Progress in Artificial Intelligence
https://doi.org/10.1007/s13748-024-00342-z

REGULAR PAPER

Rescheduling master surgical schedules via answer set programming

Giuseppe Galatà[1] · Marco Maratea[2] · Cinzia Marte[2] · Marco Mochi[3]

Received: 26 February 2024 / Accepted: 20 August 2024

Abstract
The problem of finding a Master Surgical Schedule (MSS) consists of scheduling different specialties to the operating rooms of a hospital clinic. To produce a proper MSS, each specialty must be assigned to some operating room. The number of assignments is different for each specialty and can vary during the considered planning horizon. Realizing a satisfying schedule is of upmost importance for a hospital clinic: recently, a compact solution based on the logic-based methodology of Answer Set Programming (ASP) to the MSS problem has been introduced and tested on synthetic data, with satisfying results. However, even more important is to be able to (i) reschedule efficiently in case a computed schedule cannot be fully implemented due to unavailability, and (ii) test the obtained solution on real data. In this paper, we design and implement a rescheduling solution based on ASP, and test both our scheduling and rescheduling solutions on real data from ASL1 Liguria in Italy. The experiments show that our ASP solutions provide satisfying results, also when tested on real data.

Keywords Healthcare · Scheduling · Answer set programming

1 Introduction

Digital Health, defined as the usage of information and communication technologies in medicine and in the management processes of healthcare, arose several years ago, but has gained increasing importance in recent years, thanks to new technologies and also due to new challenges such as an aging society, the COVID-19 pandemic and the need to reduce high costs. One of the major problems related to modern hospitals are long waiting lists that reduce patients' satisfaction and the level of care offered to them. The Master Surgical Schedule (MSS) consists of cycles of recurrent slots of surgery types, each belonging to a specialty. Its recurrent and predictable nature makes the MSS well suitable to be scheduled weeks or even months in advance. An MSS can be optimized according to various parameters, such as expected demand for each specialty, resource conflicts, bed availability and so on. The MSS specifies which specialty is assigned to each operating room in a particular day and session. The administrative practices of surgical departments on this task can have a large impact on hospital costs, patient outcomes and on the overall efficiency of a hospital. Many papers have analyzed this problem (see for example [21, 22, 31, 36, 39]); in particular, the introduction of an effective MSS led to efficiency gains at the operating room department: at Beatrix hospital, the annual budget for operating room hours is reduced from 12,848 h to 9,972 h (22.4% reduction) while the patients operated increased by 7.7% in 2007 respect to 2006, using the same capacity as at the same time surgery duration decreases by 9.0% [37]. The MSS is often considered as an already available input in many healthcare problem solutions but, due to the different aspects that need to be taken into account for computing a valid schedule and the presence of a number of works at the state of the art dealing uniquely with it, the MSS is an interesting combinatorial problem that deserves its own interest. Going in some more details, the MSS problem is the task of assigning the specialties to the available operating rooms in the different days and sessions, taking into account that not all the specialties need to be assigned the same amount of time and that,

✉ Cinzia Marte
cinzia.marte@unical.it

✉ Marco Mochi
marco.mochi@edu.unige.it

Giuseppe Galatà
giuseppe.galata@surgiq.com

Marco Maratea
marco.maratea@unical.it

[1] SurgiQ srl, Genova, Italy

[2] University of Calabria, Rende, Italy

[3] University of Genoa, Genova, Italy

Published online: 04 October 2024

during the considered days, the amount of time each specialty should be assigned can vary. The aim of the MSS is to support the hospital to organize the resources and plan the different specialties in the next weeks/months. In particular, by developing a MSS early a hospital can properly manage the personnel and the resources, thus leading to a reduction of the costs. Moreover, by helping the hospital to manage the surgeries and reducing the surgery waiting list, a proper solution to the MSS problem is vital to improve the degree of patients' satisfaction. Complex combinatorial problems, possibly involving optimizations, such as the MSS problem, are usually the target applications of logic-based knowledge representation and reasoning languages such as Answer Set Programming (ASP). Indeed ASP, thanks to its readability and the availability of efficient solvers, e.g., CLINGO [25], has been successfully employed for solving hard combinatorial problems in several research areas, and it has been also employed to solve many scheduling problems [1, 4, 10, 11, 14, 32], also in industrial contexts (see, e.g., [18, 20, 35] for detailed descriptions of ASP applications).

In this paper,[1] we first present an informal description and a precise mathematical formulation of the MSS problem. We then apply ASP to solve the MSS problem, by presenting a compact ASP encoding obtained by modularly representing input specifications in ASP, and then running an experimental analysis on randomly generated MSS benchmarks, to preliminary check the viability of our approach. Further, the scheduling solution is adapted to the real data from ASL1 Liguria in Italy.

However, it may be the case that a previously computed schedule can not be effectively implemented due to a sudden unavailability of ORs on some days, or to limitations or changes related to specialties. In this case, there is the need to reschedule the solution, minimizing the changes to the remaining part of the schedule. Thus, we have designed and implemented a rescheduling solution based on ASP for dealing with these issues, based on a mathematical formulation of the rescheduling we have defined. Finally, we have tested both our scheduling and rescheduling solutions on the real data from ASL1 Liguria in Italy: results using the state-of-the-art ASP solver CLINGO show that ASP is a suitable solving methodology for both tasks, also when dealing with real data and in comparison to alternative logic-based formalisms, like Integer Linear Programming using GUROBI.

This paper is a revised and extended version of [23], whose main additions correspond to the parts of the paper that deal with rescheduling, i.e., Sects. 5 and 6.2. Moreover, the background (Sect. 3) has been expanded, and the related work (Sect. 7) has been extended with some considerations about rescheduling. Further, it is structured as follows. Sections 2 begins with an informal description of the MSS problem, followed by its formal mathematical representation. Then, Sect. 3 provides background information on ASP syntax, semantics, and programming methodology. Section 4 shows our ASP encoding along with the results of the experimental evaluation. Section 5 delves into the rescheduling aspect of the MSS problem, presenting firstly the informal definition of the problem, then its mathematical formulation, and lastly the related encoding. Section 6 adapts and tests the solutions on real data. The paper ends by discussing related work and conclusions in Sect. 7 and 8, respectively.

2 Master surgical scheduling

In this section, we focus on the *Master Surgical Scheduling* (MSS) problem. Firstly, we provide an informal description of the problem and, then, its mathematical formulation.

2.1 MSS problem definition

With the computation of a MSS, a hospital can see on which days, sessions, and operating rooms (ORs) each specialty will do the surgeries. This is important since by looking at the MSS the hospital can manage personnel and resources in advance. To schedule the MSS a hospital should evaluate the percentage of time that needs to be assigned to each specialty. This percentage should fall within a tolerated range to allow for some flexibility in the scheduling and to better respond to the patients' needs. The percentage of assignments is evaluated as the number of times each specialty is assigned a session divided by the total number of sessions available in the period considered. To produce a proper schedule, the solution must assign the specialties taking into account the percentage targets and the allowed errors of each specialty. At most n sessions are associated with each day, where n is equal to the maximum number of sessions that could be assigned to an OR. Each session is identified by an id. For example, in a hospital with the maximum number of daily sessions equal to 2, day 1 will be linked to sessions 1 and 2, while day 2 will be linked to sessions 3 and 4, and so on for all the remaining days. Each session is then linked to the ORs and the scheduler must assign a specialty to each session. Hospitals could desire that the target assignment of each specialty vary, e.g., on a monthly or weekly basis. Another aspect that could change during the considered period and between the ORs is the sessions. Typically, each OR is allocated for two sessions, morning and afternoon, each day. However, some ORs may be allocated into a different number of sessions, either more or fewer, or even utilized for just one session. In particular, the single-session solution could be employed when a specialty requires specific resources and the time to

[1] This is an extended and revised version of a paper appearing in the CEUR proceedings of the AIxIA 2023 Workshop on Artificial Intelligence for Healthcare [23].

prepare them is long enough that changing the specialty at mid-day would be a waste of time. Moreover, some ORs could be unavailable on some days and a proper solution must be able to consider these unavailability.

