An ASP-based Approach to Master Surgical Scheduling

Linda Cademartori[1], Giuseppe Galatà[2], Carola Lo Monaco[1], Marco Maratea[1], Marco Mochi[1] and Marco Schouten[3]

[1]University of Genoa, Genova, Italy
[2]SurgiQ srl, Genova, Italy
[3]KTH Royal Institute of Technology in Stockholm, Sweden

Abstract

The problem of finding Master Surgical Schedules (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 rooms. 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. A poorly scheduled MSS may lead to unbalanced specialties availability and increase patients' waiting list, negatively affecting both the administrative costs of the hospital and the patient satisfaction. In this paper, we present a compact solution based on Answer Set Programming (ASP) to the MSS problem. We tested our solution on different scenarios: experiments show that our ASP solution provides satisfying results in short time, also when compared to other logic-based formalisms. Finally, we describe a web application we have developed for easy usage of our solution.

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 the modern hospitals are long waiting lists that reduce patients' satisfaction and the level of care offered to them. The Master Surgical Schedule (MSS) represents which specialty is assigned to each operating room in a particular day and session. The administrative practices of surgical departments, such as deciding which operating rooms are assigned to the specialties, 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 [1, 2, 3, 4]);

CILC 2022: 37th Italian Conference on Computational Logic, June 29 – July 1, 2022, Bologna, Italy
4518264@studenti.unige.it (L. Cademartori); giuseppe.galata@surgiq.com (G. Galatà); 4416766@studenti.unige.it (C. Lo Monaco); marco.maratea@unige.it (M. Maratea); marco.mochi@edu.unige.it (M. Mochi); schouten@kth.se (M. Schouten)
http://www.star.dist.unige.it/~marco/ (M. Maratea); https://marcomochi.github.io/website/ (M. Mochi)
0000-0002-1948-4469 (G. Galatà); 0000-0002-9034-2527 (M. Maratea); 0000-0002-5849-3667 (M. Mochi)

in particular, the introduction of an effective MSS lead to efficiency gains at the operating room department: At Beatrix hospital the annual budget for operating room hours is reduced from 12,848 hours to 9,972 hours (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% [5]. 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, 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, 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 AI languages such as Answer Set Programming (ASP). Indeed ASP, thanks to its readability and the availability of efficient solvers, e.g., CLINGO [6], has been successfully employed for solving hard combinatorial problems in several research areas, and it has been also employed to solve many scheduling problems [7, 8, 9, 10, 11], also in industrial contexts (see, e.g., [12, 13, 14] for detailed descriptions of ASP applications).

In this paper we present a 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, obtained by varying the number of days and trying different scenarios, created with realistic sizes and parameters inspired from data seen in literature. Results using the state-of-the-art ASP solver CLINGO show that ASP is a suitable solving methodology also for the MSS problem, since we are able to solve optimally instances of the MSS problem in few seconds even considering planning horizon up to 180 days. We also compare the performance of our ASP solution to those of top performing Max-SAT, Pseudo-Boolean and ILP solvers run on instances obtained by automated translation of ASP encoding and instances: Results show that CLINGO, run on the ASP encoding contribution of this paper and employing an optimization algorithm based on unsatisfiable cores, is almost always the best option. Finally, we describe the implementation of a web application we have developed in order to support users in the usage of our solution. The application allows for inserting the main parameters of the problem, running CLINGO on the encoding without actually installing nothing locally, and showing results graphically.

The paper is structured as follows. Sections 2 and 3 present an informal description of the MSS problem, and its precise mathematical formulation, respectively. Then, Section 4 shows our ASP encoding, whose experimental evaluation is presented in Section 5. Section 6 describes the implementation of our web application. The paper ends by discussing related work and conclusions in Section 7 and 8, respectively.

