheduling problems in this context: this is because ASP combines an intuitive semantics [10] with the availability of efficient solvers [11, 12].

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, instead, we deal with the real case of ASL1 Liguria for computing weekly schedules, which include data from the cities of Sanremo, Imperia, and Bordighera in the Liguria region, Italy. We present adaptations of the ASP encodings employed so far in the literature to include real requirements of such cases. We then define three scenarios: in the first scenario, our goal is to test whether our solution is able to replicate the schedule that the hospital indeed followed, while the last 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 schedule, thus indicating that ASP produces satisfying results also when tested on real, challenging data.

The paper is structured as follows. Section 2 describes the target problem in an informal way, whose ASP encoding is presented in Section 4, after having introduced syntax and semantics of ASP in Section 3. Section 5 presents the results of our experiments on the defined scenarios. The paper ends in Section 6 and 7 by discussing related work and by drawing conclusions and possible topics for further research, respectively.

2. Problem Description

This section provides the description and the requirements of the ORS problem as implemented in ASL1 Liguria, Italy, which is a local health authority consisting of three hospitals: Bordighera, Sanremo, and Imperia. The elements of the waiting list are called *registrations*. Each registration links a particular surgical procedure, with a duration, to a patient, to a specialty, and to a type of hospitalization.

The overall goal of the ORS problem is to assign the maximum number of registrations to the ORs. As the first requirement, the assignments must guarantee that the sum of the duration of surgeries assigned to a particular OR does not exceed the opening time of the OR itself. Of the three hospitals, Bordighera had two ORs available from 07:30 to 13:30, while Imperia and Sanremo had five ORs available from 07:30 to 20:00.

Moreover, registration 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. Therefore, the solution must ensure that the number of patients requiring a bed for a particular specialty is not bigger than the number of available beds for that surgery in every day. The number of available beds for the different

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

specialties of the hospitals is based on the originally available number of beds less the number of beds already occupied by the hospitalized patients. The total number of beds available for each specialty in Imperia and Sanremo are presented in Tables 1 and 2, while Bordighera has no beds availability. Finally, there are two aspects that are more specific to the data we considered for the ORS problem: the priority levels and an implied rule derived from the data. 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 unified in a unique concept: *priority*. In our settings, we introduced four different priority categories, namely P_1, P_2, P_3, and P_4. The first one gathers the patients originally assigned by the hospital: 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 P_2 over P_3 and P_3 over P_4. Regarding the implied rule, by analyzing the considered original data, we derived the limited usage of an OR. Thus, we decided to introduce a specific rule to avoid the usage of that particular OR.

3. Background on Answer Set Programming

Answer Set Programming (ASP) [5] 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 [5, 10]. 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 an 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 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:

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

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 $a_1 \mid \ldots \mid a_n$ 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* [13] 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 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* 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* [14] 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) [14].

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 new atom not appearing elsewhere in the program, meaning that the atom p can be chosen as true.

4. ASP Encoding for the Scheduling Problem

Starting from the specifications in Section 2, here we present our compact and efficient ASP solution for the scheduling problem, organized in two subsections, presenting the encoding to replicate the schedule of the hospital and the encodings to improve the schedule of the hospital, respectively. In turn, each subsection contains two paragraphs with input and output data models, and the ASP encoding, respectively. The ASP encoding is based on the input language of CLINGO [15].

