An ASP based Solution for Operating Room Scheduling with Surgical Teams in Hospital Environments*

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

[1] DEMACS, University of Calabria, Rende, Italy,
dodaro@mat.unical.it
[2] SurgiQ srl, Italy
giuseppe.galata@surgiq.com,ivan.porro@surgiq.com
[3] DIBRIS, University of Genova, Genova, Italy
muhammad.kamrankhan@edu.unige.it,marco.maratea@unige.it

Abstract. The optimization of daily operating room surgery schedule can be problematic because of many constraints, like to determine the start time of different surgeries and allocating the required resources, including the availability of surgical teams for complete surgical procedures. Recently, Answer Set Programming (ASP) has been successfully employed for addressing and solving real-life scheduling and planning problems in the health-care domain. In this paper we present an enhanced solution using ASP for scheduling operating rooms taking explicitly into consideration availability of surgical teams, that include a surgeon and an anesthetist in different specialties for the entire duration of the surgery. We tested our solution on different benchmarks with realistic parameters for schedule's length up to the target 5-days planning. The results of our experiments show that ASP is a suitable methodology for solving also such enhanced problem.

1 Introduction

Hospitals, whose production output is service, often come across issues of long waiting times, surgeries cancellation for patients and even worst resource overload occur frequently. Within every hospital, Operating Rooms (ORs) are an important unit. As indicated by [30], the ORs account for approximately 33% of the total hospital budget because it includes high staff costs (e..g, surgeons, anaesthetists, nurses) and material cost. Nowadays, in most modern hospitals, long surgical waiting lists are present because of inefficient planning. Therefore, it is extremely important to improve the efficiency of ORs to enhance the survival rate and satisfaction of patients, thereby improving the overall quality of

** Corresponding author.

healthcare system.
The management of ORs basically consists of both planning for providing the date of surgery for each patient, taking into account the availability of ORs and surgeons, and determining the sequence of operations in each OR each day, taking into account the availability of different resources. The Operating Room Scheduling (ORS) [1, 6, 29, 30] problem is the task of assigning patients to ORs by considering specialties, surgery durations, shift durations, beds availability and, most importantly the availability of surgical teams for the entire duration of the surgery. Further, the solution must prioritise patients based on health urgency.
In recent years a solution based on Answer Set Programming (ASP) [8, 21, 22] was proposed and is used for solving such problems [13, 14], together with other similar scheduling problems in this context (e.g., Nurse Scheduling [15, 3]), given its intuitive semantics [9] and the availability of efficient solvers put forward by the ASP Competition series (see, e.g., [10, 18, 19]. We have recently enhanced the previous solutions by incorporating beds management [12]. However, the drawback with these solutions is that they do not consider availability of surgical teams which are an important part of the surgical process. In this paper we improve our basic solution [13, 14] following another direction, and we present an enhanced encoding that takes into explicit account the availability of surgical teams for planning surgical procedure. The problem is expressed in ASP as modular additions to previous, more limited encoding, of ASP rules implementing the surgical teams, and then efficient solvers like CLINGO [17] are used to solve the resulting ASP encoding. Results for planning horizons up to the target 5-days planning, obtained on different benchmarks and scenario with realistic parameters for a small-medium sized Hospital, are positive and inline with Hospital needs, and further confirm that ASP is a suitable methodology for solving scheduling problems in this context.

The paper is structured as follows. Section 2 presents needed preliminary about ASP. Then, Section 3 describes the target problem in an informal way, whose ASP encoding is presented in Section 4. Section 5 shows the results of our experiments. The paper ends in Section 6 and 7 by discussing related work, and by showing conclusions and possible topics for further research.

2 Answer Set Programming

Answer Set Programming (ASP) is a programming paradigm developed in the field of non monotonic reasoning and logic programming. It is a form of declarative programming oriented towards difficult primarily NP-hard search problems and is based on the stable model (answer set) semantics. This section presents the syntax and semantics of the ASP language [9] in two separate paragraphs, and then a widely used short-cut in the third paragraph.

