An ASP-based Approach to Scheduling Pre-Operative Assessment Clinic

Simone Caruso[1] [0000−0002−2724−4342], Giuseppe Galatà[2] [0000−0002−1948−4469], Marco Maratea[1] [0000−0002−9034−2527]⋆, Marco Mochi[1,2] [0000−0002−5849−3667], and Ivan Porro[2] [0000−0002−0601−8071]

[1] University of Genoa, Genova, Italy, {name.surname}@unige.it
[2] SurgiQ srl, Genova, Italy, {name.surname}@surgiq.com

Abstract. The problem of scheduling Pre-Operative Assessment Clinic (PAC) consists of assigning patients to a day for the exams needed before a surgical procedure, taking into account patients with different priority levels, due dates, and operators availability. Realizing a satisfying schedule is of upmost importance for a clinic, since delay in PAC can cause delay in the subsequent phases, causing a decrease in patients' satisfaction. In this paper, we divide the problem in two sub-problems: In the first sub-problem patients are assigned to a day taking into account a default list of exams; then, in the second sub-problem, having the actual list of exams needed by each patient, we use the results of the first sub-problem to assign a starting time to each exam. We first present a mathematical formulation for both problems. Then, we present solutions based on Answer Set Programming (ASP): The first solution is a genuine ASP encoding of the sub-problems, while the second introduces domain-specific optimizations. Experiments show that both solutions provide satisfying results in short time, while the second is able to prove optimality faster.

Keywords: Healthcare · Pre-Operative Assessment Clinic Scheduling · Answer Set Programming

1 Introduction

The Pre-Operative Assessment Clinic (PAC) scheduling problem is the task of assigning patients to a day, in which the patient will be examined and prepared to a surgical operation, taking in account patients with different priority levels, due dates, and operators availability. The PAC consists of several exams needed by patients to ensure they are well prepared for their operation. This allows patients to stay at home until the morning of the surgery, instead of being admitted to the hospital one or two days before the scheduled operation; moreover, reducing waiting time between the exams increase patient satisfaction [34] and avoid the cancellation of the surgery [22].

The problem is divided into two sub-problems [19]: In the first sub-problem, patients are assigned to a day taking into account a default list of exams, and the solution has to schedule patients before their due date and prioritize the assignments to patients

⋆ Corresponding author.

with higher priority. In the second sub-problem, the scheduler assigns a starting time to each exam needed by the patients, considering the available operators and the duration of the exams. A proper solution to the PAC scheduling problem is vital to improve the degree of patients' satisfaction and to reduce surgical complications. Complex combinatorial problems, possibly involving optimizations, such as the PAC 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 ([2, 10, 25, 27]), has been successfully employed for solving hard combinatorial problems in several research areas, and it has been also employed to solve many scheduling problems [35, 1, 15, 16, 5, 18, 4], also in industrial contexts (see, e.g., [20, 21, 37, 3] for detailed descriptions of ASP applications).

In this paper, we first present a mathematical formulation of both sub-problems: In the solution of the first sub-problem, the scheduler minimizes the number of unassigned patients. Then, we propose a solution to the second sub-problem, using as input of the problem the result of the first one and minimizing the time each patient stays at the hospital. We then apply ASP to solve the PAC scheduling problem, by presenting two ASP encodings for the sub-problems, and run an experimental analysis on PAC benchmarks with realistic sizes and parameters inspired from data seen in literature, varying the number of patients to schedule and the available operators. Overall, results using the state-of-the-art ASP solver CLINGO [24] show that ASP is a suitable solving methodology also for the PAC scheduling problem, even if often it takes a considerable amount of time to prove the optimality of the solution of the second sub-problem. Thus, we finally apply domain-specific optimizations, still expressed as ASP rules, which consider the time slots in which exams can be effectively performed by patients, that help to reduce the overall running time significantly.

The paper is then structured as follows. Sections 2 and 3 present an informal description of the problem and a precise, mathematical formulation, respectively. Then, Section 4 shows our ASP encodings for both phases, whose experimental evaluation is presented in Section 5. Domain-specific optimizations are introduced and evaluated in Section 6. The paper ends by discussing related work and conclusions in Section7 and 8, respectively.

2 Problem Description

Since the schedule of the PAC day must be scheduled as soon as possible, this problem is typically divided in two phases: In the first phase, since we deal with this sub-problem before the actual PAC day, the clinics do not know which exams each patient will require, thus, as typically done in hospitals, we consider that each patient requires a default list of exams according to his specialty, i.e. the lists of exams are equal for patients requiring the same specialty but differ among specialties, and the scheduler assigns the day of PAC without considering the starting time of the exams. Thus, in the first sub-problem the solution assigns patients overestimating the duration and the number of exams needed. In particular, all the optional exams, such as exams required by smokers or patients with diabetes, are assigned to all the patients in the first phase. The overestimation is important in order to have a second sub-problem which is not unsatisfiable.

Then, when the operation day is closer, the hospital knows exactly the exams needed by each patient and can assign the starting time of each exam. Going in more details, the first sub-problem consists of scheduling appointments in a range of days for patients requiring surgical operation. Each patient is linked to a due date, a target day, and a priority level: The due date is the maximum day in which (s)he can be assigned, the target day is the optimal day in which schedule the appointment, while the solution prioritizes patients with higher priority level. There are several exam areas, corresponding to the locations in which patients will be examined. Each exam area needs operators to be activated and has a limited time of usage. Each operator can activate three different exam areas, but they can be assigned to just one exam area for each day. The solution must assign the operators to the exam areas, to activate them, and the day of PAC to patients, ensuring that the total time of usage of each exam location is lower than its limit. Since in this first sub-problem the list of exams needed by patients is not the final list, i.e. just the first and the last exam are the same for every patient, and in the second sub-problem some exams could be added, the solution schedules patients leaving some unused time to each exam area. An optimal solution minimizes the number of unassigned patients, giving priority to patients with higher priority levels, and ties are broken by minimizing the difference between the day assigned and the target day of each patient, giving again precedence to patients with higher priority.