2. Problem Description

With the computation of an MSS, a hospital can see in 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 the personnel and the resources in advance. The MSS is thus often scheduled for long periods of time and as soon as possible, to be able to assign the surgery to the patients in time and to properly organize the personnel. To schedule the MSS a hospital should evaluate the percentage of time that needs to be assigned to each specialty and the allowed errors for such a period of time, in order to better respond to the patients' needs. The percentage of assignments is evaluated as the number of times each specialty is assigned 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 to 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 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 all the ORs and the scheduler must assign a specialty to each session. Since the MSS is planned for a long period of time, hospitals could desire that the target assignment of each specialty is respected not for all the considered days, but may vary, e.g., on a monthly or weekly basis. Another aspect that could change during the considered period and between the ORs are the sessions. The usage of each OR is often splitted in two sessions for each day but, sometimes, some ORs can be split in a different number of sessions, higher or used even for just one session. In particular, the single-session solution could be used when a specialty requires particular resources and the time to prepare them is long enough that changing the specialty at mid day would be a waste of time. Moreover, some ORs could be unavailable in some days and the scheduler 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, that we will show later in the paper, considering 90 days and fixed target value for each month. Moreover, we considered a hospital with 10 ORs, each splitted in 2 sessions in each day, and 5 specialties (these numbers corresponding 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.

3. Mathematical formulation of the MSS problem

In this section, we provide a mathematical formulation of the basic version of the problem (called Scenario A later).

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

Definition 1. *Let*

- *day be a constant that is equal to the number of days considered;*
- *max_session be a constant that is equal to the maximum number of session associated to an operating room in a day;*
- *s_count be a constant that is equal to day* $\times$ *max_session and represents the number of sessions that must be assigned to each operating room;*
- $D = \{t : t \in [1..day]\}$ *be the set of all days;*
- $DD = \{(d_1,d_2)_1,\ldots,(d_1,d_2)_n\}$ *be a set of n pair of days such that for every pair* d_2 *is greater than* d_1*;*
- $OR = \{o_1,\ldots,o_m\}$ *be a set of m operating rooms;*
- $SP = \{sp_1,\ldots,sp_k\}$ *be a set of k specialties;*
- $S = \{s_1,\ldots,s_{s\_count}\}$ *be a set of s_count sessions id;*
- $\delta : OR \times SP \mapsto \{0,1\}$ *be a function associating an operating room to a specialty such that* $\delta(o,sp) = 1$ *if the operating room o can be assigned to the specialty sp, and* 0 *otherwise;*
- $\rho : OR \times D \mapsto S$ *be a function associating an operating room and a day to a session id such that* $\rho(o_n,d_m) \geq max\_session * d_m$ *- (max_session-1) and* $\rho(o_n,d_m) \leq max\_session * d_m$*;*
- $\varepsilon : SP \times D \times D \mapsto \mathbb{N}$ *be a function associating a specialty, a starting day and an ending day to a value representing the percentage target to reach from the starting day to the ending day;*
- $\omega : SP \times D \times D \mapsto \mathbb{N}$ *be a function associating a specialty, a starting day and an ending day to a value representing the maximum error that is allowed from the starting day to the ending day;*
- $\zeta : SP \times D \times D \mapsto \mathbb{N}$ *be a function associating a specialty, a starting day and an ending day to a value representing the percentage of times that a session has been assigned to the specialty.*

Let $mss : OR \times S \times SP\ D \mapsto \{0,1\}$ *be a function such that* $mss(o,s,sp,d) = 1$ *if the session* s *in the day* d *and in the operating room* o *is assigned to the specialty* sp*, and* 0 *otherwise. Moreover, for a given mss, let* $A_{mss} = \{(o,s,sp,d) : o \in OR, s \in S, sp \in SP, d \in D, mss(o,s,sp,d) = 1\}$.

Then, given sets OR, SP, S, D, DD and functions δ*,* ρ*,* ε*,* ω*,* ζ*, the MSS problem is defined as the problem of finding a schedule x, such that*

(c_1) $|\{\rho(o,d) = s\}|=1 \quad \forall o \in OR, \forall d \in D, \forall s \in S;$

(c_2) $|\{sp : mss(o,s,sp,d) = 1\}|=1 \quad \forall o \in OR, \forall s \in S, \forall sp \in SP, \forall d \in D, \rho(o,d) = s;$

(c_3) $|\{mss(o,s,sp,d) = 1\}|=0 \quad \forall o \in OR, \forall s \in S, \forall sp \in SP, \forall d \in D, \rho(o,d) = s, \delta(or,sp) = 0;$

(c_4) $|\{mss(o,s,sp,d) = 1\}|=0 \quad \forall o \in OR, \forall s \in S, \forall sp \in SP, \forall d \in D, \rho(o,d) \neq s;$