4.1. Scheduling Problem

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

- The constant `time_disp` represents the opening time of the ORs.
- Instances of `registration(NOSOLOGICAL,P,SP,REG,DUR,RICOV,IN,OUT)` represent the registration of the patient identified by in ID (`NOSOLOGICAL`) with priority level (`P`), and the requested specialty (`SP`) characterized by a type of hospitalization (`REG`), the duration of the surgery (`DUR`) plus the information regarding if the patient is already hospitalized (`RICOV`), and the number of days the patient requires a bed, before (`IN`) and after (`OUT`) the surgery, respectively.
- Instances of `mss(OR,SP,DAY)` represent which specialty (`SP`) is assigned to an OR (`OR`) in a day (`DAY`).

```
1 {x(NOSOLOGICAL, P, OR, SP, DAY, DUR)} :- registration(NOSOLOGICAL, P, SP, _, DUR, _, _, _),
      mss(OR, SP, DAY).
2 :- x(NOSOLOGICAL, _, _, _, DAY1, _), x(NOSOLOGICAL, _, _, _, DAY2, _), DAY1 != DAY2.
3 :- x(NOSOLOGICAL, _,OR1, _, _, _), x(NOSOLOGICAL, _,OR2, _, _, _), OR1 != OR2.
4 :- mss(OR, _, DAY), #sum {DUR, NOSOLOGICAL : x(NOSOLOGICAL, _, OR, _, DAY, DUR)} > timeDisp.
5 stay(NOSOLOGICAL, SP, DAY - IN..DAY - 1) :- x(NOSOLOGICAL, _, _, SP, DAY, _),
      registration(NOSOLOGICAL, _, SP, "Ordinario", _, 0, IN, _), IN > 0.
6 stay(NOSOLOGICAL, SP, DAY..OUT + DAY) :- x(NOSOLOGICAL, _, _, SP, DAY, _),
      registration(NOSOLOGICAL, _, SP, "Ordinario", _, 0, _, OUT), OUT >= 0.
7 :- beds(N, SP, DAY), #count {NOSOLOGICAL : stay(NOSOLOGICAL, SP, DAY)} > N.
8 :- not x(NOSOLOGICAL, _,OR, _, DAY, _), givenSchedule(NOSOLOGICAL, DAY, OR).
```

Figure 1: ASP encoding for replicating the original schedule.

- Instances of `beds(N,SP,DAY)` represent the number (N) of available beds for a specialty (SP) in the day (DAY).
- Instances of `givenSchedule(NOSOLOGICAL,DAY,OR)` represent the original schedule of the hospital, characterized by the patient identified by an ID (NOSOLOGICAL) scheduled in a day (DAY) in an OR (OR).

The output is an assignment represented by an atom of the form `x(NOSOLOGICAL,P,OR,SP,DAY,DUR)`, where the intuitive meaning is that the patient identified by an ID (NOSOLOGICAL) having a priority (P), linked to a specialty (SP), is assigned to the OR OR in the day DAY with a surgery duration equal to (DUR).

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

Rule r_1 assigns an OR and a day to the registrations. Rules r_2 and $_3$ ensure that each registration is assigned just to one day and OR, respectively. Rule r_4 ensures that the sum of the duration of the surgeries assigned to every OR is lower than the time at disposal `time_disp`. Auxiliary atoms in the heads of rules r_5, r_6 and, r_8 are derived by the encoding to simplify the other rules. Rules r_5 and r_6 evaluate the days and the specialty on which every assigned patient requires a bed before and after the surgery, respectively. Rule r_7 ensures that the number of patients requiring a bed is less than the available ones. Rule r_8 ensures that the schedule of the hospital and the newly generated schedule are equal.

4.2. Improving the Original Schedule

Here we present the new rules for the ASP encodings that aim at improving the original schedule: the first one assigns the patients of the original schedule without taking into account the original date and ORs (OPT1), while the second encoding (OPT2) assigns new patients on top of the original schedule, and we take into account a constraint derived from the data.

Encodings. The ASP encoding OPT1 is made by the rules in Figure 2, where we denote with r_i the rule appearing at line i, plus the rules from r_1 to r_7 from Figure 1.

```
9  :- not x(NOSOLOGICAL, _,_, _, _, _), registration(NOSOLOGICAL, 1,_,_,_,_,_,_).
10 :~ not x(NOSOLOGICAL, _,_, _, _, _), registration(NOSOLOGICAL, 2,_,_,_,_,_,_).
       [1@3,NOSOLOGICAL]
11 :~ not x(NOSOLOGICAL, _,_, _, _, _), registration(NOSOLOGICAL, 3,_,_,_,_,_,_).
       [1@2,NOSOLOGICAL]
12 :~ not x(NOSOLOGICAL, _,_, _, _, _), registration(NOSOLOGICAL, 4,_,_,_,_,_,_).
       [1@1,NOSOLOGICAL]
```

Figure 2: ASP rules for dealing with priorities.

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

Figure 3: ASP rule that encodes a constraint from the real data.

Rule r_9 ensures that all the patients with priority 1 are assigned, while weak constraints r_{10}, r_{11}, and r_{12} minimize, with a decreasing optimization priority level, the number of not assigned patients with priority 2, 3, and 4, respectively, by paying a penalty of 1 for every not assigned patient with that priority level.

The ASP encoding OPT2 is composed, instead, of the rule in Figure 3, where we denote with r_i the rule appearing at line i, plus the rules from r_1 to r_8 from Figure 1 and rules from r_9 to r_{12} from Figure 2. Rule r_{13} ensures that the OR identified by the id "OR A" is used just by one patient, as in the original data.