Syntax. The syntax of ASP is similar to the one of Prolog. Variables are strings starting with uppercase letter and constants are non-negative integers or strings

starting with lowercase letters. A *term* is either a variable or a constant. A *standard atom* is an expression $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity n and $t_1, \ldots, t_n$ are terms. An atom $p(t_1, \ldots, t_n)$ is ground if $t_1, \ldots, t_n$ are constants. A *ground set* is a set of pairs of the form $\langle consts : conj \rangle$, where $consts$ is a list of constants and $conj$ is a conjunction of ground standard atoms. A *symbolic set* is a set specified syntactically as $\{Terms_1 : Conj_1; \cdots ; Terms_t : Conj_t\}$, where $t > 0$, and for all $i \in [1, t]$, each $Terms_i$ is a list of terms such that $|Terms_i| = k > 0$, and each $Conj_i$ is a conjunction of standard atoms. A *set term* is either a symbolic set or a ground set. Intuitively, a set term $\{X : a(X, c), p(X); Y : b(Y, m)\}$ stands for the union of two sets: the first one contains the X-values making the conjunction $a(X, c), p(X)$ true, and the second one contains the Y-values making the conjunction $b(Y, m)$ true. An *aggregate function* is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Basically, aggregate functions map multisets of constants to a constant. The most common functions implemented in ASP systems are the following:

- $\#count$, number of terms;
- $\#sum$, sum of integers.

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

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

where $a_1, \ldots, a_n$ are standard atoms, $b_1, \ldots, b_k$ are atoms, $b_{k+1}, \ldots, b_m$ are standard atoms, and $n, k, m \geq 0$. A literal is either a standard atom a or its negation $not\ a$. The disjunction $a_1 \vee \ldots \vee a_n$ is the *head* of r, while the conjunction $b_1, \ldots, b_k, not\ b_{k+1}, \ \ldots, not\ b_m$ is its *body*. Rules with empty body are called *facts*. Rules with empty head are called *constraints*. A variable that appears uniquely in set terms of a rule r is said to be *local* in r, otherwise it is a *global* variable of r. An ASP program is a set of *safe* rules, where a rule r is *safe* if both 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 $Conj$.

A *weak constraint* ω is of the form:

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

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

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

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

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

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

A model is an interpretation that satisfies all rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [16] 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) [16].

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

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

3 Problem Description

This section provides the description and the requirements of our problem. The elements of the waiting list are called registrations. Moreover, registrations are not all equal, as they can belong to different specialties and can be in the waiting list for a long period of time. All ORs are available for 5 consecutive hours (300 minutes) in a single shift while a full day consists of two shifts. Of course, the assignments must guarantee that the sum of the predicted duration of surgeries assigned to a particular OR shift does not exceed the length of the shift itself. For each registration we consider three priority score P1, P2, and P3, where P1 is for high priority or very urgent, P2 is medium priority and P3 is for low priority. Since P1 gathers high priority registrations, they must be all assigned to an OR, followed by P2 registrations over the P3. Additionally, in each specialty

Table 1. Total number of surgeons and anaesthetists in each specialty.

Specialty	Number of Surgeons	Number of anaesthetists
1	6	6
2	4	4
3	4	4
4	2	2
5	4	4
Total	20	20

Table 2. Surgeons availability for each specialty and in each day.

Days (D)	1		2		3		4		5	
Shifts (s)	1	2	3	4	5	6	7	8	9	10
Specialty (SP)	Surgeons									
1	3	3	3	3	3	3	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2
3	2	2	2	2	2	2	2	2	2	2
4	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1

(considered to be 5 as target in small-medium-sized Hospitals) surgical teams are allocated with number of surgeons and anaesthetists every day as shown in Table 1. Every surgeon works specifically for 4 hours every day; also surgeons in each specialty are assigned only to a single shift in a day, i.e., they either work in the morning or in evening shift as shown in Table 2. The anaesthetists are also linked to specialty and their work duration is 6 hours every day, but they can work together with surgeons during any shift of the day as shown in Table 3.
In our model we also assume that surgeons for each surgery is predetermined. Once a surgery is started in an OR it cannot be interrupted. Further, surgeons cannot operate on more than one patient at the same time. The overall goal is to assign the maximum number of registrations to the ORs, respecting the priorities, and taking into account the availability of respective surgical teams in a particular specialty for the complete surgery duration.