(c_5) $\zeta(sp,d_1,d_2) > 0 \quad \forall sp \in O, \forall (d_1,d_2) \in DD;$

(c_6) $|\varepsilon(sp,d_1,d_2) - \zeta(sp,d_1,d_2)| \leq \omega(sp,d_1,d_2) \quad \forall sp \in O, \forall (d_1,d_2) \in DD;$

Condition (c_1) ensures that at each operating room is assigned to a session s_{count} times. Condition (c_2) ensures that each operating room, in each day and session is assigned to exactly one specialty. Condition (c_3) ensures that no operating room is assigned to a not allowed specialty. Condition (c_4) ensures that each specialty is assigned to an operating room in the right session and day. Condition (c_5) ensures that the percentage of times a specialty is assigned is bigger than 0 in every range of days required. Condition (c_6) ensures that the percentage target of time a specialty is assigned minus the actual percentage is less than the allowed error.

Definition 2 (Distance target percentage). *Given a solution mss, let* $t_{mss} = \sum_{sp \in SP, (d_1,d_2) \in DD;} | \varepsilon(sp,d_1,d_2) - \zeta(sp,d_1,d_2) |$ *. Intuitively,* t_{mss} *represents the sum of the distance between the target percentage and the actual percentage of times each specialty is assigned to the operating rooms in the range between* d_1 *and* d_2*.*

Definition 3 (Optimal solution). *A solution mss is said to dominate a solution mss' if* $|t_{mss}| < |t_{mss'}|$*. A solution is optimal if it is not dominated by any other solution.*

4. ASP Encoding for the MSS problem

We assume the reader is familiar with syntax and semantics of ASP. Starting from the specifications in the previous section, here we present our compact and efficient ASP solution for the MSS problem, organized in two paragraphs containing input and output data model, and the ASP encoding, respectively. The ASP encoding is based on the input language of CLINGO [15]. For details about syntax and semantics of ASP programs we refer the reader to [16].

```
1 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).
2 n_session(N,START,END) :- N = #count{SID,OR,DAY : session(SID,OR,DAY), DAY >= START, DAY <
      END}, targetShare(_,_,_START,END).
3 {mss(OR,SID,SP,DAY) : operatingRoom(OR, SP)} == 1 :- session(SID,DAY,OR).
4 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).
5 :- effectiveShare(SP,PERCENTAGE,START,END), targetShare(SP,TARGET,ERROR,START,END), PERCENTAGE
      < (TARGET-ERROR).
6 :- effectiveShare(SP,PERCENTAGE,START,END), targetShare(SP,TARGET,ERROR,START,END), PERCENTAGE
      > (TARGET+ERROR).
7 :- effectiveShare(SP,PERCENTAGE,START,END), PERCENTAGE <= 0.
8 :~ effectiveShare(SP,ES,START,END), targetShare(SP,TS,ERR,START,END). [|ES-TS|@1,SP,START]
```

Figure 1: ASP encoding of the MSS problem

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

- Instances of `sessionN(OR,N,DAY)` represent the number of sessions (`N`) in which the operating room identified by an id (`OR`) is split in the day (`DAY`).
- Instances of `operatingRoom(OR,SP)` represent which specialty (`SP`) can be assigned to the operating room identified by an id (`OR`).
- Instances of `specialty(SP)` represent the different specialties identified by their id (`SP`).
- 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`.
- Instances of `day(DAY)` represent the available days.

The output is an assignment represented by an atom of the form `mss(OR,SID,SP,DAY)`, where the intuitive meaning is that the operating room with id `OR` in the session with id `SID` and in the day `DAY` is assigned the specialty `SP`.

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.

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 splitted. Rule r_2 evaluates the total number of sessions available in the range of days between the values 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. 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, rules r_5 and r_6 check that the percentage of each specialty is compatible with the target values and the allowed errors. Rule r_7 ensures that the percentage of

each specialty is bigger than 0. Finally, weak constraint r_8 minimizes the difference between the assigned and target percentage of each specialty in each period of time.

