UNIVERSITÀ DEGLI STUDI DI GENOVA
SCUOLA POLITECNICA
DIBRIS
DEPARTMENT OF INFORMATICS, BIOENGINEERING, ROBOTICS AND SYSTEMS ENGINEERING

MASTER'S DEGREE IN
COMPUTER ENGINEERING - ARTIFICIAL INTELLIGENCE AND HUMAN-CENTERED COMPUTING

Academic year 2021/2022

Artificial Intelligence Techniques for Scheduling Pre-Operative Assessment Clinic

Candidate
Simone Caruso

Supervisor
Prof. Marco Maratea

Co-Supervisors
Dott. Giuseppe Galatà
Dott. Marco Mochi

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 thesis, we divided 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, the results of the first sub-problem are used to assign a starting time to each exam. Furthermore, we present a solution to the pre-operative assessment clinic rescheduling problems. This comes into play when the scheduling solution can not be implemented for some reasons e.g. an operator is suddenly missing, therefore a new scheduling must be found moving the least possible patients and operators inside the scheduling period. The solutions presented are based on a rule-based knowledge representation and reasoning language (Answer Set Programming) and experiments show that the solutions provide satisfying results in short time. Finally, a web application has been developed with the aim of supporting the operators in using the solution realized.

Contents

List of Figures iv

List of Tables v

1 Introduction 1
- 1.1 Context and motivation 1
- 1.2 State of the art 2
- 1.3 Thesis objectives 3
- 1.4 Thesis structure 3

2 Problem description 5
- 2.1 PAC problem description 5
- 2.2 Demonstration of a problem solved 7

3 PAC problem solution 9
- 3.1 Answer Set programming 9
- 3.2 Solving methodology 12
- 3.3 First sub-problem 12
  - 3.3.1 Input 12
  - 3.3.2 ASP Encoding 13
  - 3.3.3 Output 16
- 3.4 Second sub-problem 16
- 3.5 Input 16
  - 3.5.1 ASP Encoding 16
  - 3.5.2 Output 18
- 3.6 Encoding optimizations 18
  - 3.6.1 Optimized encoding of the rst sub-problem 18
  - 3.6.2 Optimized encoding of the second sub-problem 20

4 Results of the scheduling 23
4.1 Results of the scheduling 24
4.1.1 Results of the optimized encodings 27

5 Rescheduling 31
5.1 Problem Description 31
5.1.1 Speci cation of the problem 32
5.2 ASP Encoding 33
5.3 Results of the rescheduling 37
5.3.1 First scenario: Patient requires a new day 38
5.3.2 Second scenario: Operators are not present 39
5.3.3 Third scenario: Exam areas are not available 39

6 Web Application 41
6.1 Demo 43
6.1.1 First planning page 43
6.2 Detailed planning page 44
6.2.1 Final schedule page 45

7 Related work 49

8 Conclusions 51

Bibliography 52

List of Figures

2.1 Representation of a nal scheduling 8

3.1 Resolution methodology schema 11

4.1 Number of patients assigned to each day by the scheduler with 40 patients as input 24
4.2 Representations of the assignments of the starting time of the different exam locations of each patient in a single day 26
4.3 Comparison between the average of number of daily assigned patients, in instances of 40 patients, with non-optimized and optimized encodings . . . 28
4.4 Patients distribution with the new target day generation 29

5.1 Representation of a patient moved a day before after rescheduling 38
5.2 Representation of patients moved because of missing operator 39
5.3 Representation of patients moved because of an unavailable exam area . . . 40

6.1 Representations of web application architecture 41
6.2 First planning page 43
6.3 First planning page, personnel tab 44
6.4 First planning page, providers tab 45
6.5 Detailed planning page, edit patient exams 46
6.6 Detailed planning page, bar chart 46
6.7 Detailed planning page, doughnut chart 47
6.8 Final schedule page 47
6.9 Final schedule page, graphics tab, patients chart 48
6.10 Final schedule page, graphics tab, providers chart 48

List of Tables

2.1 Input list of patients 7

4.1 Solver time in computing the tested instances 25
4.2 Percentage of assigned patients according to their priority level 25
4.3 Comparison of the solver time in computing the tested instances with the two versions of the encoding 27
4.4 Comparison of the percentage of assigned patients according to their priority level with the two versions of the encoding 28
4.5 Percentage of assigned patients, with two exam areas for each different exam and 80 patients 29
4.6 Solver time in computing the tested instances 30

5.1 Time spent by the solver in computing the instances 38

Chapter 1

Introduction

In this chapter, the motivation for which it is important to solve the Pre-Operative Assessment Clinic (PAC) scheduling problem are rst presented. Then, a section is devoted to the state of the art of the PAC problem and of some problems in the digital health domain solved with Answer Set Programming (ASP). Finally, the thesis objectives and the thesis structure are presented.

1.1 Context and motivation

In the Digital Health domain, the area in which technologies are applied to the patients health cares, there are many scheduling problems that are important to solve to improve the quality of the medical cares offered. One of these problems that is usually not or marginally considered, is the PAC scheduling problem; instead it is fundamental to guarantee the patient care as it ensures a proper preparation of patients to their operation and a delay in the PAC can cause a delay in the subsequent phases. The PAC scheduling problem, objective of this thesis, is the task of assigning patients to a day, in which the patient will be examined and prepared to a surgical operation, taking into 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 (Harnett et al. (2010)) and avoid the cancellation of the surgery (Ferschl et al. (2005)).
Hospitals, typically, rst assigns only the day in which patients should have the pre-operative assessment, since in this moment it is not known the precise list of necessary exams. Then,

after having a contact with the patient, it is de ned the nal list of exams needed and the appointment is xed. At this point it is established the precise time for each exam. Thus, we divided the problem into two sub-problems: in the rst 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 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. Since in the rst sub-problem it is considered a default list of exams that is, in general, an overestimation of the necessary exams, this solution does not guarantee to nd the global optimum of the problem. However, having two sub-problems, the computational part is simpli ed and good results are returned.
Moreover, it can happen that, due to unexpected problems (e.g. operators are missing, an exam area is suddenly unavailable), it is required to do changes in the scheduling. Giving an automated solution to this problem, that computes a new scheduling with minimal changes, ensures a proper execution of the pre-operative assessment avoiding delay and a greater satisfaction of both patients and operators.
Our solution is based on the language Answer Set Programming. Recently, in fact, a number of scheduling problems in the Digital Health domain have been ef ciently solved by using knowledge representation and reasoning language, e.g. the rule-based language ASP, but none of these problems consider the PAC problem.
Finally, it would be useful for the hospital operators having an intuitive tool, such as a web application, that allows non expert users to schedule patients graphically and automatically, and that provides a representation of the results easily readable.