4 ASP Encoding for ORS with Surgical Teams

In this section we present the input predicates and our ASP encoding, in two different subsections.

4.1 Data Model

The input data to our model is specified by means of the following atoms:

– Instances of `registration(R,P,SU,_,SP,_,_)` represent the registrations, where we outline only the main variables: with an id (`R`), a priority score (`P`)

Table 3. Anaesthetists availability for each specialty and in each day.

Days (D)	1		2		3		4		5	
Shifts (s)	1	2	3	4	5	6	7	8	9	10
Specialty (SP)	Anaesthetists									
1	6	6	6	6	6	6	6	6	6	6
2	4	4	4	4	4	4	4	4	4	4
3	4	4	4	4	4	4	4	4	4	4
4	2	2	2	2	2	2	2	2	2	2
5	4	4	4	4	4	4	4	4	4	4

, a surgery duration (SU) in minutes, and the id of the specialty (SP) it belongs to.
representing that the registration (`R`) with priority (`P`) is assigned with surgeon id (`SR`) and anaesthetist id (`AN`) to the operating room (`O`) during the shift (`S`) of the day (`D`) with a slot time (`ST`).

- Instances of `mss(O,S,SP,D)` link each operating room (`O`) to a shift (`S`) for each specialty (`SP`) and planning day (`D`), as established by the hospital Master Surgical Schedule (MSS).
- Instances of `surgeon(SR,SP,S)` represent the surgeons with an id (`SR`) for each specialty (`SP`) and shift (`S`).
- Instances of `an(AN,SP,S)` show the anaesthetists with an id (`AN`) for each specialty (`SP`) and shift (`S`).
- Instances of `time(S,ST)` show the time slots (`ST`) for each shift (`S`), i.e, each shift is divided it into `ST` time slots (which are always 30 in our setting) and each of them corresponds to a specific length expressed in minutes. As an example, each shift can be divided into 30 time slots where each time slot lasts 10 minutes. The length of each slot depends on the different scenarios. In particular, we consider 4 different variants (or scenario) of the problem, where the length of each time slot is set to 10 (scenario A), 20 (scenario B), 30 (scenario C), and 60 (scenario D) minutes, respectively. The choice of the length is injected in the encoding by specifying a constant, called `shift_duration`, that represents the total length (expressed in minutes) of the shift, e.g., `shift_duration` is set to 300 if each time slot lasts 10 minutes (30 time slots times 10 minutes), whereas it is set to 600 if each time slot lasts 60 minutes.
- Instances of `surgWT(SWT,SR,D)` represent the total work time in hours (`SWT`) for surgeons with id (`SR`) for each day (`D`).
- Instances of `anWT(AWT,AN,D)` represent the total work time in hours (`AWT`) for anaesthetists with id (`AN`) for each day (`D`).

The output is stored in an assignment represented by atom of the following form:

```
x(R,P,SR,AN,O,S,D,ST)
```