5. Experimental Results

In this section, we report the results of an empirical analysis of the MSS problem via ASP (second paragraph). For the problem, data have been randomly generated using parameters inspired by literature and real world data (first paragraph). A third paragraph compares results obtained with alternative logic-based formalisms. 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 [15] 5.4.0, using parameters *--opt-strategy=usc* for faster optimization and *--parallel-mode 4* for parallel execution. This setting is the result of a preliminary analysis (but presented later in Table 3) done also with other parameters, i.e., the default configuration and the one having *--restart-on-model* for optimization. The time limit was set to 30 seconds. Encodings and benchmarks employed in this section can be found at: http://www.star.dist.unige.it/~marco/RuleMLRR2022/material.zip .

MSS benchmarks. 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 specialities. 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 will 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 will 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.

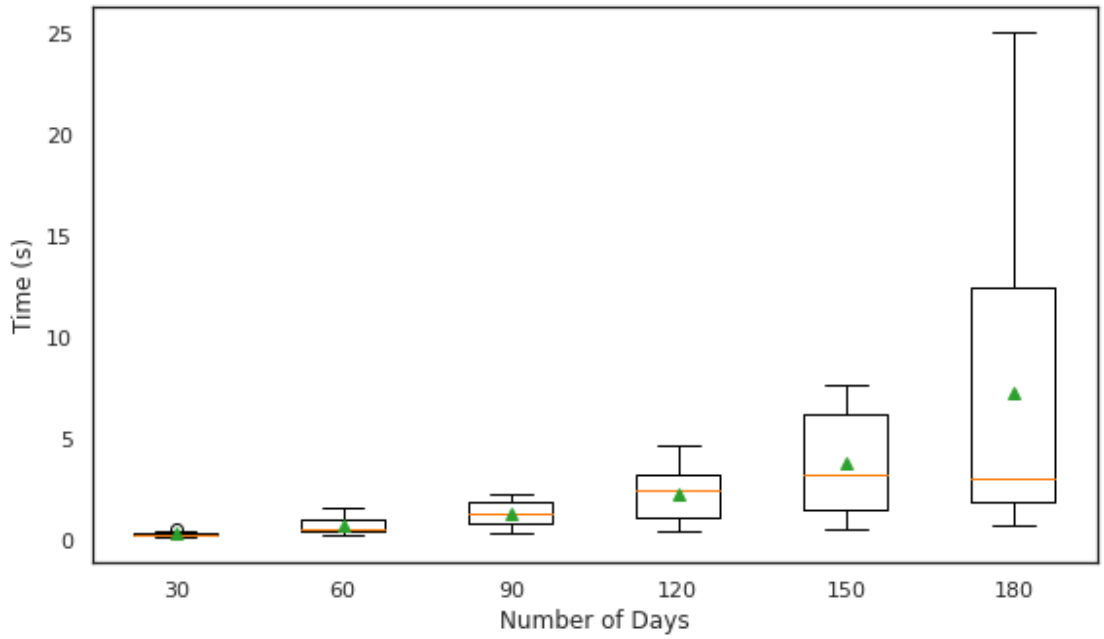

Figure 2: Results obtained by solving 10 instances per group of days in Scenario A. The box starts from the first quartile and ends at the third quartile. The mean time is represented by the (green) triangle, while the (orange) line represents the median value.

Results of our MSS solution. First, we tested the performances of the scheduler in the basic scenario (Scenario A). The results for this scenario are shown in Figure 2, which represents the range of seconds required to reach the optimal solution in all the 10 instances tested with the different number of days considered, identified by the minimum and maximum times for solving the instances in the set, together with the mean and the median time. From the figure it can be seen that the scheduler is able to optimally schedule the MSS in a mean time of less than 10 seconds 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 seconds on average, from the figure it can be noted that even in the worst case, the scheduler is able to find the optimal solution in less than 30 seconds.

Then, we tested the performance of the scheduler in the Scenario B. Testing the scheduler with the 10 instances with 90 days in this scenario we found that the scheduler was able to reach the optimal solution on average in 3 seconds, that is a time that is very near 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.

Having evaluated now the performance in Scenario A and B, we then tested the scheduler in Scenario C and D. Table 2 reports the time required by each instance in Scenario A and in these more constrained scenarios, on 90 days planning horizon.