In the second sub-problem, patients are linked to their real exams, so the solution has to assign the starting time of each exam, having the first sub-problem already assigned the day. The input consists of registrations, exams needed by patients and the exam areas activated. Exams are ordered, so the solution must assign the starting time of each exam respecting their order and their duration, by considering that each exam area can be used by one patient at a time. Finally, the solution minimizes the difference between the starting time of the first exam and the last exam of each patient.

3 Formalization of the PAC Scheduling problem

In this section, we provide a mathematical formulation of the two sub-problems.

Definition 1 (first PAC sub-problem). *Let*

- R *be a finite set of registrations;*
- $E = \{e_1, \ldots, e_m\}$ *be a set of m exams;*
- $EL = \{el_1, \ldots, el_n\}$ *be a set of n exam locations;*
- O *be a finite set of operators;*
- $D = \{t : t \in [1..14]\}$ *be the set of all days;*
- $\delta : R \mapsto \{1,2,3,4\}$ *be a function associating a registration to a priority;*
- $\rho : R \times E \mapsto \mathbb{N}$ *be a function associating a registration and an exam to a duration such that for a registration r and an exam e if* $\rho(r,e) > 0$ *then the registration r requires the exam e;*
- $\varepsilon : D \mapsto \mathbb{N}$ *be a function associating a day to the number of registration assigned in the day d;*
- $\zeta : \mathbb{N} \times oe2 \mapsto \mathbb{N}$ *be a function such that* $\zeta(n) = (ts * \varepsilon(n)) - (oe2 * \varepsilon * (\varepsilon - 1))$*;*
- $\omega : R \mapsto \mathbb{D}$ *be a function associating a registration to a due date;*

- $\lambda : R \mapsto \mathbb{D}$ *be a function associating a registration to a target day;*
- $\sigma : E \mapsto \mathbb{EL}$ *be a function associating an exam to the exam location;*
- $\Delta : EL \times D \mapsto \mathbb{N}$ *be a function associating an exam location to the maximum sum of exams lengths assignable to the exam location;*
- $\tau : EL \times D \mapsto \mathbb{N}$ *be a function associating an exam location and a day to the required number of operators to be activated, such that* $\tau(el,d) = n$ *if the exam location el in the day d requires n operators to be activated;*
- $\theta : O \times EL \times D \mapsto \{0,1\}$ *be a function such that* $\theta(o,el,d) = 1$ *if the operator o is assigned to the exam location el in the day d, and* 0 *otherwise;*
- $oe1$ *be a constant used to decrease the maximum sum of exam lengths assignable to the exam locations;*
- $oe2$ *be a constant used to decrease the maximum sum of exams duration assignable to a day;*
- ts *be a constant that is equal to the number of time slots.*

Let $x : R \times D \mapsto \{0,1\}$ *be a function such that* $x(r,d) = 1$ *if the registration r is assigned to the day d, and* 0 *otherwise. Moreover, for a given x, let* $A_x = \{(r,d) : r \in R, d \in D, x(r,d) = 1\}$.

Then, given sets R, E, EL, O,D, and functions δ, ρ, ε, ω, λ, σ, Δ, τ, θ, *the first PAC sub-problem is defined as the problem of finding a schedule x, such that*

(c_1) $|\{d : x(r,d) = 1\}| \leq 1 \quad \forall r \in R, d \leq \omega(r)$;
(c_2) $|\{d : x(r,d) = 1\}| = 0 \quad \forall r \in R, d > \omega(r)$;
(c_3) $|\{o : \theta(o,el,d) = 1\}| = \tau(el,d) \quad \forall (r,d) \in Ax, \rho(r,e) > 0, el = \sigma(e)$;
(c_4) $|\{el : \theta(o,el,d)\}| \leq 1 \quad \forall o \in O, \forall d \in D$;
(c_5) $\sum_{x(r,d)=1, \sigma(e)=el} \rho(r,e) \leq \Delta(el,d)/oe1 \quad \forall el \in EL, t \in T$;
(c_6) $\sum_{x(r,d)=1, \forall e \in E} \rho(r,e) \leq \zeta(\varepsilon(d), oe2) \quad \forall d \in D$;

Condition (c_1) ensures that each registration is assigned at most one time in the days before the due date associated to the registration. Condition (c_2) ensures that each registration is not assigned in a day after the due date associated to the registration. Condition (c_3) ensures that for each exam location used by at least one registration, the required number of operators are assigned Condition (c_4) ensures that each operator is assigned to at most one exam location in each day. Condition (c_5) ensures that the sum of all the durations of the exams associated to an exam location in a day is less or equal to the maximum time assignable to that exam location divided by the value $oe1$. Condition (c_6) ensures that the sum of all exam durations in a day is less than a value obtained with the number of registrations assigned in the day and the value oe2.

Definition 2 (Unassigned registrations). *Given a solution x, let* $U_x^{pr} = \{r : r \in R, \delta(r) = pr, r \notin A_x\}$. *Intuitively,* U_x^{pr} *represents the set of registrations of priority pr that were not assigned to any day.*

Definition 3 (Distance target day). *Given a solution x, let* $t_x^{pr} = \sum_{x(r,d) \in A_x, \delta(r)=pr} | d - \lambda(r) |$. *Intuitively,* t^{pr_x} *represents the sum of the distance between the day assigned to the registrations of priority pr and the target day associated.*

Definition 4. *A solution x is said to dominate a solution x' if* $|U_x^{pr}| < |U_{x'}^{pr}|$ *for the biggest pr for which* $|U_x^{pr}| \neq |U_{x'}^{pr}|$ *or if* $|U_x^{pr}| = |U_{x'}^{pr}|$ *for all the pr and* $|t_x^{pr}| < |t_{x'}^{pr}|$ *for the biggest pr for which* $|t_x^{pr}| \neq |t_{x'}^{pr}|$.