representing that the registration (`R`) with priority (`P`) is assigned with surgeon id (`SR`) and anaesthetist id (`AN`) to the operating room (`O`) during the shift (`S`) of the day (`D`) with a slot time (`ST`).

```
(r1) {x(R,P,SR,AN,O,S,D,ST): (ST+SU) <= shift_duration} :-
         registration(R,P,SU,_,SP,_,_),  mss(O,S,SP,D), surgeon(SR,SP,S), an(AN,SP,S),
         time(S,ST).
(r2) :- registration(R,_,_,_,_,_,_), #count{R,SR,AN,O,S,D,ST : x(R,P,SR,AN,O,S,D,ST)}>1.
(r3) :- x(R1,_,_,_,O,S,D,ST), x(R2,_,_,_,O,S,D,ST), R1 != R2.
(r4) :- #count{R:x(R,_,_,_,_,O,S,_,ST), registration(R,_,SU,_,_,_,_), T>=ST, T<ST+SU}>1,
         mss(O,S,_,_), time(S,T).
(r5) :- #count{R:x(R,_,SR,_,_,S,_,ST)} > 1, surgeon(SR,_,S), time(S,ST).
(r6) :- #count{R:x(R,_,SR,_,_,S,_,ST), registration(R,_,SU,_,_,_,_), T>=ST, T<ST+SU}>1,
         surgeon(SR,_,S), time(S,T).
(r7) :- #count{R:x(R,_,_,AN,_,S,_,ST)} > 1, an(AN,_,S), time(S,ST).
(r8) :- #count{R:x(R,_,_,AN,_,S,_,ST), registration(R,_,SU,_,_,_,_), T>=ST, T<ST+SU}>1,
         an(AN,_,S), time(S,T).
(r9) :- #sum{SU,R:x(R,_,SR,_,_,_,D,_), registration(R,_,SU,_,_,_,_)} > SWT,
         surgWT(SWT,SR,D).
(r10) :- #sum{SU,R:x(R,_,_,AN,_,_,D,_), registration(R,_,SU,_,_,_,_)} > AWT,
         anWT(AWT,AN,D).
(r11) :- #count{R:x(R,1,_,_,_,_,_,_)} < totRegsP1.
(r12) :~ M=#count{R:x(R,2,_,_,_,_,_,_)}, N=totRegsP2-M. [N@3].
(r13) :~ M=#count{R:x(R,3,_,_,_,_,_,_)}, N=totRegsP3-M. [N@2].
```

Fig. 1. ASP encoding of the ORS problem with surgical teams.

4.2 Encoding

The related ASP encoding is shown in Figure 1, and is described in this subsection. The encoding is based on the Guess&Check programming methodology.

Rule (r_1) guesses an assignment for the registrations, surgeons and anaesthetists to an OR in a given day, shift and with a time slot among the ones permitted by the MSS for the particular specialty the registrations, surgeons and anaesthetists belongs to, such that the registrations assigned with a slot time and surgery duration should be less than `shift_duration` of OR. We recall that `shift_duration` is a constant that depends on the different scenario considered (see Section 4.1).

After guessing an assignment for the registrations, the encoding presents constraints, related to general and work-time requirements, and to registrations of priority 1, to discard some unwanted assignments. All such constraints are explained in the following paragraphs, together with weak constraints for dealing with the optimization on registrations having priority 2 and 3.

General requirements. Rule (r_2) checks that same registration with same surgeon and anaesthetist should not be assigned more than once in different OR or shifts at the same time. Rule (r_3) ensures that different registrations cannot be assigned in the same OR and shift at the same time slot. Rule (r_4) checks that different registrations cannot be assigned at different times ,in the same OR and shift until the end time of the previous surgery to avoid time overlapping. Rule (r_5) shows that the same surgeon cannot be assigned at the same time slot in different ORs in the same shift. Rule (r_6) checks that the same surgeon cannot be assigned at different times in different OR in the same shift until the end of previous surgery. Rule (r_7) ensures that the same anaesthetist cannot be assigned at the

same time slot in different ORs in the same shift. Rule (r_8) checks that the same anaesthetist cannot be assigned to different times to different ORs in the same shift until the previous surgery is finished.

Work time requirements. Rules (r_9) and (r_{10}) are related to the work time of surgeons and anaesthetist to impose that the total number of registrations assigned to a surgeon cannot increase her/his total work hours in a day; similarly for anaesthetist the rule ensures that the total number of registrations assigned should not increase the total work hours in a day.

Priorities and optimization. Finally, since we model a priority based system, we want to be sure that every registration having priority 1 is assigned first, then we assign as much as possible the others, giving preference to registrations having priority 2 over those having priority 3. This is accomplished through constraint (r_{11}) for priority 1 and the weak constraints (r_{12}) and (r_{13}) for priority 2 and 3, where `totRegsP1`, `totRegsP2` and `totRegsP3` are constants representing the total number of registrations having priority 1, 2 and 3, respectively.

5 Experimental Results

5.1 Slot Interval Analysis

This section reports about the results of an empirical analysis of the ORS problem with surgical teams. The ASP system used is CLINGO [17]. We have performed different slot interval analysis on the performance of our encoding and employed ASP solver. As described in Section 4.1, the dimension of the slot interval determines the time sensitivity of our encoding, and four different scenarios have been considered: A, B, C, and D for slot interval of 10, 20, 30, and 60 minutes, respectively.