From the table we can see that the timing obtained by 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 in the Scenario A. In the Scenario D, the scheduler obtained

Table 2
Time required for each instances in the different Scenarios and considering 90 days

Instance #	Time (s) Scenario A	Time (s) Scenario C	Time (s) Scenario D
1	1.7	1.7	1.7
2	0.3	0.2	0.3
3	1.8	1.1	4.3
4	2.0	3.0	2.0
5	0.9	2.2	0.9
6	1.9	1.0	2.1
7	0.8	0.6	0.6
8	2.2	1.6	2.3
9	0.9	0.8	0.9
10	0.9	5.2	0.8
Mean	1.3	1.7	1.5

the optimal solution almost in the same time as in the Scenario A for all but one instance: Indeed, the third instance requires 4 seconds instead of 2 seconds to reach the optimal solution (from a preliminary analysis, this harder instance corresponds to a setting in which a higher number of sessions is set to an OR assigned to only one specialty with low target).

Overall, we can say that also in Scenario C and D the scheduler is able to reach highly satisfying results, also when compared to the basic Scenario A.

Comparison to alternative logic-based formalisms. In the following, we present an empirical comparison of our ASP-based solution with alternative logic-based approaches, obtained by applying automatic translations of ASP instances. In more detail, we used the ASP solver WASP [17], with the option `-pre=wbo`, which converts ground ASP instances into pseudo-Boolean instances in the wbo format [18]. Then, we used the tool PYPBLIB [19] to encode wbo instances as MaxSAT instances.

Then, we considered three state-of-the-art MaxSAT solvers, namely MAXHS [20], OPEN-WBO [21], and RC2 [22], and the industrial tool for solving optimization problems GUROBI [23], which is able to process instances in the wbo format. Concerning CLINGO, we used *(i)* its default configuration (CLINGO-DEF); *(ii)* the option `restart-on-model` (CLINGO-ROM); and *(iii)* the option `-opt-strategy=usc` (CLINGO-USC). The latter enables the usage of algorithm OLL [24], which is the same algorithm employed by the MaxSAT solver RC2.

The experiments were executed on Scenario A considering the 10 instances with 30 days horizon, with a timeout of 30 seconds. Results are reported in Table 3, where for each solver and instance we report the ranking obtained by each solver, counting optimal solutions. The solver is in the first position if it finds the solution in the shortest time; a dash means that the solver did not compute the solution before the time limit. As a general observation, CLINGO-USC obtains the best performance overall, since it is the first to find the optimal solution in all but one instance. The performance of CLINGO-ROM is in general slightly worse than the one of CLINGO-USC, even if in the majority of the instances the required time to reach the optimal solution is similar to the time required by CLINGO-USC. GUROBI is able to reach the optimal solutions before

Table 3
Comparison of ASP solution with alternative logic-based solutions.

Instance	CLINGO-DEF	CLINGO-ROM	CLINGO-USC	MaxHS	OPEN-WBO	RC2	GUROBI
1	4	3	2	6	5	-	1
2	4	2	1	-	-	-	3
3	4	2	1	-	-	-	3
4	4	2	1	-	-	-	3
5	4	2	1	-	-	-	3
6	4	2	1	-	-	-	3
7	4	2	1	-	-	-	3
8	4	2	1	-	-	-	3
9	4	2	1	-	-	-	3
10	2	2	1	-	-	-	3

CLINGO-USC and CLINGO-ROM in the first instance while in all the other instances it ranked third. Concerning MaxSAT solvers, we observe that both OPEN-WBO and MAXHS are able to reach the optimal solution before the time limit just in the first instance, while in all the other instances they can not obtain the optimal solution before the time limit. RC2 can not return an optimal solution in any of the instances evaluated. Concerning CLINGO-DEF, we obtain the optimal solution in all the instances but, without using any of the available options, the solutions are obtained in more time than the other options and GUROBI, but for instance 10.

6. Web Application

After having presented our solution and compared it with other solvers, we have wrapped the encoder and the CLINGO solver inside a NodeJS architecture and developed a simple graphical user interface (GUI) to configure the different inputs of the problem. By developing the web app, we want to reduce the burden related to the installation and the proper usage of the ASP-based solution, mainly for non-technical users. Moreover, even if our solution was able to solve all the tested instances in less than 30 seconds, without a proper interface even the best solution could be discarded because of the difficulties caused by the technology itself.

In particular, the first page of the web app, shown in Figure 3, is devoted to the definition of the characteristics of the problem. It allows setting:

- The number of months to schedule.
- The number of sessions per each ORs.
- The number of ORs.
- The starting day of the MSS.
- The different specialties, each with a specific target and error.
- The timeout of the scheduler.