Definition 5 (second PAC sub-problem). *Let*

- $T = \{t : t \in [1..ts]\}$ *be the set of all time slots;*
- $\beta : R \times E \mapsto \mathbb{N}$ *be a function associating a registration and an exam to a value corresponding to the order in which the exam must be assigned.*
- $\gamma : EL \mapsto \mathbb{N}$ *be a function associating an exam location to the starting time of the exam location;*
- $\xi : EL \mapsto \mathbb{N}$ *be a function associating an exam location to the ending time of the exam location;*
- $\mu : EL \mapsto \mathbb{N}$ *be a function associating an exam location and a day to the maximum number of registration that can be assigned concurrently.*

Let $x : R \times E \times EL \times T \mapsto \{0,1\}$ *be a function such that* $x(r,e,el,t) = 1$ *if the registration r and the exam e are assigned to the exam location el in the time slot t, and* 0 *otherwise. Moreover, for a given x let* $A_x = \{(r,e,el,t) : r \in R, e \in E, el \in EL, d \in D, x(r,e,el,t) = 1\}$.

Then, given sets R, E, EL, T, and functions ρ, β, γ, ξ, μ, *the second PAC sub-problem is defined as the problem of finding a schedule x, such that*

(c_7) $|\{t : x(r,e,el,t) = 1, t \in T\}|=1 \quad \forall e \in E, \forall r \in R, \rho(r,e) > 0, \sigma(e) = el;$
(c_8) $\gamma(el) \leq t \leq \xi(el) - \beta(r,e) \quad \forall (r,e,el,t) \in A_x.$
(c_9) $x(r,e,el,t) = 0 \quad \forall (r,e',el,t') \in A_x, \forall e \in E, \rho(r,e') = d, \forall t \in T, t' \leq t < t' + d;$
(c_{10}) $t > t' \quad \forall (r,e,el,t) \in A_x, (r,e',el,t') \in A_x, \beta(r,e) > \beta(r,e');$
(c_{11}) $|\{r : x(r,e,el,t) = 1, r \in R, e \in E\}| = \mu(el) \quad \forall el \in EL, t \in T;$

Condition (c_7) ensures that each exam is assigned exactly once. Condition (c_8) ensures that each exam is assigned after the starting time of the required exam location and before the closing time of the required exam location minus the duration of the exam. Condition (c_9) ensures that for each registration each exam is assigned after that the exam before is ended. Condition (c_{10}) ensures that each exam is assigned after the exams with lower order. Condition (c_{11}) ensures that the number of exams assigned to a location is lower than the maximum availability for each location in any time slot.

Definition 6 (Time in hospital). *Given a solution x, let*

$$m_x = \sum_{r \in R, x(r,0,el,t) \in A_x, x(r,23,el,t') \in A_x} t' + \rho(r,23) - t.$$

Intuitively, m_x *represents the sum of the difference between the ending time of the last exam and the starting time of the first exam of each registration.*

Definition 7. *A solution x is said to dominate a solution x' if* $m_x < m_{x'}$.

4 ASP Encoding

In this section we present the ASP encoding for the two sub-problems, in two separate sub-sections.

```
{x(RID,PR,TOTDUR,DAY) : day(DAY), DAY < DUEDATE} 1 :- reg(RID,PR,TARGET,TOTDUR,DUEDATE).
:- x(RID,_,_,DAY), exam(RID,FORNID,_), not examLoc(FORNID,_,_,DAY,_).
res(RID,FORNID,DAY,DUR) :- x(RID,PR,_,DAY), exam(RID,FORNID,DUR).
:- N1 = #count{FORNID: res(RID,FORNID,_,_)},N2 = #count{FORNID: exam(RID,FORNID,_,_)},
     x(RID,_,_,_), N1 != N2.
:- #sum{DUR, RID: res(RID,FORNID,DAY,DUR)} > NHOURS/oe1, examLoc(FORNID,_,NHOURS,DAY,N).
:- #sum{TOTDUR, RID: x(RID,_,TOTDUR,DAY)} = M, #count{RID: x(RID,_,_,DAY)} = N, day(DAY), M
     > ((ts*N)-(oe2*N*(N+1))), N>1.
{operator(ID, FORNID, DAY) : operators(ID, FORNID, DAY)} == NOP :- examLoc(FORNID, NOP, _,
     DAY,_), res(REGID, FORNID, DAY, _).
:- operator(ID,FORNID1,DAY), operator(ID,FORNID2,DAY), FORNID1 < FORNID2.
unassignedP1(N) :- M = #count {RID: x(RID,1,_,_)}, N = totRegsP1 - M.
unassignedP2(N) :- M = #count {RID: x(RID,2,_,_)}, N = totRegsP2 - M.
unassignedP3(N) :- M = #count {RID: x(RID,3,_,_)}, N = totRegsP3 - M.
unassignedP4(N) :- M = #count {RID: x(RID,4,_,_)}, N = totRegsP4 - M.
:~ unassignedP1(N). [N@8]
:~ unassignedP2(N). [N@7]
:~ unassignedP3(N). [N@6]
:~ unassignedP4(N). [N@5]
:~ x(RID,1,_,DAY), reg(RID,_,TARGET,_,_). [|DAY-TARGET|@4,RID]
:~ x(RID,2,_,DAY), reg(RID,_,TARGET,_,_). [|DAY-TARGET|@3,RID]
:~ x(RID,3,_,DAY), reg(RID,_,TARGET,_,_). [|DAY-TARGET|@2,RID]
:~ x(RID,4,_,DAY), reg(RID,_,TARGET,_,_). [|DAY-TARGET|@1,RID]
```

Fig. 1. ASP encoding of the first sub-problem

4.1 ASP Encoding for the first PAC sub-problem

We assume the reader is familiar with syntax and semantics of ASP. Starting from the specifications in the previous section, here we present the ASP encoding for the first sub-problem, based on the input language of CLINGO [23]. For details about syntax and semantics of ASP programs we refer the reader to [9].

