UNIVERSITY OF GENOVA

PHD PROGRAM IN COMPUTER SCIENCE AND SYSTEMS ENGINEERING

Application of Artificial Intelligence declarative methods for Solving Operating Room Scheduling problems in Hospital Environments

by

Muhammad Kamran Khan

Thesis submitted for the degree of *Doctor of Philosophy* (34° cycle)

September 2022

Prof. Marco Maratea	Supervisor
Dr. Giuseppe Galatà	Co-Supervisor
Prof. Giorgio Delzanno	Head of the PhD program

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

Abstract

Digital health is a relatively new but already important field in which digitalization meets the need to automatically and efficiently solve problems in healthcare to improve the quality of life for patients. The need to efficiently solve some of these problems has become even more pressing due to the Covid-19 pandemic that significantly increased stress and demand on hospitals. Hospitals have long waiting lists, surgery cancellations, and even worse, resource overload—issues that negatively impact the level of patient satisfaction and the quality of care provided. Within every hospital, operating rooms (ORs) are an important unit. The Operating Room Scheduling (ORS) problem is the task of assigning patients to operating rooms, taking into account different specialties, lengths and priority scores of each planned surgery, operating room session durations, and the availability of beds for the entire length of stay both in the Intensive Care Unit and in the wards. A proper solution to the ORS problem is of primary importance for the quality of healthcare service and the satisfaction of patients in hospital environments. In this thesis, we provide several contributions to the ORS problem. We first present a solution to the problem based on Knowledge Representation and Reasoning via modeling and solving approaches using Answer Set Programming (ASP). This first basic solution builds on a previous solution but takes into account explicitly beds and ICU units because in the pandemic we understood how important and limiting they were. Moreover, we also present an ASP solution for the rescheduling problem, i.e., when the off-line schedule cannot be completed for some reasons, and a further extension where surgical teams are also considered. Another technical contribution is a second solution for the basic ORS problem with beds and an ICU unit, whose modeling departs from the guidelines previously used and shows efficiency improvements. Finally, we introduce a web framework for managing ORS problems via ASP that allows a user to insert the main parameters of the problem, solve a specific instance, and show results graphically in real time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Digital health is a relatively new but already important field in which digitalization meets the need for automatically and efficiently solving problems in healthcare. The need to efficiently solve some of these problems has become even more pressing due to the Covid-19 pandemic that significantly increased stress and demand on hospitals. Hospitals have long waiting times, surgery cancellations, and even worse, resource overload, issues that negatively impact the level of patient satisfaction and the quality of care provided. Within every hospital, Operating Rooms (ORs) are an important unit. Due to the personnel involved and the needed infrastructure, ORs entail significant costs. Moreover, the cost per minute for an unused OR can easily go over €30 (Macario 2010). In general, ORs account for approximately 33% of the overall budget in hospitals (Meskens et al. 2013). Other than ORs, another central element when dealing with ORs-related problems are beds: For example, between Q1 2010/11 and Q3 2018/19, the total number of NHS hospital beds decreased by 12%, from 144,455 to 127,589 [1]. The latest official UK National Health Service (NHS) data shows [2] that in Q2 2019/20 (from July to September 2019), 70 out of 202 trusts had an average bed occupancy $> 90\%$. Hospitals cannot operate at or close at 100% occupancy, as spare bed capacity is needed to accommodate variations in demand and ensure that patients can flow through the system. Thus, a proper solution to the enriched problem in which both ORs and beds are explicitly considered is of primary importance for the healthcare service quality and the satisfaction of patients in hospital environments. The Operating Room Scheduling (ORS)

[1] https://www.nuffieldtrust.org.uk/resource/hospital-bed-occupancy

[2] See data stored on https://www.england.nhs.uk/statistics/statistical-work-areas/bed-availability-and-occupancy/bed-data-overnight/

problem is the task of assigning patients to operating rooms, taking into account different specialties, lengths and priority scores of each planned surgery, as well as operating room session durations. Enhancements include the availability and management of beds for the entire length of stay both in the Intensive Care Unit and in the wards, and of surgical teams involving anesthetists and surgeons.

1.2 State-of-the-art

There are a significant number of approaches in the literature for solving problems in digital health related to ORs. Aringhieri et al. (2015a) addressed the joint OR planning (MSS) and scheduling problem, described as the allocation of OR time blocks to specialties together with the subsets of patients to be scheduled within each time block over a one-week planning horizon. They developed a 0-1 linear programming formulation of the problem and used a two-level meta-heuristic to solve it. Its effectiveness was demonstrated through numerical experiments carried out on a set of instances based on real data and resulted, for benchmarks of 80-100 assigned registrations, in a 95-98% average OR utilization rate, for a number of ORs ranging from 4 to 8. The execution times were around 30-40 seconds. In Landa et al. (2016), the same authors introduced a hybrid two-phase optimization algorithm which exploits neighborhood search techniques combined with Monte Carlo simulation, in order to solve the joint advance and allocation scheduling problem, taking into account the inherent uncertainty of surgery durations. In both the previous works, the authors solve the bed management part of the problem limited to weekend beds while assuming that each specialty has its own post-surgery beds from Monday to Friday with no availability restriction. In Aringhieri et al. (2015b), some of the previous authors face the bed management problem for all the days of the week, with the aim to level the post-surgery ward bed occupancies during the days using a Variable Neighbourhood Search approach. What concerns, instead, the availability of surgical teams, Meskens et al. (2013) considered the surgical teams in the computation of an OR schedule, and developed a model using Constraint Programming (CP) with multiple constraints such as availability, staff preferences and affinities among surgical teams. They optimize the use of ORs by minimizing makespan and maximizing affinities among surgical team members. The effectiveness of their proposed method for improving surgical cases was evaluated using real data from a hospital. Hamid et al. (2019) incorporated the decision-making styles (DMS) of the surgical team to improve the compatibility level by considering constraints such as the availability of material resources, priorities of patients, and availability, skills, and competencies of the surgical team. They developed a multi-objective mathematical

model to schedule surgeries. Two metaheuristics, namely Non-dominated Sorting Genetic Algorithm and Multi-Objective Particle Swarm Optimization, were developed to find pareto-optimal solutions. Xiang et al. (2015) proposed an Ant Colony Optimization (ACO) approach to surgical scheduling taking into account all resources in the entire process of a surgery. The problem was represented as an extended multi-resource constrained flexible job shop scheduling problem, which was solved using a two-level hierarchical graph to integrate sequencing job and allocating resources. To evaluate the efficiency of ACO, a Discrete Event System (DES) model of an OR system was developed in the simulation platform SIMIO. Monteiro et al. (2015) developed a comprehensive multi-objective mathematical model using epsilon-constraint method coupled to the CPLEX solver. Vijayakumar et al. (2013) used Mixed Integer Programming (MIP) model for multi-day, multi-resource, patient-priority-based surgery scheduling. A First Fit Decreasing Algorithm was developed. From a solution time perspective, their model took hours and in most cases was unable to optimally solve the problem. Belkhamsa et al. (2018) proposed two meta heuristics, an Iterative Local Search (ILS) approach and Hybrid Genetic Algorithm (HGA) to solve a daily surgery scheduling problem. Zhou et al. (2016) developed an Integer Programming model for optimal surgery schedule of assigning patients to different resources in any surgical stage. They used Lagrangian Relaxation algorithm and solved the sub problem by using branch and bound. They verified their model using real data instances from an Hospital. A common issue with all such solutions seem to be computation time and scalability; moreover, rescheduling and the development of web applications are rarely considered.
A preliminary solution to the basic ORS problem based on ASP but not considering beds, ICU units and surgical teams is presented in Dodaro et al. (2018, 2019).

1.3 Contribution of the thesis

The main contributions of the thesis are the following:

- We provide an ASP encoding for solving the ORS problem involving beds management by means of a modular addition of ASP rules starting from the specification of the problem. This has resulted in the article "An ASP-based Solution for Operating Room Scheduling with Beds Management", published in Proc of the 3rd International Joint Conference on Rules and Reasoning(RuleML+RR 2019), LNCS 11784, pages 67-81, 2019.

- We run an experimental analysis assessing the good performance of our ASP solution and a scalability analysis w.r.t. scheduling length, considering a number of scenarios w.r.t. beds availability. This has been added as a part of the article "Operating Room (Re)Scheduling with Bed Management via ASP", published in Theory and Practice of Logic Programming (TPLP), Vol. 22(2), pages 229–253, 2022.
DOI:https://doi.org/10.1017/S1471068421000090

- We provide an ASP encoding and an experimental analysis for the ORS problem enriched with the management of surgical teams composed of anesthetists and surgeons. This has resulted in the publication of the article "An ASP-based Solution for Operating Room Scheduling with Surgical Teams in Hospital Environments", in Post-Proc. of the 19th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2020), LNAI 12414, pages 192–204, 2021.

- We present an alternative ASP encoding and an experimental analysis for the ORS problem enriched with the management of surgical teams that has been developed in collaboration with Prof. Martin Gebser from the University of Klagenfurt.

- We provide a further ASP encoding and an experimental analysis for the rescheduling of the basic ORS problem with beds, which comes into play when the scheduling cannot be implemented due to unexpected events, e.g., the unavailability of a patient. This work has been inserted as part of the article "Operating Room (Re)Scheduling with Bed Management via ASP", published in Theory and Practice of Logic Programming (TPLP), Vol. 22(2), pages 229–253, 2022.
DOI:https://doi.org/10.1017/S1471068421000090

- We describe a Graphical User Interface that employs our ASP solution to produce real-time scheduling of operating rooms and to help users make use of it with minimal effort by allowing them to impose the values of the main parameters of the problem. Also, this work has been inserted as part of the already mentioned article "Operating Room (Re)Scheduling with Bed Management via ASP", published in Theory and Practice of Logic Programming (TPLP), Vol. 22(2), pages 229–253, 2022.
DOI:https://doi.org/10.1017/S1471068421000090

1.4 Structure of the thesis

The thesis is structured as follows: **Chapter 2** contains needed preliminaries about ASP, presenting the syntax of the language, and then syntactic shortcuts are employed. Then, **Chapter 3** presents the problem description of the basic ORS problem and of the enhancements then considered in the thesis. The related ASP encoding involving bed management in the wards and ICU unit is presented in **Chapter 4.** The same chapter highlights the Knowledge Representation (KR) capabilities of ASP to represent specifications in a natural and intuitive way. Furthermore, the also chapter presents experimental analysis considering several scenarios with respect to scheduling length, beds abundance and shortage. **Chapter 5** presents two different encodings as ASP solutions for the ORS problem with surgical teams. The first encoding shows a solution following the guidelines of previous work while the alternative encoding is a computationally optimized solution. **Chapter 6** presents the experimental analysis of the two ASP solutions for the ORS problem with surgical teams with respect to OR, surgeons and anesthetists work time efficiencies. **Chapter 7** provides a further ASP encoding and experimental analysis for the rescheduling of the basic ORS problem, which comes into play when the scheduling cannot be implemented due to unexpected events, e.g., the unavailability of a patient. **Chapter 8** presents a Graphical User Interface for our ASP solution, which allows easy user interface to produce real-time scheduling of operating rooms. **Chapter 9** presents the state-of-the-art literature on different techniques for solving the ORS problem, and it also presents other scheduling problems where ASP has been employed. **Chapter 10** finally presents a conclusion by recapping the contributions of this thesis. The chapter then presents opportunities for future work, i.e., the open issues and promising research directions related to ORS problem with bed management and surgical teams, and beyond.