1.2 State of the art

Regarding the PAC problem some studies have been done, but in general the focus is on how to improve the PAC organization and reduce waiting time, as for example, the work in Edward et al. (2008) that used two simulation models to analyse the dif culties of planning in the context of PAC and to determine the resources needed to reduce waiting times and long access times.
Currently, despite the importance of having a solution to the PAC scheduling problem to guarantee appropriate medical cares to patients, there are few solutions for this problem and some of the languages that have proven to be ef cient in solving scheduling problems have not been used yet. For example, the ASP language that has been successfully used for

solving hard combinatorial and application scheduling problems in several research areas among which the Healthcare domain (see, e.g., Alviano et al. (2020) for a recent survey).

1.3 Thesis objectives

The general thesis objective is to develop a system able to solve the pre-operative assessment clinic scheduling and rescheduling problem. Thus, the main objectives of the thesis are:

- to nd a solution to the PAC scheduling problem: to model the two sub-problems of the pre-operative assessment using a rule-based language and tools based on a modeling and solving approach;
- to nd a solution for the rescheduling problem, studying the possible scenarios for which it is required and to model the problem using the same approach of the PAC scheduling problem;
- to perform an experimental analysis of the solutions realized, generating synthetic data and studying the quality of the results, the performance and the scalability of the solutions;
- to develop a web application to support non-expert users in using the solutions proposed with an intuitive graphical interface.

1.4 Thesis structure

In Chapter 2, it is presented a detailed description of the PAC scheduling problem, how hospitals deal with this problem and consequentially, our solution. Moreover, it is shown an example of a small problem solved using an approach based on two sub-problems.
In Chapter 3, after a presentation of the ASP language, the language used in this thesis, the expected input and output are de ned, and then, the two encodings of the rst and second sub-problem are presented with a detailed description of all the rules. Furthermore, it is proposed an optimized version of the two encodings.
In Chapter 4, we present the results of the experimental analysis of the PAC scheduling problem. Different scenarios have been considered and we generated data to test our solutions varying the number of patients to schedule with realistic numbers inspired from data seen in literature. Then, it is made a comparison between the optimized and non-optimized encodings.

In Chapter 5, the rescheduling problem is presented. Firstly, it is given a detailed description of the problem and the considered scenarios for rescheduling; then it is shown the encoding realized using ASP and, nally, the results of the experimental analysis of the considered scenarios are shown.
In Chapter 6, it is presented the web application developed to support the operators in using the solutions proposed, its architecture and the frameworks used, then it is shown a demonstration of the application working.
Finally, in Chapter 7 and Chapter 8, the related work and the conclusions are, respectively, mentioned.

Chapter 2

Problem description

2.1 PAC problem description

The Pre-Operative Assessment Clinic 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. This problem is typically organized in two phases by hospitals, in the rst phase it is assigned a day to patients, based on an estimation of the necessary exams and then, in the second phase, after having had a contact with the patient, the appointment is xed, it is determined the nal list of the necessary exams and nally, the time of each exam. This solution, organized in two phases, does not guarantee to nd the global optimum of the problem, having to do an estimation of the exams needed, however accurate, we have to consider more exams of those needed (overestimation), so that the two phases are aligned and avoiding unsatis able solution in the second phase due to too many patients assigned in the same day. However, the two phases are required by hospitals, thus we divided the problem in two sub-problems: in the rst sub-problem, since, as said before, the clinics do not know which exams each patient will require, 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. Thus, in the rst 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 rst phase. 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 rst 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 rst sub-problem the list of exams needed by patients is not the nal list, i.e. just the rst 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 rst 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 rst exam and the last exam of each patient.

Patient id	Priority	Due date
0	2	9
1	1	6
3	1	3
4	3	11
5	1	3
6	3	12
7	2	9
8	2	7
9	3	12
10	4	15
11	4	12
12	4	13
13	1	6
14	3	12
15	3	12
16	3	9
17	2	7
18	1	5
19	3	12

Table 2.1 Input list of patients

2.2 Demonstration of a problem solved

It is following reported an example of a small problem solved with the two phases solution. In the rst sub-problem, as said before, we have in input a list of patients, in this case 20, to be assigned in 4 days. Each patient has his/her relative information about priority and due date as shown in table 2.1. Moreover, it is de ned for each patient his/her speciality, so that we can also specify the linked overestimated list of exams. The solution of this sub-problem assigns to each patient a day out of the 4 available, in which the pre-operative assessment is done; therefore, once the day is established, it is, then, drawn up the nal list of exams and this becomes the input of the second sub-problem. As result, it is xed the starting time and ending time of each exam, ensuring that there are no overlaps between exams provided by the same exam areas.

In gure 2.1, it is reported the nal result of the problem presented, on the y-axis we have the patients id, on the x-axis the time line and each colored block represents an exam identi ed by the id of the exam area that provides it. However, as said before, this solution does not guarantee to nd the global optimum of the problem. In fact, as results show, one patient out

(a) Day 1

(b) Day 2

(c) Day 3

(d) Day 4

Figure 2.1 Representation of a nal scheduling

of the 20 in the input list has not been assigned due to the overestimation done in the rst sub-problem, but looking at the nal results, the patient could be assigned, for example, in day 1. In effect, the exams requested by the patient were with id 0, 1, 4, 6, 10, 11, 23 that were possible to be scheduled without overlapping with the other exams with the same id.

Chapter 3

PAC problem solution

3.1 Answer Set programming

Answer Set Programming (ASP) is a well-known declarative programming language, developed in the field of reasoning and logic programming (Calimeri et al. (2020)). We recall that an ASP program is made of (a combination of):

1. facts of the form `head.`;
2. normal rules of the form `head :- body.`;
3. choice rules of the form `lowerBound {atoms} upperBound :- body.`;
4. constraints of the form `:- body.`;
5. weak constraints of the form `:~ body. [weight@level,terms]`

where `head` is an atom, `atoms` is a set of atoms, and `body` is a set of (possibly negated) atoms, also including aggregate functions, such as `#sum`, and `terms` is a sequence of terms, i.e., variables (strings starting with uppercase letter) or constants (non-negative integers or strings starting with lowercase letters). Atoms can be made over terms. The semantics is given in terms of its *answer sets*, that is, sets A of ground atoms, where atoms in A are said to be true (false, otherwise), such that: (1.) `head` is in A; (2.) whenever the `body` is true (i.e., all positive atoms are in A and all negated atoms are not in A), `head` is in A; (3.) whenever the `body` is true, a range, between lower bound and upper bound, of atoms in `atoms` are in A (if the lower bound is equal to the upper bound, the choice rule can be also written as `{atoms} == bound :- body.`); (4.) the `body` must be false. Moreover, weak constraints of the form (5.) allow expressing preferences among answer sets, where `level` represents the

priority and weight is a numerical costs that is paid whenever the body of a weak constraint is true w.r.t. an answer set. Overall, the preferred weak constraints are the ones with the lowest costs at the highest levels.

Example of fact (1).
As example of fact, it is reported an instance of the reg(RID,PR,TARGET,TOTDUR,DUEDATE) atom:

```
reg(1,4,9,36,12).
```

the reg atom represents a registration characterized by an id (1), a priority (4), a target day (9), the total duration of the exams (36) and the due date (12).