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

- Instances of `reg(RID,PR,TARGET,TOTDUR,DUEDATE)` represent the registrations, characterized by an id (`RID`), the priority level (`PR`), the ideal day in which the patient should be assigned (`TARGET`), the sum of the durations of the exams needed by the patient (`TOTDUR`), and the due date (`DUEDATE`).
- Instances of `exam(RID,FORNID,DUR)` represent the exams needed by the patients identified by an id (`RID`), the exam area (`FORNID`), and the duration (`DUR`).
- Instances of `examLoc(FORNID,NOP,NHOURS,DAY,N)` represent the exam areas, characterized by an id (`FORNID`), which requires `NOP` operators to be activated, which is active for a certain time (`NHOURS`) in a day (`DAY`), and can be concurrently assigned up to `N` patients.
- Instances of `operators(ID,FORNID,DAY)` represent the operators, characterized by an id (`ID`), who can be assigned to the exam ares (`FORNID`) in a day `(DAY)`.
- Instances of `day(DAY)` represent the available days.

The output is an assignment represented by an atom of the form `x(RID,PR,TOTDUR,DAY)`, where the intuitive meaning is that the exams of registration with id `RID` and priority level `PR` is assigned to the day `DAY` and has a total duration of exams equal to `TOTDUR`.

```
1 {x(RID,FORNID,ST,ST+DUR,DAY) : examLoc(FORNID,DAY,FORNST,FORNET,_), time(ST), ST >= FORNST,
      ST <= FORNET-DUR} = 1 :- reg(RID,DAY), esame(RID,FORNID,DUR).
2 :- x(RID,FORNID1,ST1,_,_), x(RID,FORNID2,ST2,_,_), phase(FORNID1,ORD1), phase(FORNID2,ORD2),
      ORD2 < ORD1, ST1 < ST2.
3 :- #count{FORNID: x(RID,FORNID,ST,ET,DAY), T >= ST, T < ET} > 1, reg(RID,DAY), time(T).
4 :- #count{FORNID: x(RID,FORNID,ST,ET,DAY), T >= ST, T < ET} > N, examLoc(FORNID,DAY,_,_,N),
      time(T).
5 :~ reg(RID,_), x(RID,0,ST,_,_), x(RID,23,_,ET,_). [ET-ST@1, RID]
```

Fig. 2. ASP encoding of the second sub-problem

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

Rule r_1 assigns registrations to a day. The assignment is made assigning a day that is before the due date. Rule r_2 checks that every registration is assigned to a day with all the exams area needed to be activated. Rule r_3 derives an auxiliary atom that is used later in other rules. In particular, the new atom is used to get the duration of the visit for each patient and for each exam area. Then, rule r_4 checks that the number of needed exams and the number of `res` atoms created are the same. Rule r_5 is used to ensure that each exam area is used for a total amount of time that is lower than its limit divided by the oe1 constant, in order to overestimate the required time for the visits. Rule r_6 is used to be sure to not assign too many patients in the first sub-problem to a particular day. So, it overestimates the time needed by each patient, the degree of the overestimation can be changed by using different oe2 values. Rule r_7 assigns operators to the required exam areas. Rule r_8 checks that each operator is assigned to just one exam area in every day. Rules from r_9 to r_{12} are needed to derive auxiliary atoms that are used later on in optimization. In particular, they are used to count how many patients with different priorities are not assigned to a day. Weak constraints from r_{13} and r_{16} are used to minimize the number of unassigned registrations according to their priority. Finally, weak constraints from r_{17} and r_{20} minimize the difference between the assigned and target day of each patient, giving precedence to higher priorities.

4.2 ASP Encoding for the second PAC sub-problem

Data Model. The input data is the same of the first sub-problem for the atoms `exam` and `time`, while other atoms are changed:

- Instances of `reg(RID,DAY)` represent the registrations, characterized by an id (`RID`) assigned to a day (`DAY`).
- Instances of `examLoc(FORNID,DAY,FORNST,FORNET,N)` represent the exam areas, characterized by an id (`FORNID`), which in a day (`DAY`) has a starting time and closing time respectively equals to `FORNST` and `FORNET`, which is active for a certain value of time (`NHOURS`) in a day (`DAY`), and can provide the exam to `N` patients.
- Instances of `phase(FORNID, ORD)` represent the order (`ORD`) of the exams provided by the exam area characterized by an id (`FORNID`).

The output is represented by an atom of the form `x(RID,FORNID,ST,ET,DAY)`, where the intuitive meaning is that the exam of the registration with id `RID` is in exam area `FORNID`, starts at time `ST` and ends at time `ET`, on the day `DAY`.

Encoding. The encoding consists of the rules reported in Figure 2. Rule r_1 assigns a starting and an ending time to each exam needed by every patient, checking that the time in which is assigned is inside the opening time of the required exam area. Rule r_2 ensures that the order between the exams is respected. Rules r_3 checks that each patient is assigned to at most one exam for every time slot. Then, rule r_4 checks that each exam area provides the exam to at most `N` patients for every time slot. Finally, rule r_5 minimizes the difference between the ending time of the last exam and the starting time of the first exam of each patient.