Once the user is satisfied with the inserted data, by clicking on the "START PLANNING" button, she can start the scheduling. The web app processes the data to transform them to a format that allows the compatibility with the CLINGO solver and, once the solver finds the optimal

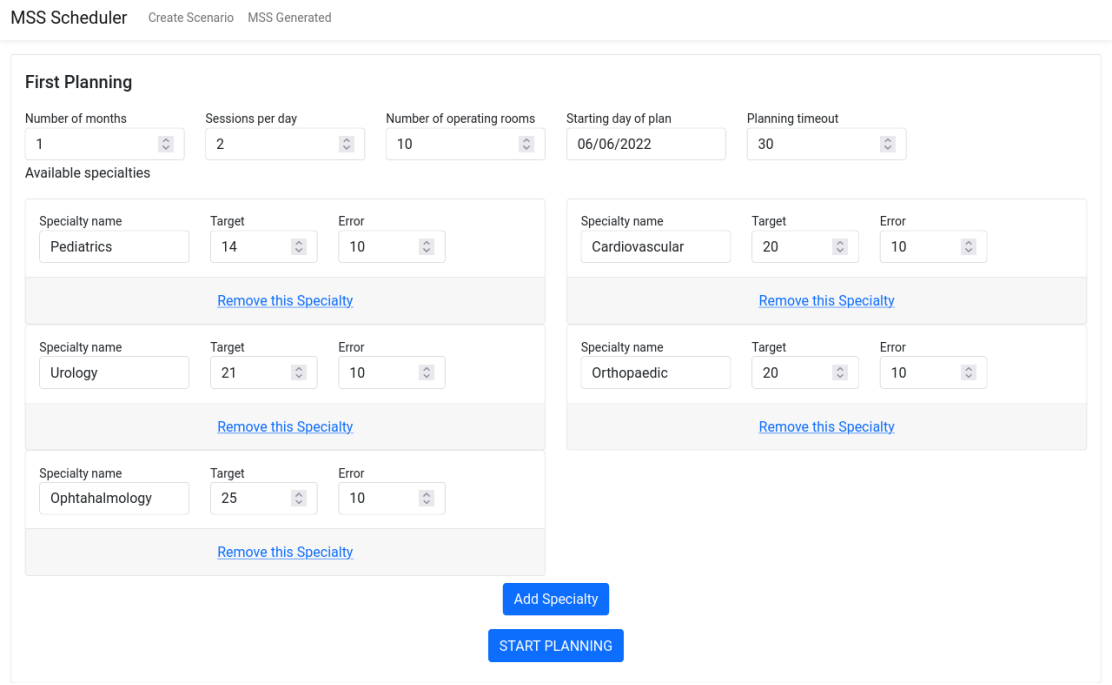

Figure 3: The first page of the web app. From this page the user can define the inputs of the problem.

solution or the timeout time is reached, the user is redirected to the second page. In the second page, reported in Figure 4, there are two cards available. In the first card, called "Table", is shown a properly processed result obtained by the solver (assuming that at least a solution is provided, even if not optimal; however, this is not the case of all our analysis). The card shows the MSS obtained by the solver in a table format. In each row of the table there are the day and the linked session plus the information regarding the specialties assigned to the different ORs, thus mimicking the MSS output of Table 1. In the second card, called "Graphics", are shown the graphs comparing the target and the actual percentage of time each specialty is assigned for every range of days considered. This card helps the user to evaluate the quality of the result in a simple way.

7. Related Work

The section is organized in two paragraphs: the first presents works that highlights the importance of solving the MSS problem and alternative methods for solving the problem, while the second mentions works in which ASP has been already successfully employed to closely related scheduling problems.

Solving the MSS problem. In [4] is presented a literature review on how different Operations Research techniques can be applied to the surgical planning. Presenting the different approaches to the MSS problem, the authors pointed out that a more efficient MSS can improve the usage

Figure 4: The second page of the web app. From this page the user can evaluate the quality of the MSS obtained by the solver.

of the different resources involved (such as wards, that we do not take into account). [5] 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 [3], 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. The work in [25] 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. Differently from our work, they did not consider a target value for the different specialities. The authors of [26] 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. In [27], the authors addressed the MSS problem by proposing a cyclic schedule for the frequently performed surgical procedures, maximizing the operating room utilization.