Example of normal rules (2.).
As example of normal rule, we imagine we have assigned to each patient a day and we want to assign the same day to his or her exams:

```
res(RID,FORNID,DAY,DUR) :- x(RID,PR,_,DAY), exam(RID,FORNID,DUR).
```

the head, res(RID,FORNID,DAY,DUR), is true whenever the body is true; in this example, the body is made by x(RID,PR,_,DAY) and exam(RID,FORNID,DUR). Therefore, for each exam FORNID of a person RID it is assigned the day DAY.

Example of choice rules (3).
Consider that we want to assign a day to each patient. In the following choice rule:

```
0 {x(RID,PR,TOTDUR,DAY) : day(DAY), DAY < DUEDATE} 1 :-
    reg(RID,PR,TARGET,TOTDUR,DUEDATE).
```

for each patient reg(RID,PR,TARGET,TOTDUR,DUEDATE), we assign a day from those that are possible day(DAY), such that DAY is less then (before) the due date of the patient. In this rule the lower bound is zero, and upper bound is equal to 1.

Example of constraints (4).
In the following constraint:

```
:- x(RID,_,_,DAY), exam(RID,FORNID,_), not examLoc(FORNID,_,_,DAY,_).
```

Figure 3.1 Resolution methodology schema

we state that we cannot assign a patient on a day DAY if an exam location required by his/her exams (FORNID), is not true, in other words, if the exam location is not working on a day, we cannot assign on that day a patient that requires an exam provided by that exam location.

Example of weak constraints (5).

```
:    x(RID,1,_,DAY),  reg(RID,_,TARGET,_,_).  [|DAY-TARGET|@4,RID]
```

This weak constraint minimize the difference between the patient assigned day DAY and the target day TARGET. In this way each patient is assigned as near as possible to his/her target day. This is expressed by [|DAY-TARGET|@4,RID], where |DAY-TARGET| corresponds to the weight (what we want to minimize), 4 is the level, useful to give a priority when we have more weak constraints, and RID to apply the weak constraint for each patient with different id.

3.2 Solving methodology

In figure 3.1, it is reported a representation of a solution based on a declarative programming approach as our ASP solution. It is made up of five steps:

- Problem: it is the problem formulation to be modelled and solved.
- Encoding: it is the formal representation of the problem, using ASP. It is made up of a set of rules as presented above.
- Solver: it takes in input the encoding of the problem and generate the answer set.
- AnswerSet: it is the output of the solver and it corresponds to the set of atoms that satisfy all the rules of the encoding.
- Solution: it is the solution of the problem.

3.3 First sub-problem

3.3.1 Input

The input of the first sub-problem is the following:

- 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 area (FORNID) in a day (DAY).
- Instances of day(DAY) represent the available days.

- Two constant oe1 and oe2 used to overestimate the patients assignment and they represent a trade-off between the risk of having unsatis able solution in the second sub-problem and the number of patients assignable (the greater the risk, the greater the number of patients).

3.3.2 ASP Encoding

Starting from the speci cations in the previous section, here we present the ASP encoding for the rst sub-problem.

For all the patients, assign a day that is at least one day before their due date:

```
0 {x(RID,PR,TOTDUR,DAY) : day(DAY), DAY < DUEDATE} 1 :-
    reg(RID,PR,TARGET,TOTDUR,DUEDATE).
```

In this rule, for all the registrations reg(RID,PR,TARGET,TOTDUR,DUEDATE), it is assigned a day from those that are possible day(DAY) and that is at least one day before the due date. Thus, a new atom x(RID,PR,TOTDUR,DAY) is de ned that presents the registration with the corresponding assigned day.

A patient cannot be assigned in a day in which an exam area that (s)he requires is not working:

```
:- x(RID,_,DAY,_,_), exam(RID,FORNID,_), not examLoc(FORNID,_,_,DAY,_).
```

This constraint ensures that in the assigned day of each patient, all the exam areas required by his/her exams are present; therefore we check that the atom examLoc required by at least one exam (same FORNID) is not false.

Save for each exam the corresponding patient assigned day:

```
res(RID,FORNID,DAY,DUR) :- x(RID,PR,_,DAY), exam(RID,FORNID,DUR).
```

In this rule it is de ned 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.

Overestimation of the daily total amount of time used of each exam location:

```
4 :- #sum{DUR, RID: res(RID,FORNID,DAY,DUR)} > NHOURS/oe1,
     examLoc(FORNID,_,NHOURS,DAY,N).
```

It is ensured 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.

Overestimation of the daily patients assignment:

```
5 :- #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.
```

To be sure to not assign too many patients in the rst sub-problem to a particular day, therefore to avoid to have unsatis able solution in the second sub-problem because of not enough time to schedule all the exams without overlapping, it is overestimated the time needed by each patient. The degree of the overestimation can be changed by using different oe2 values.

Assign operators to the working exam areas:

```
6 {operator(ID, FORNID, DAY) : operators(ID, FORNID, DAY)} == NOP :-
     examLoc(FORNID, NOP, _, DAY,_), res(RID, FORNID, DAY, _).
7 :- operator(ID,FORNID1,DAY), operator(ID,FORNID2,DAY), FORNID1 < FORNID2.
```

The exam areas, to be activated, require a given number of operators. Thus, in the rst rule reported, it is assigned, for each exam area required at least by one patient, the number of operators needed (NOP). Then, in the second rule, it is checked that the operators are assigned to just one exam location in every day. This is done stating that FORNID1 cannot be less then FORNID2, that is a an other way of saying that they cannot be different but with better performance.

Count the number of patients that are not assigned:

```
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.
```

These rules 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. The total registration for each priority totRegsP1, totRegsP2, totRegsP3, totRegsP4 are values given in the input.

Minimization of the not assigned patients:

```
:   unassignedP1(N). [N@8]
:   unassignedP2(N). [N@7]
:   unassignedP3(N). [N@6]
:   unassignedP4(N). [N@5]
```

These weak constraints are used to minimize the number of unassigned registrations according to their priority.

Minimization of the distance between the patient assigned day and his/her target day:

```
:   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]
```

Finally, since we want to assign patients as near as their target day (an optimal day in which schedule the appointment), it is minimized the difference between the assigned and target day of each patient, giving precedence to higher priorities.

3.3.3 Output

The output of the first sub-problem is an assignment represented by two atoms, one is 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, the other one is operator(ID,FORNID,DAY) meaning that the operator with id ID is assigned to the exam location FORNID in the day DAY.

3.4 Second sub-problem

3.5 Input

The assignment patient-day, output of the first sub-problem, becomes the input of the second sub-problem. More formally, the input of the second sub-problem is made by the atoms exam and time that are the same of the first sub-problem and the following atoms:

- 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,_) 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).

3.5.1 ASP Encoding

The encoding of the second sub-problem consists of the rules reported following.

For each patient exams, assign a starting time in one of the available time slots:

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

In this rules, it is assigned a starting and an ending time to each exam needed by every patient, it is also checked that the time in which the exam is assigned is inside the opening time of the required exam area.