5 Experimental Results

In this section, we report the results of an empirical analysis of the PAC scheduling problem via ASP. For the first sub-problem, data have been randomly generated using parameters inspired by literature and real world data, then the results of the first sub-problem have been used as input for the second sub-problem. 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 [23] 5.4.0, using parameters *--restart-on-model* for faster optimization and *--parallel-mode 8* for parallel execution. This setting is the result of a preliminary analysis done also with other parameters, e.g., `--opt-strategy=usc` for optimization. The time limit was set to 300 seconds for both sub-problems.

PAC benchmarks. Data are based on the sizes and parameters of a typical middle sized hospital, with 24 different exam areas. For the benchmarks we considered the constants $oe1$ and $oe2$ equal to 2 and 5, respectively. The values of the constants $oe1$ and $oe2$ are used to overestimate the required resources, by adding limits to the assignments to exam locations, for avoiding solutions of the first sub-problem that could lead to unsatisfiable problems in the second sub-problem. Thus, we set the two variables in a safe range, while a hospital could decide to decrease the values of oe1 and oe2 to increase the number of patients assignable in the first sub-problem. The solution schedules patients in a range of 14 days, for each day there are 60 time slots, thus the constant ts is set to 60, corresponding to 5 minute per time slot. To test scalability we generated 3 different benchmarks of different dimensions. Each benchmark was tested 5 times with different randomly generated input.

In particular, each patient is linked to a surgical specialty, and needs a number of exams between 5 and 13, according to the specialty, while the duration of each exam varies between 3 and 6 time slots. The priorities of the registrations have been generated from an even distribution of four possible values (with weights of 0.25 for registrations having priority 1, 2, 3, and 4, respectively). For all the benchmarks, there are 24 exam areas and the operators, that are 35, can be assigned to 3 different exam areas. So, by increasing the number of patients while maintaining fixed the number of operators, we tested different scenarios with low, medium and high requests.

For the second sub-problem, we used the results of the first sub-problem as input. Thus, the number of patients and the exam locations activated depend on the assignment of the solution of the first sub-problem. Patients require all the same first and last exam, while the other exams required by each patient are linked to an order that is randomly assigned and that must be respected by the scheduler. In the second sub-problem

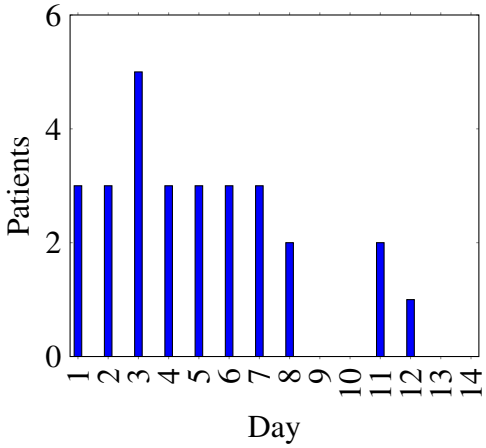

Fig. 3. Number of patients assigned to each day by the scheduler with 40 patients as input

Table 1. Percentage of assigned patients according to their priority level

Total #Patients	%P1 assigned	%P2 assigned	%P3 assigned	%P4 assigned
40	94%	85%	77%	66%
60	90%	69%	20%	13%
80	67%	22%	14%	10%

clinics know the actual list of exams needed by patients: To simulate this scenario, we randomly added and discarded the optional exams assigned to patients in the first sub-problem. For example, optional exams are needed by patients that are over 65 years old or smokers. 5 instances for each benchmark have been generated, each corresponding to the assignments of 14 days.

Results for the first sub-problem. The first optimization criteria in the PAC scheduling sub-problem is to assign as many patients as possible, starting from patients with higher priority. Our solution is able to assign a day to 201 patients out of 217 patients with highest priority; moreover, the scheduler is able to assign all or all but one patients with the highest priority in 12 out of 15 instances tested. Instances with 80 patients are more difficult, since in this scenario the number of operators is not enough to deal with the high number of patients.

In Table 1 are summarized the results obtained in this first sub-problem, in particular, the table shows the average number of patients from the 5 instances assigned with 40, 60, and 80 patients according to their priority level.

The second optimization criteria is to have an assigned day that is as much near as possible to the target day. This optimization criteria is able to assign patients with higher priority near to their target day while; instead, for patients with lower priorities, quality decreases. This is due to two reasons: The first one is that there are more optimization criteria with higher priorities, then, the scheduler already tries to assign as

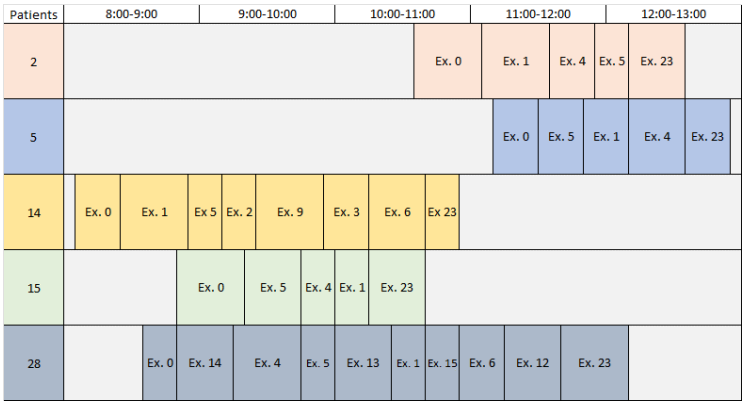

Fig. 4. Representations of the assignments of the starting time of the different exam locations of each patient in a single day

many patients as possible, without taking into account the target days of the patients, and the second one is that some patients have a target day in a day without their exam locations available, so, even in an optimal solution the assigned day to some patients would not be in the target day.