5.2 Benchmarks

For each scenario, the characteristics of the tests are as follows:

- 4 different benchmarks, with a planning period of 1, 2, 3 and 5 working days;
- For each benchmark the total number of randomly generated registrations were 350 for 5 days, 210 for 3 days, 140 for 2 days and 70 for 1 days;
- 5 specialties;
- 20 surgeons assigned to the 5 specialties;
- 20 anaesthetists assigned to the 5 specialties;
- 4 hours of work time in a day for each surgeon;
- 6 hours of work time in a day for each anaesthetist;
- 10 ORs distributed among the specialties;
- 5 hours morning and afternoon shifts for each OR summing up to 500, 300, 200 and 100 hours of OR available time for the four benchmarks;

Table 4 shows the distribution of the total number of randomly generated registrations for each benchmark of 5, 3, 2 and 1 day, for each specialty, together with the distribution of ORs for each specialty.

Table 4. Total number of randomly generated registrations for each benchmark.

Specialty	Registrations				ORs
	5-day	3-day	2-day	1-day	
1	80	48	32	16	3
2	70	42	28	14	2
3	70	42	28	14	2
4	60	36	24	12	1
5	70	42	28	14	2
Total	350	210	140	70	10

5.3 Results

Experiments have been run on a HP 630 Notebook with Intel(R) Core(TM) i3 CPU M380@2.53GHz. Results of the experiments are reported for scenario A in Table 5, for scenario B in Table 6, for scenario C in Table 7 and for scenario D in Table 8, respectively.
Each benchmark was tested 10 times with different randomly generated inputs. A time limit of 300 seconds was set for each experiment. In each table averages for 10 instances for each benchmark are reported. The first three columns show the number of assigned registrations out of the generated ones for each priority P1, P2 and P3, while the last three columns show a measure of the total time occupied by the assigned registrations as a percentage of the total OR time available (indicated as OR time Eff in the tables) and the total percentage of surgeons and anesthetists working time (indicated respectively as Surg WT Eff and Anest WT Eff in the tables).
As we can see in scenario A (Table 5) with slot interval of 10 minutes we obtain results only for schedules up to 3 days, while in the case of the 5-day benchmark the computation time exceeds our time limit of 300 seconds on all instances.
Scenario B (Table 6) details the scheduling results with slot intervals of 20 minutes. It can be seen that OR efficiency is 75% while the Surgeons and Anesthetists WT efficiency remain greater than 90% and 60%, respectively, for all benchmarks in this scenario.
In scenario C (Table 7) with a slot interval of 30 minutes, the OR efficiency is around 76% while the Surgeons and Anesthetists WT efficiency are enhanced up to 95% and 63% for all benchmarks respectively.
In scenario D (Table 8) with a slot interval of 60 minutes, OR efficiency is almost 79% while the Surgeons WT efficiency is further enhanced to more than 95%, and Anesthetists WT efficiency is up to 65%.
In all the evaluated benchmarks for different scenarios we can observe that the OR efficiency and the anesthetists efficiency are limited by the fact that we reached the ceiling of the surgeons maximum working hours. Considering that in our setup we had one anesthetist for each surgeon, the ratio between anesthetist and surgeon efficiencies is the same that the ratio between their maximum working time of the surgeons and the anesthetists, i.e., 2/3. In a real application, this would be a useful information for the hospital manager to quantify the

Table 5. Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario A.

Bench.	P1	P2	P3	Total	OR time Eff	Surg WT Eff	Anest WT Eff
5 days	-	-	-	-	-	-	-
3 days	43.1/43.1	53.0/81.6	26.2/85.3	122.3/210.0	73.5%	91.8%	61.2%
2 days	29.9/29.9	37.0/54.7	17.7/55.4	84.6/140.0	74.7%	93.4%	62.3%
1 day	13.4/13.4	22.8/28.0	8.9/28.6	46.1/70.0	75.6%	94.5%	62.9%

Table 6. Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario B.