Ensure that the order between the exams is respected:

```
2 :- x(RID,FORNID1,ST1,_,_), x(RID,FORNID2,ST2,_,_), phase(FORNID1,ORD1),
     phase(FORNID2,ORD2), ORD2 < ORD1, ST1 < ST2.
```

Since some exams are needed by others, it has been assigned to each exam an order value (phase(FORNID,ORD)). Exams with a lower order value must be scheduled before those with higher order value, thus in this rule, it is ensured, for each patient, that the starting time of an exam is lower than all the other exams with higher order value.

Ensure that there are no exam overlapping for each patient:

```
3 :- #count{FORNID: x(RID,FORNID,ST,ET,DAY), T >= ST, T < ET} > 1,
     reg(RID,DAY), time(T).
```

In this rule, it is checked that each patient is assigned to at most one exam for every time slot stating that the number of exams assigned in a time slot cannot be greater than 1.

Each exam area provides the exam to at most N patients for every time slot:

```
4 :- #count{FORNID: x(RID,FORNID,ST,ET,DAY), T >= ST, T < ET} > N,
     examLoc(FORNID,DAY,_,_,N), time(T).
```

Then, since there can be more than one exam areas that can provide an exam (examLoc), ensure that for each time slot no more than N patients are assigned to an exam area.

Minimize the total duration of the pre-operative assessment:

```
5 :    reg(RID,_), x(RID,0,ST,_,_), x(RID,23,_,ET,_). [ET-ST@1, RID]
```

Finally, this weak constraint minimizes the difference between the ending time of the last exam and the starting time of the rst exam of each patient.

3.5.2 Output

The output of the second sub-problem is represented by an atom of the form: exam(RID,FORNID,ST,ET,DAY), where the meaning is that the exam of the registration with RID is in exam area FORNID, starts at time ST and ends at time ET, on the day DAY.

3.6 Encoding optimizations

The solutions proposed above is able to assign a good number of patients and to schedule the exams optimally, and has been presented at the 6th workshop of the Italian planning and scheduling and published in the post-proceedings of the XXth international conference Caruso et al. (2021), and to appear in the Volume 13196 of the LNCS post-proceedings of the AI*IA 2021 conference. However, in the rst sub-problem solution it is required to set some constants for which it can be dif cult to nd a good trade off between assigning more patients and the risk to have an unsatis able solution in the second phase. Recalling that before the second phase the patients' appointments are xed, having an unsatis able solution in the second phase is not acceptable because this means that the solver is not able to satisfy all the constraints of this sub-problem, therefore it must be computed a new solution of the rst sub-problem and all the appointments could change. This is not acceptable in practice. Thus, we worked on an optimized version of both the sub-problems encoding. In the rst one, we modi ed how the patients are assigned in order to have an easier way of scheduling patients (no constants) and more patients scheduled in every day but always ensuring the satis ability of the solution, instead in the second sub-problem we worked on improving the performance.
In the next sections, the two encoding with the new rules and those modi ed are presented.

3.6.1 Optimized encoding of the rst sub-problem

In order to improve the patients assignment we have realized an optimized version of the encoding presented before. More in detail, in this version we introduced the time slots, so that we can assign a possible starting time and ending time of the PAC for each patient. In this way, we can ensure that for each assigned patient it is possible to schedule the two critical exams, i.e the rst and the last one, without overlapping with the rst and the last of

other patients. These two exams are de ned as critical because they are the only required by each patient and that must be scheduled in a precise order, i.e as rst and last. Therefore, in order to introduce the time slots, the atom day(DAY) became time(DAY,T) and represent the time slots in each available day, then we modi ed the input atom reg(RID, PR,TARGET,TOTDURATION,DUEDATE) adding the total duration of his/her pre-operative assessment (TOTDURATION). Following, we report the modi ed rules of the encoding. While, starting from the encoding reported in 3.3.2, rule 2 and the rules from the rule 6 to rule 19 remain the same, the choice rule (rule 1) has been modi ed and two more constraints have been introduced.

For all the patients, assign a day that is at least one day before their due date and a starting time:

```
1 0 {x(RID,PR,DAY,ST,ST+TOTDURATION) : time(DAY,ST), DAY < DUEDATE, ST <=
      60-TOTDURATION} 1 :- reg(RID, PR, TARGET,TOTDURATION, DUEDATE).
```

In this rule, for all the registrations reg(RID,PR,TARGET,TOTDUR,DUEDATE), it is now assigned not only a day but also a time slot that represents the starting time of the pre-operative assessment from those that are possible (time(DAY,ST)). Thus, a new atom x(RID,PR,DAY,ST,ST+TOTDURATION) is de ned that presents the registration with the corresponding assigned day and the possible starting time and ending time.

Check that the rst and the last exam can be assigned without overlapping with the same exams of different patients:

```
3 :- #count{RID: x(RID,_,DAY,ST,_),exam(RID,0,DUR), T >= ST, T < ST+DUR} > N,
      examLoc(0,_,_,DAY,N), time(DAY,T).
4 :- #count{RID: x(RID,_,DAY,_,ET),exam(RID,23,DUR), T < ET, T >= ET-DUR} > N,
      examLoc(23,_,_,DAY,N), time(DAY,T).
```

As said before, in order to ensure that the patient assignment, in this sub-problem, do not return solutions unsatis able in the second sub-problem, we check that the rst and the last exam, identi ed with id 0 and 23 (atoms exam(RID,0,DUR) and exam(RID,23,DUR)) can be assigned in a time slot without overlapping with other exams with the same id. More in detail, we check that, in each time slots, there are less then N patients assigned to the same

exam area, where N is the number of available exam areas able to provide the same exam. These two exams, in fact, are the only one that must be done by every patient and they must be scheduled respectively as rst and last while other exams can change the order; thus these exams are the bottle neck of the problem and giving an assignment to them allow to have an estimation of the patients assignable without decreasing too much the performance of the scheduler.

3.6.2 Optimized encoding of the second sub-problem

In this section it is reported the optimization done on the encoding of the second sub-problem. Starting from the encoding presented in 3.5.1, we performed some domain speci c optimizations with the aim of improving performance.
The rst optimization is the pruning of exams' starting time slots. This optimization rely on the knowledge of the PAC domain and the possibility of pruning impossible solutions, in fact, 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.

Then, a second optimization has been done on the minimization adding a lower bound. As it can be seen in the encoding in section 3.5.1, we minimized 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 rst 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 optimized encoding of the second sub-problem is following reported. The rules from rule 2 to rule 4 remain the same as in the encoding reported in 3.5.1, while others have been changed.

Pruning of exams' starting time slots:

```
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.
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.
```

In these rules two new atoms forbiddenAfter and forbiddenBefore are de ned 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.