Overall, the MSS problem takes as input the number of ORs and specialties, the number of days to consider for the scheduling, the number of sessions for each day, and the different target values for each specialty, and computes the assignment of the different specialties to the available ORs of a hospital in the considered planning horizon. An optimal solution minimizes the difference between the percentage of usage of each specialty and the target value of each period. An example of MSS is presented in Table 1. In particular, the table is the result obtained by our solution, which we will show later in the paper, considering 90 days and a fixed target value for each month. Moreover, we considered a hospital with 10 ORs, organized in 2 sessions for each day, and 5 specialties (these numbers correspond to hospitals of small-medium size in Italy) SP1 ... SP5: the table shows the MSS for the first 7 days of the solution. In particular, each row represents a day and the sessions linked to that day, the columns report the ORs and the intersection shows the specialty assigned to the OR in that day and session.

2.2 MSS mathematical formulation

Fix the sets D of *days*, OR of *operating rooms*, and SP of *specialties*. Let n be the maximum number of sessions assignable to an operating room in a single day. Consequently, the total number of sessions to be allocated to each operating room during the specified period is given by $m = n \cdot |D|$. Let $S = \{s_1, \ldots, s_m\}$ be the set comprising the *sessions* to be assigned. As previously explained, we may examine shorter intervals rather than the entire duration of $|D|$ days. To this aim, we partition the set D into smaller subsets δ in such a way that the elements within each subset follow a consecutive order, without any gaps or interruptions. Let Δ be the set collecting these subsets. Moreover, let:

- $\tau : SP \times OR \rightarrow \{0, 1\}$ be the function such that $\tau(x, y) = 1$ if the specialty x can be assigned to the operating room y, 0 otherwise;
- $\rho_n : OR \times D \times S \rightarrow \{0, 1\}$ be the function associating an operating room with the appropriate sessions for the given day. Specifically, $\rho_n(y, d, s_i) = 1$ if $i \in \{n \cdot d - n + 1, \ldots, n \cdot d\}$, 0 otherwise;
- $\epsilon : SP \times \Delta \rightarrow [0, 1]$ be the *target function*, indicating the desired percentage goal to attain for the schedule of a particular specialty within a specified range of days;
- $\omega : SP \times \Delta \rightarrow [0, 1]$ be the *deviation function*, defining the admissible error for the schedule of a particular specialty within a specified range of days from the desired value.

To exploit only suitable tuples, we consider the set

$$R = \rho_n^{-1}(1) = \{(y, d, s) \in OR \times D \times S \mid \rho_n(y, d, s) = 1\}$$

which collects the right assignments of sessions concerning a given day for an operating room. Referring to the example presented in Table 1 and considering only the operating room 1 on days 1 to 3, the set R encompasses the tuples (OR1,1,1), (OR1,1,2), (OR1,2,3), (OR1,2,4), (OR1,3,5), and (OR1,3,6).

Then, we define the notion of *scheduling*, which links together an OR, a day, a session, and a specialty.

Definition 1 (Scheduling) A scheduling σ is a function of the form $\sigma : R \rightarrow SP$ that associates to each OR, day, and session, a specialty.

Table 1 Example of MSS generated by our solution

Day	Session	OR1	OR2	OR3	OR4	OR5	OR6	OR7	OR8	OR9	OR10
1	1	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	2	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
2	3	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	4	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
3	5	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	6	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
4	7	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	8	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
5	9	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	10	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP2	SP1	SP4
6	11	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP2	SP1	SP4
	12	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP2	SP1	SP4
7	13	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP2	SP1	SP4
	14	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP2	SP1	SP4

The set $\Sigma = \{(y, d, s, x) \in OR \times D \times S \times SP \mid x = \sigma(y, d, s) \wedge \tau(x, y) = 1\}$ collects all the tuples eligible as scheduling.

Referring once more to the example presented in Table 1 and focusing exclusively on operating room 1 on days 1 to 3, the set Σ comprises the tuples (OR1,1,1,SP5), (OR1,1,2,SP5), (OR1,2,3,SP5), (OR1,2,4,SP5), (OR1,3,5,SP5), and (OR1,3,6,SP5).

Finally, before formally defining the MSS problem, we introduce the *temporal distribution function* of the form $\zeta : SP \times \Delta \times \Sigma \rightarrow [0, 1]$, representing the percentage of time that a specialty has been assigned within a given range of days in a scheduling.

Definition 2 (MSS Problem) The MSS problem is defined as the problem of finding a set ψ of tuples $\mathbf{t} = (x, y, d, s) \in \Sigma$ that satisfies the following conditions:

(c_1) $\forall y \in OR,\ \forall d \in D,\ \forall s \in S\ |x\ (x, y, d, s) \in \psi| = 1$;
(c_2) $\forall \delta \in \Delta,\ \forall x \in SP\ |x : (x, y, d, s) \in \psi \wedge d \in \delta \wedge (y, d, s) \in R| \geq 1$;
(c_3) $\forall x \in SP,\ \forall \delta \in \Delta\ |\epsilon(x, \delta) - \zeta(x, \delta, \psi)| \leq \omega(x, \delta)$.

The specified conditions are necessary to enforce the following constraints: (c_1) ensures that each operating room, in each day and session, is assigned to exactly one specialty; (c_2) ensures that for each range δ of days, each specialty is assigned at least once to some operating room in δ; (c_3) ensures that the difference between the target function and the temporal distribution function remains within the specified tolerance. Moreover, to assess different solutions concerning the variance between the target function and the temporal distribution function, we introduce the following definition.

Definition 3 (Dominating Solution) Given a solution ψ for the MSS problem, let

$$t_\psi = \sum_{x \in SP, \delta \in \Delta} \mid \epsilon(x, \delta) - \zeta(x, \delta, \psi) \mid$$

be the sum of the differences between the target function and the temporal distribution function. A solution ψ dominates a solution ψ' if $t_\psi \prec t_{\psi'}$.

Consequently, we define the notion of *optimal solution*.

Definition 4 *(Optimal solution)* A solution is optimal if it is not dominated by any other solution.

3 Background

Answer Set Programming (ASP) [7] is a logic-based programming paradigm developed in the field of non-monotonic reasoning. In this section, we first overview the language of ASP, by presenting syntax and semantics. Then, we describe the ASP programming methodology and outline the main differences with respect to other logic-based formalisms. About the language, more detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in [7, 9]. Hereafter, we assume the reader is familiar with basic logic programming conventions.

3.1 Syntax and semantics

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:

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

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* [8] 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.

The following example illustrates a rule comprising a #count aggregate, followed by a constraint and a weak constraint:

```
n_session(N,START,END) :- N = #count{SID,OR,DAY :
     session(SID,OR,DAY), DAY >= START, DAY < END},
     targetShare(_,_,_START,END).
:- effectiveShare(SP,PERCENTAGE,START,END), PERCENTAGE
     <= 0.
:~ effectiveShare(SP,ES,START,END),
     targetShare(SP,TS,START,END). [|ES-TS|@1,SP,START]
```

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) w.r.t. 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* [19] 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) [19].

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.

The meaning of the rules provided at the end of the previous paragraph about the ASP syntax will be provided in the next section, which presents the ASP encoding for the MSS problem.

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.

An example of a choice rule is:

```
{mss(OR,SID,SP,DAY) : operatingRoom(OR, SP)} == 1 :-
     session(SID,DAY,OR).
```

3.2 Programming methodology

Fig. 1 depicts a representation of a solution based on a logic-based declarative programming approach, as our ASP solution, consisting of five blocks as described above.

- Problem: this block represents the problem description or formulation to be modeled and solved.
- Encoding: this block involves the formal representation of the problem, using ASP in our case, based on the informal description of the problem or the precise mathematical formulation provided.
- Solver: this block takes the encoding of the problem as input and generates the answer sets.
- AnswerSet: this block represents the output of the solver and corresponds to the set of atoms that satisfy all the rules of the encoding, according to the semantics given above.
- Solution: this block is the solution of the problem, in which the answer sets are interpreted as solutions of the input problem.