Chapter 2

Background on ASP

Answer Set Programming (ASP) Brewka et al. (2011) is a programming paradigm developed in the field of nonmonotonic reasoning and logic programming. In this section, we will overview the language of ASP. More detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in Alviano et al. (2015); Brewka et al. (2011); Calimeri et al. (2020); Gebser et al. (2015). Hereafter, we assume the reader is familiar with logic programming conventions.

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

- *#count*, number of terms;

- *#sum*, sum of integers.

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

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

where $a_1, \ldots, a_n$ are standard atoms, $b_1, \ldots, b_k$ are atoms, $b_{k+1}, \ldots, b_m$ are standard atoms, and $n, k, m \geq 0$. A literal is either a standard atom a or its negation *not a*. The disjunction $a_1 \lor \ldots \lor a_n$ is the *head* of r, while the conjunction $b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$ is its *body*. Rules with empty body are called *facts*. Rules with empty head are called *constraints*. A variable that appears uniquely in set terms of a rule r is said to be *local* in r, otherwise it is a *global* variable of r. An ASP program is a set of *safe* rules, where a rule r is *safe* if the following conditions hold: *(i)* for each global variable X of r there is a positive standard atom ℓ in the body of r such that X appears in ℓ; and *(ii)* each local variable of r appearing in a symbolic set $\{Terms : Conj\}$ also appears in a positive atom in *Conj*.
A *weak constraint* Buccafurri et al. (2000) ω is of the form:

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

where w and l are the weight and level of ω, respectively. (Intuitively, $[w@l]$ is read "as weight w at level l", where weight is the "cost" of violating the condition in the body, whereas levels can be specified for defining a priority among preference criteria). An ASP program with weak constraints is $\Pi = \langle P, W \rangle$, where P is a program and W is a set of weak constraints. A standard atom, a literal, a rule, a program or a weak constraint is *ground* if no variables appear in it.

Semantics. Let P be an ASP program. The *Herbrand universe* U_P and the *Herbrand base* B_P of P are defined as usual. The ground instantiation G_P of P is the set of all the ground instances of rules of P that can be obtained by substituting variables with constants from U_P. An *interpretation* I for P is a subset I of B_P. A ground literal ℓ (resp., *not* ℓ) is true w.r.t. I if $\ell \in I$ (resp., $\ell \notin I$), and false otherwise. An aggregate atom is true w.r.t. I if the evaluation of its aggregate function (i.e. the result of the application of f on the multiset S) with respect to I satisfies the guard; otherwise, it is false.

A ground rule r is *satisfied* by I if at least one atom in the head is true w.r.t. I whenever all conjuncts of the body of r are true w.r.t. I.
A model is an interpretation that satisfies all rules of a program. Given a ground program G_P and an interpretation I, the *reduct* of G_P w.r.t. I is the subset G_P^I of G_P obtained by deleting from G_P the rules in which a body literal is false w.r.t. I. An interpretation I for P is an *answer set* (or stable model) for P if I is a minimal model (under subset inclusion) of G_P^I (i.e. I is a minimal model for G_P^I) Faber et al. (2011); Ferraris (2011). For a detailed discussion on the semantics of ASP programs with aggregates, we refer the reader to Alviano et al. (2015).
Given a program with weak constraints $\Pi = \langle P, W \rangle$, the semantics of Π extends from the basic case defined above. Thus, let $G_\Pi = \langle G_P, G_W \rangle$ be the instantiation of Π; a constraint $\omega \in G_W$ is violated by an interpretation I if all the literals in ω are true w.r.t. I. An *optimum answer set* for Π is an answer set of G_P that minimizes the sum of the weights of the violated weak constraints in G_W in a prioritized way.

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

Chapter 3

Problem Description

In this section, we provide an informal description of the ORS problem and its main requirements, organized into paragraphs containing the basic problem, the addition of bed management and the inclusion of surgical teams, respectively.

Basic ORS problem. As we already said in the introduction, most modern hospitals are characterized by a very long surgical waiting list, often worsened, if not altogether caused, by inefficiencies in operating room planning. A very important factor is represented by the availability of beds in the wards and, if necessary, in the ICU for each patient for the entire duration of their stay. This means that hospital planners have to balance the need to use the OR time with the maximum efficiency with an often reduced bed availability.
In this thesis, the elements of the waiting list are called *registrations*. Each registration links a particular surgical procedure, with a predicted surgery duration and length of stay in the ward and in the ICU, to a patient. The overall goal of the ORS problem is to assign the maximum number of registrations to the operating rooms (ORs), taking into account the availability of beds in the associated wards and in the ICU. This approach entails that the resource optimized is the one, between the OR time and the beds, that represents the bottleneck in the particular scenario analyzed.
As first requirement of the ORS problem, the assignments must guarantee that the sum of the predicted duration of surgeries assigned to a particular OR session does not exceed the length of the session itself: this is referred in the following as *surgery requirement*. Moreover, registrations are not all equal: they can be related to different medical conditions and can be in the waiting list for different periods of time. These two factors are unified in one concept: *priority*. Registrations are classified according to three different priority categories, namely P_1, P_2 and P_3. The first one gathers either very urgent registrations or the ones that have

been in the waiting list for a long period of time; it is required that these registrations are all assigned to an OR. Then, the registrations of the other two categories are assigned to the top of the ORs capacity, prioritizing the P_2 over the P_3 ones (*minimization*).

Bed management. Regarding the bed management part of the problem, we have to ensure that a registration can be assigned to an OR only if there is a bed available for the patient for the entire LOS. In particular, we have considered the situation where each specialty is related to a ward with a variable number of available beds exclusively dedicated to the patients associated to that specialty. This is referred in the following as *ward bed requirement*. The ICU is a particular type of ward that is accessible to patients from any specialty. However, only a small percentage of patients is expected to need to stay in the ICU. This requirement will be referred as the *ICU bed requirement*. Obviously, during their stay in the ICU, the patient does not occupy a bed in the specialty's ward.
In our model, a patient's LOS has been divided in the following phases:

- a LOS in the ward before surgery, in case the admission is programmed a day (or more) before the surgery takes place; and
- the LOS after surgery, which can be further subdivided into the ICU LOS and the following ward LOS.

Surgical teams. Additionally, in each specialty (considered to be 5 as target in small-medium-sized Hospitals) surgical teams are allocated with number of surgeons and anaesthetists every day as exemplified in Table 3.1. Tables 3.2 and 3.3, instead, show how many surgeons/anesthetists are available in each shift and for each specialty. However, surgeons assigned to a shift in a day are different from the ones assigned for the other shift of the same day, while the same anesthetists cover both shifts of the same day. Every surgeon works specifically for a number of hours every day; also surgeons in each specialty are assigned only to a single shift in a day, i.e., they either work in the morning (represented as shift 1, 3, 5, 7 and 9) or in evening shift (represented as shift 2, 4, 6, 8 and 10) as shown in Table 3.2. The anaesthetists are also linked to specialty and they also work for a fixed number of hours every day, but they can work together with surgeons during any shift of the day as shown in Table 3.3. In our model, we also assume that once a surgery is started in an OR it cannot be interrupted. Further, surgeons cannot operate on more than one patient at the same time. The overall goal is to assign the maximum number of registrations to the ORs, respecting the priorities, and taking into account the availability of respective surgical teams in a particular specialty for the complete surgery duration.

Table 3.1 Total number of surgeons and anaesthetists in each specialty.

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

Table 3.2 Surgeons availability for each specialty and in each day.

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

Table 3.3 Anaesthetists availability for each specialty and in each day.

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

Chapter 4

ASP solution for the ORS problem with bed management

```
(r1)  {x(R,P,O,S,D)} :- registration(R,P,_,_,SP,_,_), mss(O,S,SP,D).
(r2)  :- x(R,P,O,S1,_), x(R,P,O,S2,_), S1 != S2.
(r3)  :- x(R,P,O1,S,_), x(R,P,O2,S,_), O1 != O2.
(r4)  surgery(R,SU,O,S) :- x(R,_,O,S,_), registration(R,_,SU,_,_,_,_).
(r5)  :- x(_,_,O,S,_), duration(N,O,S), #sum{SU,R:surgery(R,SU,O,S)}>N.
(r6)  stay(R,(D-A)..(D-1),SP) :- registration(R,_,_,LOS,SP,_,A),
            x(R,_,_,_,D), A>0.
(r7)  stay(R,(D+ICU)..(D+LOS-1),SP) :- registration(R,_,_,LOS,SP,ICU,_),
            x(R,_,_,_,D), LOS>ICU.
(r8)  stayICU(R,D..(D+ICU-1)) :- registration(R,_,_,_,_,ICU,_),
            x(R,_,_,_,D), ICU>0.
(r9)  :- #count {R: stay(R,D,SP)}>AV, SP>0, beds(SP,AV,D).
(r10) :- #count {R: stayICU(R,D)}>AV, beds(0,AV,D).
(r11) :- N = totRegsP1 - #count{R: x(R,1,_,_,_)}, N > 0.
(r12) :~ N = totRegsP2 - #count{R: x(R,2,_,_,_)}. [N@3]
(r13) :~ N = totRegsP3 - #count{R: x(R,3,_,_,_)}. [N@2]
```

Figure 4.1 ASP encoding of the ORS problem.

Starting from the specifications in the previous section, here the ASP encoding of the basic ORS scheduling problem with bed management is is described in the ASP language, in particular following the input language of CLINGO, in a first subsection, while results of an experimental analysis on a number of scenarios regarding beds availability is presented in a second subsection.

It is important to emphasize here that, albeit CLINGO is compliant with the ASP-Core2 Calimeri et al. (2013) input language, it supports a richer syntax and slightly different semantics, see Gebser et al. (2015) for a formal description of the language. The next two sub-sections present the data model and the encoding itself, respectively.

4.1 ASP Encoding

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

- Instances of *registration(R,P,SU,LOS,SP,ICU,A)* represent the registrations, characterized by an id (*R*), a priority score (*P*), a surgery duration (*SU*) in minutes, the overall length of stay both in the ward and the ICU after the surgery (*LOS*) in days, the id of the specialty (*SP*) it belongs to, a length of stay in the ICU (*ICU*) in days, and finally a parameter representing the number of days in advance (*A*) the patient is admitted to the ward before the surgery.
- Instances of *mss(O,S,SP,D)* link each operating room (*O*) to a session (*S*) for each specialty (*SP*) and planning day (*D*) as established by the hospital Master Surgical Schedule (MSS).
- The OR sessions are represented by the instances of the predicate *duration(N,O,S)*, where *N* is the session duration.
- Instances of *beds(SP,AV,D)* represent the number of available beds (*AV*) for the beds associated to the specialty *SP* in the day *D*. The ICU is represented by giving the value 0 to *SP*.