For each patient exams, assign a starting time in one of the available time slots using the pruning of exams' starting time slots:

```
{x(RID,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,PR,DAY), exam(RID,FORNID,DUR).
```

The two new atoms de ned before are used in this guess rule, so that the starting time of each exam is after the value computed in forbiddenBefore and after the difference between lastTimeSlot , that corresponds to the last time slot, and the value computed in forbiddenAfter , in fact, this value represent the minimum number of time slots required after an exam (to perform all the remaining exams), therefore the starting time of an exam must be before the last time slot available minus the duration of the following exams.

Minimization with lower bound:

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

This rule 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 rst 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.

Chapter 4

Results of the scheduling

In this chapter, we report the results of an empirical analysis of the PAC scheduling problem via ASP. For the rst sub-problem, data have been randomly generated using parameters inspired by literature and real world data, then the results of the rst sub-problem have been used as input for the second sub-problem. After that, it is reported a comparison with the results obtained with the optimized encodings. The experiments were run on a AMD Ryzen 5 3600 CPU @ 3.60GHz with 16 GB of physical RAM. The ASP system used was CLINGO (Gebser et al. (2016a)) 5.4.0, using parameters --restart-on-model 1 for faster optimization and --parallel-mode 6 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 5 minutes 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 rst sub-problem that could lead to unsatis able 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 rst sub-problem. The solution schedules patients in a range of 14 days, but the target date, i.e the optimal day in which schedule the appointment, is set among the rst 7 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.

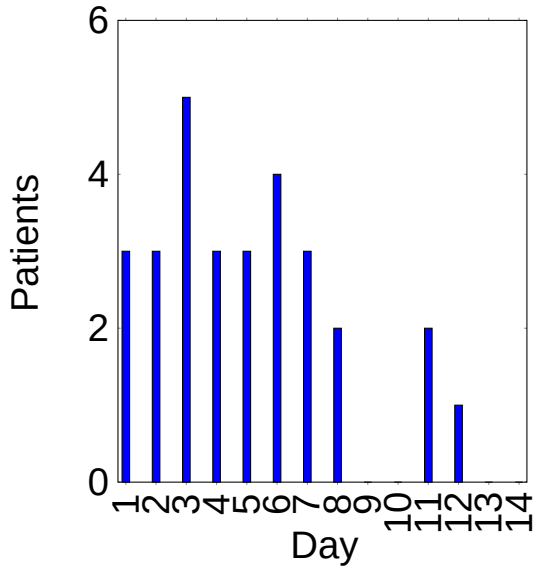

Figure 4.1 Number of patients assigned to each day by the scheduler with 40 patients as input

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 xed 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 rst sub-problem as input. Thus, the number of patients and the exam locations activated depend on the assignment of the solution of the rst sub-problem. Patients require all the same rst 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 rst 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 assignment of 14 days.

4.1 Results of the scheduling

Table 4.1 Solver time in computing the tested instances

Total #Patients	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
40	11s	24s	74s	120s	1s
60	300s	3s	265s	300s	300s
80	300s	300s	300s	300s	300s

Table 4.2 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%	74%	22%	16%
80	94%	30%	13%	15%

Results for the rst sub-problem. The rst optimization criteria in the PAC scheduling sub-problem is to assign as many patients as possible, starting from patients with higher priority. In Table 4.1, it is reported the time spent by the solver to compute the solutions. As it is shown, the solver is able to nd the optimum, on average, in less than 60 seconds with instances with 40 patients, instead with a higher number of patients, it is still able to nd a solution but the time limit is reached. In particular, it can be noticed that instance 2 with 60 patients is solved in few seconds, instead other with the same dimension took longer. This happened because of synthetic data, and, more in details, in 8 days out of 14 it was not possible to schedule any patients due to not enough operators to activate the required exam areas. On average, the number of scheduled patients per instance is 30, even in instances with more patients. On instances with a higher number of patients the performance decreases, the time limit is reached and the number of assigned patients remain, on average, 30. Instances with 80 patients are more dif cult, since in this scenario the number of operators is not enough to deal with the high number of patients.

Overall, 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.

In Table 4.2 are summarized the results obtained in this rst 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 close 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 rst one is that there are more optimization criteria with higher

Figure 4.2 Representations of the assignments of the starting time of the different exam locations of each patient in a single day

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 would not be in the target day.
Figure 4.1 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 4.1, 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 rst sub-problem. The solution minimizes the difference between the ending time of the last exam and the starting time of the rst 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 14 out of 15 instances tested. While the average total duration of the exams for each patient is 31 time slots, the solution nds a schedule that allows patients to have an average time in hospital that is just 32 time slots.

Table 4.3 Comparison of the solver time in computing the tested instances with the two versions of the encoding

	Instance 1		Instance 2		Instance 3		Instance 4		Instance 5	
Tot #Patients	BASE	OPT	BASE	OPT	BASE	OPT	BASE	OPT	BASE	OPT
40	11s	0.5s	24s	0.7s	74s	0.6s	120s	6.4s	1s	1.2s
60	300s	300s	3s	2.6s	265s	300s	300s	300s	300s	300s
80	300s	300s	300s	300s	300s	44s	300s	300s	300s	9s

The results are obtained on average in 122, 154, 141 seconds in the instances with 40, 60 and, 80 patients, respectively. The time does not grow with the number of patients, in fact, as said before, the total number of patients assigned in the rst sub-problem (which is the input of this sub-problem) is, on average, 30 also in instances with 60 or 80 patients. Figure 4.2 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 rst 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 4.2 the scheduler is able to assign all 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, e.g., each exam location is assigned to at most one patient for each time slot.

4.1.1 Results of the optimized encodings

The same instances presented before have been used to test the optimized encodings. Results show that there is a high improvement in the quality of the results and in the performance with the optimized encodings. In the following, we present the results obtained.
In Table 4.3 it is reported a comparison of the solver time in computing the instances between the two versions of the encoding of the rst sub-problem: the basic encoding (BASE in the table), i.e the one without optimizations, and the optimized encoding (OPT). The average time is 2s, 241s, 191s for instances with 40, 60 and 80 patients respectively, that is a small improvement with the results obtained with the non-optimized encoding.
However, the quality of the results is clearly improved, as table 4.4 shows: the percentage of patients assigned is higher for all the instances, in particular the problem scales better with the growing of the number of patients. In fact, instances with 60 and 80 patients still have a percentage greater than the 70% even for patients with low priority (priority 3 and 4) and the average of the number of patients scheduled has increased from 30, obtained with both the instances with 60 and 80 with the non-optimized encoding, to 45 for instances with 60 patients and 60 for instances with 80.

Table 4.4 Comparison of the percentage of assigned patients according to their priority level with the two versions of the encoding

	Priority 1		Priority 2		Priority 3		Priority 4	
Tot #Patients	BASE	OPT	BASE	OPT	BASE	OPT	BASE	OPT
40	94%	94%	85%	88%	77%	84%	66%	84%
60	90%	92%	74%	80%	22%	77%	16%	77%
80	94%	97%	30%	92%	13%	74%	15%	74%

(a) Basic encoding

(b) Optimized encoding

Figure 4.3 Comparison between the average of number of daily assigned patients, in instances of 40 patients, with non-optimized and optimized encodings

In gure 4.3 it is reported a comparison between the basic and optimized encoding on the number of daily scheduled patients with instances of 40 patients. As it is shown, the number of patients assigned daily is clearly higher and the patients are more concentrated in the rst 7 days due to the fact that we generated the patients' target dates in the rst week and one optimization level is to assign patients as close as possible to their target day.

