

Scheduling Pre-Operative Assessment Clinic via Answer Set Programming

Simone Caruso¹, Giuseppe Galatà², Marco Maratea¹, Marco Mochi^{1,2} and Ivan Porro²

¹DIBRIS, University of Genova, Genova, Italy

²SurgiQ srl, Italy

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 utmost 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, and present the results of a first preliminary analysis of the two problems based on Answer Set Programming (ASP). In the first sub-problem patients are assigned to a day taking into account a default list of exams; then, the second sub-problem, having the actual list of exams needed by each patient, use the result of the first sub-problem to assign a starting time to each exam.

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 [1] and avoid the cancellation of the surgery [2].


The problem is divided into two sub-problems [3]: 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 prioritizing the assignments to patients with higher priority. In the second sub-problem, the scheduler assigns a starting time to each exam needed by 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,

IPS-2021: 9th Italian Workshop on Planning and Scheduling

✉ 4493864@studenti.unige.it (S. Caruso); giuseppe.galatà@surgiq.com (G. Galatà); marco.maratea@unige.it (M. Maratea); marco.mochi@unige.it (M. Mochi); ivan.porro@surgiq.com (I. Porro)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

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 availability of efficient solvers ([4, 5, 6, 7]) has been successfully employed for solving hard combinatorial problems in several research areas, and it has been also employed to solve many scheduling problems [8, 9, 10, 11, 12, 13, 14], also in industrial contexts (see, e.g., [15, 16, 17, 18] for detailed descriptions of ASP applications).

In this paper, we apply ASP to solve the PAC scheduling problem. In the solution of 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 results of the first one and minimizing the time each patient stays at the hospital. We have finally run a first experimental analysis on PAC benchmarks with realistic sizes and parameters inspired from data seen in literature, scenarios for the first sub-problem, varying the number of patients to schedule and the available operators. Overall, results using the state-of-the-art ASP solver CLINGO [19] of this preliminary analysis show that ASP is a suitable solving methodology also for PAC scheduling problem.

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, dealing with this sub-problem before the actual PAC day, the clinics don't know which exam each patient will require, thus, in the first sub-problem, as typically done in hospitals, we consider that each patient requires a default list of exams according to his specialty, 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. 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 to have a second sub-problem that 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 to 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 limit 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 assign 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

```

1 {x(RID,PR,TOTDUR,DAY) : day(DAY), DAY < DUEDATE} 1 :- reg(RID,PR,TARGET,TOTDUR,DUEDATE).
2 :- x(RID,_,_,DAY), exam(RID,FORNID,_), not examLoc(FORNID,_,_,DAY,_).
3 res(RID,FORNID,DAY,DUR) :- x(RID,PR,_,DAY), exam(RID,FORNID,DUR).
4 :- N1 = #count{FORNID: res(RID,FORNID,_,_)}, N2 = #count{FORNID: exam(RID,FORNID,_,_)},
   x(RID,_,_,_), N1 != N2.
5 :- #sum{DUR, RID: res(RID,FORNID,DAY,DUR)} > NHOORS/2, examLoc(FORNID,_,NHOORS,DAY,N).
6 :- #sum{TOTDUR, RID: x(RID,_,TOTDUR,DAY)} = M, #count{RID: x(RID,_,_,DAY)} = N, day(DAY), M >
   ((60*N)-(5*N*(N+1))), N>1.
7 {operator(ID, FORNID, DAY) : operators(ID, FORNID, DAY)} == NOP :- examLoc(FORNID, NOP, _,
   DAY,_), res(REGID, FORNID, DAY, _).
8 :- operator(ID,FORNID1,DAY), operator(ID,FORNID2,DAY), FORNID1 < FORNID2.
9 unassignedP1(N) :- M = #count {RID: x(RID,1,_,_)}, N = totRegsp1 - M.
10 unassignedP2(N) :- M = #count {RID: x(RID,2,_,_)}, N = totRegsp2 - M.
11 unassignedP3(N) :- M = #count {RID: x(RID,3,_,_)}, N = totRegsp3 - M.
12 unassignedP4(N) :- M = #count {RID: x(RID,4,_,_)}, N = totRegsp4 - M.
13 :~ unassignedP1(N). [N@8]
14 :~ unassignedP2(N). [N@7]
15 :~ unassignedP3(N). [N@6]
16 :~ unassignedP4(N). [N@5]
17 :~ x(RID,1,_,DAY), reg(RID,_,TARGET,_,_). [|DAY-TARGET|@4,RID]
18 :~ x(RID,2,_,DAY), reg(RID,_,TARGET,_,_). [|DAY-TARGET|@3,RID]
19 :~ x(RID,3,_,DAY), reg(RID,_,TARGET,_,_). [|DAY-TARGET|@2,RID]
20 :~ x(RID,4,_,DAY), reg(RID,_,TARGET,_,_). [|DAY-TARGET|@1,RID]