The presence of a clear programming methodology is, arguably, one of the advantages that ASP offers with respect to other logic-based paradigms. Others include: (i) The ASP high-level specifications are declarative and often appreciated even by non-experts since they found them readable, differently from the specifications employed by the other paradigms, e.g., SAT and CP. (ii) There are free and open source systems (like the mentioned CLINGO, or WASP [2]), whose performances are often comparable to the ones of industrial tools for ILP like, e.g. CPLEX, or to GUROBI, or SAT solvers (as shown in our experiments). (iii) ASP allows

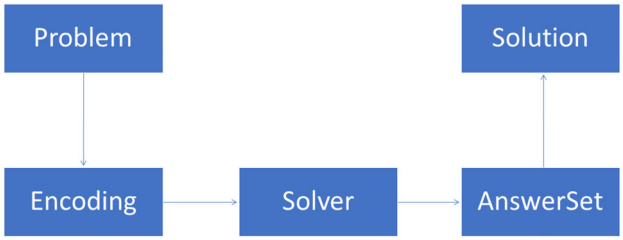

Fig. 1 Programming methodology schema

for easily expressing and reasoning on multi-objective and multi-level optimizations, which is not the case for, e.g., optimization variants of SAT such as Max-SAT (unless weights having exponential gaps are applied).

On the other hand: (*a*) Being a declarative approach, there is less control on the solving, which is delegated to an ASP solver. (*b*) Some CP constraints, sush as *alldifferent*, that may be useful in applications, are not part of the language, and can not be expressed in a compact way; ILP allows for expressing quadratic, non-linear, functions in the optimization statement, which are not part of the ASP standard and ASP solver can not solve (at least in a direct way).

4 ASP solution for the MSS problem

Starting from the specifications in the previous section, here we present our compact and efficient ASP solution for the MSS problem, and the results of an experimental analysis performed on randomly generated benchmarks. The ASP encoding is based on the input language of CLINGO [24].

Data model. The input data is specified by means of the following atoms in relation to the mathematical formulation:

- instances of `day(D)` represent the set D of available days;
- the constant `max_session` corresponds to the constant n and represents the maximum number of sessions that can be assigned to the ORs in a day;
- the constant `s_count` represents the maximum number of sessions that can be assigned to each operating room through all days. It is evaluated by multiplying the constant `max_session` with the constant `d_count`, that represents the number of days considered in the scheduling and corresponds to the constant m;
- instances of `operatingRoom(OR,SP)` represent which specialty `SP` can be assigned to the operating room identified by an id `OR` and represents pairs for which the function τ assumes value 1;
- instances of `specialty(SP)` represent the different specialties identified by their id `SP` and corresponds to the set S;
- instances of `targetShare(SP, TARGET,ERROR, START,END)` represent for each specialty `SP` the target percentage `TARGET` of utilization and the maximum distance allowed to the target value `ERROR` in the range of days between `START` and `END` and models together information described by the functions ϵ and ω;
- instances of `sessionN(OR,N,D)` represent the number of sessions `N` in which the operating room identified by an id `OR` is split in the day `D`;

The output, is an assignment represented by atoms of the form `mss(OR,SID,SP,D)`, where the intuitive meaning is that the operating room with id `OR` in the session with id `SID` and in the day `D` is assigned the specialty `SP` and corresponds to the tuples contained in the set ψ.

Encoding. The related encoding is shown in Fig. 2, and is described next. To simplify the description, we denote as r_i the rule appearing at line i of Fig. 2.

Auxiliary atoms in the heads of rules r_1, r_2 and, r_4 are derived by the encoder to simplify the other rules. In particular, rule r_1 assigns the correct session ids to each operating room for all the days considered. The assignment is made assigning an id such that the number of ids assigned in each active day is equal to the number of sessions in which the operating room is split. Rule r_2 evaluates the total number of sessions available in the range of days between start and end. This value is then used to evaluate the percentage of assignment of each specialty. Rule r_3 assigns one of the possible specialties to a session of every operating room, satisfying condition c_1. Rule r_4 derives an atom that represents the assignment percentage of each specialty. In particular, it counts the number of sessions linked to each specialty and divides it by the total number of sessions that are available in that period. Then, rule r_5 ensures that the percentage of each specialty is bigger than 0, encoding condition c_2. Rules r_6 and r_7 check that the percentage of each specialty is compatible with the target values and the allowed errors, encoding condition c_3. Finally, weak constraint r_8 minimizes the difference between the assigned and target percentage of each specialty in each period of time.

Benchmarks and experiments. Here we report the results of an empirical analysis of the MSS problem via ASP performed on synthetic benchmarks, in which data have been randomly generated using parameters inspired by literature and real world data. The experiments were run on a AMD Ryzen 5 2600 CPU @ 3.40GHz with 16 GB of physical RAM. The ASP system used was CLINGO [24] 5.4.0, employing parameters *--opt-strategy=usc* for faster optimization and *--parallel-mode 6* for parallel execution. This setting is the result of a preliminary analysis done also with other parameters, i.e., the default configuration and the one having

```
session(SID,DAY,OR) :- operatingRoom(OR,_), sessionN(OR,N,DAY), SID=1..s_count, SID >=
    ((max_session*DAY)-(max_session-1)), SID<=((max_session*DAY)-(max_session-N)), not
    inactive(OR,DAY).
n_session(N,START,END) :- N = #count{SID,OR,DAY : session(SID,OR,DAY), DAY >= START, DAY <
    END}, targetShare(_,_,_START,END).
{mss(OR,SID,SP,DAY) : operatingRoom(OR, SP)} == 1 :- session(SID,DAY,OR).
effectiveShare(SP,PERCENTAGE,START,END) :- SESSION = #count{ OR,SID,DAY : mss(OR,SID,SP,DAY), D
    >= START, D < END}, n_session(N,START,END), specialty(SP), PERCENTAGE = ((SESSION*100) / N).
:- effectiveShare(SP,PERCENTAGE,START,END), PERCENTAGE <= 0.
:- effectiveShare(SP,PERCENTAGE,START,END), targetShare(SP,TARGET,ERROR,START,END), PERCENTAGE
    < (TARGET-ERROR).
:- effectiveShare(SP,PERCENTAGE,START,END), targetShare(SP,TARGET,ERROR,START,END), PERCENTAGE
    > (TARGET+ERROR).
:~ effectiveShare(SP,ES,START,END), targetShare(SP,TS,ERR,START,END). [|ES-TS|@1,SP,START]
```

Fig. 2 ASP encoding of the MSS problem

--restart-on-model for optimization. The time limit was set to 30 s.

Data are based on the sizes and parameters of a typical middle sized hospital, with 5 different specialties and 10 ORs. Each specialty is associated with a target value for each month and an error, that is equal to 10 for all the specialties. Each specialty can be assigned to just some randomly selected ORs and the target value is assigned by dividing the number of ORs in which the specialty can be assigned to the total number of ORs, and adding to the result a random value in the range between -5 and 5. To test our solution we considered four different scenarios. In the first scenario, that we call Scenario A, we considered to have the constant $max\_session$ equal to 2, while the constant d_count has values from 30 to 180. Moreover, in this scenario the target value for each specialty is equal for each month. For this scenario, we considered 10 instances, each with different target values for all the specialties, for each range of days considered. In particular, we tested the scalability of the scheduler by considering an increasing number of days: 30, 60, 90, 120, 150 and, 180. Then, we generated a second scenario, that we call Scenario B, that is based on the Scenario A considering 90 days. The difference with Scenario A is that for each month the target value is increased or decreased by a random value between -2 and 2, thus for each specialty there are three different target values. Changes in the target values could be done by the hospital manager because of different availability of doctors or due to the increase of the surgeries of some specialty. For the third and fourth scenario, named Scenario C and D, respectively, we again considered a planning horizon fixed to 90 days. The constant $max\_session$ is equal to 2 for the Scenario C, while for the Scenario D is equal to 3. This means that, in the fourth scenario, one randomly selected operating room is splitted in three sessions. The difference between the Scenario C and the others is that, for 5 days, three ORs are unavailable, meaning that no session can be assigned to them during that days. The scenarios C and D aim thus at evaluating what is the impact of limiting the usage of the ORs, or changing the number of sessions, respectively.