Bench.	P1	P2	P3	Total	OR time Eff	Surg WT Eff	Anest WT Eff
5 days	71.2/71.2	99.4/140.1	38.3/138.7	208.9/350.0	75.1%	93.8%	62.5%
3 days	40.9/40.9	61.6/85.3	25.2/83.8	127.7/210.0	75.3%	94.1%	62.8%
2 days	28.2/28.2	41.1/56.1	14.9/55.7	84.2/140.0	75.0%	93.7%	62.5%
1 day	12.5/12.5	23.1/29.7	8.9/27.8	43.5/70.0	75.5%	94.4%	62.9%

excess of anesthetists and reorganize their numbers or their working times.

Overall we obtained satisfying results but for the 5-day schedule length for the more fine-grained time interval of Scenario A. In order to further investigate the issue, we moved on a different dimension and tested the Scenario A configuration with half the number of registrations (35 instead of 70 for each planning day), surgeons (10 instead of 20), anesthetists (10 instead of 20) and ORs (5 instead of 10). With these numbers we can reach acceptable solutions after 60 seconds of computation time (see Table 9) for every benchmark, including the 5-day one. In the previous analysis the issue was caused by the excessive computation required, especially in the grounding phase, by such a number of registrations, surgeons and anesthetists with the finer time sensitivity of 10 minutes.

6 Related Work

In this section we will discuss some relevant works related to this research. Meskens et al [30] considered the surgical teams in the computation of an OR schedule, and developed a model using Constraint Programming (CP) with multiple constraints such as availability, staff preferences and affinities among surgical teams. They optimize the use of ORs by minimizing makespan and maximizing affinities among surgical team members. The effectiveness of their proposed method for improving surgical cases was evaluated using real data from an hospital. Hamid et al [29] incorporated the decision-making styles (DMS) of the surgical team to improve the compatibility level by considering constraints such as the availability of material resources, priorities of patients, and availability, skills, and competencies of the surgical team. They developed a multi-objective

Table 7. Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario C.

Bench.	P1	P2	P3	Total	OR time Eff	Surg WT Eff	Anest WT Eff
5 days	71.9/71.9	99.0/139.8	44.1/138.3	215.0/350.0	76.0%	95.2%	63.3%
3 days	41.7/41.7	66.9/84.8	21.6/83.5	130.2/210.0	76.1%	95.1%	63.5%
2 days	27.9/27.9	42.7/53.8	16.9/58.3	87.5/140.0	76.2%	95.2%	63.5%
1 day	14.2/14.2	23.0/29.4	6.7/26.4	43.9/70.0	76.2%	95.1%	63.5%

Table 8. Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario D.

Bench.	P1	P2	P3	Total	OR time Eff	Surg WT Eff	Anest WT Eff
5 days	68.7/68.7	109.6/143.8	46.9/137.5	224.8/350.0	79.0%	98.8%	65.8%
3 days	41.8/41.8	65.1/ 81.9	25.5/86.3	132.4/210.0	78.7%	98.4%	65.6%
2 days	27.5/27.5	46.5/ 54.5	14.6/58.0	87.7/140.0	79.2%	98.9%	65.9%
1 day	13.3/13.3	23.1/27.5	8.3/29.2	44.7/70.0	78.3%	97.8%	65.2%

mathematical model to schedule surgeries. Two metaheuristics, namely Non-dominated Sorting Genetic Algorithm and Multi-Objective Particle Swarm Optimization, were developed to find pareto-optimal solutions. Xiang et al [34] proposed an Ant Colony Optimization (ACO) approach to surgical scheduling taking into account all resources in the entire process of a surgery. The problem was represented as an extended multi-resource constrained flexible job shop scheduling problem, which was solved using a two-level hierarchical graph to integrate sequencing job and allocating resources. To evaluate the efficiency of ACO, a Discrete Event System (DES) model of an OR system was developed in the simulation platform SIMIO. Monteiro et al. [31] developed a comprehensive multi-objective mathematical model using epsilon-constraint method coupled to the CPLEX solver. Vijayakumar et al. [33] used Mixed Integer Programming (MIP) model for multi-day, multi-resource, patient-priority-based surgery scheduling. First Fit Decreasing algorithm was developed. From a solution time perspective, their model took long hours or in most cases was unable to optimally solve the problem. Belkhamsa et al. [7] proposed two meta heuristics, an Iterative Local Search (ILS) approach and Hybrid Genetic Algorithm (HGA) to solve a daily surgery scheduling problem. Zhou et al. [35] developed an In-

Table 9. Averages of the results for 5, 3, 2 and 1 day benchmarks with slot interval 10 minutes and reduced number of registrations.