The output is an assignment represented by atoms of the form *x(R,P,O,S,D)*, where the intuitive meaning is that the registration *R* with priority *P* is assigned to the OR *O* during the session *S* and the day *D*. It is important to emphasize here that the priority *P* is not actually needed in the output, however it is included because it improves the readability of the encoding presented in the subsequent section.

Encoding. The related encoding is shown in Figure 4.1, and is described in the following. Rule (r_1) guesses an assignment for the registrations to an OR in a given day and session among the ones permitted by the MSS for the particular specialty the registration belongs to. The same registration should not be assigned more than once, in different OR sessions. This is assured by constraints (r_2) and (r_3). Note that in our setting there is no requirement that every registration must actually be assigned.

Surgery requirement. With rules (r_4) and (r_5) we impose that the total length of surgery durations assigned to a session is less than or equal to the session duration.
Rules (r_6)-(r_{10}) deal with the presence and management of beds. In particular, rule (r_6) assigns a bed in the ward to each registration assigned to an OR, for the days before the surgery. Rule (r_7) assigns a ward bed for the period after the patient was dismissed from the ICU and transferred to the ward. Rule (r_8) assigns a bed in the ICU.

Ward bed requirement. Rule (r_9) ensures that the number of patients occupying a bed in each ward for each day is never larger than the number of available beds.

ICU bed requirement. Finally, rule (r_{10}) performs a similar check as the one in rule (r_9), but for the ICU.

Minimization. We remind that we want to be sure that every registration having priority 1 is assigned, then we assign as much as possible of the others, giving precedence to registrations having priority 2 over those having priority 3. This is accomplished through constraint (r_{11}) for priority 1 and the weak constraints (r_{12}) and (r_{13}) for priority 2 and 3, respectively, where *totRegsP1*, *totRegsP2*, and *totRegsP3* are constants representing the total number of registrations having priority 1, 2 and 3, respectively.
Minimizing the number of unassigned registrations could cause an implicit preference towards the assignments of the registrations with shorter surgery durations. To avoid this effect, one can consider to minimize the idle time; however, this is in general slower from a computational point of view and often unnecessary, since the preference towards shorter surgeries is already mitigated by our three-tiered priority schema.

Remark. We note that, given that the MSS is fixed, our problem and encoding could be decomposed by considering each specialty separately in case the beds are not a constrained resource, as will be the case for one of our scenario. We decided not to use this property because (i) this is the description of a practical application that is expected to be extended over time and to correctly work even if the problem becomes non-decomposable, e.g. a (simple but significant) extension in which a room is shared among specialties leads to a problem which is not anymore decomposable, and (ii) it is not applicable to all of our scenario. Additionally, even not considering this property at the level of the encoding, the experimental analysis that we will present is already satisfactory for our use case even when the decomposition could be applied.

Table 4.1 Bed availability for each specialty and in each day in scenario A.

Specialty	Monday	Tuesday	Wednesday	Thursday	Friday
0 (ICU)	40	40	40	40	40
1	80	80	80	80	80
2	58	58	58	58	58
3	65	65	65	65	65
4	57	57	57	57	57
5	40	40	40	40	40

4.2 Experimental analysis

In this section, we report about the results of an empirical analysis of the ORS encoding. Data have been randomly generated but having parameters and sizes inspired by real data. Experiments were run on a Intel Core i7-7500U CPU @ 2.70GHz with 7.6 GB of physical RAM. The ASP system used was CLINGO Gebser et al. (2016), version 5.5.2, with the "−−restart−on−model" option enabled.

4.2.1 ORS benchmarks

The employed encoding is composed by the ASP rules $(r_1),\ldots,(r_{13})$ from Figure 4.1. The test cases we have assembled are based on the requirements of a typical small-medium size Italian hospital, with five surgical specialties to be managed over the widely used 5-day planning period. Three different scenarios were assembled. The first one (scenario A) is characterized by an abundance of available beds, so that the constraining resource becomes the OR time. For the second one (scenario B), we reduced the number of beds, in order to test the encoding in a situation with plenty of OR time but few available beds. Scenario B is pushed further in scenario C, where the number of beds is further reduced, to test our encoding also in this extreme situation. Each scenario was tested 10 times with different randomly generated inputs.
The characteristics of the tests are the following:

- 3 different benchmarks, comprising a planning period of 5 working days, and different numbers of available beds, as reported in Table 4.1, Table 4.2 and Table 4.3 for scenario A, B, and C, respectively;
- 10 ORs, unevenly distributed among the specialties;

Table 4.2 Bed availability for each specialty and in each day in scenario B.

Specialty	Monday	Tuesday	Wednesday	Thursday	Friday
0 (ICU)	4	4	5	5	6
1	20	30	40	45	50
2	10	15	23	30	35
3	10	14	21	30	35
4	8	10	14	16	18
5	10	14	20	23	25

Table 4.3 Bed availability for each specialty and in each day in scenario C.

Specialty	Monday	Tuesday	Wednesday	Thursday	Friday
0 (ICU)	4	4	5	5	6
1	10	15	20	25	30
2	7	10	11	14	18
3	7	10	13	16	20
4	4	6	8	11	13
5	6	9	12	15	18

- 5 hours long morning and afternoon sessions for each OR, summing up to a total of 500 hours of ORs available time for each benchmark;
- 350 generated registrations, from which the scheduler will draw the assignments. In this way, we simulate the common situation where a hospital manager takes an ordered, w.r.t. priorities, waiting list and tries to assign as many elements as possible to each OR.

The surgery durations have been generated assuming a normal distribution, while the priorities have been generated from a uneven distribution of three possible values (with weights respectively of 0.20, 0.40 and 0.40 for registrations having priority 1, 2 and 3, respectively). The lengths of stay (total LOS after surgery and ICU LOS) have been generated using a truncated normal distribution, in order to avoid values less than 1. In particular for the ICU, only a small percentage of patients have been generated with a predicted LOS while the large majority do not need to pass through the ICU and their value for the ICU LOS is fixed to 0. Finally, since the LOS after surgery includes both the LOS in the wards and in the ICU, the value generated for the ICU LOS must be less than or equal to the total LOS after surgery.
The parameters of the test have been summed up in Table 4.4. In particular, for each specialty (1 to 5), we reported the number of registrations generated, the number of ORs assigned to the specialty, the mean duration of surgeries with its standard deviation, the mean LOS after

Table 4.4 Parameters for the random generation of the scheduler input.

Specialty	Reg.	ORs	Surgery Duration (min) mean (std)	LOS (d) mean (std)	ICU (%)	ICU LOS (d) mean (std)	LOS (d) before surgery
1	80	3	124 (59.52)	7.91 (2)	10	1 (1)	1
2	70	2	99 (17.82)	9.81 (2)	10	1 (1)	1
3	70	2	134 (25.46)	11.06 (3)	10	1 (1)	1
4	60	1	95 (19.95)	6.36 (1)	10	1 (1)	0
5	70	2	105 (30.45)	2.48 (1)	10	1 (1)	0
Total	350	10					

Table 4.5 Scheduling results for the scenario A benchmark.

Assigned Registrations					
Priority 1	Priority 2	Priority 3	Total	OR Time Eff.	Bed Occupancy Eff.
62 / 62	132 / 150	72 / 138	266 / 350	96.6%	52.0%
72 / 72	128 / 145	64 / 133	264 / 350	95.6%	51.0%
71 / 71	132 / 132	69 / 147	272 / 350	96.7%	96.7%
66 / 66	138 / 142	57 / 142	261 / 350	96.2%	50.7%
79 / 79	119 / 130	67 / 141	265 / 350	96.0%	51.9%
67 / 67	131 / 131	66 / 152	264 / 350	96.6%	53.8%
66 / 66	121 / 132	69 / 152	256 / 350	96.0%	49.8%
69 / 69	130 / 135	68 / 146	267 / 350	96.8%	51.6%
60 / 60	139 / 153	59 / 137	258 / 350	96.8%	50.8%
68 / 68	138 / 142	57 / 139	263 / 350	95.2%	51.3%

the surgery with its standard deviation, the percentage of patients that need to stay in the ICU, the mean LOS in the ICU with its standard deviation and, finally, the LOS before the surgery (i.e. the number of days, constant for each specialty, the patient is admitted before the planned surgery is executed).

4.2.2 Results

Results of the experiments are reported for scenario A in Table 4.5, for scenario B in Table 4.6, and for scenario C in Table 4.7.
A time limit of 60 seconds was given and each scenario was run 10 times with different input registrations. No run manages to reach the optimal solution within the chosen timeout. However, the quality of the solution improves only marginally even if the timeout is extended up to 5 minutes, which is the largest timeout value we have tried in view of a practical

Table 4.6 Scheduling results for the scenario B benchmark.

Assigned Registrations					
Priority 1	Priority 2	Priority 3	Total	OR Time Eff.	Bed Occupancy Eff.
62 / 62	106 / 150	13 / 138	181 / 350	66.3%	92.7%
72 / 72	77 / 145	43 / 133	192 / 350	67.5%	94.2%
71 / 71	80 / 132	38 / 147	189 / 350	68.2%	96.1%
66 / 66	81 / 142	41 / 142	188 / 350	71.4%	93.4%
79 / 79	90 / 130	20 / 141	189 / 350	69.0%	94.1%
67 / 67	95 / 131	25 / 152	187 / 350	66.5%	93.9%
66 / 66	92 / 132	30 / 152	188 / 350	71.8%	94.1%
69 / 69	84 / 135	36 / 146	189 / 350	68.7%	92.7%
60 / 60	91 / 153	34 / 137	185 / 350	69.7%	94.1%
68 / 68	82 / 142	35 / 139	185 / 350	69.3%	95.1%

Table 4.7 Scheduling results for the scenario C benchmark.