For the basic scenario (Scenario A) it turned out that the scheduler is able to optimally schedule the MSS in a mean time of less than 10 s even considering 180 days of planning horizon, which is a remarkable result. Moreover, besides being able to reach an optimal solution in less than 10 s on average, the scheduler is able to always find the optimal solution in less than 30 s. About Scenario B, we found that the scheduler was able to reach the optimal solution on average in 3 s, that is a time that is very close to the time required in Scenario A. Thus, this analysis reveals that even changing the target values in each month for all the specialties, our solution maintains very good performance. Results of Scenario C is almost equal to the original one. So, even if three ORs are unavailable for 5 days, the scheduler is able to compute the optimal solution in the same time required by Scenario A. Finally, in the Scenario D the scheduler obtained the optimal solution almost in a similar time as in the Scenario A for all but one instance: indeed, the third instance requires 4 s instead of 2 s to reach the optimal solution.

A more detailed account of the results for our MSS solution on random benchmarks can be found in Sect. 5 of [23] and in Sect. 6 of [29].

5 Rescheduling

In this section, we focus on the *Rescheduling Master Surgical Schedule* (RMSS) problem. We start by providing both an informal and a formal description of the problem. Following that, we present an ASP encoding that relates to this formulation.

5.1 RMSS problem description

In the context of Digital Health, effectively addressing unforeseen challenges and adapting to changes is of vital importance. In particular, when dealing with scheduling

Table 2 Example of RMSS generated by our solution. The specialties in italic represent specialties that were not previously assigned in that session, OR, and day. The specialties in bold represent specialties that were previously assigned in ORs that become unavailable in those sessions

Day	Session	OR1	OR2	OR3	OR4	OR5	OR6	OR7	OR8	OR9	OR10
1	1	**SP5**	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	2	**SP5**	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
2	3	SP5	**SP5**	*SP5*	*SP5*	SP2	SP5	*SP5*	*SP5*	SP1	SP4
	4	SP5	**SP5**	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
3	5	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	6	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
4	7	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	8	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
5	9	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP3	SP1	SP4
	10	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP2	SP1	SP4
6	11	SP5	SP5	SP3	SP3	SP2	**SP5**	SP4	SP2	SP1	SP4
	12	SP5	SP5	SP3	SP3	SP2	**SP5**	SP4	SP2	SP1	SP4
7	13	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP2	SP1	SP4
	14	SP5	SP5	SP3	SP3	SP2	SP5	SP4	SP2	SP1	SP4

problems, this means that a previously valid solution may become impossible to follow. Usually, the changes that invalidates a previously scheduled MSS are those that modify the structure and organization of the hospital for a significant period of time: e.g. the decision to open or close a wing of the hospital, the decision to acquire or dismiss specialized equipment for some OR, the increase or decrease of the expected surgical procedures for a given specialty, long term damages to an OR and so on. Consequently, implementing rescheduling solutions becomes essential to provide solutions that can react to these changes rapidly while maintaining the original solution as much as possible.

In the context of the MSS problem we focus on following three possible scenarios:

1. Some ORs are no longer available on some days;
2. Some specialty significantly changes the required target value;
3. Some specialties can be assigned to a limited number of days in a certain range.

Thus, the RMSS problem consists of starting from a previously valid solution (meaning all the associations between specialties and operating rooms and sessions) and reassigning these associations according to the new requirements that emerged, while still respecting all the unchanged constraints. As in the MSS problem, the optimization criteria of highest importance are the ones that allow generating a scheduling of high quality (i.e. reducing the changes of specialty for the same OR on the same day). However, with lower importance, two additional optimization criteria are introduced for the rescheduling problem:

(i) The first optimization aims to minimize the difference in terms of the temporal distribution of the assignment of all the specialties between the scheduling of the basic problem and the rescheduling version, respectively;
(ii) The second optimization aims to minimize the changes between a solution for the MSS and the RMSSproblems.

We point out that (i) is set as the second highest criteria for rescheduling, because changing significantly the percentage of time assigned to a specialty carries a heavy burden on the organizational level, i.e. requiring more nurses and doctors than expected. Whereas, (ii) requires that the solution for the RMSSproblem should try to assign the specialty to the same sessions and ORs when possible with respect to the starting solution of the MSS.

To illustrate an example of the RMSS problem, let's consider as initial MSS the one provided in Table 1. We will illustrate a solution to the rescheduling problem for Scenario 1. In this scenario, let's assume that due to maintenance works, OR1, OR2, and OR6 are unavailable on days 1, 2, and 3, respectively. Consequently, there is a decrease in the total assignment of specialty SP5. Initially, SP5 was assigned to 30% of the total sessions, but with the unavailability, it would now be assigned to 26% of the sessions. Moreover, other specialties have seen an increase in their percentage of assignments. Specialty SP3 was previously assigned to 25% of the sessions while, due to the unavailability of the ORs, its total assignments went up to 28%. A proper solution would aim to increase the percentage of sessions to which SP5 is assigned while maintaining similarity to the original MSS. In Table 2 is presented a possible solution to this problem. As shown in the table, to compensate for the unavailability, specialty SP5 has been assigned to 4 new sessions on the second day. This adjustment brings the total percentage of sessions

assigned to SP5 back to 30%, as required. Moreover, having been replaced by SP5 in 3 sessions, SP3's new total assignment in the rescheduling solution returned to 25%.

5.2 RMSS mathematical formulation

As previously discussed, the rescheduling derives from specific circumstances that make some input tuples unusable. Accordingly, to identify these tuples, we introduce the *unusable* function of the form $\upsilon : OR \times D \rightarrow \{0, 1\}$, such that $\upsilon(y, d) = 1$ if the operating room y is no longer available on day d, 0 otherwise. Consequently, in the rescheduling problem, we replace the set R with the set $\tilde{R}$ defined as:

$$\tilde{R} = \{(y, d, s) \in OR \times D \times S \mid (y, d, s) \in R \wedge \upsilon(y, d) = 0\}.$$

Referring to the example presented in Table 2, and specifically focusing on day 1 and operating rooms 1 to 3, the set $\tilde{R}$ includes the tuples (OR2,1,1), (OR2,1,2), (OR3,1,1), and (OR3,1,2). Moreover, from now on, we refer to the target function and the deviation function as $\tilde{\epsilon}$ and $\tilde{\omega}$, respectively, highlighting that these functions may assume different values with respect to the original scheduling.

We are now able to define a *rescheduling*.

Definition 5 *(Rescheduling)* A rescheduling $\tilde{\sigma}$ is a function of the form $\tilde{\sigma} : \tilde{R} \rightarrow SP$ that associates to each OR, day, and session, a specialty.

Accordingly, the set $\tilde{\Sigma} = \{(y, d, s, x) \in OR \times D \times S \times SP \mid x = \tilde{\sigma}(y, d, s) \wedge \tau(x, y) = 1\}$ collects all the tuples eligible for the rescheduling. Referring again to the example presented in Table 2, and specifically focusing on day 1 and operating rooms 1 to 3, the set $\tilde{\Sigma}$ includes the tuples (OR2,1,1,SP5), (OR2,1,2,SP5), (OR3,1,1,SP3), and (OR3,1,2,SP3).

Another scenario behind the rescheduling problem is the limitation on the number of days within which a specialty x can be assigned to an operating room. To tackle this scenario, we introduce the *limitation function* of the form $\lambda : SP \times \Delta \rightarrow \mathbb{N}_0$ such that

$$\begin{cases} \lambda(x, \delta) = n \in \mathbb{N} & \text{if } x \text{ is limited} \\ \lambda(x, \delta) = 0 & \text{otherwise.} \end{cases}$$

Now we can formally define the rescheduling problem.

Definition 6 *(RMSS problem)* The Rescheduling Master Surgical Schedules (RMSS) problem is defined as the problem of finding a set $\tilde{\psi}$ of tuples $\mathbf{t} = (x, y, d, s) \in \tilde{\Sigma}$ that satisfies conditions (c_1), (c_2) and (c_3) of Sect. 2, evaluated over $\tilde{\psi}, \tilde{\epsilon}, \tilde{\omega}$, and condition

(c_4) $\forall x \in SP, \forall \delta \in \Delta$ if $\lambda(x, \delta) > 0$ it holds that $|x : (x, y, d, s) \in \tilde{\psi} \wedge d \in \delta| < \lambda(x, \delta)$.