Solving scheduling problems 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* [28, 29, 10], 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, and further extended to include bed management [9]. More recent problems include the *Chemotherepy Treatment Scheduling* problem [30], 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. Often problems in which an MSS needs to be computed, including those dealing with the Operating Room Scheduling problem mentioned above, consider the MSS as an input of the problem; however, as we have seen in this paper and by the presence of a number of works at the state-of-the-art dealing uniquely with the problem, the MSS is per se of interest and deserves devoted solutions, to be possibly integrated with other problem solutions building on it.

Concerning scheduling problems beyond the healthcare domain, ASP encoding were proposed for the following problems: *Incremental Scheduling Problem* [31], where the goal is to assign jobs to devices such that their executions do not overlap one another; *Team Building Problem* [7], where the goal is to allocate the available personnel of a seaport for serving the incoming ships; the work in [32], where, in the context of routing driverless transport vehicles, the setup problem of routes such that a collection of transport tasks is accomplished in case of multiple vehicles sharing the same operation area is solved via ASP, in the context of car assembly at Mercedes-Benz Ludwigsfelde GmbH, and the recent survey paper by Falkner et al. [13], where industrial applications dealt with ASP are presented, including those involving scheduling problems.

8. Conclusion

In this paper, we have presented an analysis of the MSS problem modeled and solved with ASP. We started from an informal description of the problem, formulated it in precise mathematical terms, and then presented our ASP solution. Results on synthetic benchmarks show that the ASP solution is able to optimally solve the MSS problem even when considering large planning horizons, up to 6 months. Moreover, solving more difficult scenarios, in which, e.g., targets and number of sessions change within the planning horizon, reduce just slightly the performance of the scheduler. We also compared our solution to other logic-based languages and tools for solving combinatorial problems, on instances obtained by automatic transformation of ASP instances: The analysis shows that, on instances of our basic scenario, our solution with CLINGO employing optimization algorithms based on unsatisfiable cores [33] has the best performance. For what concerns future works, we are currently working on extending our experiments. Moreover, we would like to implement and test optimization algorithms (see, e.g., [34]), and to investigate re-scheduling solutions, that may come into play when the MSS scheduling can not be implemented for some reasons, e.g., sudden unavailability of ORs. Finally, we plan to propose our benchmarks to future ASP Competitions [35].

References

[1] C. Van Riet, E. Demeulemeester, Trade-offs in operating room planning for electives and emergencies: A review, Operations Research for Health Care 7 (2015) 52–69. doi:https://doi.org/10.1016/j.orhc.2015.05.005, proc. of ORAHS 2014.

[2] Y. B. Ferrand, M. J. Magazine, U. S. Rao, Managing operating room efficiency and responsiveness for emergency and elective surgeries—a literature survey, IIE Transactions on Healthcare Systems Engineering 4 (2014) 49–64. arXiv:https://doi.org/10.1080/19488300.2014.881440.

[3] B. Spratt, E. Kozan, Waiting list management through master surgical schedules: A case study, Operations Research for Health Care 10 (2016) 49–64. URL: https://www.sciencedirect.com/science/article/pii/S2211692316300042. doi:https://doi.org/10.1016/j.orhc.2016.07.002.

[4] G. Francesca, R. Guido, Operational research in the management of the operating theatre: a survey., Health care management science 14,1 (2001) 89–114. doi:doi:10.1007/s10729-010-9143-6.

[5] van Oostrum, Jeroen, Applying Mathematical Models to Surgical Patient Planning, Ph.D. thesis, E, 2009. URL: http://hdl.handle.net/1765/16728.

[6] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, Artificial Intelligence 187 (2012) 52–89.

[7] F. Ricca, G. Grasso, M. Alviano, M. Manna, V. Lio, S. Iiritano, N. Leone, Team-building with answer set programming in the Gioia-Tauro seaport, Theory and Practice of Logic Programming 12 (2012) 361–381.

[8] D. Abels, J. Jordi, M. Ostrowski, T. Schaub, A. Toletti, P. Wanko, Train scheduling with hybrid ASP, in: LPNMR, volume 11481 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 3–17.

[9] 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.