5. Experimental Results

In this section, we report the results of an empirical analysis conducted using the defined ASP encodings, on the three scenarios previously defined. For all the settings we used original data. In particular, for the first scenario, we used data corresponding to a weekly schedule of the hospital. While, for OPT1 and OPT2, we used data corresponding to a weekly schedule of the hospital plus, in order to augment the number of patients to schedule, a random selection of patients scheduled in the future by the hospital. The experiments were run on a Apple M1 CPU @ 3.22 GHz with 8 GB of physical RAM. The ASP system used was CLINGO [15] 5.6.2, using parameters *--restart-on-model* for faster optimization and *--parallel-mode 4* for parallel execution. To let the solution be used in a real-time context, the time-limit was set to 10 seconds. Encodings and benchmarks employed in this section can be found at: http://www.star.dist.unige.it/~marco/CILC2023/material.zip .

5.1. Benchmarks

Data Description. To test our solution for scheduling surgeries, we utilized data from hospitals of ASL1 of Liguria region, Italy. The hospitals in ASL1 serve a population of around 213 thousand 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.

The initial data set contained several issues commonly found in real-world data, including incomplete information, inaccuracies, and inconsistencies between files. In order to address these issues, we performed data cleaning procedures such as correcting typos, filling in missing values by deriving data from other files, removing duplicate rows of the same patient, dropping columns with non-meaningful information, and creating new columns by merging information from different columns.

Through these cleaning procedures, we were able to obtain a more accurate and compact representation of the original data.

Tested Settings. Having presented the data, now we will present the different settings 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 solution could have improved patient's assignment and optimized resource allocation. In particular, in OPT2, the solution had to schedule the original patients in the same way done by the hospital and try to schedule as many new additional patients as possible. In OPT1, we considered both the original patients and additional patients but did not impose constraints requiring the ASP solution to replicate the hospital's schedule. 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_{13} in Figure 3). 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 introduced the concept of priority. In particular, originally scheduled patients were assigned to priority 1. The patients with priority 1 are forced to be assigned, freely in OPT1 while following the original schedule in OPT2. Then, we randomly select 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 3 to patients assigned at least 3 weeks later. The number of additional patients the solution will try

Table 3
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 4
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 %

Table 5
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 %

to schedule is linked to the original number of scheduled patients. Each setting 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. OPT1 and OPT2 enable us to assess the potential impact of our solution in terms of reducing patient waiting lists and optimizing resource allocation.

5.2. Results of 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 the implemented rules and the obtained resources availability. In Tables 3, 4, and 5 can be seen the original percentage usage of the ORs obtained by the three hospitals of Bordighera, Imperia, and Sanremo, respectively. These results represent the benchmarks to compare with in OPT1 and OPT2.

5.3. Results of OPT1

Tables 6, 7, and 8 show the results obtained in this setting in terms of the percentage of patients assigned in 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 increase the number of assigned patients

Table 6
Number of assigned patients in Bordighera grouped by their priority level.

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

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

P1	P2	P3	P4
143/143	112/120	109/130	48/108
143/143	112/120	109/130	47/108
143/143	112/120	109/130	45/108
143/143	112/120	109/130	45/108
143/143	112/120	109/130	46/108
143/143	112/120	109/130	47/108
143/143	112/120	109/130	46/108
143/143	112/120	109/130	43/108
143/143	112/120	109/130	43/108
143/143	112/120	109/130	44/108

significantly 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 9. 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 4, it can be seen a comparison between the obtained percentage usage of the ORs with the ASP 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.

Table 8
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

Table 9
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%

5.4. Results of OPT2

The results obtained in this setting are summarized for each hospital in Figure 5. In particular, from the figure, it can be noted the advantage of using the ASP solution. Indeed, while both settings assign additional patients, the difference between them lies in the usage of ASP (OPT1) to assign the originally scheduled patients. Therefore, all the additional patients assigned in OPT1 result from a more efficient schedule for the original patients. However, not all comparisons seem to suggest better results with OPT1: we will now delve into the results of each hospital in more detail. In particular, in Bordighera, the second and the third settings assign the same number of patients. This seems to suggest that the assignments done by the ASP solution didn't improve the original schedule but, as previously mentioned, in OPT2 the schedule assigns the patients as in the original schedule and in this hospital, 1 patient is assigned to an OR is not used in the OPT1, since it was used just by one patient. Therefore, even if using fewer resources, the solution of OPT1 is still able to match the performance of OPT2, meaning that the patients were scheduled more appropriately.