Assigned Registrations					
Priority 1	Priority 2	Priority 3	Total	OR Time Eff.	Bed Occupancy Eff.
62 / 62	43 / 150	12 / 138	117 / 350	43.1%	85.8%
71 / 71	41 / 132	6 / 147	118 / 350	42.9%	93.2%
66 / 66	40 / 142	11 / 142	117 / 350	42.5%	92.0%
79 / 79	38 / 130	7 / 141	124 / 350	44.0%	93.8%
67 / 67	42 / 131	9 / 152	118 / 350	41.9%	89.8%
69 / 69	39 / 135	13 / 146	121 / 350	45.3%	94.4%
60 / 60	48 / 153	10 / 137	118 / 350	45.3%	91.2%
68 / 68	38 / 143	13 / 139	119 / 350	44.6%	91.5%

use of the program. For this reason, we decided to keep the timeout value at 60 seconds, which makes the program adapt to be used also for quick testing of "what-if" scenarios and simulations. For each satisfiable instance out of the 10 runs executed, the tables report in the first three columns the number of the assigned registrations out of the generated ones for each priority, and in the remaining two columns a measure of the total time occupied by the assigned registrations as a percentage of the total OR time available (indicated as "OR Time Eff." in the tables) and the ratio between the beds occupied after the planning to the available ones before the planning (labeled as "Bed Occupancy Eff." in the tables). As a general observation, these results show that our solution is able to utilize efficiently whichever resource is more constrained: on scenario A, our solution manages to reach a very high efficiency, over 95%, in the use of OR time, while in scenario B achieves an efficiency

of bed occupancy between 92% and 95%, and over 85% even in the extreme case represented by Scenario C. The same set of generated registrations was used in each scenario, so that the differences in the results can be ascribed only to the different bed configurations. Taking into consideration a practical use of this solution, the user would be able to individuate and quantify the resources that are more constraining and take the appropriate actions. This means that the solution can also be used to test and evaluate "what if" scenarios.
Finally, in Figure 4.2 we (partially) present the results achieved on one instance (i.e. the first instance of Table 4.5, Table 4.6, and Table 4.7) with 350 registrations for 5 days. Each bar represents the total number of available beds for specialty 1, as reported in Table 4.1 for the plot at the top, Table 4.2 for the middle one, and Table 4.3 for the bottom one, for each day of the week, from Monday through Friday. The colored part of the bars indicates the amount of occupied beds while the gray part the beds left unoccupied by our planning.

4.2.3 Scalability Analysis

We have performed a scalability analysis on the performance of employed ASP solver and encoding for ORS with bed management w.r.t schedule length.

Evaluation. The characteristics of the tests for each scenario are the following:

- We consider 7 different benchmarks with planning period of 1, 2, 3, 5, 7, 10, and 15 working days;
- For each benchmark the total number of randomly generated registrations were 70 per day, i.e. 70, 140, 210, 350, 490, 700, and 1050 for 1, 2, 3, 5, 7, 10, and 15 days, respectively;
- 5 specialties for each benchmark;
- 10 ORs unevenly distributed among the specialties;
- 5 hours long morning and afternoon shifts for each OR summing to 100, 200, 300, 500, 700, 1000, and 1500 hours of OR available time for the 7 benchmarks;
- An execution time of 60 seconds was given to each instance.

Table 4.8 shows the distribution of the total number of randomly generated registrations for each benchmark. Further, this table also shows the distribution of ORs for each speciality, i.e to speciality 1 three ORs are assigned, to speciality 2, 3 and 5, two ORs are assigned, while to speciality 4 only one OR is allocated. Each speciality is considered as a ward with variable number of available beds.

Table 4.8 Total number of randomly generated registrations for each benchmark

Specialty	Registrations							ORs
	15-day	10-day	7-day	5-day	3-day	2-day	1-day	
1	240	160	112	80	48	32	16	3
2	210	140	98	70	42	28	14	2
3	210	140	98	70	42	28	14	2
4	180	120	84	60	36	24	12	1
5	210	140	98	70	42	28	14	2
Total	1050	700	490	350	210	140	70	10

Moreover, we kept also for this analysis the parameters for the random generation of scheduler input from Table 4.4.

Scenario A. The results of scenario A are reported in Table 4.9, that shows averages of results for satisfiable instances with abundance of available beds. It should be noted that each benchmark (15, 10, 7, 5, 3, 2 and 1 day) represents here the average of the satisfiable runs with different randomly generated inputs.
As we can see, Table 4.9 contains seven columns, where the first column shows the name of the benchmark for which the test is performed, the columns from the second to the fourth show the average number of the assigned registrations out of the generated ones for each priority (P1, P2 and P3, respectively), while the last two columns show the mean for the OR time and the bed occupancy efficiency.
As we can observe from the results, all considered benchmarks achieved an overall OR time efficiency greater than 80%, but for the case with 15 days, where some degradation is visible, for which the OR time Efficiency is 67.6%.

Scenario B. The results of the scalability analysis for scenario B are reported in Table 4.10 that shows averages of results for all satisfiable instances generated, and is organized as Table 4.9. From Table 4.10 it can be observed that all the considered benchmarks achieve efficiency of bed occupancy greater than 90%, in particular between 92% and 96%, being able, as in the previous scenario, to optimize the constrained resource.

Table 4.9 Averages of the results for 15, 10, 7, 5, 3, 2 and 1 day benchmarks for Scenario A.

Benchmark	Priority 1	Priority 2	Priority 3	Total	OR time Eff.	Bed Occupancy Eff.
15 days	211 / 211	261 / 421	70 / 418	542 / 1050	67.6%	62.5%
10 days	142 / 142	240 / 277	76 / 281	458 / 700	84.4%	61.6%
7 days	105 / 105	166 / 189	68 / 196	339 / 490	89.8%	53.3%
5 days	68 / 68	131 / 139	65 / 143	264 / 350	96.2%	51.6%
3 days	44 / 44	78 / 82	36 / 84	158 / 210	96.5%	36.9%
2 days	28 / 28	53 / 57	22 / 55	108 / 140	95.9%	27.6%
1 day	13 / 13	27 / 29	13 / 28	53 / 70	93.4%	15.3%

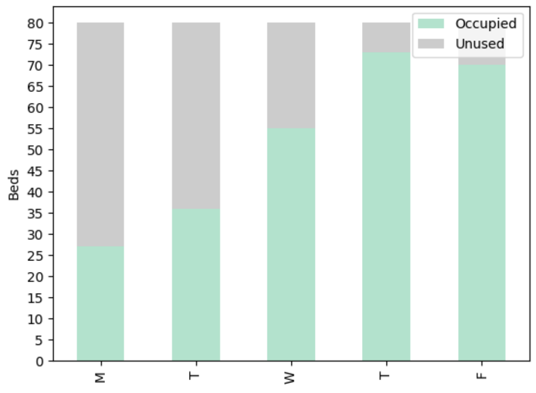
Table 4.10 Averages of the results for 15, 10, 7, 5, 3, 2 and 1 day benchmarks for Scenario B.

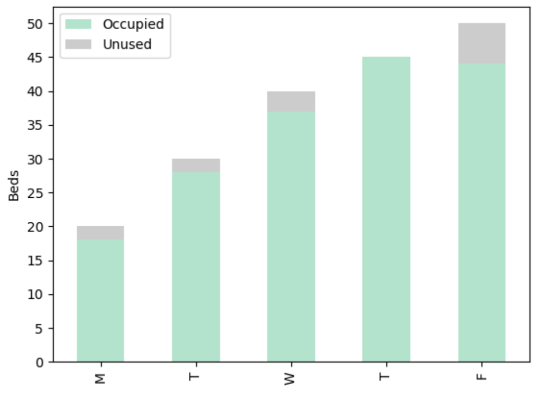
Benchmark	Priority 1	Priority 2	Priority 3	Total	OR time Eff.	Bed Occupancy Eff.
15 days	208 / 208	189 / 423	107 / 419	504 / 1050	62.7%	96.9%
10 days	144 / 144	166 / 281	56 / 275	366 / 700	68.0%	96.5%
7 days	95 / 95	136 / 197	30 / 198	261 / 490	70.6%	94.0%
5 days	68 / 68	88 / 139	31 / 143	187 / 350	68.8%	94.0%
3 days	41 / 41	58 / 85	19 / 84	118 / 210	73.3%	93.3%
2 days	30 / 30	35 / 56	6 / 54	71 / 140	66.7%	92.6%
1 day	16 / 16	25 / 28	6 / 26	47 / 70	70.4%	99.4%

Scenario C. The results for third scenario are reported in Table 4.11, which is organized as Tables 4.9 and 4.10.
From the results, we can note that although the total number of available beds if further reduced, for all benchmarks we achieve efficiency of bed occupancy greater than 85%, even for the extreme case of 15 days planning length, and overall between 86% and 95%.

Table 4.11 Averages of the results for 15, 10, 7, 5, 3, 2 and 1 day benchmarks for Scenario C.

Benchmark	Priority 1	Priority 2	Priority 3	Total	OR time Eff.	Bed Occupancy Eff.
15 days	206 / 206	64 / 407	32 / 437	302 / 1050	37.8%	96.4%
10 days	135 / 135	91 / 289	13 / 276	239 / 700	44.0%	96.3%
7 days	101 / 101	49 / 194	13 / 195	163 / 490	43.2%	91.6%
5 days	68 / 68	41 / 140	10 / 143	119 / 350	44.5%	91.9%
3 days	43 / 43	25 / 83	4 / 84	72 / 210	45.0%	91.4%
2 days	27 / 27	19 / 57	3 / 56	49 / 140	43.9%	85.9%
1 day	13 / 13	22 / 29	3 / 28	38 / 70	70.4%	99.4%





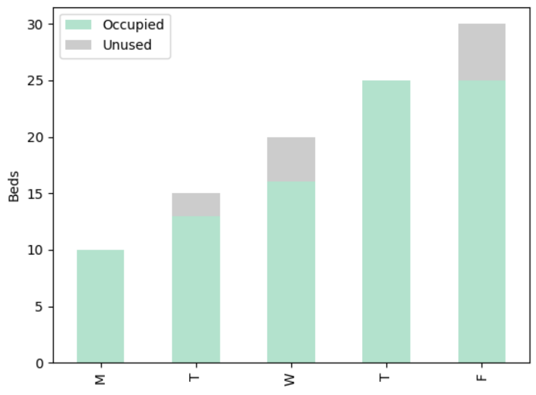


Figure 4.2 Example of bed occupancy of the ward corresponding to specialty 1 for 5-day scheduling. The plot at the top corresponds to the first instance of scenario A, the one in the middle to the first instance of scenario B. Finally, the one at the bottom corresponds to the first instance of scenario C.

Chapter 5

ASP solutions for the ORS problem with surgical teams

In this chapter, we present two ASP solutions for the basic ORS problem enriched with surgical teams, whose specifications are provided in Chapter 3, in two separate sections. The first encoding extends those previously presented, while the second is based on a new modeling idea.

5.1 First ASP solution

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

- Instances of `registration(R,P,SU,_,SP,_,_)` represent the registrations, where we outline only the main variables: with an id (`R`), a priority score (`P`) , a surgery duration (SU) in minutes, and the id of the specialty (SP) it belongs to.

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

- Instances of `mss(O,S,SP,D)` link each operating room (`O`) to a shift (`S`) for each specialty (`SP`) and planning day (`D`), as established by the hospital Master Surgical Schedule (MSS).

- Instances of `surgeon(SR,SP,S)` represent the surgeons with an id (`SR`) for each specialty (`SP`) and shift (`S`).

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

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

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

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

Encoding. The related ASP encoding is shown above in Figure 5.1, and is described in this section. The encoding is based on the Guess&Check programming methodology.
Rule (r_1) guesses an assignment for the registrations, surgeons and anaesthetists to an OR in a given day, shift and with a time slot among the ones permitted by the MSS for the particular specialty the registrations, surgeons and anaesthetists belongs to, such that the registrations assigned with a slot time and surgery duration should be less than `shift_duration` of OR. We recall that `shift_duration` is a constant that depends on the different scenario considered (see Section 5.1).