The latter condition is required to ensure that for the specialties that are limited in the rescheduling problem, the bound is satisfied.

We conclude this section by examining the two additional optimization criteria introduced in Sect. 5.1. To this end, we introduce the following elements essential for analyzing a solution.

Definition 7 Let ψ be a solution for the MSS problem and $\tilde{\psi}$ a solution for the RMSSproblem. We define

(1) $\eta_{\psi,\tilde{\psi}} = \sum_{x \in SP, \delta \in \Delta} | \zeta(x, \delta, \psi) - \zeta(x, \delta, \tilde{\psi}) |$, and

(2) $c_{\psi,\tilde{\psi}} = |(x, y, d, s) \in \psi \wedge (x, y, d, s) \in \tilde{\psi}|$.

With (1) we assess the difference in terms of temporal distribution function between a solution ψ and $\tilde{\psi}$ for the MSS and RMSSproblems, respectively; with (2), we evaluate the changes between the original solution of the MSS problem and the new one.

Finally, we define the notion of dominating solution for the RMSSproblem.

Definition 8 Let ψ represent a solution of the MSS problem and $\tilde{\psi}_1$ and $\tilde{\psi}_2$ denote two solutions of the RMSSproblem. We say that $\tilde{\psi}_1$ dominates $\tilde{\psi}_2$ if

- $t_{\tilde{\psi}_1} < t_{\tilde{\psi}_2}$, or if
- $t_{\tilde{\psi}_1} = t_{\tilde{\psi}_2} \Rightarrow \eta_{\psi,\tilde{\psi}_1} < \eta_{\psi,\tilde{\psi}_2}$, or if
- $t_{\tilde{\psi}_1} = t_{\tilde{\psi}_2}$ and $\eta_{\psi,\tilde{\psi}_1} = \eta_{\psi,\tilde{\psi}_2} \Rightarrow c_{\psi,\tilde{\psi}_1} > c_{\psi,\tilde{\psi}_2}$.

Definition 9 A solution $\tilde{\psi}$ is optimal if it is not dominated by any other rescheduling solution.

5.3 ASP encoding for the RMSS problem

We now present the encoding of the RMSSproblem.

Data model. The input data is specified by means of atoms and constants outlined in Sect. 4, supplemented by the following atoms:

- instances of `unusable(OR,D)` represent an operating room `OR` that is unavailable on day `D` and model the function υ;
- instances of `limit(SP,START,END,N)` represent the limitation `N` concerning a specialty `SP` within the timeframe from `START` to `END` and model the function λ;
- instances of `newTARGETshare`, operating similarly to the atom `targetShare` (introduced in Sect. 4), represent the potential new values that the target function may assume in the rescheduling and model the information described by the functions $\tilde{\epsilon}$ and $\tilde{\omega}$;

```
1  newsession(SID,DAY,OR) :- operatingRoom(OR,_), sessionN(OR,N,DAY),SID = 1..s_count, SID >=
       ((max_session*DAY)-(max_session-1)), SID <= ((max_session*DAY)-(max_session-N)), not
       inactive(OR,DAY), not unusable(OR,DAY).
2  n_newsession(N,START,END) :- N = #count{SID,OR,DAY : newsession(SID,DAY,OR), DAY>=START,
       DAY<END}, newTARGETShare(_, _, _,START, END).
3  rmss(OR,SID,SP,DAY) :- mss(OR,SID,SP,DAY), not unusable(OR,DAY), unusable(_,_).
4  {rmss(OR, SID, SP, DAY) : operatingRoom(OR, SP)} == 1 :- newsession(SID, DAY, OR).
5  newEFFshare(SP, PERCENTAGE, START, END) :- SESSION = #count{ OR,SID,DAY : rmss(OR,SID,SP,DAY),
       DAY>=START, DAY<END}, n_newsession(N, START, END), specialty(SP), PERCENTAGE =
       ((SESSION*100) / N).
6  :- newEFFshare(SP, PERCENTAGE, _,_), PERCENTAGE = 0.
7  :- newEFFshare(SP, PERCENTAGE, START, END), newTARGETshare(SP,TARGET,ERROR,START,END),
       PERCENTAGE < (TARGET-ERROR).
8  :- newEFFshare(SP, PERCENTAGE, START,END), newTARGETshare(SP,TARGET,ERROR,START,END),
       PERCENTAGE > (TARGET+ERROR).
9  :- #count{SP,OR,DAY: rmss(OR,SID,SP,DAY), DAY>=START, DAY<=END} >= N, limit(SP,START,END,N).
10 :~ newEFFshare(SP, ES, START, END), newTARGETshare(SP,TS,ERR,START,END). [|ES-TS|@3, SP, START]
11 :~ effectiveShare(SP, ES, START, END), newEFFshare(SP, NES, START, END). [|ES-NES|@2, SP,
       START]
12 :~ mss(OR, SID, SP, DAY), not rmss(OR, SID, SP, DAY), not unusable(OR,DAY). [1@1,OR,SID]
```

Fig. 3 ASP encoding of the RMSS problem

- instances of `mss(OR, SID, SP, D)` represent the solution of the old planning, including the operating room `OR`, session `SID`, specialty `SP`, and day `D`.

The output is the rescheduling, consisting of atoms of the form `rmss(OR, SID, SP, D)`, representing an operating room `OR` linked to a session `SID` for a specialty `SP` on a day `D` and corresponds to the tuples contained in the set $\tilde{\psi}$.

Encoding. The ASP encoding is shown in Fig. 3. To simplify the description, we denote as r_i the rule appearing at line i of Fig. 3. Essentially, the new encoding mirrors the one for the scheduling, in addition to operating rooms no longer available represented by the predicate `unusable`, alongside the new criteria introduced to attain the optimal solution. Therefore, rules r_1, r_2, and r_5 behave, respectively, as rules r_1, r_2, and r_4 described in Sect. 4. Rule r_3 aims at maintaining the same schedules for all the days that have not an unusable operating room. Rule r_4 assigns one of the possible specialties to a session of every operating room, satisfying condition c_1. Rules r_6 encodes condition c_2, rules r_7 and r_8 encode condition c_3 of the RMSS problem, whereas rule r_9 models condition c_4. Finally, the optimal solution is achieved through the application of rules r_{10}, r_{11}, and r_{12}, prioritized in decreasing order. Specifically, r_{10} focuses on minimizing the difference between the temporal distribution and target function of each specialty in each period of time. Following that, with lower priority, rule r_{11} aims to minimize the difference in terms of temporal distribution function between the old and the new solution. Lastly, rule r_{12} minimizes the changes between the old and the new solution.

6 Adaptation to and results on real data

In this section, we present the results obtained by using real data and the modification done to use the MSS solution with the new data. The section is split into two parts: in the first one, we present the results obtained for the MSS problem, following two directions, i.e., trying to replicate the same MSS used by the real hospital, and trying to increase the quality of the MSS, that needs adaptations to the encoding. Producing an MSS that tries to assign the same specialty in all the sessions of an OR in a day is crucial for reducing the costs of a hospital since it allows for a reduction in the work needed to prepare the ORs for different specialties. Moreover, keeping the specialties to the same percentage of assignments, if possible, eases the organization of the personnel turnover. The second part, instead, presents the results of the RMSS solution on the scenarios mentioned in Sect. 5.1.