In Imperia, the total number of patients assigned in OPT2 is actually higher than the one assigned in OPT1 but, as can be seen in Table 10, the schedule provided by OPT1 is of higher

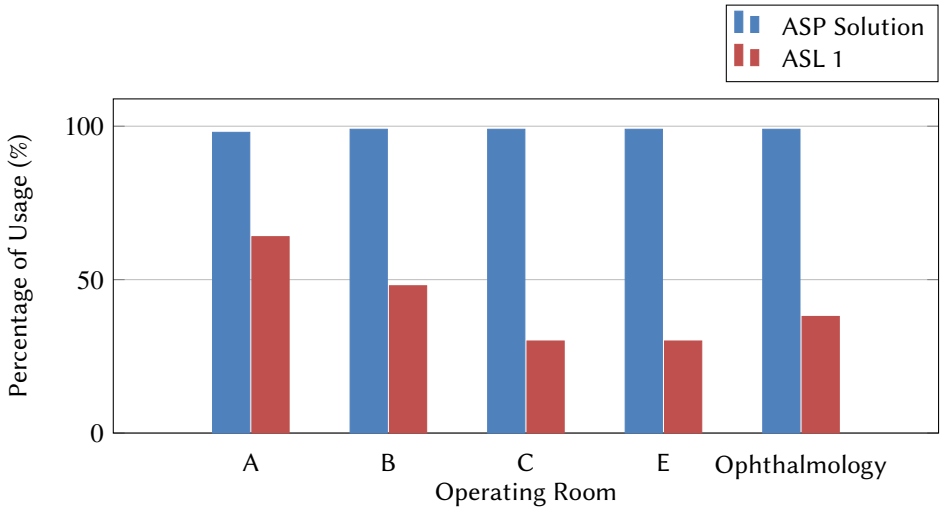

Figure 4: Comparison of the ORs usage in Imperia between the ASP solution of OPT1 and the ASL1 schedule.

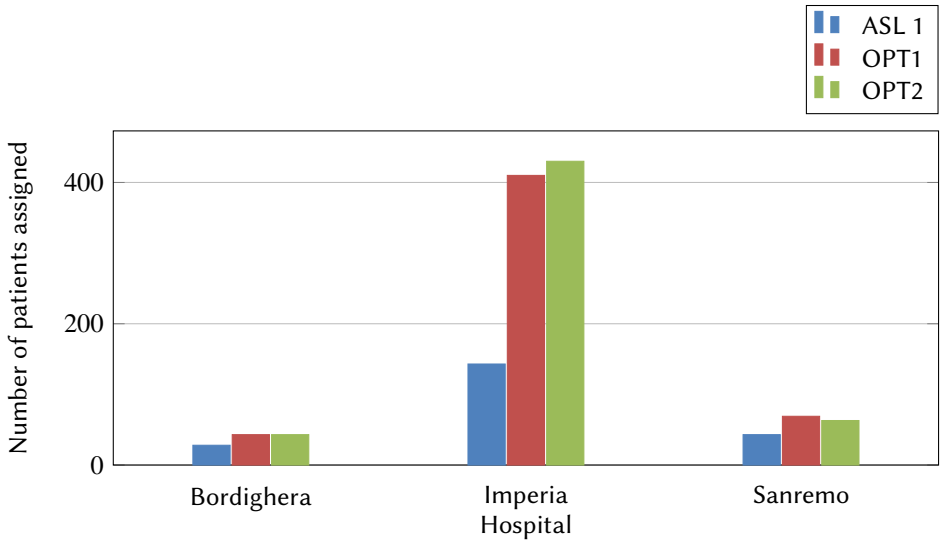

Figure 5: Comparison on the number of patients scheduled by ASL1, and by OPT1 and OPT2 solutions.

quality, since it is able to assign more patients with higher priority, even if at the detriment of patients with lower priorities. Finally, in Sanremo, the schedule done by OPT1 outperforms that of OPT2 by assigning more patients while not decreasing the quality of the solution. As such, considering all the priorities, OPT1 has assigned at least the same number of patients as OPT2 in each priority group.

Table 10
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%

6. Related Work

This section is organized into two paragraphs: The first one is focused on existing works that use different techniques to solve the ORS problem, with a focus on the works using real data. While the second presents some works in which ASP has been employed in the healthcare domain.