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

Figure 5.1 ASP encoding of the ORS problem with surgical teams.

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

General requirements. Rule (r_2) checks that same registration with same surgeon and anaesthetist should not be assigned more than once in different OR or shifts at the same time. Rule (r_3) ensures that different registrations cannot be assigned in the same OR and shift at the same time slot. Rule (r_4) checks that different registrations cannot be assigned at different times ,in the same OR and shift until the end time of the previous surgery to avoid time overlapping. Rule (r_5) shows that the same surgeon cannot be assigned at the same time slot in different ORs in the same shift. Rule (r_6) checks that the same surgeon cannot be assigned at different times in different OR in the same shift until the end of previous surgery. Rule (r_7) ensures that the same anaesthetist cannot be assigned at the same time slot in different ORs in the same shift. Rule (r_8) checks that the same anaesthetist cannot be assigned to different times to different ORs in the same shift until the previous surgery is finished.

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

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

5.2 Alternative ASP solution

In this section, for the same problem solved in the previous section about the basic ORS problem with surgical teams, we present an alternative solution still based on ASP but whose modeling principles depart from those used in the previous encoding. This solution has been designed during the (remote) PhD collaboration with Prof. Martin Gebser of Klagenfurt University in Austria.

Data Model. The input data is the same specified in the previous chapter. The output is an assignment represented by atoms of the form *schedule(R,L,O,P,D,S,r(oproom,I))*, where the intuitive meaning is that the registration *R* with priority *L* and operation duration equal to *O* for the specialty *P* during the session *S* and the day *D* in the operating room *I*. It is important to emphasize here that the priority *L* is not actually needed in the output, however it is included because it improves the readability of the encoding presented in the subsequent paragraph.

Encoding. The related encoding is shown in Figure 5.2, and is described in the following. Rules r_4, r_9, and r_{21} are defined to reduce the number of fields of the atoms involved in the rules. Rules from r_1 to r_3 define the atom resource for the ORs, surgeons and anaesthetists which, starting from the mss atoms, get all the sessions, days, and specialty in which these resources exist.

```
(r1)  resource(r(oproom,I),P,D,S) :- mss(I,S,P,D).
(r2)  resource(r(surgeon,I),P,D,S) :- resource(r(oproom),P,D,S),
    surgeon(I,P,S).
(r3)  resource(r(anaesthetist,I),P,D,S) :- resource(r(surgeon),P,D,S),
    an(I,P,S).
(r4)  resource(r(K),P,D,S) :- resource(r(K,I),P,D,S).
(r5)  capacity(r(surgeon,I),D,C) :- surgeryTime(C,I,D),
    #sum+{shift_duration,S : resource(r(anaesthetist),P,D,S),
    resource(r(surgeon,I),P,D,S)} > C.
(r6)  capacity(r(anaesthetist,I),D,C) :- anaesthetistWT(C,I,D),
    #sum+{shift_duration,S : resource(r(anaesthetist,I),P,D,S)} > C.
(r7)  feasible(R,L,O,P,D,S) :- resource(r(anaesthetist),P,D,S),
    registration(R,L,O,_,P,_,_), O <= shift_duration.
(r8)  feasible(R,L,O,P,D,S, r(K,I)) :- resource(r(K,I),P,D,S),
    feasible(R,L,O,P,D,S).
(r9)  feasible(R,L,O,P) :- feasible(R,L,O,P,D,S).
(r10) {occupation(R,P,D-2..D-1; schedule(R,L,O,P,D,S) :
    feasible(R,L,O,P,D,S)} 1 :- feasible(R,L,O,P).
(r11)  {schedule(R,L,O,P,D,S,r(K,I)) : feasible(R,L,O,P,D,S, r(K,I))} = 1
    :- resource(r(K,I),P,D,S), schedule(R,L,O,P,D,S).
(r12)  :- capacity(X,D,C), #sum+{O,R: schedule(R,L,O,P,D,S,X)} > C.
(r13)  beds(200,P,D) :- mss(_,_,P,D).
(r14)  :- beds(N,P,D) :- #count{R: occupation(R,P,D)} > N.
(r15)  overlap(R1,O1,R2,O2) :- schedule(R1,L1,O1,P1,D,S,X),
    schedule(R2,L2,O2,P2,D,S,X), R1 < R2.
(r16)  {ordering(R1,R2,O2)} :- overlap(R1,O1,R2,O2).
(r17)  ordering(R2,R1,O1) :- overlap(R1,O1,R2,O2), not ordering(R1,R2,O2.
(r18)  end(R,0..O-1) :- feasible(R,L,O,P).
(r19)  end(R,Q) :- ordering(R1,R,O),end(R1,Q1), Q = Q1+O, Q <=
    shift_duration.
(r20)  :- end(R,shift_duration).
(r21)  schedule(R,L,O,P) :- schedule(R,L,O,P,D,S).
(r22) :- registration(R,1,O,_,P,_,_), not schedule(R,1,O,P).
(r23) :~ registration(R,L,O,_,P,_,_), not schedule(R,L,O,P), 1 < L.
    [1@5-L,R]
```

Figure 5.2 ASP encoding optimized of the ORS problem.

Rule r_7 get all the possible days and sessions in which each registration can be assigned. Then r_8 use the atom defined in rule r_7 to get all the operating rooms in which the registration can be assigned and all the possible surgeon and anaesthetists that can be assigned to the surgery. Rules r_{10} guess a day and a session for each possible registration from the possible days and sessions assignable to the registration. Rule r_{11} guess the resources for each assigned registration. Rule r_{12} imposes that the total length of surgery durations assigned to each resource is less than the maximum capacity of the considered resource. Rules r_{13} and r_{14} define 200 beds for each specialty and day and checks that the number of assigned registration to the considered days and specialties are lower than 200. Rule r_{15} get all the registrations assigned to the same day, session and resource. Then, with rules r_{16} and r_{17} is assigned an order between the two registrations. Rules r_{18}, r_{19} and, r_{20} define the ending time of each assigned surgery. In particular, r_{19} ensure that each surgery is started and completed after the surgery before, while r_{20} checks that the ending time of each surgery is before the ending time of the considered session. Rule r_{22} impose that every registration with priority level equal to 1 is assigned. With rule r_{23} we minimize the number of registration with priority L with weight equal to 5-L.

Chapter 6

Analysis of the two ASP solutions for the ORS problem with surgical teams

This chapter first presents the dimensions on which the analysis has been performed, and then a (comparative) analysis of the two ASP encoding for the ORS problem with surgical teams presented in the previous chapter.

6.1 Benchmarks

We have performed different slot interval analysis on the performance of our encoding and employed ASP solver. As described in Section 5.1, the dimension of the slot interval determines the time sensitivity of our encoding, and four different scenarios have been considered: A, B, C, and D for slot interval of 10, 20, 30, and 60 minutes, respectively.
For each scenario, the characteristics of the tests are as follows:

- 4 different benchmarks, with a planning period of 1, 2, 3 and 5 working days;
- For each benchmark the total number of randomly generated registrations were 350 for 5 days, 210 for 3 days, 140 for 2 days and 70 for 1 days;
- 5 specialties;
- 20 surgeons assigned to the 5 specialties;
- 20 anaesthetists assigned to the 5 specialties;
- 4 hours of work time in a day for each surgeon;

Table 6.1 Total number of randomly generated registrations for each benchmark.

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

- 6 hours of work time in a day for each anaesthetist;
- 10 ORs distributed among the specialties;
- 5 hours morning and afternoon shifts for each OR summing up to 500, 300, 200 and 100 hours of OR available time for the four benchmarks;

Table 6.1 shows the distribution of the total number of randomly generated registrations for each benchmark of 5, 3, 2 and 1 day, for each specialty, together with the distribution of ORs for each specialty.

6.2 Experimental Results

For simplicity we refer to the ASP encoding in Chapter 5.1 as Encoding 1, and to the second ASP encoding in Chapter 5.2 as Encoding 2.
The averages of results with the experiments for encoding 1 and 2 are reported for scenario A in Table 6.2 for scenario B in Table 6.3, for scenario C in Table 6.4 and for scenario D in Table 6.5, respectively. Each benchmark was tested 10 times with different randomly generated inputs (see part 3 for details). A time limit of 300 seconds was set for each experiment in both encodings. In each table averages for 10 instances for each benchmark are reported, and assignments to P1, P2, P3 priorities, total assignments, OR, Surgeons and Anesthetists efficiency are reported.
As we can see in scenario A Table 6.2 with slot interval of 10 minutes. For encoding 1 we obtain results only for schedules up to 3 days, while in the case of the 5-days benchmark the computation time exceeds our time limit of 300 seconds on all instances. However, with encoding-2 other than the 1, 2 and 3-days benchmark we also obtained schedules for 5 days benchmark. It can also be seen that for encoding 2 the OR efficiency is 74% while Surgeons and Anesthetists WT efficiency remains greater than 91% and 60% for all benchmarks.

For Scenario B Table 6.3 details the scheduling results for encoding 1 and 2 with slot intervals of 20 minutes. It can be seen that OR efficiency for both the encodings is 75% while the Surgeons and Anesthetists WT efficiency remain greater than 90% and 60%, respectively, for all benchmarks in this scenario.
In scenario C Table 6.4 with a slot interval of 30 minutes, the OR efficiency for encoding 1 and 2 is around 76% while the Surgeons and Anesthetists WT efficiency for both the encodings are up to 94% and 63% for all benchmarks respectively.
In scenario D Table 6.5 with a slot interval of 60 minutes, OR efficiency for both encodings is almost 79% while the Surgeons WT efficiency is more than 95%, and Anesthetists WT efficiency is up to 65%.