Figure 4.4 Patients distribution with the new target day generation

Furthermore, following the good results obtained, we tested an other possibility of assigning the target day. It is also used in practice to put the target day some days before the due date of the patient. We put 3 days before. Therefore, we generated and tested, 5 more instances with 40, 60 and 80 patients. As it is shown in gure 4.4 in this way patients are more distributed among the 14 days, while the total number of assigned patients is similar. All the instances presented since now, have been generating with a number of exam areas able to provide the same exam equal to one, so, we generated 15 more instances to test how the solution was able to deal with more exam areas providing the same exam.

Instance	Priority 1	Priority 2	Priority 3	Priority 4
1	16%	14%	19%	45%
2	100%	88%	100%	94%
3	77%	85%	95%	89%
4	100%	100%	100%	100%
5	100%	86%	83%	94%

Table 4.5 Percentage of assigned patients, with two exam areas for each different exam and 80 patients

In table 4.5, it is reported the percentage of assigned patients with two exam areas for each different exam. The scheduler is able to assign almost all the patients in all the instances, but the rst one that, due to a unlucky generation, in 10 out of the 14 available days it is not possible to schedule any patients because there are not enough operators able to work on

Table 4.6 Solver time in computing the tested instances

Total #Patients	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
40	1.7s	2.2s	17s	10s	2.7s
60	23s	40s	21s	14s	12s
80	21s	35s	13s	0.5s	33s

necessary exam areas. In conclusion, as the results show, the solution of the rst sub-problem is able to assign a good number of patients, and has a good scalability with the increase of the number of total patients (it has been tested up to 80 patients) and with the number of same exam areas available.

Finally, all the instances above presented and used to test the solution of the rst sub-problem, have been used as input of the second one. As the number of patients is increased, thanks to the optimized encoding of the rst sub-problem, the performance of the basic encoding of the second sub-problem decreased. In fact, the encoding without any optimization reached in all the instances the time limit of 300 seconds and produced low quality results, with several time slots of waiting time between the exams. Instead, the optimized encoding is able to nd the optimum in all the instances in few seconds, as shown in table 4.6, without waiting time between the exams.

Chapter 5

Rescheduling

5.1 Problem Description

A solution to the Pre-Operative Assessment Clinic rescheduling problem is vital in a hospital context. In fact, it can happen that, due to some reasons, it is not possible to implement a scheduling. Therefore, it is necessary to nd a solution in order to guarantee the proper execution of the pre-operative assessment for every patient assigned in that scheduling. We considered three different scenarios in which rescheduling can be required:

- some patients can't be present on the assigned day,
- some operators are missing for some days,
- exam locations are unavailable for some days.

Thus, given a scheduling and the information of why it is not possible to implement it, the PAC rescheduling problem consists in moving patients and operators in order to nd a new assignment that takes into account the new information and the constraints of the initial problem, i.e the PAC scheduling problem (Chapter 2.1) or, in particular, we considered the rescheduling after the second sub-problem.
The optimization of the rescheduling problem consists in minimizing the changes done to the initial scheduling, according to some criteria: the priority is not to change the day in which patients were assigned and, in case it is not possible, to minimize the distance between the new and the old assigned day. Then, if a patient is rescheduled in the same day than we want not to change the exams starting time, otherwise, since the day is changed it is not important to maintain the same exams starting time, thus it is minimized the patient's total length of the pre-operative assessment as it was done in the second sub-problem of the PAC scheduling

problem (Chapter 3.4). Finally, if possible, operators are assigned in the same exam areas they were initially assigned.

5.1.1 Speci cation of the problem

Input

The input data is the same of the PAC scheduling problem for the atoms exam, examLoc (of the second sub-problem), operators and phase, while other atoms are changed:

- Instances of reg(RID,PR,DUEDATE) represent the registration of a patient, characterized by an id (RID), the priority level (PR), and the duedate (DUEDATE).
- Instances of x(RID,FORNID,ST,ET,DAY) represent a planned exam, characterized by a patient id (RID), the exam location (FORNID), the starting time and ending time of the exam (ST and ET), and the day (DAY).
- Instances of operator(ID,FORNID,DAY) represent assigned operator, characterized by an id (ID), the exam location in which is assigned (FORNID) and the day (DAY).
- Instances of forbidden(RID,DAY) represent a patient (RID) cannot be present in a day (DAY).
- Instances of notAvailableExamLoc(FORNID,DAY) represent an exam location (FORNID) is not available in a day (DAY).
- Instances of notAvailableOperator(ID,DAY) represent an operator (ID) is not available in a day (DAY).
- Instances of time(DAY,T) represents the available time slots (T) in a day (DAY).

Output

The output is a new assignment represented by two atoms, one is of the form y(RID,PR,TOTDUR,DAY) where the meaning is that the exam of registration with RID and priority level PR is assigned to the day DAY and has a total duration of exams equal to TOTDUR. The other one is operatorY(ID,FORNID,DAY) meaning that the operator with ID is assigned on the exam location FORNID in the day DAY.

5.2 ASP Encoding

The ASP encoding of the PAC rescheduling problem we realized is made by: the rules 5 and 6 presented in the optimized encoding of the second sub-problem (Chapter 3.6.2) and not reported again, and the rules presented in the following.
De ne an auxiliary atom representing the possible starting time of each exam:

```
allowedTime(RID,FORNID,ST..ts-ET) :-
    forbiddenBefore(RID,FORNID,ST),forbiddenAfter(RID,FORNID,ET).
```

In this rule, it is de ned an auxiliary atom allowedTime(RID,FORNID,ST..ts-ET) that represents all the possible starting time of each exam. This is done starting from the time slot ST until the time slot ts-ET (both de ned above in the atoms forbiddenBefore , forbiddenAfter), where ts is the total amount of time slots available.

De ne an auxiliary atom representing the possible day in which each patient can be assigned:

```
allowedDay(RID,DAY):- reg(RID,_,DUEDATE), time(DAY,_), not forbidden(RID,_),
    DAY < DUEDATE.
```

In this rule, a new atom is de ned allowedDay(RID,DAY) that represents the days in which each patient can be assigned. It is computed as the available time slots (time(DAY,_)) before the due date of the patient that are not forbidden (i.e. those in which the patient cannot be present).