Solutions to the Operating Room Scheduling Problem In [16] is presented a comprehensive review of the different approaches to the ORS problem. It presents different methods to solve it and different definitions of the problem. Two studies that proposed solutions to the ORS problem by testing them with real data are [3] and [17]. In the former, the problem of scheduling surgical interventions over a one-week planning horizon was analyzed, considering several departments that share a fixed number of ORs and post-operative beds. The problem was addressed using a two-phase method with the aim of minimizing patient waiting times and maximizing hospital resource utilization. In the latter, the problem addressed consists of two interconnected sub-problems. In the first sub-problem, the decisions concerned the assignment of a date for the intervention and an operating block to a set of patients to be operated on in a given planning horizon. The second sub-problem aims to determine the sequence of selected patients daily and in each OR. To solve the entire problem, a hybrid two-phase optimization algorithm was devised that exploits the potential of neighborhood search combined with Monte Carlo simulation. Moreover, [4] incorporated the decision-making styles (DMS) of the surgical team to improve the compatibility level by considering constraints such as the availability of material resources, priorities of patients, and availability, skills, and competencies of the surgical team. They developed a multi-objective mathematical model to schedule surgeries. Two metaheuristics, namely Non-dominated Sorting Genetic Algorithm and Multi-Objective Particle Swarm Optimization, were developed to find pareto-optimal solutions. To evaluate the effectiveness of their solution they used data coming from an hospital based in Iran.

Solving Healthcare Domain Problems with ASP. ASP has been successfully used for solving hard combinatorial and application scheduling problems in several research areas. In the Healthcare domain (see, e.g., [18] for a recent survey), the first solved problem was the *Nurse Scheduling Problem* [19, 20, 21], where the goal is to create a scheduling for nurses working in hospital units. Then, the problem of assigning operating rooms to patients, denoted as *Operating Room Scheduling* [8, 9], has been treated, and further extended to include bed management [22, 23]. More recent problems include the *Chemotherapy Treatment Scheduling* problem [24], in which patients are assigned a chair or a bed for their treatments, and the *Rehabilitation Scheduling Problem* [25], which assigns patients to operators in rehabilitation sessions. In [26] and in [27]

is proposed a solution to a problem split into two phases. In the former, the problem consists to assign a date to the patients in the first phase and the time for the exams in the second phase. In the latter, the problem consists of assigning a date to visit or therapy for multiple recurrent exams to chronic patients. The problem is split into two sub-problems to increase the performance of the solution using the Benders' decomposition method.

7. Conclusion

In this paper we have adapted and tested on real data ASP encodings for the ORS problem. The evaluated scenarios show that our ASP encoding (i) is able to mimic the original schedule implemented in the hospitals of ASL1 Liguria, and (ii) could have greatly improved such schedule in terms of efficiency. Future works include the definition of an encoding for the rescheduling problem, which takes place when the original schedule can not be implemented for some reason, and the development of a web application for the easy use of our solution.

References

[1] N. Meskens, D. Duvivier, A. Hanset, Multi-objective operating room scheduling considering desiderata of the surgical team, Decis. Support Syst. 55 (2013) 650–659. URL: https://doi.org/10.1016/j.dss.2012.10.019. doi:10.1016/j.dss.2012.10.019.

[2] A. Abedini, H. Ye, W. Li, Operating room planning under surgery type and priority constraints, Procedia Manufacturing 5 (2016) 15–25.

[3] R. Aringhieri, P. Landa, P. Soriano, E. Tànfani, A. Testi, A two level metaheuristic for the operating room scheduling and assignment problem, Computers & Operations Research 54 (2015) 21–34.

[4] M. Hamid, M. M. Nasiri, F. Werner, F. Sheikhahmadi, M. Zhalechian, Operating room scheduling by considering the decision-making styles of surgical team members: A comprehensive approach, Computers & Operation Research 108 (2019) 166–181.

[5] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Communications of the ACM 54 (2011) 92–103.

[6] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: Proceedings of the Fifth International Conference and Symposium , Seattle, Washington, August 15-19, 1988 (2 Volumes), MIT Press, 1988, pp. 1070–1080.

[7] M. Gelfond, V. Lifschitz, Classical Negation in Logic Programs and Disjunctive Databases, New Generation Comput. 9 (1991) 365–386.