Table 6.2 Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario A with encoding 1 and 2

Encoding	Bench.	P1	P2	P3	Total	OR Eff.	Surgeons WT Efficiency	Anesthetists WT Efficiency
1	5 days	-	-	-	-	-	-	-
	3 days	43.1 / 43.1	53 / 81.6	26.2 / 85.3	122.3 / 210.0	73.5%	91.8%	61.2%
	2 days	29.9 / 29.9	37.0 / 54.7	17.7 / 55.4	84.6 / 140.0	74.7%	93.4%	62.3%
	1 day	13.4 / 13.4	22.8 / 28.0	8.9 / 28.6	46.1 / 70.0	75.6%	94.5%	62.9%
2	5 days	70.5 / 70.5	91.7 / 144.2	48.3 / 135.3	210.5 / 350.0	74.3%	92.8%	61.9%
	3 days	43.1 / 43.1	60.5 / 81.6	22.1 / 85.3	125.7 / 210.0	74.7%	93.4%	62.5%
	2 days	29.9 / 29.9	43.6 / 54.7	10.9 / 55.4	84.4 / 140.0	75.9%	94.9%	63.2%
	1 day	13.4 / 13.4	22.9 / 28.0	9.8 / 28.6	44.1 / 70.0	75.5%	94.3%	62.9%

Table 6.3 Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario B with encoding 1 and 2

Encoding	Bench.	P1	P2	P3	Total	OR Eff.	Surgeons WT Efficiency	Anesthetists WT Efficiency
1	5 days	71.2 / 71.2	99.4 / 140.1	38.3 / 138.7	208.9 / 350.0	75.1%	93.8%	62.5%
	3 days	40.9 / 40.9	61.6 / 85.3	25.2 / 83.8	127.7 / 210.0	75.3%	94.1%	62.8%
	2 days	28.2 / 28.2	41.1 / 56.1	14.9 / 55.7	84.2 / 140.0	75.0%	93.7%	62.5%
	1 day	12.5 / 12.5	23.1 / 29.7	8.9 / 27.8	43.5 / 70.0	75.5%	94.4%	62.9%
2	5 days	71.2 / 71.2	95.5 / 140.1	46.8 / 138.7	213.5 / 350.0	75.3%	94.2%	62.8%
	3 days	40.9 / 40.9	67.0 / 85.3	21.1 / 83.8	129 / 210.0	76.3%	95.3%	63.6%
	2 days	28.2 / 28.2	44.8 / 56.1	12.1 / 55.7	85.1 / 140.0	75.9%	94.8%	63.2%
	1 day	12.5 / 12.5	23.1 / 29.7	8.9 / 27.8	43.5 / 70.0	75.3%	94.1%	62.7%

Table 6.4 Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario C with encoding 1 and 2

Encoding	Bench.	P1	P2	P3	Total	OR Eff.	Surgeons WT Efficiency	Anesthetists WT Efficiency
1	5 days	71.9 / 71.9	99.0 / 139.8	44.1 / 138.3	215.0 / 350.0	76.0%	95.2%	63.3%
	3 days	41.7 / 41.7	66.9 / 84.8	21.6 / 83.5	130.2 / 210.0	76.1%	95.1%	63.5%
	2 days	27.9 / 27.9	42.7 / 53.8	16.9 / 58.3	87.5 / 140.0	76.2%	95.2%	63.5%
	1 day	14.2 / 14.2	23.0 / 29.4	6.7 / 26.4	43.9 / 70.0	76.2%	95.1%	63.5%
2	5 days	71.9 / 71.9	96.2 / 139.8	46.3 / 138.3	214.4 / 350.0	75.8%	94.7%	63.1%
	3 days	41.7 / 41.7	67.4 / 84.8	21.4 / 83.5	130.5 / 210.0	76.5%	95.6%	63.7%
	2 days	27.9 / 27.9	42.9 / 53.8	16.6 / 58.3	87.3 / 140.0	76.3%	95.4%	63.8%
	1 day	14.2 / 14.2	23.0 / 29.4	6.7 / 26.4	43.9 / 70.0	76.5%	95.6%	63.8%

Table 6.5 Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario D with encoding 1 and 2

Encoding	Bench.	P1	P2	P3	Total	OR Eff.	Surgeons WT Efficiency.	Anesthetists WT Efficiency
1	5 days	68.7 / 68.7	109.6 / 143.8	46.9 / 137.5	224.8 / 350.0	79.0%	98.8%	65.8%
	3 days	41.8 / 41.8	65.1 / 81.9	25.5 / 86.3	132.4 / 210.0	78.7%	98.4%	65.6%
	2 days	27.5 / 27.5	46.5 / 54.5	14.6 / 58.0	87.7 / 140.0	79.2%	98.9%	65.9%
	1 day	13.3 / 13.3	23.1 / 27.5	8.3 / 29.2	44.7 / 70.0	78.3%	97.8%	65.2%
2	5 days	68.7 / 68.7	100.2 / 143.8	53.6 / 137.5	222.5 / 350.0	79.0%	98.7%	65.8%
	3 days	41.8 / 41.8	66.9 / 81.9	23.2 / 86.3	131.9 / 210.0	78.9%	98.6%	65.7%
	2 days	27.5 / 27.5	47.1 / 54.5	13.5 / 58.0	88.1 / 140.0	79.2%	99.1%	66.0%
	1 day	13.3 / 13.3	23.3 / 27.5	8.1 / 29.2	44.7 / 70.0	78.8%	98.5%	65.7%

Appendix A further reports very detailed results about metrics like CPU time, answer sets, optimization and assigned registrations for the experiments performed on both Encoding 1 and 2 just showed in Tables 6.2-6.5.

Analysis with different optimization strategies. We have performed different slot interval analysis for scenario A, B, C, D on the performance of Encoding 1 and 2 by setting the solver options to –restart-on-model (denoted as rom), –opt-strategy=bb,1 (denoted as bb1) and Multithreading (denoted as -t4) in tables below.
The results of the experiments for encoding 1 and 2 with solver options (rom, bb1, t4) are reported for scenario A in Table 6.6, for scenario B in Table 6.7, for scenario C in Table 6.8and for scenario D in Table 6.9, respectively. Each benchmark was tested 10 times with different randomly generated inputs. A time limit of 300 seconds was set for each experiment.

In each table averages for 10 instances for each benchmark are reported. Each table consists of six columns, with the first column show the name of the benchmark, the second column reports the option used, then further for each encoding we have reported as an average a column for Assigned (P1, P2, P3 and total assigned registrations) while the second column reports (OR, surgeon work time and anesthetist work time) efficiencies.
As we can see in scenario A (Table 6.6) with slot interval of 10 minutes. For encoding 1 with all options we obtain results only for schedules up to 3 days, while in the case of the 5-days benchmark the computation time exceeds our time limit of 300 seconds on all instances. However, with encoding-2 other than the 1, 2 and 3-days benchmark we also obtained schedules for 5 days benchmark. For 1 day benchmark both the encodings for all options performs almost the same. It can also be seen that in case of 2, 3 and 5 day benchmark for both encoding 1 and 2 with option (bb,1 and t4) the number of assigned P2 registrations are increased (shown in bold). It can also be seen that for option (bb,1) and (t4) with encoding 2 the OR efficiency is greater than 70% while Surgeons and Anesthetists WT efficiency remains greater than 91% and 60% for all benchmarks.

Table 6.6 Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario A with encoding 1 and 2

		Encoding 1		**Encoding 2**	
Bench	Option	Assigned [P1, P2, P3,Total]	Efficiency (%) [OR, SWT, AWT]	Assigned [P1, P2, P3, Total]	Efficiency (%) [OR, SWT, AWT]
1	rom	[13.4 , 22.8, 8.9, 46.1]	[75.6, 94.5, 62.9]	[13.4, 22.9, 9.8, 44.1]	[75.5, 94.3, 62.9]
	bb1	[13.4, 22.9, 7.3, 43.6]	[74.8, 93.4, 62.3]	[13.4, 22.9, 7.8, 44.1]	[75.4, 94.2, 62.8]
	t4	[13.4, 22.9, 7.3, 43.6]	[74.5, 93.4, 62.2]	[13.4, 22.9, 7.8, 44.1]	[75.1, 90.9, 57.1]
2	rom	[29.9, 37.0, 17.7, 84.6]	[74.7, 93.4, 62.3]	[29.9, 43.6, 10.9, 84.4]	[75.9, 94.9, 63.2]
	bb1	[29.9, 43.3, 0, 73.2]	[68.0, 85.1, 57.0]	[29.9, 43.5, 8.3, 81.7]	[73.3, 91.7, 61.1]
	t4	[29.9, 43.5, 2.6, 76.0]	[69.8, 87.3, 58.2]	[29.9, 43.5, 8.4, 81.8]	[73.6, 91.9, 61.3]
3	rom	[43.1, 53.0, 26.2, 122.3]	[73.5, 91.8, 61.2]	[43.1, 60.5, 22.1, 125.7]	[74.7, 93.4, 62.5]
	bb1	[43.1, 65.0, 0, 108]	[67.7, 84.7, 56.4]	[43.1, 65.7, 5.3, 114.1]	[70.0, 87.5, 58.3]
	t4	[43.1, 64.6, 0, 107.7]	[67.4, 84.3, 56.2]	[43.1, 65.6, 7.8, 116.5]	[71.0, 88.8, 59.2]
5	rom	-	-	[70.5, 91.7, 48.3, 210.5]	[74.3, 92.8, 61.9]
	bb1	-	-	[70.5, 118, 4.8, 193.7]	[70.3, 87.8, 58.5]
	t4	-	-	[70.5, 117.9, 5.1, 193.5]	[70.4, 88.0, 58.7]

In scenario B (Table 6.7) with slot interval of 20 minutes, for encoding 1 we obtain results for a complete 5 day schedule for all options (rom, bb1 and t4). It can also be seen that for both encoding 1 and 2 with option (bb1 and t4) the number of assigned P2 registrations are increased (shown in bold). Further, for option (bb,1) and (t4) with encoding 1 the OR efficiency is greater than 66% while Surgeons and Anesthetists WT efficiency remains greater than 82% and 55% for all benchmarks. For encoding 2 with option (bb,1) and (t4) the OR efficiency is greater than 69% while the Surgeons and Anesthetists WT efficiency remains greater than 86% and 57%. In scenario C (Table 6.8) and scenario D (Table 6.9) with slot interval of 30 and 60 minutes. It can be seen that for both encoding 1 and 2 with option (bb,1 and t4) the number of assigned P2 registrations are increased (shown in bold). It can also be seen that for option (bb,1) and (t4) for both encoding 1 and 2 the OR efficiency remains same for all benchmarks and is greater than 65% while Surgeons and Anesthetists WT efficiency remains greater than 82% and 55% for all benchmarks.