```

Figure 1: ASP encoding of the first sub-problem

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 priority 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 the order and their duration, considering that each exam area can be used by one patient at a time. Moreover, the solution minimizes the difference between the starting time of the first exam and the last exam of each patient.

3. ASP Encoding

In this section we first present the ASP encoding for the first sub-problem, and then the main elements of the second sub-problem, in two separate sub-sections.

3.1. 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 [20]. For details about syntax and semantics of ASP programs we refer the reader to [21].

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

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

The output is an assignment represented by 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 his/her exams equal to TOTDUR.

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 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 minus 30 time slots, 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. 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 be 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} minimizes the difference between the assigned day and the target day of each patient, prioritizing the difference of patients with higher priority.

3.2. 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:

```

1 {x(RID, FORNID, ST, ST+DURATA, DAY) : examLoc(FORNID, DAY, FORNST, FORNET, _) , time(ST), ST >= FORNST,
   ST <= FORNET-DURATA} = 1 :- reg(RID, DAY), esame(RID, FORNID, DURATA).
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]

```

Figure 2: ASP encoding of the second sub-problem

- Instances of `reg(RID, DAY)` represent the registrations, characterized by an id (RID) assigned to the day (DAY).
- Instances of `examLoc(FORNID, DAY, FORNST, FORNET, N)` represent the exam areas, characterized by an id (FORNID), which in the day (DAY) has a starting time and closing time respectively equals to (FORNST) and (FORNET), which is active for a certain value of time (N) in the day (DAY), and can provide the exam to a value (N) of 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 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 the exam area FORNID, starts at the starting time ST and ends at the ending 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.

4. 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 [20] 5.1.0, 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 on the same instances where we tested also other parameters, e.g., `-opt-strategy=usc` for optimization. The time limit was set to 300 seconds for the first and the second sub-problems.

Table 1

Percentage of assigned patients according to their priority level

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

4.1. PAC benchmarks

Data are based on the sizes and parameters of a typical middle sized hospital, with 24 different exam areas. The solution schedules patients in a range of 14 days, for each day there are 60 time slots, 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 in all the benchmarks, can be assigned to 3 different exam areas. So, 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 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. There are 5 instances for each benchmark, each corresponding to the assignments of 14 days.

4.2. 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. The solution has more difficulties on the instances with 80 patients, 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 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

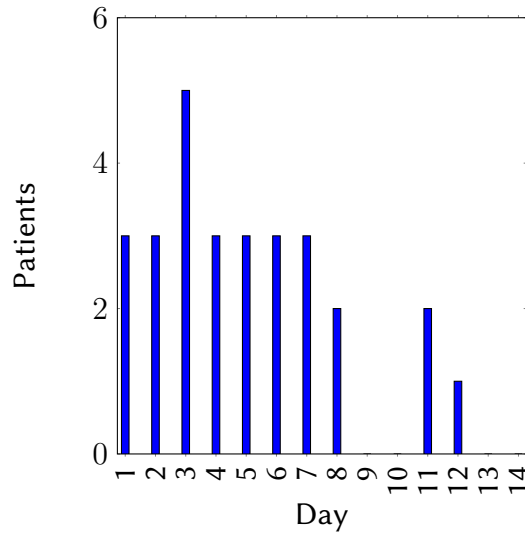


Figure 3: Number of patients assigned to each day by the scheduler with 40 patients as input

to the target day. This optimization criteria is able to assign patients with higher priority near their target day while, for patients with lower priorities, the scheduler is not able to reach great results. 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 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 wouldn't 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 by 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 in another day, because that day is the only day with the exam locations they required.