[8] C. Dodaro, G. Galatà, M. Maratea, I. Porro, Operating room scheduling via answer set programming, in: AI*IA, volume 11298 of *LNCS*, Springer, 2018, pp. 445–459.

[9] C. Dodaro, G. Galatà, M. Maratea, I. Porro, An ASP-based framework for operating room scheduling, Intelligenza Artificiale 13 (2019) 63–77.

[10] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, T. Schaub, ASP-Core-2 input language format, Theory and Practice of Logic Programming 20 (2020) 294–309.

[11] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, Artificial Intelligence 187 (2012) 52–89.
[12] M. Alviano, G. Amendola, C. Dodaro, N. Leone, M. Maratea, F. Ricca, Evaluation of disjunctive programs in WASP, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), LPNMR, volume 11481 of *LNCS*, Springer, 2019, pp. 241–255.
[13] F. Buccafurri, N. Leone, P. Rullo, Enhancing Disjunctive Datalog by Constraints, IEEE Transactions on Knowledge and Data Engineering 12 (2000) 845–860.
[14] W. Faber, G. Pfeifer, N. Leone, Semantics and complexity of recursive aggregates in answer set programming, Artificial Intelligence 175 (2011) 278–298.
[15] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: ICLP (Technical Communications), volume 52 of *OASICS*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 2:1–2:15.
[16] Şeyda Gür, T. Eren, Application of operational research techniques in operating room scheduling problems: Literature overview, Journal of Healthcare Engineering 2018 (2018).
[17] P. Landa, R. Aringhieri, P. Soriano, E. Tànfani, A. Testi, A hybrid optimization algorithm for surgeries scheduling, Operations Research for Health Care 8 (2016) 103–114.
[18] M. Alviano, R. Bertolucci, M. Cardellini, C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, M. Mochi, V. Morozan, I. Porro, M. Schouten, Answer set programming in healthcare: Extended overview, in: IPS and RCRA 2020, volume 2745 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2745/paper7.pdf.
[19] M. Alviano, C. Dodaro, M. Maratea, An advanced answer set programming encoding for nurse scheduling, in: AI*IA, volume 10640 of *LNCS*, Springer, 2017, pp. 468–482.
[20] C. Dodaro, M. Maratea, Nurse scheduling via answer set programming, in: LPNMR, volume 10377 of *LNCS*, Springer, 2017, pp. 301–307.
[21] M. Alviano, C. Dodaro, M. Maratea, Nurse (re)scheduling via answer set programming, Intelligenza Artificiale 12 (2018) 109–124.
[22] C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, I. Porro, An ASP-based solution for operating room scheduling with beds management, in: P. Fodor, M. Montali, D. Calvanese, D. Roman (Eds.), Proceedings of the Third International Joint Conference on Rules and Reasoning (RuleML+RR 2019), volume 11784 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 67–81.
[23] G. Galatà, M. Maratea, M. Mochi, V. Morozan, I. Porro, An asp-based solution to the operating room scheduling with care units, in: R. D. Benedictis, M. Maratea, A. Micheli, E. Scala, I. Serina, M. Vallati, A. Umbrico (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, volume 3065 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-3065/paper8_183.pdf.
[24] C. Dodaro, G. Galatà, A. Grioni, M. Maratea, M. Mochi, I. Porro, An ASP-based solution to the chemotherapy treatment scheduling problem, Theory and Practice of Logic Programming 21 (2021) 835–851.
[25] M. Cardellini, P. D. Nardi, C. Dodaro, G. Galatà, A. Giardini, M. Maratea, I. Porro, A two-phase ASP encoding for solving rehabilitation scheduling, in: S. Moschoyiannis, R. Peñaloza, J. Vanthienen, A. Soylu, D. Roman (Eds.), Proceedings of the 5th International

Joint Conference on Rules and Reasoning (RuleML+RR 2021), volume 12851 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 111–125.

[26] S. Caruso, G. Galatà, M. Maratea, M. Mochi, I. Porro, Scheduling pre-operative assessment clinic with answer set programming, Journal of Logic and Computation (2023). URL: https://doi.org/10.1093/logcom/exad017. doi:10.1093/logcom/exad017, exad017.

[27] P. Cappanera, M. Gavanelli, M. Nonato, M. Roma, Logic-based benders decomposition in answer set programming for chronic outpatients scheduling, TPLP (To Appear) abs/2305.11969 (2023). URL: https://doi.org/10.48550/arXiv.2305.11969. doi:10.48550/arXiv.2305.11969. arXiv:2305.11969.