Table 6.7 Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario B with encoding 1 and 2

		Encoding 1		**Encoding 2**	
Bench	Option	Assigned [P1, P2, P3,Total]	Efficiency (%) [OR, SWT, AWT]	Assigned [P1, P2, P3, Total]	Efficiency (%) [OR, SWT, AWT]
1	rom	[12.5, 23.1, 8.9, 43.5]	[75.5, 94.4, 62.9]	[12.5, 23.1, 8.9, 43.5]	[75.3, 94.1, 62.7]
	bb1	[12.5, 23.1, 7.9, 43.5]	[74.5, 94.2, 62.8]	[12.5, 23.1, 7.9, 43.5]	[75.3, 94.2, 62.8]
	t4	[12.5, 23.1, 7.0, 42.6]]	[73.8, 92.2, 61.5]	[12.5, 23.1, 7.9, 43.5]]	[75.0, 93.7, 62.5]
2	rom	[28.2, 41.4, 14.9, 84.2]	[75.0, 93.7, 62.5]	[28.2, 44.8, 12.1, 85.1]	[75.9, 94.8, 63.2]
	bb1	[28.2, 44.8, 0, 73.0]	[67.7, 84.6, 56.4]	[28.2, 44.9, 1.5, 74.6]	[69.5, 86.8, 57.9]
	t4	[28.2, 44.7, 1.2, 74.1]]	[68.3, 85.3, 56.9]	[28.2, 45.0, 6.4, 79.5]	[73.3, 91.6, 61.0]]
3	rom	[40.9, 61.6, 25.2, 127.7]	[75.3, 94.1, 62.8]	[40.9, 67.0, 21.1, 129.0]	[76.3, 95.3, 63.6]
	bb1	[40.9, 70.4, 0, 111.3]	[69.0, 86.2, 57.5]	[40.9, 70.8, 0.4, 112.1]	[69.3, 86.6, 57.7]
	t4	[40.9, 70.4, 0, 111.3]]	[68.8, 86.0, 57.3]	[40.9, 70.6, 3.9, 115.4]	[71.2, 89.0, 59.4]
5	rom	[71.2, 99.4, 38.3, 208.9]	[75.1, 93.8, 62.5]	[71.2, 95.5, 46.8, 213.5]	[75.3, 94.2, 62.8]
	bb1	[71.2, 117.6, 0, 188.8]	[69.9, 87.3, 58.2]	[71.2, 118.5, 2.2, 191.5]	[70.6, 88.2, 58.8]
	t4	[71.2, 93.6, 15.9, 180.7]	[66.0, 82.6, 55.0]	[71.2, 118.0, 2.6, 191.7]	[70.6, 88.2, 59.0]]

Table 6.8 Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario C with encoding 1 and 2

		Encoding 1		**Encoding 2**	
Bench	Option	Assigned [P1, P2, P3,Total]	Efficiency (%) [OR, SWT, AWT]	Assigned [P1, P2, P3, Total]	Efficiency (%) [OR, SWT, AWT]
1	rom	[14.2, 23.0, 6.7, 43.9]	[76.2, 95.1, 63.5]	[14.2, 23.0, 6.7, 43.9]	[76.5, 95.6, 63.8]
	bb1	[14.2, 23.0, 6.5, 43.7]	[75.9, 94.8, 63.2]	[14.2, 23.0, 6.7, 43.9]	[75.9, 94.8, 63.2]
	t4	[14.2, 23.0, 6.4, 43.6]	[75.6, 94.5, 63.0]	[14.2, 23.0, 6.7, 43.9]	[76.2, 95.2, 63.5]
2	rom	[27.9, 42.7, 16.9, 87.5]	[76.2, 95.2, 63.5]	[27.9, 42.9, 16.6, 87.3]	[76.3, 95.4, 63.8]
	bb1	[27.9, 44.1, 0, 72.0]	[65.5, 82.0, 54.6]	[27.9, 44.1, 1.7, 73.7]	[67.0, 83.7, 55.8]
	t4	[27.9, 44.1, 6.1, 78.1]	[70.1, 87.6, 58.4]	[27.9, 44.1, 10.6, 82.6]	[74.0, 92.5, 61.6]
3	rom	[41.7, 66.9, 21.6, 130.2]	[76.1, 95.1, 63.5]	[41.7, 67.4, 21.4, 130.5]	[76.5, 95.6, 63.7]
	bb1	[41.7, 71.1, 0, 112.8]	[68.6, 85.7, 57.1]	[41.7, 71.3, 0.6, 113.6]	[69.0, 86.2, 57.5]
	t4	[41.7, 71.1, 0, 112.8]	[68.7, 85.9, 57.2]	[41.7, 71.2, 2.6, 115.2]	[69.9, 87.4, 58.3]
5	rom	[71.9, 99.0, 44.1, 215.0]	[76.0, 95.2, 63.3]	[71.9, 96.2, 46.3, 214.4]	[75.8, 94.7, 63.1
	bb1	[71.9, 118.7, 0, 190.6]	[70.0, 87.4, 58.2]	[71.9, 119.0, 1.8, 192.5]	[70.4, 88.0, 58.7
	t4	[71.9, 118.5, 0, 190.4]	[69.7, 87.1, 58.1]	[71.9, 118.7, 1.7, 192.3]	[70.6, 88.2, 58.8]

Table 6.9 Averages of the results for 5, 3, 2 and 1 day benchmarks for Scenario D with encoding 1 and 2

		Encoding 1		**Encoding 2**	
Bench	Option	Assigned [P1, P2, P3,Total]	Efficiency (%) [OR, SWT, AWT]	Assigned [P1, P2, P3, Total]	Efficiency (%) [OR, SWT, AWT]
1	rom	[13.3, 23.1, 8.3, 44.7]	[78.3, 97.8, 65.2]	[13.3, 23.3, 8.1, 44.7]	[78.8, 98.5, 65.7]
	bb1	[13.3, 23.3, 8.0, 44.6]	[78.5, 98.1, 65.4]	[13.3, 23.3, 8.1, 44.7]	[78.7, 98.3, 65.5]
	t4	[13.3, 23.3, 8.1, 44.7]	[78.5, 98.1, 65.4]	[13.3, 23.3, 8.1, 44.7]	[78.5, 98.1, 65.4]
2	rom	[27.5, 46.5, 14.6, 87.7]	[79.2, 98.9, 65.9]	[27.5, 47.1, 13.5, 88.1]	[79.2, 99.1, 66.0]
	bb1	[27.5, 47.1, 0, 74.6]	[70.1, 87.7, 58.4]	[27.5, 47.1, 0.6, 75.2]	[70.4, 88.0, 58.7]
	t4	[27.5, 47.1, 0, 74.6]	[70.0, 87.5, 57.4]	[27.5, 47.1, 10.0, 84.6]	[78.0, 97.5, 65.0]
3	rom	[41.8, 65.1, 25.5, 132.4]	[78.7, 98.4, 65.6]	[41.8, 66.9, 23.2, 131.9]	[78.9, 98.6, 65.7]
	bb1	[41.8, 71.2, 0, 113.0]	[71.4, 89.2, 59.5]	[41.8, 71.2, 0.2, 113.2]	[71.5, 89.4, 59.6]
	t4	[41.8, 71.2, 1.6, 114.6]	[72.7, 90.8, 60.6]	[41.8, 71.2, 4.3, 117.3]	[74.0, 92.5, 61.6]
5	rom	[68.7, 109.6, 46.9, 224.8]	[79.0, 98.8, 65.8]	[68.7, 100.2, 53.6, 222.5]	[79.0, 98.7, 65.8]
	bb1	[68.7, 125.9, 0, 194.6]	[72.5, 90.6, 60.4]	[68.7, 125.7, 1.1, 195.5]	[72.8, 91.0, 60.7]
	t4	[68.7, 125.9, 0, 194.6	[72.7, 90.7, 60.5]	[68.7, 125.7, 0.9, 195.3]	[72.7, 91.0, 60.6]

Chapter 7

Rescheduling for the ORS problem with bed management

The rescheduling procedure is applied to a previously planned schedule, i.e. we start from an already created schedule (old schedule or x-schedule) that could not be executed fully till the end due to some reasons, e.g. some patients could not be operated in their assigned slots or the patients may delete their registration. In such a situation all those postponed registrations (or surgeries) must be reallocated to one of the next slots in the remaining part of the original planning period (new schedule or y-schedule). The planning period we consider is 5 days, and we consider the basic ORS setting with bed management (Chapter 4).
Once planned, a speciality schedule does not influence other specialties so it makes sense to reschedule one specialty at a time. Since we already have the initial schedule for the planning period of 5 days, we assumed that in day 2, a number of registrations from specialty 1 had to be postponed to the next day. So we have to reschedule these registrations in the remaining available three days, i.e. day 3, 4 and 5.
In order to insert the postponed registrations in the new schedule (y-schedule) we have to make sure that the start of the schedule leaves enough available OR time by automatically dropping the necessary registrations from the old schedule; the choice of the registrations to be removed will begin from the last day, i.e. day 5 of the planning period and from registrations in the priority 3 category.
The next two sub-sections will show ASP encoding for the rescheduling problem, together with the needed changes in the data model, and the results of the experimental analysis we performed.

```
(rr1)  {y(R,P,O,S,D)} :- registration(R,P,_,_,SP,_,_), mss(O,S,SP,D).
(rr2)  :- y(R,P,O,S1,_), y(R,P,O,S2,_), S1 != S2.
(rr3)  :- y(R,P,O1,S,_), y(R,P,O2,S,_), O1 != O2.
(rr4)  surgery(R,SU,O,S) :- y(R,_,O,S,_), registration(R,_,SU,_,_,_,_).
(rr5)  :- y(_,_,O,S,_), #sum{SU,R: surgery(R,SU,O,S)}>N, duration(N,O,S).
(rr6)  stay(R,(D-A)..(D-1),SP) :- registration(R,_,_,LOS,SP,_,A),
         y(R,_,_,_,D), A>0.
(rr7)  stay(R,(D+ICU)..(D+LOS-1),SP) :- registration(R,_,_,LOS,SP,ICU,_),
         y(R,_,_,_,D), LOS>ICU.
(rr8)  stayICU(R,D..(D+ICU-1)) :- registration(R,_,_,_,_,ICU,_),
         y(R,_,_,_,D), ICU>0.
(rr9)  :- #count{R: stay(R,D,SP)} > AV, SP>0, beds(SP,AV,D).
(rr10) :- #count{R: stayICU(R,D)} > AV, beds(0,AV,D).
(rr11) :- not y(R,P,_,_,_),x(R,P,_,_,2).
(rr12) totReg(L,4) :- M = #count{R:y(R,1..2,_,_,_),x(R,1..2,_,_,_)},
         P = #count{R : x(R,1..2,_,_,_)}, L=P-M.
(rr13) totReg(L,3) :- M = #count{R:y(R,3,_,_,_),x(R,3,_,_,3..4)},
         P = #count{R : x(R,3,_,_,3..4)}, L=P-M.
(rr14) totReg(L,2) :- M = #count{R:y(R,3,_,_,5),x(R,3,_,_,5)},
         P = #count{R : x(R,3,_,_,5)}, L=P-M.
(rr15) :~ totReg(L,V). [L@V]
(rr16) :~ y(R,_,_,_,D),x(R,_,_,_,OldD),DF = |D - OldD|. [DF@1, R]
```

Figure 7.1 ASP encoding of the ORS problem without beds-management *(Rescheduling)*

7.1 ASP Encoding

Input. The input data is specified by means all of the atoms described in Section 4.1 and by atoms of the form *x(R,P,O,S,D)*. The latter represent a solution to the ORS problem as computed by the encoding described in Section 5.2. Moreover, in the following we assume that atoms of the form *mss(O,S,SP,D)* include only elements from day 3 to day 5, i.e. $3 \leq D \leq 5$.

Output. The output of the new schedule is represented by atoms of the form *y(R,P,O,S,D)*, representing that the registration R of the patient with priority P is assigned to an operating room O in a shift S of day D.