For each patient that cannot be present on his/her (old) assigned day, assign a new starting time and day to all his/her exams:

```
1{y(RID,FORNID,ST,ST+DUR,DAY): allowedTime(RID,FORNID,ST),
    allowedDay(RID,DAY)}1 :- reg(RID,_,DUEDATE), exam(RID,FORNID,DUR),
    forbidden(RID,_).
```

In this choice rule, a new starting time and day are assigned to all the exams of patients that can not be present on their (old) assigned day. These patients are characterized by the atom forbidden(RID,_) true and their new starting time and day are chosen from those available

and de ned in the atoms allowedDay(RID,DAY) and allowedTime(RID,FORNID,ST)

De ne a new atom that represents the days in which there are changes:

```
changedDay(DAY) :- y(RID,_,_,_,DAY), x(RID,_,_,_,DAY1), DAY != DAY1.
changedDay(DAY) :- notAvailableExamLoc(_,DAY).
changedDay(DAY) :- notAvailableOperator(_,DAY).
```

Since the objective of the rescheduling is to minimize the difference between the new and the starting scheduling, it is convenient, in order to increase performance, to nd a new assignment of the exams starting time only in the days with changes (for example in a day in which a new patient is inserted) and to copy the initial assignments in the other days. Thus, in rules 7, 8 and 9, a new atom is de ned (changedDay(DAY)) that represents the days in which there are changes that can be because: it is assigned a new patient (rule 7), an exam location is not available (rule 8) and an operator is not present (rule 9).

Assign a new starting time and day to each exam of patients assigned in days with changes:

```
1{y(RID,FORNID,ST,ST+DUR,DAY): allowedTime(RID,FORNID,ST),
    allowedDay(RID,DAY)}1 :- reg(RID,_,DUEDATE), exam(RID,FORNID,DUR), not
    forbidden(RID,_), x(RID,_,_,_,DAY1), changedDay(DAY1).
```

In this rule, it is de ned a new assignment for all the exams of patients that in the initial scheduling were assigned in days that in the new scheduling have changes, therefore in the days in which the atom (changedDay(DAY)) is true.

Assign the starting time of the exams of patients assigned in day without any changes:

```
y(RID,FORNID,ST,ET,DAY) :- x(RID,FORNID,ST,ET,DAY), not changedDay(DAY), not
    forbidden(RID,_).
```

As said before, in order to increase performance, we want to make the minimum choices possible, thus in the days without changes we can just copy the initial scheduling. In this

rule, it is defined a new atom y(RID,FORNID,ST,ET,DAY) that is equal to the input atom x(RID,FORNID,ST,ET,DAY) when the atom changedDay(DAY) is not true and the patient does not need rescheduling (not true forbidden(RID,_)).

Ensure that, for each patient, all his/her exams are assigned in the same day:

```
:- y(RID,_,_,_,DAY1), y(RID,_,_,_,DAY2), DAY1 > DAY2.
```

This rule ensures that for each patient all the exams are scheduled in the same day. As said above, the atom y(RID,_,_,_,DAY) represents an assignment of an exam, therefore there cannot be two y atoms with the same RID but different day DAY. For, with the same meaning but better performance, with DAY1 > DAY2

Common rules with the PAC scheduling problem:

```
:- y(RID,FORNID1,ST1,_,_), y(RID,FORNID2,ST2,_,_), phase(FORNID1,ORD1),
   phase(FORNID2,ORD2), ORD2 < ORD1, ST1 < ST2.
:- #count{FORNID: y(RID,FORNID,ST,ET,_), T>= ST, T<ET} > 1, reg(RID,_,_),
   time(_,T).
:- #count{RID: y(RID,FORNID,ST,ET,DAY), T>= ST, T<ET} > N,
   examLoc(FORNID,_,DAY,_,_,N), time(_,T).
{operatorY(ID, FORNID, DAY) : operator(ID, FORNID, DAY)} == NOP :-
   examLoc(FORNID,NOP, DAY,_,_,_), exam(RID,FORNID,SESSION),
   y(RID,_,_,_,DAY).
:- operatorY(ID, FORNID1, DAY), operatorY(ID, FORNID2, DAY), FORNID1 <
   FORNID2.
```

The rules reported above are in common with the encoding of the second sub-problem of the PAC scheduling problem (Chapter 3.5.1) but instead of the x(REGID,FORNID,ST,ET,DAY) atom we have y(RID,FORNID,ST,ET,DAY) and, instead of the operator(ID,FORNID,DAY) atom we have operatorY(ID,FORNID,DAY).

Compute the amount of changes in the operators assignment after rescheduling:

```
changedOperators(N) :- #count{ID: operatorY(ID,FORNID1,DAY),
    operators(ID,FORNID2,DAY), FORNID1 != FORNID2} = N.
```

In this rule, it is computed the number of changes in the operators assignment with respect to the initial scheduling and the new atom changedOperators(N) is de ned. This atom will be used later in the optimization.

Compute, for each patient, the minimal total duration of his/her pre-operative assessment:

```
cost(RID, TOT) :- TOT = #sum{DUR,FORNID : exam(RID, FORNID,DUR)},
    y(RID,_,_,_,DAY1), x(RID,_,_,_,DAY2), DAY1 != DAY2.
```

In this rule, a new atom cost(RID, TOT) is de ned in which it is reported, for each patient, the minimum total duration of his/her pre-operative assessment. Since, this atom will be used later in the optimization only for the patients rescheduled in a different day from the initial scheduling, it is computed only when the day after rescheduling (DAY1) is different from the initial scheduled day (DAY2).

Optimization, minimize the difference between the patient assigned day before and after rescheduling:

```
#minimize{|DAYX-DAY|@4,RID: y(RID,_,_,_,DAY),x(RID,_,_,_,DAYX),
    changedDay(DAY)}.
```

This weak constraint minimizes the difference between the patient assigned day before and after rescheduling, this is applied only if the atom changedDay(DAY) is true, otherwise we just copy the starting scheduling. This weak constraint has the higher priority in the optimization.
Optimization, minimize the difference between exams starting time before and after rescheduling:

```
#minimize{|STx-STy|@3,RID,FORNID:
    y(RID,FORNID,STy,_,DAY),x(RID,FORNID,STx,_,DAY)}.
```

This rule minimizes the difference between exams starting time before and after rescheduling. This is applied only for the patients assigned, after rescheduling, in the same day of the initial scheduling.

Optimization, minimize the total duration of the PAC for patients rescheduled in a new day:

```
#minimize{|ET-ST-TOT|@2,RID: y(RID,0,ST,_,_), y(RID,23,_,ET,_),
    cost(RID,TOT)}.
```

As done for the optimization of the second sub-problem in section 3.5.1, here we apply the minimization with lower bound. For patients rescheduled in a new day with respect to the initial scheduling, it is minimized the time spent in the hospital, 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.
Optimization, minimize the difference in operators assignment:

```
#minimize{N@1: changedOperators(N)}.
```

Finally, it is minimized the difference between the operators' assigned exam area before and after rescheduling. More in detail, it is minimized the value N of the atom changedOperators(N) computed in rule 17.

5.3 Results of the rescheduling

In this chapter, the results of an empirical analysis of the PAC rescheduling problem are presented. As for the PAC scheduling problem, the ASP system used was clingo (Gebser et al. (2016a) 5.4.0), using parameter --restart-on-model for faster optimization and --parallel-mode 6 for parallel execution. The time limit was set to 60 seconds.