[10] M. Alviano, C. Dodaro, M. Maratea, Nurse (re)scheduling via answer set programming, Intelligenza Artificiale 12 (2018) 109–124.

[11] 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.

[12] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Magazine 37 (2016) 53–68.

[13] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, Künstliche Intelligenz 32 (2018) 165–176.

[14] P. Schüller, Answer set programming in linguistics, Künstliche Intelligence 32 (2018) 151–155.

[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] 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.

[17] M. Alviano, G. Amendola, C. Dodaro, N. Leone, M. Maratea, F. Ricca, Evaluation of disjunctive programs in WASP, in: LPNMR 2019, volume 11481 of *LNCS*, Springer, 2019, pp. 241–255.

[18] Olivier Roussel and Vasco Manquinho, Input/Output Format and Solver Requirements for the Competitions of Pseudo-Boolean Solvers, 2012.

[19] C. Ansótegui, T. Pacheco, J. Pon, Pypblib, 2019. URL: https://pypi.org/project/pypblib/.

[20] P. Saikko, J. Berg, M. Järvisalo, LMHS: A SAT-IP hybrid maxsat solver, in: SAT 2016, volume 9710 of *LNCS*, Springer, 2016, pp. 539–546. URL: https://doi.org/10.1007/978-3-319-40970-2_34. doi:10.1007/978-3-319-40970-2_34.

[21] R. Martins, V. M. Manquinho, I. Lynce, Open-wbo: A modular maxsat solver,, in: SAT 2014, volume 8561 of *LNCS*, Springer, 2014, pp. 438–445. URL: https://doi.org/10.1007/978-3-319-09284-3_33. doi:10.1007/978-3-319-09284-3_33.

[22] A. Ignatiev, A. Morgado, J. Marques-Silva, RC2: an efficient maxsat solver, J. Satisf. Boolean Model. Comput. 11 (2019) 53–64. URL: https://doi.org/10.3233/SAT190116.

[23] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual, 2021. URL: https://www.gurobi.com.

[24] A. Morgado, C. Dodaro, J. Marques-Silva, Core-Guided MaxSAT with Soft Cardinality Constraints, in: CP 2014, Springer, Lyon, France, 2014, pp. 564–573.

[25] T. R. Bovim, M. Christiansen, A. N. Gullhav, T. M. Range, L. Hellemo, Stochastic master surgery scheduling, European Journal of Operational Research 285 (2020) 695–711.

[26] I. Marques, M. E. Captivo, N. Barros, Optimizing the master surgery schedule in a private hospital, Operations Research for Health Care 20 (2019) 11–24. URL: https://www.sciencedirect.com/science/article/pii/S2211692318300225.

[27] J. van Oostrum, M. van Houdenhoven, J. Hurink, E. Hans, G. Wullink, G. Kazemier, A master surgical scheduling approach for cyclic scheduling in operating room departments, OR Spectrum = OR Spektrum 30 (2008) 355–374. doi:10.1007/s00291-006-0068-x.

[28] C. Dodaro, M. Maratea, Nurse scheduling via answer set programming, in: LPNMR, volume 10377 of *LNCS*, Springer, 2017, pp. 301–307.

[29] 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.

[30] 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.

[31] M. Balduccini, Industrial-size scheduling with ASP+CP, in: J. P. Delgrande, W. Faber (Eds.), Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings, volume 6645 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 284–296.

[32] M. Gebser, P. Obermeier, T. Schaub, M. Ratsch-Heitmann, M. Runge, Routing driverless

transport vehicles in car assembly with answer set programming, Theory and Practice of Logic Programming 18 (2018) 520–534.

[33] M. Alviano, C. Dodaro, Unsatisfiable core analysis and aggregates for optimum stable model search, Fundam. Informaticae 176 (2020) 271–297. URL: https://doi.org/10.3233/FI-2020-1974. doi:10.3233/FI-2020-1974.

[34] E. DiRosa, E. Giunchiglia, M. Maratea, A new approach for solving satisfiability problems with qualitative preferences, in: M. Ghallab, C. D. Spyropoulos, N. Fakotakis, N. M. Avouris (Eds.), ECAI, volume 178 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2008, pp. 510–514.

[35] M. Gebser, M. Maratea, F. Ricca, The seventh answer set programming competition: Design and results, Theory and Practice of Logic Programming 20 (2020) 176–204.