4.3. 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 solution 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 find a schedule that allows patients to have an average time in hospital that is just

Patients	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00						
2				Ex. 0	Ex. 1	Ex. 4	Ex. 5	Ex. 23			
5					Ex. 0	Ex. 5	Ex. 1	Ex. 4	Ex. 23		
14	Ex. 0	Ex. 1	Ex. 5	Ex. 2	Ex. 9	Ex. 3	Ex. 6	Ex. 23			
15			Ex. 0	Ex. 5	Ex. 4	Ex. 1	Ex. 23				
28		Ex. 0	Ex. 14	Ex. 4	Ex. 5	Ex. 13	Ex. 1	Ex. 15	Ex. 6	Ex. 12	Ex. 23

Figure 4: Representations of the assignments of the starting time of the different exam location of each patient in a single day

38.5 time slots.

The results are obtained on average in 152, 186, 220 seconds respectively in the instances with 40, 60 and, 80 patients. In Figure 4 there are all 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 the patients optimally; indeed, all patients have no waiting time between each exam and thus the time spent in the hospital is reduced to the minimum, while respecting the constraints of the sub-problem. As an example, each exam location is assigned to at most one patient for each time slot.

5. Related Work

The work in [3] used two simulation models to analyse the difficulties of planning in the context of PAC and to determine the resource 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. [22] 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. [1], [2], [23], and, [24] 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 for PAC, implementing it has proved to be an important tool to avoid the cancellation of the surgeries for medical reasons and to significantly reduce the risk associated with the surgery.

ASP has been already used as a tool for solving scheduling problems [15, 25], also in the healthcare domain [18]. In this paper, we focus on a problem that was not addressed in previous works by using ASP.

6. Conclusions and Current Work

In this paper, we have presented a preliminary analysis of the PAC scheduling problem modeled and solved with ASP. We started from the problem as presented in other works and utilized parameters that can be found in other works. Results on synthetic data shows that the solution is able, considering the different constraints, to assign a high number of patients with higher priority. We are currently working on extending our preliminary experiments. Moreover, we would like also to implement and test other solving procedures, e.g., [26, 27, 28, 29], considering the relation between ASP and SAT procedures [30, 31], whose goal would be to improve the current results. Finally, we plan to add this solution into a platform of solutions for scheduling problems in healthcare, similarly to, e.g., [32] in the context of SMT solving.

References

- [1] M. P. Harnett, D. Correll, S. Hurwitz, A. Bader, D. Hepner, Improving Efficiency and Patient Satisfaction in a Tertiary Teaching Hospital Preoperative Clinic, *Anesthesiology* 112 (2010) 66–72.
- [2] M. Ferschl, A. Tung, B. Sweitzer, D. Huo, D. Glick, Preoperative Clinic Visits Reduce Operating Room Cancellations and Delays, *Anesthesiology* 103 (2005) 855–859.
- [3] G. M. Edward, S. F. Das, S. G. Elkhuisen, P. J. M. Bakker, J. A. M. Hontelez, M. W. Hollmann, B. Preckel, L. C. Lemaire, Simulation to analyse planning difficulties at the preoperative assessment clinic, *BJA: British Journal of Anaesthesia* 100 (2008) 195–202.
- [4] M. Alviano, G. Amendola, C. Dodaro, N. Leone, M. Maratea, F. Ricca, Evaluation of disjunctive programs in WASP, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), *LPNMR*, volume 11481 of *LNCS*, Springer, 2019, pp. 241–255.
- [5] F. Calimeri, M. Gebser, M. Maratea, F. Ricca, The design of the fifth answer set programming competition, *CoRR abs/1405.3710* (2014). URL: <http://arxiv.org/abs/1405.3710>. arXiv:1405.3710.
- [6] M. Gebser, M. Maratea, F. Ricca, The design of the seventh answer set programming competition, in: M. Balduccini, T. Janhunen (Eds.), *LPNMR*, volume 10377 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 3–9.
- [7] 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.
- [8] 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.