Bench.	P1	P2	P3	Total	OR time Eff	Surg WT Eff	Anest WT Eff
5 days	35.5/35.5	52.3/68.3	17.0/71.2	104.8/175.0	74.4%	93.1%	62.0%
3 days	19.0/19.0	31.9/41.2	12.9/40.8	63.8/105.0	74.4%	93.0%	62.0%
2 days	14.2/14.2	20.8/28.6	6.9/26.3	40.9/70.0	73.0%	91.3%	60.8%
1 day	5.5/5.5	10.4/15.1	3.6/14.4	19.5/35.0	70.7%	89.5%	58.9%

teger Programming model for optimal surgery schedule of assigning patients to different resources in any surgical stage. They used Lagrangian Relaxation algorithm and solved the subproblem by using branch and bound. They verified their model using real data instances from a hospital. A common issue with all such solutions seem to be computation time and scalability.

About, instead, other scheduling problems in which ASP have been proficiently employed: Nurse Scheduling Problem [3, 4, 15], where the goal is to create a scheduling for nurses working in hospitals; Team Building Problem [32], where the goal is to allocate the available personnel of a seaport for serving the incoming ships; the Conference Paper Assignment Problem [5], which deals with the problem of assigning reviewers in the PC to submitted conference papers; and scheduling production materials between storage locations and assembly station [20].

7 Conclusions

In this paper we employed ASP for solving ORs problem with surgical teams. The results of our experiments confirm that ASP is a suitable methodology for addressing planning and scheduling problems in health-care system. We presented the results of an experimental analysis on several directions to check scalability of our solution in terms of efficiency, considering shift duration, surgeons and anesthetist working hours. This solution achieved satisfied ORs, surgeons' and anaesthetists' efficiency also for the planning length of 5 days. As a future work we would like first to investigate other parameters and configurations, e.g., with less anesthetists for which we expect that the efficiency of the anesthetists would increase. We also want to integrate the extension of the ORS model with beds management with the one presented in this paper, in order to have a more complete unified solution. We also plan to compare to alternative methods, assuming this is possible (i.e., availability of alternative solutions), and viable (i.e., very same problem solved). Finally, through results are satisfying, we plan to work also on improving performance by both evaluating more solvers, e.g., WASP [2], other than Clingo actually used, and employing SAT techniques (e.g., [28, 26, 27, 25, 11]), given the strong existing relation between ASP and SAT [24, 23].