6.1 Results on MSS

Replicate the results with real data. After having tested our solution in different scenarios with synthetic data in Sect. 4, we wanted to test it with real data. To accomplish this, we used data from ASL1 Liguria, Italy, already used in [34] in the context of another scheduling problem. ASL1 is a local health authority consisting of three hospitals: Bordighera, Sanremo, and Imperia. Each hospital has between 2 and 5 ORs and the number of patients visited in a typical month ranges between 100 patients of Bordighera to 500 patients of Imperia. We have made the decision to proceed with the test

```
1  specialty("IMPERIA CARDIOLOGIA").
2  operatingRoom("SALA D","IMPERIA
       CARDIOLOGIA").
3  operatingRoom("SALA EP","IMPERIA
       CARDIOLOGIA").
4  specialty("IMPERIA CHIRURGIA GENERALE").
5  operatingRoom("SALA A","IMPERIA CHIRURGIA
       GENERALE").
6  operatingRoom("SALA B","IMPERIA CHIRURGIA
       GENERALE").
7  operatingRoom("SALA C","IMPERIA CHIRURGIA
       GENERALE").
8  operatingRoom("SALA D","IMPERIA CHIRURGIA
       GENERALE").
9  operatingRoom("SALA E","IMPERIA CHIRURGIA
       GENERALE").
10 specialty("IMPERIA CHIRURGIA GENERALE DH
       SURGERY").
11 operatingRoom("SALA A","IMPERIA CHIRURGIA
       GENERALE DH SURGERY").
12 ...
```

Fig. 4 Example of input derived from the data for Imperia hospital

using data from the hospital of Imperia since it is the hospital with more ORs, specialties, and patients. In particular, the considered data from the hospital of Imperia consists of the list of surgeries of the patients for the months of March, April, and May, 2019. The Imperia hospital makes use of 9 different ORs and there are 8 different specialties.

In this case, we wanted to assign the ORs to the different specialties of the hospital with the same percentages of assignments. In order to complete these tests, we derived the number of ORs, number of specialties, possible assignments of specialties to ORs, and percentage of assignments of each specialty from the real data. Then, we used this information as input, of which an excerpt obtained from the Imperia hospital can be seen in Fig. 4. We tested the solution considering 90 days in a different, real setting where, differently from the synthetic data, e.g., the number of ORs is larger than the number of specialties. The solution was able to find an optimal solution in less than a second. This means that the solution managed to assign the different ORs as requested by the hospital very rapidly and, moreover, the error was equal to 0 percentage points. This allows us to confirm the goodness of our solution, that is able to replicate what happened in the hospital.

Additions to consider real data. In the following, we present an addition to the already presented solution that we did to increase the quality of the MSS produced by the hospital and that allows us to derive the percentage of assignments to assign to each specialty. Indeed, in the rest of the work, we consider the target of assignment of the specialties as an input. Here, we want to consider the case in which a hospital wants to derive the assignments considering the number of expected patients and caring about the quality of the MSS. To produce a better MSS, we want to obtain an MSS in which the ORs are assigned, as much as possible, to the same specialty for all the sessions in a day, every week the specialties are assigned to the same day, and the percentage of time assigned to each specialty every month does not change. Indeed, analyzing the real data, we found that many ORs were assigned to different specialties on consecutive days and, even if not needed, looking at the number of patients, the percentage of time assigned to the different specialties varies a lot during the months.

Starting from the ASL1 data, we derived the number of patients assigned each month for every specialty, considering it as a prediction of future needs, and used this information as a new input for a modified version of the encoder in Fig. 2.

Data model. The input data is the same as the one presented in Sect. 4 but for the atom `targetShare(SP,TARGET, ERROR,START,END)` that is not used and is replaced by an atom `needed(SP, EXPECTED, START, END)` that represents for each specialty `SP` the expected percentage (`EXPECTED`) of utilization needed for patients of that specialty in the range of days between `START` and `END`. The output is the same as in Sect. 4.

Encoding. The related encoding is the same as presented in Fig. 2 but without the rules appearing in lines 5, 6, and, 7 and the weak constraint in line 8, plus the rules in Fig. 5. In the following, we describe such additional rules.

In particular, rule r_9 ensures that the percentage of sessions assigned to each specialty is more than the expected percentage. Rule r_{10} is used to derive the difference between the percentage of sessions assigned to a specialty and the expected percentage for the specialty in which the difference is bigger than a value equal to 10. Weak constraint r_{11} minimizes the assignments of two different specialties to the same OR and on the same day in different sessions. Weak constraint r_{12} minimizes the assignments of two different specialties to the same OR in a one-week time gap. Weak constraint r_{13} minimizes the difference in the percentage of sessions assigned to each specialty each month. Finally, weak constraint r_{14} minimizes the value obtained with rule r_{10}, thus, it reduces the unnecessary assignments.

Testing the solution with the new encoding with the data of the Imperia hospital, we obtained an optimal solution in 5.8 s. Thus, the solution is able to obtain a MSS in which for every day, each OR has just one specialty assigned, and every week the specialties are assigned to the same day. Moreover, the difference in the percentage of sessions assigned to each specialty every month is 0, while keeping the value higher than the expected need.

Upon testing the new solution with real data, we can affirm its viability as a valid option for generating a MSS, especially when the hospital prioritizes enhancing the overall quality of the MSS over achieving specific assignments for each specialty.

```
:- effectiveShare(SP, PERCENTAGE, START, END), needed(SP,TARGET,START,END), PERCENTAGE < TARGET.
slack(SP, PERCENTAGE-TARGET, START) :- effectiveShare(SP, PERCENTAGE, START, _), needed(SP,
    TARGET, START, _), PERCENTAGE > TARGET + 10.
:~ mss(OR, SID1, SP1, DAY), mss(OR, SID2, SP2, DAY), SID2 > SID1, SP1 != SP2. [1@4, DAY, OR]
:~ mss(OR, _, SP, DAY),not mss(OR, _, SP, DAY+7), session( _, DAY+7, OR). [1@3, OR, DAY]
:~ effectiveShare(SP, P1, START, _), effectiveShare(SP, P2, START+30, _). [|P1-P2|@2,SP, START]
:~ slack(SP, SL, ST). [SL@1, SL, ST]
```

Fig. 5 Added rules to the MSS encoding to deal with real data

Table 3 Sum of the differences between the new target values and the assigned temporal distribution values of every specialty (Target), expressed in percentage points, and the difference, expressed in percentage, between the original MSS and the new solution (Change) in the rescheduled instances in the three scenarios

Instance	Scenario 1		Scenario 2		Scenario 3	
	Target	Change	Target	Change	Target	Change
1	2	1%	0	4%	0	1%
2	2	1%	0	1%	0	1%
3	3	1%	1	4%	0	1%
4	3	1%	2	9%	0	1%
5	3	3%	3	7%	0	1%
6	3	2%	2	10%	0	1%
7	3	6%	6	14%	0	1%
8	3	6%	4	11%	0	1%

6.2 Results on RMSS

Benchmarks. As for the MSS problem, the ASP system used was CLINGO [24], ver. 5.4.0, using *--parallel-mode 6* for parallel execution. The time limit was set to 10 s.

The rescheduling problem starts from the result of a scheduling problem with some additional information. Thus, we started from the results obtained using the encoding presented in Fig. 2 and the real data of the Imperia hospital presented in the previous subsection. Depending on the scenario, we added the unavailability of an OR for some days, new target values for the specialty, or a limited number of days that some specialty can be assigned to, as presented in Sect. 5.1. To test the scalability of our solution, for each scenario, we tested different settings requiring more changes to the original solution. We tested 8 instances for each scenario, increasing the unavailability according to the scenario. In particular, in each instance, we changed the unavailability of the ORs, the difference of the required target value, and the limit of usage of the different specialties, respectively. In particular, in Scenario 1, in every instance, there are two more unusable ORs. In Scenario 2, for each instance, a modified target value for a specialty is added, and finally, in Scenario 3, every instance increases the limit of usage of a specialty or adds a limit to another specialty.

Results. We present the results obtained from testing our solution to the RMSS problem, analyzing each scenario defined in Sect. 5.1 individually. A summary of these results is provided in Table 3, where each row denotes an instance and each column represents a distinct scenario. Moreover, within each column, the analysis is further subdivided into two subcolumns: *Target*, denoting the total sum of the differences, in terms of percentage points, between the new target value and the assigned temporal distribution value, evaluated for each specialty, and *Change*, expressing the difference in percentage between the original scheduling and the rescheduling.

First scenario: some ORs are no longer available. In this scenario, we consider that an OR can no longer be assigned on some days as originally scheduled. This affects the entire schedule due to the impact on the total time assigned to the different specialties.