- [9] 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.
- [10] C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, I. Porro, An ASP-based solution for operating room scheduling with beds management, in: RuleML+RR, volume 11784 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 67–81.
- [11] C. Dodaro, G. Galatà, M. Maratea, I. Porro, Operating room scheduling via answer set programming, in: AI*IA, volume 11298 of *LNCS*, Springer, 2018, pp. 445–459.
- [12] M. Alviano, C. Dodaro, M. Maratea, Nurse (re)scheduling via answer set programming, *Intelligenza Artificiale* 12 (2018) 109–124.
- [13] C. Dodaro, M. Maratea, Nurse scheduling via answer set programming, in: LPNMR, volume 10377 of *LNCS*, Springer, 2017, pp. 301–307.
- [14] 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.
- [15] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, *AI Magazine* 37 (2016) 53–68.
- [16] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, *KÄijnstliche Intelligenz* 32 (2018) 165–176.
- [17] P. Schüller, Answer set programming in linguistics, *Künstliche Intelligenz* 32 (2018) 151–155. URL: <https://doi.org/10.1007/s13218-018-0542-z>. doi:10.1007/s13218-018-0542-z.
- [18] M. Alviano, R. Bertolucci, M. Cardellini, C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, M. Mochi, V. Morozan, I. Porro, M. Schouten, Answer set programming in healthcare: Extended overview, in: IPS and RCRA 2020, volume 2745 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: <http://ceur-ws.org/Vol-2745/paper7.pdf>.
- [19] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, *Artificial Intelligence* 187 (2012) 52–89.
- [20] 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.
- [21] 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.
- [22] C. Stark, A. Gent, L. Kirkland, Improving patient flow in pre-operative assessment, *BMJ Open Quality* 4 (2015). doi:10.1136/bmjquality.u201341.w1226.
- [23] H. Tariq, R. Ahmed, S. Kulkarni, S. Hanif, O. Toolsie, H. Abbas, S. Chilimuri, Development, functioning, and effectiveness of a preoperative risk assessment clinic, *Health Services Insights* 2016 (2016) 1. doi:10.4137/HSI.S40540.
- [24] C. L. Woodrum, M. Wisniewski, D. J. Triulzi, J. H. Waters, L. H. Alarcon, M. H. Yazer, The effects of a data driven maximum surgical blood ordering schedule on preoperative blood ordering practices, *Hematology* 22 (2017) 571–577. arXiv:<https://doi.org/10.1080/10245332.2017.1318336>.
- [25] 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. URL: <https://doi.org/10.1017/S1471068418000182>. doi:10.1017/S1471068418000182.

- [26] E. Giunchiglia, M. Maratea, A. Tacchella, D. Zambonin, Evaluating search heuristics and optimization techniques in propositional satisfiability, in: R. Goré, A. Leitsch, T. Nipkow (Eds.), International Joint Conference on Automated Reasoning (IJCAR 2001), volume 2083 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 347–363.
- [27] E. Giunchiglia, M. Maratea, A. Tacchella, Dependent and independent variables in propositional satisfiability, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), JELIA, volume 2424 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 296–307.
- [28] E. Giunchiglia, M. Maratea, A. Tacchella, (In)Effectiveness of look-ahead techniques in a modern SAT solver, in: F. Rossi (Ed.), CP, volume 2833 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 842–846.
- [29] E. D. Rosa, 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.
- [30] E. Giunchiglia, M. Maratea, On the Relation Between Answer Set and SAT Procedures (or, Between cmodels and smodels), in: ICLP, volume 3668 of *LNCS*, Springer, 2005, pp. 37–51.
- [31] E. Giunchiglia, N. Leone, M. Maratea, On the relation among answer set solvers, *Ann. Math. Artif. Intell.* 53 (2008) 169–204.
- [32] A. Armando, C. Castellini, E. Giunchiglia, M. Idini, M. Maratea, TSAT++: an open platform for satisfiability modulo theories, *Electronic Notes in Theoretical Computer Science* 125 (2005) 25–36.