#Patients to reschedule	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
1	4.861	6.004	6.185	4.932	3.213
2	6.039	8.207	7.465	8.62	2.945
4	8.55	60.06	8.437	60.099	3.342

Table 5.1 Time spent by the solver in computing the instances

Since the rescheduling problem consists of, given a scheduling that cannot be implemented, move operators and patients to nd a new assignment, the initial scheduling was taken from the results of the PAC scheduling problem.

5.3.1 First scenario: Patient requires a new day

In this scenario, it is not possible for one (or more) patient to have the pre-operative assessment on his/her planned day in the initial scheduling. Thus, a new day and a new starting time for all her/his exams must be found, while other patients should have no changes but, in principle, they can be moved in order to leave space for the patients to reschedule.

(a) Before rescheduling (assigned to day 2) (b) After rescheduling (assigned to day 1)

Figure 5.1 Representation of a patient moved a day before after rescheduling

In gure 5.1 it is reported an example of a patient, in this case patient with id 12 (on y-axis), that in the initial scheduling was assigned in a day in which he could not be present, thus after rescheduling, it has been moved to the day before.
We tested 5 instances with 1, 2 and 4 patients to be rescheduled and in 13 out of 15 of them, as shown in Table 5.1, the scheduler was able to assign all the patients optimally in few seconds respecting all the constraints. In fact, by inspecting the re-scheduling results, in the new assignment there is no waiting time between the exams and no overlapping, moreover the patients that did not have to be rescheduled have no changes in their assignment.

5.3.2 Second scenario: Operators are not present

In this scenario, it is considered the case in which an operator (or more) is not available for some days. If, in those days, it is possible to replace the operator so that the exam area in which (s)he was assigned can still be activated then no changes are needed; otherwise, patients requiring that exam area must be rescheduled in a new day.

(a) Day before rescheduling (b) Day after rescheduling

Figure 5.2 Representation of patients moved because of missing operator

In gure 5.2 it is reported an example of the changes done in a day, after rescheduling. Since an operator was missing, it was not possible to activate the exam area with id 6; thus all the patients requiring that exam area have been rescheduled. As it is shown in gure 5.2b, two patients (38 and 30) have been rescheduled in the day shown and both require the exam area with id 6. Moreover, the gure shows that there is some free space (i.e. waiting time), between some exams. This space cannot be reduced because the patients won't be able to fully complete all their exams without overlapping and they could not be assigned in an other day because the day distance is the higher priority in the minimization. Furthermore, since we are using synthetic data, it can happen to have combinations that lead to unsatis able instances. In fact, one instance tested in this scenario returned unsatis able because, due to a missing operator and, as a consequence, the impossibility to activate all the necessary exam areas, it was not possible to reschedule some patients.

5.3.3 Third scenario: Exam areas are not available

In this scenario, it is supposed that an (or more) exam area is not available. Thus, the patients requiring it must be rescheduled in a new day. As in the previous scenario, patients not

interested by the unavailability should not be moved. However, since we want all the patients to be assigned, every patient can have changes in their appointment if necessary.

(a) Day before rescheduling (b) Day after rescheduling

Figure 5.3 Representation of patients moved because of an unavailable exam area

In gure 5.3, it is reported an example of this scenario, in which the exam area with id 4 has been put unavailable for three days. As it is shown in gure 5.3b patients requiring that exam area have been moved, but non-rescheduled patients still have the same starting time for each exam, as a consequence, in the optimum, one rescheduled patient has some waiting time between two exams. In fact, higher priority is given to have the same exams starting time for patients not moved and then minimize the total length of those, instead, moved to a new day.

Chapter 6

Web Application

The solution provided in this thesis should be used by hospital operators, so we developed a web application to allow to intuitively use the ASP solutions and CLINGO through a graphical interface.

Figure 6.1 Representations of web application architecture

The system architecture is reported in gure 6.1. The graphical interface has been developed using the framework angular, as backend we used nodejs and graphql for data queries and it is used a wrapper to call the solver (clingo).
The work ow is the following:

- the user selects a period to plan; data are shown accordingly to the selection, and then (s)he can start the rst planning. This corresponds to the solution of the rst sub-problem of the PAC scheduling problem.
- Once the rst planning is completed, the result is stored and shown. Then, the operator has to select all the exams necessary for each patient. Now, the detailed planning can start. This corresponds to the second sub-problem of the PAC problem.
- The nal, obtained, scheduling is stored and shown with the precise starting time of each exam.

Figure 6.2 First planning page

6.1 Demo

6.1.1 First planning page

In gure 6.2 the rst page is shown; here the operator can select the period to plan and the solver timeout. In this page, data are read-only because the application has been thought integrated with the hospital database, therefore with updated data. Three tabs allow to check the elements involved in the planning:

- Patients tab, to monitor all the patients with a target date included in the planning period (Figure 6.2).
- Personnel tab, to monitor the operators available in the selected period (Figure 6.3). A drop down menu allow to select which operator to inspect.
- Providers tab, to check all the exams area enabled in the selected time period (Figure 6.4). A drop down menu allow to select the exam area to inspect.

Then, clicking on the "START PLANNING" button, the solver is called through the wrapper, the result is the solution of the rst PAC sub-problem. A loading screen will appear and once the solver nishes the planning (optimum found or timeout reached), the user is redirected to the detailed planning page in which it is shown the result.

Figure 6.3 First planning page, personnel tab

6.2 Detailed planning page

In the detailed planning page the user can look at the results of the rst planning, that can be selected through a drop down menu. At this point the operator has to select the precise exams each patient needs. Thus, by clicking on the edit icon a menu will appear listing all the patient exams where a check box allows to select which one is necessary as shown in gure 6.5.
Moreover, a second tab called "Graphics", displays a bar chart (gure 6.6) in which the bars represent the number of patient for each day of the scheduled period and four doughnuts (gure 6.7) chart that represent the percentage of patient scheduled over the total patient, grouped by the patients priority.
Finally, clicking on the "START PLANNING" button, again the solver is called and the detailed planning with the precise starting time of all the exams is computed. Once the solver terminates, the user is redirected to the nal page.

Figure 6.4 First planning page, providers tab

6.2.1 Final schedule page

As shown in gure 6.8, the nal schedule page is made up of a drop down menu, from which it can be selected the scheduled period, and two tabs: patients info tab and graphics tab. In the patients info tab, patients are sorted in a table and clicking on the icon in the "Show Exams" column a table will appear listing all the exams and their starting time.
In gure 6.9 and gure 6.10 are reported the charts shown in the graphics tab, that are two gantt chart, where the rst one reports on the x-axis the timeline and on the y-axis the patients and each segment corresponds to an exam, instead the second one reports the same chart with the exam areas on the y-axis.

Figure 6.5 Detailed planning page, edit patient exams

Figure 6.6 Detailed planning page, bar chart