In all the instances tested, as can be seen in Table 3, the total sum of the differences between the target values and the temporal distribution values is at most 3 percentage points, meaning that the solution is able to reschedule the specialties following the initial target values for all but three specialties, missing the correct target value by just 1 percentage point. We point out that, in instances with at most 10% of days with unusable ORs, i.e., instance 1 to 4, the solution of the ASP encoding of the RMSS problem differs from the solution of the MSS problem only of 1%; whereas, increasing the percentage of unusable days and ORs up to 20%, the difference between the solutions is just the 6%. This means that, even when the setting of the ORs changes and the target values are still to be matched, the rescheduling solution is able to recreate a new schedule by changing just the minimum necessary number of sessions.

Second scenario: changes in target values. Here we simulate a scenario where, due to changes in future needs, one or more specialties adjust their target values. This adjustment necessitates increasing or decreasing the number of sessions allocated to those specialties accordingly. Due to this change, it derives that to reach the new target values, the difference between the solution of the MSS and RMSS problems will be larger when compared to the first scenario. Indeed, allocating more/fewer sessions to the specialties will require some significant changes to the original solution.

In the first 6 instances tested, we can see that the total difference between the target values and the temporal distribution value is at most 3 percentage points. This means that

the solution is able to reschedule the specialties according to the new target values assigned in this scenario, keeping, in all of these instances, more than 90% of the original MSS solution unchanged; in the last two instances, instead, the quality of the solution, meaning the difference between the target value and the temporal distribution, and the difference between the solution of the MSS and RMSS, is affected by the increasing difference between the original and the new target value. Indeed, in this case, the difference is 6 and 4 percentage points, respectively. However, taking into accounts the increasing difference between the original target values and the new ones, it should be noted that the RMSS solutions are able to maintain the same solution for more than 85% of the considered sessions.

Third scenario: limit on the usage of specialties. In this third scenario, we examine a situation where some specialties have a constraint on the total number of days they can be allocated. The best solution changes only the assignments regarding the limited specialties, maintaining unchanged the other two parameters of optimizations.

Despite that the solver is not able to prove the optimality in 10 s, the results show that the ASP encoding computes an optimal solution over all the instances. Indeed, as reported in Table 3, the solutions obtained with all the instances follow the required target values, and the difference between the MSS and the RMSS solutions regards only the limited specialties, that have to be rescheduled, keeping the remaining assignment unchanged.

6.3 Comparison to alternative logic-based formalisms

In the following, we present an empirical comparison of the original solution presented in Sect. 5 on different alternative logic-based formalisms, obtained by applying automatic translations of ASP instances. With this analysis, we want to compare ASP performance to that of other, possibly commercial, solutions. In more detail, we used the ASP solver WASP [2], with the option `–pre=wbo`, which converts ground ASP instances into pseudo-Boolean instances in the wbo format [30]. Then, we used the tool PYPBLIB [5] to encode wbo instances as MaxSAT instances. Moreover, in order to provide a fair comparison, given that other approaches can not deal with multi level optimization, we also processed our ASP instances using WASP with the option `–pre=1parse`, which collapses all weak constraint levels into one single level using exponential weights. In this way, the costs found by the different approaches can be compared.

For the comparison, we considered two state-of-the-art MaxSAT solvers, namely MAXHS [33] and OPEN- WBO [28], and the industrial ILP tool for solving optimization problems GUROBI [26], which is able to process instances in wbo format. Concerning ASP, we used CLINGO with the option `restart-on-model`, that was the solver and the option used in Sect. 6.2, but now run on ground instances.

Table 4 Comparison of the rescheduling ASP solution using CLINGO with the option `restart-on-model` (CLINGO- ROM in the table) and the alternative logic-based solutions GUROBI, on wbo instances, MAXHS and OPEN- WBO. The value in each cell represents if the solver was able to find the optimal solution in less than 10 s (OPT), a percentage value representing the gap to the optimal solution, or a dash in case the solver was not able to find any solution before the timeout

Instance	CLINGO- ROM	GUROBI	MAXHS	OPEN- WBO
1	0.05%	OPT	-	60%
2	OPT	OPT	OPT	OPT
3	OPT	OPT	-	OPT
4	OPT	OPT	-	53%
5	0.03%	OPT	-	215%
6	0.02%	OPT	-	468%
7	0.03%	OPT	-	167%
8	0.04%	OPT	-	167%

The experiment was executed on the 8 instances of Scenario 1 presented in Sect. 6. Results are reported in Table 4, 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, computed as the ratio between the difference of the sub-optimal and optimal solutions, and the optimal one (as a concrete example, consider Instance 6 for OPEN- WBO in Table 4: the sub-optimal solution computed by the solver is 193069 while the optimal solution is 33960, thus the percentage gap is $\frac{193069-33960}{33960}$. The results obtained show that using GUROBI is possible to obtain the optimal solution in all the instances before the timeout. Concerning CLINGO- ROM, it is possible to see that, while the optimal solution is reached in 4 instances, in all the other instances the obtained result is very close to the optimal solution. Indeed, in all the remaining instances, the gap of the solution found using ASP to the optimal one is at most 0.05%. Concerning OPEN- WBO, we can see that while it is able to find the optimal solution in two instances, when it does not find the optimal solution, the found solution is of low quality. This is due to the solving algorithm it employes. Finally, MAXHS can solve just one instance, optimally, but it is not able to find any solution to the other instances. To sum up, it is possible to state that GUROBI and CLINGO- ROM are the two solvers that perform better in this comparison: GUROBI always finds the optimal solution, while CLINGO- ROM computes either the optimal solution or a solution that has a small gap to it.

7 Related work

The section is organized in two subsections: the first presents works that highlight the importance of solving the (R)MSS problem, and alternative methods for solving such problems, with a focus on the works using real data. The second subsection, instead, mentions works in which ASP has been already successfully employed to closely related scheduling problems, with focus on works in which rescheduling solutions have been devised.

7.1 Solving the MSS problem

In [22] is presented a literature review on how different Operations Research techniques can be applied to surgical planning. Presenting the different approaches to the MSS problem, the authors pointed out that a more efficient MSS can improve the usage of the different resources involved (such as wards, that we do not take into account). Some works were able to use real data to test their solutions; among them, [37] shows the benefit of implementing an effective MSS in a regional hospital in the Netherlands. In particular, thanks to the suggestion of the solution proposed, the hospital was able to reduce the budget while increasing the number of patients operated. In this work, the MSS is evaluated as a cyclic schedule composed of different individual surgical case types. Thus, the MSS is composed by a sequence of surgeries instead of blocks of specialties. Moreover, the MSS is planned for 3 weeks only. In [36], the authors proposed a solution to the MSS problem and the surgical case assignments problem formulating it using a mixed integer nonlinear programming approach. They compared their solutions to the historical data of an Australian public hospital. Differently from our work, the solution proposed by the authors maximizes the number of patients operated instead of focusing on target values required by the hospital. [27] used a mixed integer linear programming model to address the problem. They used the required surgeries of the week to assign the ORs to the different specialties and considered a fixed (two) number of sessions for each day. The tests done in this work are conducted on real data provided by a medium-sized Portuguese private hospital. In [38], the authors addressed the MSS problem by proposing a cyclic schedule for the frequently performed surgical procedures, maximizing the operating room utilization. In this work, the solution was tested with data from the Erasmus Medical Center in Rotterdam, The Netherlands.

The work in [6] used a simulation-optimization approach to solve the MSS problem. In particular, they used a two-stage stochastic optimization model and a discrete-event simulation model to handle uncertainty such as the surgery duration. They tested the solution with generated synthetic data; further, they did not consider a target value for the different specialties.

All such works focused on the scheduling problem, while they did not consider rescheduling.

7.2 Solving scheduling problems in healthcare with ASP

ASP has been successfully used for solving hard combinatorial and application scheduling problems in several research areas. In the healthcare domain, the first solved problem was the *Nurse Scheduling Problem* [3, 17], where the goal is to create a scheduling for nurses working in hospital units. Then, the problem of assigning ORs to patients, denoted as *Operating Room Scheduling*, has been treated [15], and further extended to include bed management [14]. More recent problems include the *Chemotherepy Treatment Scheduling* problem [13], in which patients are assigned a chair or a bed for their treatments, and the *Rehabilitation Scheduling Problem* [11], which assigns patients to operators in rehabilitation sessions. In both recent works, real data were used to test the solutions. Moreover, for all the works mentioned up to now, also rescheduling solutions have been defined, implemented and tested [4, 13, 16].