Figure 3 reports the result obtained by the scheduler with one of the instances with 40 patients as input. What can be seen from the graph is that some patients are assigned in days 11 and 12, while in days 9 and 10 there are no patients assigned. This can be explained by the fact that the scheduler tries to assign as many patients as possible and do not try to assign as soon as possible the patients. Moreover, in some days patients can not be assigned due to the unavailability of the exam locations required. In particular, in the assignments in Figure 3, patients that are assigned in day 11 could not be assigned to another day, because that day is the only day with the exam locations they required.

Results for the second sub-problem. In the second sub-problem the solution assigns the starting time of each exam of the patients. The input is taken from the results obtained in the first sub-problem. The solution minimizes the difference between the ending time of the last exam and the starting time of the first exam. While minimizing this value the solution tries to minimize the time spent in the hospital by all patients. In this sub-problem the scheduler is able to reach an optimal solution in 13 out of 15 instances tested. While the average total duration of the exams for each patient is 37 time slots, the solution finds a schedule that allows patients to have an average time in hospital that is just 38.5 time slots.

The results are obtained on average in 152, 186, 220 seconds in the instances with 40, 60 and, 80 patients, respectively. Figure 4 represents the starting times assigned to each patient in a particular day. The patients that must be scheduled are the same that are assigned by the first sub-problem in this day; the scheduler of the second sub-problem minimizes the waiting times of each patient.

As can be seen in Figure 4 the scheduler is able to assign all patients optimally; indeed, all patients have no waiting time between each exam and thus the time spent

```
6 forbiddenAfter(RID,FORNID1,ST+DUR1) :- reg(RID,_,_), exam(RID,FORNID1,DUR1),
      phase(FORNID1,ORD1), #sum{DUR2: exam(RID,FORNID2,DUR2), phase(FORNID2,ORD2), ORD2 >
      ORD1 } = ST.
7 forbiddenBefore(RID,FORNID1,ST) :- reg(RID,_,_), exam(RID,FORNID1,DUR1),
      phase(FORNID1,ORD1), #sum{DUR2: exam(RID,FORNID2,DUR2), phase(FORNID2,ORD2), ORD2 <
      ORD1 } = ST.
8 {x(REGID,FORNID,ST,ST+DUR,DAY): examLoc(FORNID,DAY,FORNST,FORNET,_),
      forbiddenAfter(RID,FORNID,FORB1), forbiddenBefore(RID,FORNID,FORB2), time(ST), ST >=
      FORNST, ST <= FORNET-DUR, ST <= lastTimeSlot-FORB1, ST > FORB2 } = 1 :-
      reg(RID,PRI,DAY), exam(RID,FORNID,DUR).
```

Fig. 5. Optimized encoding for pruning the sessions' starts

in the hospital is reduced to the minimum, while respecting the constraints of the sub-problem, e.g., each exam location is assigned to at most one patient for each time slot.

6 Domain Specific Optimizations

Results show that ASP is a suitable methodology for solving the PAC problem. However, in the second sub-problem, we noted that our encoder is able to find the optimal solution in every instance in a short time but then the solver needs a large amount of time to prove optimality. For this reason, we decided to perform some domain specific optimizations, presented in the following two paragraphs, in order to decrease the grounding and planning time with the aim of improving performance. These optimizations rely on the knowledge of the PAC domain and the possibility of pruning impossible solutions already in the grounding process, as the results in the third paragraph show.

Pruning of exams' starting time slots. As shown in Figure 2, in r_1 the starting time of the exams is guessed between all the available daily time slots, expressed by the atom `time(ST)`. Given that the exams must be assigned following an order, it is known the minimum number of time slots that each patient need to stay before and after each exam. Thus, the guess rule can be improved by reducing the number of possible starting time slots of each exam with the following constraints:

- an exam cannot start in a time slot if the remaining time slots are less than the minimum amount of time slots required to complete all the following exams;
- an exam cannot start in a time slot if the time slots before are less than the minimum amount of time slots required to complete the previous exams.

The encoding for pruning the exams' starting time slots is reported in Figure 5. In rules r_6 and r_7 two new atoms `forbiddenAfter` and `forbiddenBefore` are defined as the minimum amount of time slots needed by each patient after (before) each exam. The minimum amount of time slots required by the exams after (before) each exam is obtained by computing the sum of the duration of the exams with the greater (lower) phase value. In r_8 the two new atoms are used in the guess rule, so that the starting time is after the value computed by the rule r_6 and after the difference between `lastTimeSlot`, that corresponds to the last time slot, and the value computed by r_7.

```
9  cost(RID, TOT) :- TOT = #sum{DUR,FORNID : exam(RID,FORNID,DUR)}, reg(RID,_,_).
10 :∼ x(RID,0,ST,_,_),x(RID,23,_,ET,_), cost(RID,TOT), ET-ST-TOT >= 0. [ ET-ST-TOT@1, RID]
```

Fig. 6. Optimized minimization rule

Table 2. Comparison of the mean time required to reach the optimal solution in the three scenarios with the different versions of the encoding for the second sub-problem

#Patients	ENC (s)	ENC+OPT1 (s)	ENC+OPT2 (s)	ENC+OPT1+OPT2 (s)
40	152	54	5	1
60	186	54	5	1
80	220	77	7	2
Percentage Optimal	86,7%	100%	100%	100%

Minimization with lower bound. As it can be seen in Figure 2, rule r_5 minimizes the time spent in the hospital by each patient, computed as the difference between the ending time of the last exam and the starting time of the first exam. However, the time spent in the hospital by each patient cannot be lower than the sum of the duration of all the required exams. Therefore, the minimization rule can be improved by computing the minimum time required by each patient and using it as a lower bound, so that solutions below this value are pruned.