References

1. Abedini, A., Ye, H., Li, W.: Operating room planning under surgery type and priority constraints. Procedia Manufacturing **5**, 15–25 (2016)
2. Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F.: Evaluation of disjunctive programs in WASP. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) LPNMR. Lecture Notes in Computer Science, 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. Amendola, G., Dodaro, C., Leone, N., Ricca, F.: On the application of answer set programming to the conference paper assignment problem. In: AI*IA. Lecture Notes in Computer Science, vol. 10037, pp. 164–178. Springer (2016)
6. Aringhieri, R., Landa, P., Soriano, P., Tànfani, E., Testi, A.: A two level metaheuristic for the operating room scheduling and assignment problem. Computers & Operations Research **54**, 21–34 (2015)
7. Belkhamsa, M., Jarboui, B., Masmoudi, M.: Two metaheuristics for solving no-wait operating room surgery scheduling problem under various resource constraints. Comput. Ind. Eng. **126**, 494–506 (2018)
8. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011)
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 and Practice of Logic Programming **20**(2), 294–309 (2020)
10. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: The design of the fifth answer set programming competition. CoRR **abs/1405.3710** (2014), `http://arxiv.org/abs/1405.3710`
11. Di Rosa, E., Giunchiglia, E., Maratea, M.: A new approach for solving satisfiability problems with qualitative preferences. In: Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N.M. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 178, pp. 510–514. IOS Press (2008)
12. 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.) RuleML+RR. Lecture Notes in Computer Science, vol. 11784, pp. 67–81. Springer (2019)
13. 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)
14. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: An ASP-based framework for operating room scheduling. Intelligenza Artificiale **13**(1), 63–77 (2019)
15. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: LPNMR. LNCS, vol. 10377, pp. 301–307. Springer (2017)
16. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1), 278–298 (2011)
17. 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)
18. Gebser, M., Maratea, M., Ricca, F.: The design of the seventh answer set programming competition. In: Balduccini, M., Janhunen, T. (eds.) LPNMR. Lecture Notes in Computer Science, vol. 10377, pp. 3–9. Springer (2017)
19. Gebser, M., Maratea, M., Ricca, F.: The seventh answer set programming competition: Design and results. Theory Pract. Log. Program. **20**(2), 176–204 (2020)
20. Gebser, M., Obermeier, P., Schaub, T., Ratsch-Heitmann, M., Runge, M.: Routing driverless transport vehicles in car assembly with answer set programming. Theory Pract. Log. Program. **18**(3-4), 520–534 (2018)
21. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the Fifth International Conference and Symposium , Seattle, Washington, August 15-19, 1988 (2 Volumes). pp. 1070–1080. MIT Press (1988)

22. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Comput. **9**(3/4), 365–386 (1991)
23. Giunchiglia, E., Leone, N., Maratea, M.: On the relation among answer set solvers. Ann. Math. Artif. Intell. **53**(1-4), 169–204 (2008)
24. Giunchiglia, E., Maratea, M.: On the Relation Between Answer Set and SAT Procedures (or, Between cmodels and smodels). In: ICLP. LNCS, vol. 3668, pp. 37–51. Springer (2005)
25. Giunchiglia, E., Maratea, M.: Solving optimization problems with DLL. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 141, pp. 377–381. IOS Press (2006)
26. Giunchiglia, E., Maratea, M., Tacchella, A.: Dependent and independent variables in propositional satisfiability. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA. Lecture Notes in Computer Science, vol. 2424, pp. 296–307. Springer (2002)
27. Giunchiglia, E., Maratea, M., Tacchella, A.: (In)Effectiveness of look-ahead techniques in a modern SAT solver. In: Rossi, F. (ed.) CP. Lecture Notes in Computer Science, vol. 2833, pp. 842–846. Springer (2003)
28. Giunchiglia, E., Maratea, M., Tacchella, A., Zambonin, D.: Evaluating search heuristics and optimization techniques in propositional satisfiability. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) IJCAR. vol. 2083, pp. 347–363. Springer (2001)
29. Hamid, M., Nasiri, M.M., Werner, F., Sheikhahmadi, F., Zhalechian, M.: Operating room scheduling by considering the decision-making styles of surgical team members: A comprehensive approach. Computers & Operation Research **108**, 166–181 (2019)
30. Meskens, N., Duvivier, D., Hanset, A.: Multi-objective operating room scheduling considering desiderata of the surgical team. Decis. Support Syst. **55**(2), 650–659 (2013). https://doi.org/10.1016/j.dss.2012.10.019, `https://doi.org/10.1016/j.dss.2012.10.019`
31. Monteiro, T., Meskens, N., Wang, T.: Surgical scheduling with antagonistic human resource objectives. International Journal of Production Research **53**(24), 7434–7449 (2015)
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 and Practice of Logic Programming **12**(3), 361–381 (2012)
33. Vijayakumar, B., Parikh, P.J., Scott, R., Barnes, A., Gallimore, J.: A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. European Journal of Operation Research **224**(3), 583–591 (2013)
34. Xiang, W., Yin, J., Lim, G.: An ant colony optimization approach for solving an operating room surgery scheduling problem. Comput. Ind. Eng. **85**, 335–345 (2015)
35. Zhou, B., Yin, M., Lu, Z.: An improved lagrangian relaxation heuristic for the scheduling problem of operating theatres. Comput. Ind. Eng. **101**, 490–503 (2016)