In [12] and [10], it is proposed a solution to a problem of scheduling chronic outpatients' clinical pathways, which is 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 a visit or a therapy for multiple recurrent exams to chronic patients. The problem is split into two sub-problems to increase the performance of the solution using Benders' decomposition method.

8 Conclusion

In this paper, we have presented an analysis of the MSS and RMSS problems, as defined in the paper, modeled and solved with ASP. For both problems, we started from an informal description of the problem, then formulated in precise mathematical terms, and finally presented our ASP solutions. We focused our analysis on real data from ASL1 Liguria in Italy, and considered a number of tasks: for the MSS problem we also needed to adapt the encoding to the real data on a task. Results demonstrate the overall efficacy of our solution for both scheduling and rescheduling also when tested on real data, also when compared to other logic-based formalisms. For what concerns future works, it would be interesting to analyze other scenarios in which some of our constraints are relaxed, e.g., in which an OR can be shared among specialties, and/or added, e.g., in which some ORs are reserved to emergency situations.

Encodings and benchmarks employed in this paper can be found at: https://github.com/MarcoMochi/PAI-HC2023-mss.

Funding Open access funding provided by Università della Calabria within the CRUI-CARE Agreement.

Declarations

Conflict of interest The authors have no relevant financial or nonfinancial interests to disclose.

References

1. Abels, D., Jordi, J., Ostrowski, M., Schaub, T., Toletti, A., Wanko, P.: Train scheduling with hybrid ASP. In: LPNMR. Lecture notes in computer science, vol. 11481, pp. 3–17. Springer (2019)
2. 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)
3. 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)
4. Alviano, M., Dodaro, C., Maratea, M.: Nurse (re)scheduling via answer set programming. Intelligenza Artificiale **12**(2), 109–124 (2018)
5. Ansótegui, C., Pacheco, T., Pon, J.: Pypblib (2019), https://pypi.org/project/pypblib/
6. Bovim, T.R., Christiansen, M., Gullhav, A.N., Range, T.M., Hellemo, L.: Stochastic master surgery scheduling. Eur. J. Oper. Res. **285**(2), 695–711 (2020)
7. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011)
8. Buccafurri, F., Leone, N., Rullo, P.: Enhancing disjunctive datalog by constraints. IEEE Trans. Knowl. Data Eng. **12**(5), 845–860 (2000)
9. 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 Pract. Logic Program. **20**(2), 294–309 (2020)
10. Cappanera, P., Gavanelli, M., Nonato, M., Roma, M.: Decomposition approaches for scheduling chronic outpatients' clinical pathways in answer set programming. J. Logic Comput. **33**, exad038 (2023). https://doi.org/10.1093/logcom/exad038
11. 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). LNCS, vol. 12851, pp. 111–125. Springer (2021)
12. Caruso, S., Galatà, G., Maratea, M., Mochi, M., Porro, I.: Scheduling pre-operative assessment clinic with answer set programming. J. Log. Comput. **34**(3), 465–493 (2024)
13. Dodaro, C., Galatà, G., Grioni, A., Maratea, M., Mochi, M., Porro, I.: An ASP-based solution to the chemotherapy treatment scheduling problem. Theory Pract. Logic Program. **21**(6), 835–851 (2021)
14. 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). LNCS, vol. 11784, pp. 67–81. Springer (2019)
15. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: Operating room scheduling via answer set programming. In: AI*IA. LNCS, vol. 11298, pp. 445–459. Springer (2018)
16. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: An ASP-based framework for operating room scheduling. Intelligenza Artificiale **13**(1), 63–77 (2019)
17. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: LPNMR. LNCS, vol. 10377, pp. 301–307. Springer (2017)
18. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Mag. **37**(3), 53–68 (2016)
19. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artif. Intell. **175**(1), 278–298 (2011)
20. Falkner, A.A., Friedrich, G., Schekotihin, K., Taupe, R., Teppan, E.C.: Industrial applications of answer set programming. Künstl. Intell. **32**(2–3), 165–176 (2018)
21. Ferrand, Y.B., Magazine, M.J., Rao, U.S.: Managing operating room efficiency and responsiveness for emergency and elective surgeries—a literature survey. IIE Trans. Healthcare Syst. Eng. **4**(1), 49–64 (2014)
22. Francesca, G., Guido, R.: Operational research in the management of the operating theatre: a survey. Health Care Manag. Sci. **14**(1), 89–114 (2001). https://doi.org/10.1007/s10729-010-9143-6
23. Galatà, G., Maratea, M., Mochi, M.: Master surgical scheduling via answer set programming tested on real data. In: Calimeri, F., Dragoni, M., Stella, F. (eds.) Proceedings of the 2nd AIxIA workshop on artificial intelligence For healthcare (HC@AIxIA 2023) co-located with the 22nd International conference of the Italian association for artificial intelligence (AIxIA 2023). CEUR workshop proceedings, vol. 3578, pp. 130–144. CEUR-WS.org (2023)
24. 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)
25. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: from theory to practice. Artif. Intell. **187**, 52–89 (2012)
26. Gurobi Optimization, LLC: Gurobi optimizer reference manual (2021), https://www.gurobi.com
27. Marques, I., Captivo, M.E., Barros, N.: Optimizing the master surgery schedule in a private hospital. Operations research for health care **20**, 11–24 (2019), https://www.sciencedirect.com/science/article/pii/S2211692318300225
28. Martins, R., Manquinho, V.M., Lynce, I.: Open-wbo: a modular maxsat solver,. In: SAT 2014. LNCS, vol. 8561, pp. 438–445. Springer (2014). https://doi.org/10.1007/978-3-319-09284-3_33
29. Mochi, M., Galatà, G., Maratea, M.: Master surgical scheduling via answer set programming. J. Log. Comput. **33**(8), 1777–1803 (2023)

30. Olivier Roussel and Vasco Manquinho: Input/Output format and solver requirements for the competitions of pseudo-boolean solvers (2012)
31. Oostrum, J., Bredenhoff, E., Hans, E.: Suitability and managerial implications of a master surgical scheduling approach. Annals OR **178**, 91–104 (2010). https://doi.org/10.1007/s10479-009-0619-z
32. Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N.: Team-building with answer set programming in the Gioia-Tauro seaport. Theory Pract. Logic Program. **12**(3), 361–381 (2012)
33. Saikko, P., Berg, J., Järvisalo, M.: LMHS: A SAT-IP hybrid maxsat solver. In: SAT 2016. LNCS, vol. 9710, pp. 539–546. Springer (2016). https://doi.org/10.1007/978-3-319-40970-2_34
34. Scanu, M., Mochi, M., Dodaro, C., Galatà, G., Maratea, M.: Operating room scheduling via answer set programming: the case of ASL1 liguria. In: CILC 2023. CEUR workshop proceedings, vol. 3428. CEUR-WS.org (2023)
35. Schüller, P.: Answer set programming in linguistics. Künstliche Intell. **32**(2–3), 151–155 (2018)
36. Spratt, B., Kozan, E.: Waiting list management through master surgical schedules: a case study. Oper. Res. Health Care **10**, 49–64 (2016). https://doi.org/10.1016/j.orhc.2016.07.002
37. van Oostrum, Jeroen: Applying mathematical ‚models to surgical patient planning. Ph.D. thesis, E (Sep) (2009), http://hdl.handle.net/1765/16728
38. van Oostrum, J., van Houdenhoven, M., Hurink, J., Hans, E., Wullink, G., Kazemier, G.: A master surgical scheduling approach for cyclic scheduling in operating room departments. OR Spectrum = OR Spektrum **30**(2), 355–374 (2008). https://doi.org/10.1007/s00291-006-0068-x
39. Van Riet, C., Demeulemeester, E.: Trade-offs in operating room planning for electives and emergencies: a review. Oper. Res. Health Care **7**, 52–69 (2015). https://doi.org/10.1016/j.orhc.2015.05.005