The enconding with the optimized weak constraint is shown in Figure 6. In rule r_9, the minimum length of time to fully complete all the exams for each patient is computed as the sum of the duration of all the exams and the new auxiliary atom `cost(RID, TOT)` is defined. The weak constraint, in rule r_{10}, minimizes the difference between the planned total time (i.e. the difference between ending time and starting time of the last and first exam) and the lower bound previously computed, activating the weak constraint only when the difference is greater or equal than zero.

Results. In Table 2 the time to reach the optimal solution with the basic encoding, defined as ENC, and with the different optimizations, defined as OPT1 and OPT2, respectively, are reported. In particular, we define ENC+OPT1 as the encoder obtained by adding the rules in Figure 5 to the encoder defined in Figure 2 and by dropping the rule r_1 in Figure 2, ENC+OPT2 as the encoder obtained by adding the rules in Figure 6 to the encoder defined in Figure 2 and by dropping the rule r_5 in Figure 2, and ENC+OPT1+OPT2 as the encoder obtained by adding both optimizations. From Table 2, it can be noted that, while the original encoder is able to reach the optimal solution with 87% of the instances, all the encoders utilizing the different optimizations are able to reach the optimal solution on all instances. Moreover, ENC+OPT1 gives better performance than ENC, in particular, is able to prune a lot of possible solutions thanks to the new rules but, as we previously noted, some time is still spent computing solutions below the known lower bound. With ENC+OPT2, the performance increases noticeably, leading to the optimal solution in a few seconds. While adding either OPT1 and OPT2 led to better results, being able to reach an optimal solution in all the instances in up to 2 seconds, further increasing the performance.

7 Related Work

This paper is an extended and revised version of a paper appearing in the CEUR proceedings [13], having the following main improvements: (i) a mathematical formulation of the two phases of the problem (Section 3), which could be a starting point for testing other languages and tools from experts, (ii) domain-specific optimizations to the previous encoding, and a related experimental analysis, focused on improving the grounding phase of the ASP solver (Section 6), and (iii) a more complete related work section, that now includes also recent studies in which ASP has been employed to closely related scheduling problems (reported below).

The section is organized in two paragraphs: the first is focused on alternative methods for solving the PAC problem, while the second mentions works in which ASP has been employed to closely related scheduling problems.

Solving the PAC problem. The work in [19] used two simulation models to analyse the difficulties of planning in the context of PAC and to determine the resources needed to reduce waiting times and long access times. The models were tested in a large university hospital and the results were validated measuring the level of patient satisfaction. [38] used a Lean quality improvement process changing the process and the standard routine. For example, patients were not asked to move from a room to another for the visits, but patients were placed in a room, and remained there for the duration of their assessment. This and other changes to the processes led to the decrease of the average lead time for patients and to the number of patients required to return the next day to complete the visits. [34], [22], [39], and, [40] studied the importance of implementing the PAC and the positive results obtained by having less waiting time between the exams and for the visit to the hospital. In particular, while different clinics follow different guidelines, implementing PAC has proved to be an important tool to avoid the cancellation of the surgeries and to significantly reduce the risk associated with the surgery.

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 (see, e.g., [3] for a recent survey), the first solved problem was the *Nurse Scheduling Problem* [4, 18, 5], where the goal is to create a scheduling for nurses working in hospital units. Then, the problem of assigning operating rooms to patients, denoted as *Operating Room Scheduling* [16, 17], has been treated, and further extended to include bed management [15]. More recent problems include the *Chemotherepy Treatment Scheduling* problem [14], in which patients are assigned a chair or a bed for their treatments, and the *Rehabilitation Scheduling Problem* [12], which assigns patients to operators in rehabilitation sessions. The current paper is the only one which deals with the pre-operative phase, and presents a two phases approach.

Concerning scheduling problems beyond the Healthcare domain, ASP encoding were proposed for the following problems: *Incremental Scheduling Problem* [8, 11, 26, 25], where the goal is to assign jobs to devices such that their executions do not overlap one another; *Team Building Problem* [35], where the goal is to allocate the available personnel of a seaport for serving the incoming ships; and the *Conference Paper Assignment Problem* [6], which deals with the problem of assigning reviewers in the Pro-

gram Committee to submitted conference papers. Other relevant papers are Gebser et. al [28], 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. [21], 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 PAC scheduling problem modeled and solved with ASP. We started from a mathematical formulation of the problem, which considers constraints and parameters that can be found in other works, and then presented our ASP solution. The solution is further improved with domain specific optimizations. Results on synthetic data shows that the solution is able to assign a high number of patients with higher priority, and that the domain-specific optimizations help to reduce the time to prove optimality. We are currently working on extending our experiments, and comparing to other languages and tools using the mathematical formulation in Section 3. Moreover, we would like also to implement and test other solving procedures, e.g., [33, 31, 32, 36], considering the relation between ASP and SAT procedures [30, 29], whose goal would be to further improve scalability. Finally, we plan to add this solution into a platform of solutions for scheduling problems in Healthcare, similarly to, e.g., [7] in the context of SMT solving.

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: Balduccini, M., Lierler, Y., Woltran, S. (eds.) LPNMR. LNCS, vol. 11481, pp. 241–255. Springer (2019)
3. Alviano, M., Bertolucci, R., Cardellini, M., Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Mochi, M., Morozan, V., Porro, I., Schouten, M.: Answer set programming in healthcare: Extended overview. In: IPS and RCRA 2020. CEUR Workshop Proceedings, vol. 2745. CEUR-WS.org (2020), `http://ceur-ws.org/Vol-2745/paper7.pdf`
4. 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)
5. Alviano, M., Dodaro, C., Maratea, M.: Nurse (re)scheduling via answer set programming. Intelligenza Artificiale **12**(2), 109–124 (2018)
6. 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)
7. Armando, A., Castellini, C., Giunchiglia, E., Idini, M., Maratea, M.: TSAT++: an open platform for satisfiability modulo theories. Electronic Notes in Theoretical Computer Science **125**(3), 25–36 (2005)

8. Balduccini, M.: Industrial-size scheduling with ASP+CP. In: Delgrande, J.P., Faber, W. (eds.) Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6645, pp. 284–296. Springer (2011). https://doi.org/10.1007/978-3-642-20895-9_33, https://doi.org/10.1007/978-3-642-20895-9_33
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. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the Fifth Answer Set Programming Competition. Artificial Intelligence **231**, 151–181 (2016)
12. Cardellini, M., Nardi, P.D., Dodaro, C., Galatà, G., Giardini, A., Maratea, M., Porro, I.: A two-phase ASP encoding for solving rehabilitation scheduling. In: Moschoyiannis, S., Peñaloza, R., Vanthienen, J., Soylu, A., Roman, D. (eds.) Proceedings of the 5th International Joint Conference on Rules and Reasoning (RuleML+RR 2021). Lecture Notes in Computer Science, vol. 12851, pp. 111–125. Springer (2021)
13. Caruso, S., Galatà, G., Maratea, M., Mochi, M., Porro, I.: Scheduling pre-operative assessment clinic via answer set programming. In: Benedictis, R.D., Maratea, M., Micheli, A., Scala, E., Serina, I., Vallati, M., Umbrico, A. (eds.) Proceedings of the 9th Italian workshop on Planning and Scheduling (IPS'21) and the 28th International Workshop on "Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion" (RCRA'21). CEUR Workshop Proceedings, vol. 3065. CEUR-WS.org (2021)
14. Dodaro, C., Galatà, G., Grioni, A., Maratea, M., Mochi, M., Porro, I.: An ASP-based solution to the chemotherapy treatment scheduling problem. Theory and Practice of Logic Programming **21**(6), 835–851 (2021)
15. Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Porro, I.: An ASP-based solution for operating room scheduling with beds management. In: Fodor, P., Montali, M., Calvanese, D., Roman, D. (eds.) Proceedings of the Third International Joint Conference on Rules and Reasoning (RuleML+RR 2019). Lecture Notes in Computer Science, vol. 11784, pp. 67–81. Springer (2019)
16. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: Operating room scheduling via answer set programming. In: AI*IA. LNCS, vol. 11298, pp. 445–459. Springer (2018)
17. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: An ASP-based framework for operating room scheduling. Intelligenza Artificiale **13**(1), 63–77 (2019)
18. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: LPNMR. LNCS, vol. 10377, pp. 301–307. Springer (2017)
19. Edward, G.M., Das, S.F., Elkhuizen, S.G., Bakker, P.J.M., Hontelez, J.A.M., Hollmann, M.W., Preckel, B., Lemaire, L.C.: Simulation to analyse planning difficulties at the preoperative assessment clinic. BJA: British Journal of Anaesthesia **100**(2), 195–202 (02 2008)
20. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine **37**(3), 53–68 (2016)
21. Falkner, A.A., Friedrich, G., Schekotihin, K., Taupe, R., Teppan, E.C.: Industrial applications of answer set programming. Künstliche Intelligenz **32**(2-3), 165–176 (2018)
22. Ferschl, M., Tung, A., Sweitzer, B., Huo, D., Glick, D.: Preoperative Clinic Visits Reduce Operating Room Cancellations and Delays. Anesthesiology **103**(4), 855–859 (10 2005)
23. 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)
24. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence **187**, 52–89 (2012)

25. 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)
26. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. Journal of Artificial Intelligence Research **60**, 41–95 (2017)
27. Gebser, M., Maratea, M., Ricca, F.: The seventh answer set programming competition: Design and results. Theory and Practice of Logic Programming **20**(2), 176–204 (2020)
28. Gebser, M., Obermeier, P., Schaub, T., Ratsch-Heitmann, M., Runge, M.: Routing driverless transport vehicles in car assembly with answer set programming. Theory and Practice of Logic Programming **18**(3-4), 520–534 (2018). https://doi.org/10.1017/S1471068418000182, https://doi.org/10.1017/S1471068418000182
29. Giunchiglia, E., Leone, N., Maratea, M.: On the relation among answer set solvers. Ann. Math. Artif. Intell. **53**(1-4), 169–204 (2008)
30. 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)
31. 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)
32. 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)
33. 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.) International Joint Conference on Automated Reasoning (IJCAR 2001). Lecture Notes in Computer Science, vol. 2083, pp. 347–363. Springer (2001)
34. Harnett, M.P., Correll, D., Hurwitz, S., Bader, A., Hepner, D.: Improving Efficiency and Patient Satisfaction in a Tertiary Teaching Hospital Preoperative Clinic. Anesthesiology **112**(1), 66–72 (01 2010)
35. 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)
36. Rosa, E.D., 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)
37. Schüller, P.: Answer set programming in linguistics. Künstliche Intelligence **32**(2-3), 151–155 (2018). https://doi.org/10.1007/s13218-018-0542-z, https://doi.org/10.1007/s13218-018-0542-z
38. Stark, C., Gent, A., Kirkland, L.: Improving patient flow in pre-operative assessment. BMJ Open Quality **4**(1) (2015). https://doi.org/10.1136/bmjquality.u201341.w1226
39. Tariq, H., Ahmed, R., Kulkarni, S., Hanif, S., Toolsie, O., Abbas, H., Chilimuri, S.: Development, functioning, and effectiveness of a preoperative risk assessment clinic. Health Services Insights **2016**, 1 (10 2016). https://doi.org/10.4137/HSI.S40540
40. Woodrum, C.L., Wisniewski, M., Triulzi, D.J., Waters, J.H., Alarcon, L.H., Yazer, M.H.: The effects of a data driven maximum surgical blood ordering schedule on preoperative blood ordering practices. Hematology **22**(9), 571–577 (2017)