Encoding. The ASP rescheduling encoding for ORS problem is shown in Figure 7.1. In particular, rules (rr_1)-(rr_{10}) correspond to rules (r_1)-(r_{10}) from the encoding reported in Figure 4.1, where atoms over the predicate x are replaced with the ones over the predicate y. Rule (rr_{11}) states that all registration scheduled for the day 2 (i.e. the ones postponed according to our scenarios) should be rescheduled. Rules (rr_{12})-(rr_{15}) ensure that the

Table 7.1 Results for the four rescheduling scenarios

Scenario	Postponed Registr.	Total Old Registr.	Dropped Registr.	Total New Registr.
I	1	45	0	46
II	2	44	1	46
III	4	43	2	47
IV	6	41	4	47

maximum number of registrations from the old schedule should be included also in the new one. In particular, rule (rr_{12}) computes the difference between the total number of registrations with priority 1 and 2 assigned in the previous schedule and the number of registrations assigned in the current schedule, whereas rule (rr_{13}) (resp. (rr_{14})) computes the difference between the total number of registrations with priority 3 for day 3 and 4 (resp. day 5), and the number of registrations assigned in the current schedule. Such differences are then minimized by means of the weak constraint (rr_{15}). Finally, rule (rr_{16}) minimizes the total sum of the difference (in terms of number of days) between the new schedule and the old one for each registration.

7.2 Experimental Results

The results using our ASP rescheduling encoding on four scenarios (I, II, III, and IV) are summarized in Table 7.1. An execution time of 60 seconds was given for each scenario.
For our tests we started from an old schedule (x-schedule) calculated under the conditions delineated in Scenario A of the scheduling problem, in particular we took into account the results for specialty 1. Since we consider postponed registrations from a single speciality for our analysis, it should be noted that the total number of old registrations of specialty 1 with priority 1, 2 and 3 to be rescheduled in the next 3 days were 45. In the table, the first column mentions the scenario, the second shows the number of registrations that were inserted in each scenario (Postponed Registrations), the third column reports the total number of registrations from the old schedule (Total Old Registrations), while the fourth column shows the necessary number of dropped registrations from the old schedule. Finally, the last column shows the total number of registrations in the new schedule by also reassigning the postponed registrations (Total New Registrations) in the next days. Columns from the second to the fifth report the (rounded) mean of the 10 instances. Overall, our encoding dropped 0, 1, 2 and 4 priority 3 old schedule registrations for scenarios I, II, III, and IV, respectively.

These results confirm that we managed to produce a new schedule in case of disruption of the previous one even when already in its execution phase. This was accomplished by allowing the rescheduled registrations to change time but minimizing the number of changes of surgery date, which would imply a major disruption in the procedure of the hospital, in particular regarding the ICU and ward bed management. On the one hand, in a situation of minor disruption like that of Scenario I we managed to produce a new schedule without having to drop any of the previously scheduled surgeries; on the other hand, even in case of greater disruption, like in Scenario II, III, and IV, the number of scheduled surgeries in the new schedule were at least as many as in the previous schedule.

Chapter 8

GUI for ASP solution

Our scheduling solution has been planned and developed also in view of its practical utilization by medical operators e.g., OR managers and OR staff in hospitals. To this end, we are developing a web application that wraps the ASP encoding for scheduling the basic ORS setting with bed management (Chapter 4), and the CLINGO solver (see Figure 8.1). The software is a full-stack JavaScript application with a Graphical User Interface (GUI) and a Node.js back-end. The ASP facts and encoding are dynamically composed at run-time, reflecting the user choices, and are then relayed to the ASP solver through a wrapper package. This solution allows the solver to be embedded inside an easily reachable and usable web application, removing the hurdle that installing and managing the solver may represent for a non-expert user. The application currently includes:

- a registration and authentication process,
- a database for storing and retrieving previous test data, and
- a GUI to easily create and customize new test scenarios or load pre-made ones.

The GUI can be divided into the following sections: an input screen, an overall results screen, and their graphical representation for the ORs and for the bed occupancy. The input screen (see Figure 8.2) currently hosts two tables: on the left the parameters for the random generation of the registrations and on the right the bed availability. It is important to note that in a more operative stage the generation parameter will be obviously replaced by the actual registration data. In the left table the user can set the parameters for the generator. From left to right these are:

- the specialty names;

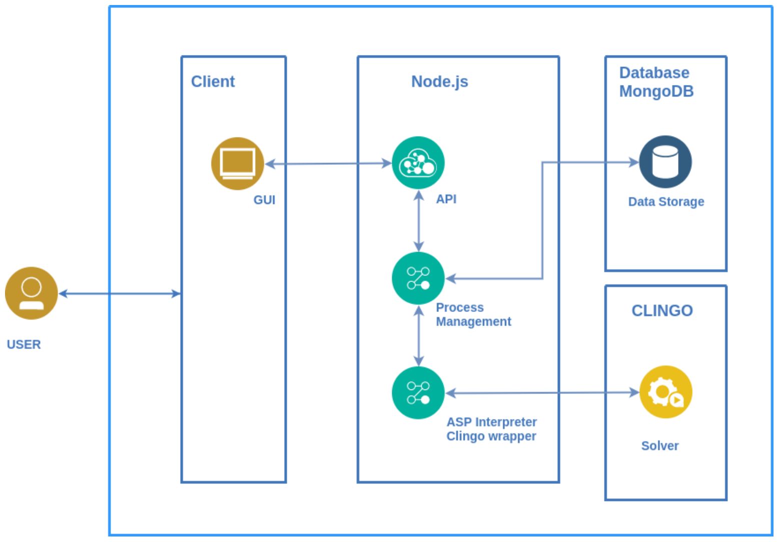
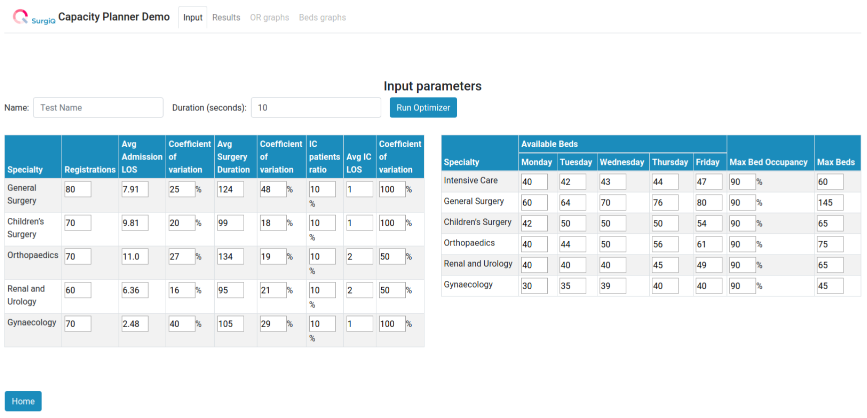


Figure 8.1 Web application architecture schema.



Input parameters

Name: Test Name Duration (seconds): 10

Specialty	Registrations	Avg Admission LOS	Coefficient of variation	Avg Surgery Duration	Coefficient of variation	IC patients ratio	Avg IC LOS	Coefficient of variation
General Surgery	80	7.91	25 %	124	48 %	10 %	1	100 %
Children's Surgery	70	9.81	20 %	99	18 %	10 %	1	100 %
Orthopaedics	70	11.0	27 %	134	19 %	10 %	2	50 %
Renal and Urology	60	6.36	16 %	95	21 %	10 %	2	50 %
Gynaecology	70	2.48	40 %	105	29 %	10 %	1	100 %

Specialty	Available Beds					Max Bed Occupancy	Max Beds
	Monday	Tuesday	Wednesday	Thursday	Friday		
Intensive Care	40	42	43	44	47	90 %	60
General Surgery	60	64	70	76	80	90 %	145
Children's Surgery	42	50	50	50	54	90 %	65
Orthopaedics	40	44	50	56	61	90 %	75
Renal and Urology	40	40	40	45	49	90 %	65
Gynaecology	30	35	39	40	40	90 %	45

Figure 8.2 Input screen for the registration generator parameter and the beds availability.

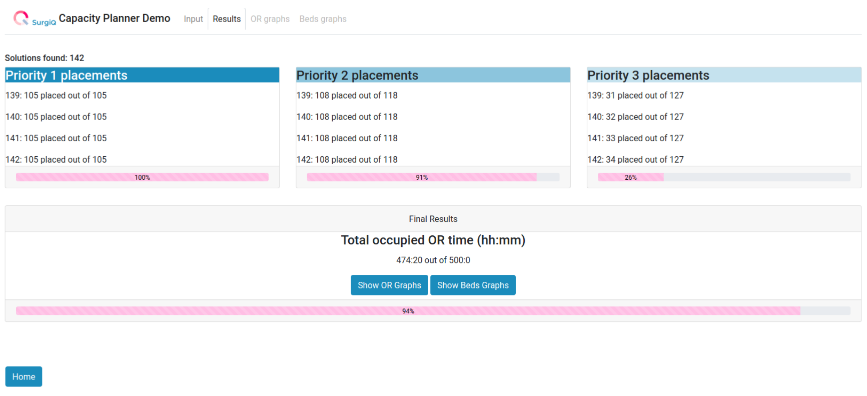
- the number of registrations we aim to assign for each specialty;
- the parameters (mean and coefficient of variation) of the Gaussian distribution used to generate the predicted LOS in the ward after the surgery;
- the parameters (mean and coefficient of variation) of the Gaussian distribution used to generate the predicted surgery lengths;
- the ratio of patients predicted to need a place in the ICU ward; and
- the parameters (mean and coefficient of variation) of the Gaussian distribution used to generate the predicted LOS in the ICU ward after the surgery.

In the right table, the user can set the number of available beds for each ward connected to a specialty. From left to right the parameters are:

- the specialty names;
- the number of available beds for each day of the planning period (in the figure a five days period is shown);
- the percentage of the maximum number of beds potentially available, reflecting the practice usually used by hospitals to reserve a bed quota for emergencies and unexpected events; and
- the total number of beds allocated to the specialty.

The number of beds tends to increase during the planning period to simulate the patients operated during the previous period that still occupy a bed at the beginning of the planning period and are gradually discharged. In the overall results screen (see Figure 8.3) the user can monitor in real-time the evolution of the process and, finally, read the final results:

- At the top of the screen there are three cards containing the number of assigned registrations out of the total, arranged according to their priority class, for each solution found by the solver engine. Each number is continuously updated during the execution whenever a new solution is found. The percentage of assigned registrations is represented by a progress bar at the bottom of each card.
- At the bottom we summarize the final results at the end of the execution. In particular, the OR time out of the total available is reported, both in absolute numbers and as a percentage through a progress bar.



Solutions found: 142

Priority 1 placements	Priority 2 placements	Priority 3 placements
139: 105 placed out of 105	139: 108 placed out of 118	139: 31 placed out of 127
140: 105 placed out of 105	140: 108 placed out of 118	140: 32 placed out of 127
141: 105 placed out of 105	141: 108 placed out of 118	141: 33 placed out of 127
142: 105 placed out of 105	142: 108 placed out of 118	142: 34 placed out of 127
100%	91%	26%

Final Results

Total occupied OR time (hh:mm)

474:20 out of 500:0

94%

Figure 8.3 Results screen.

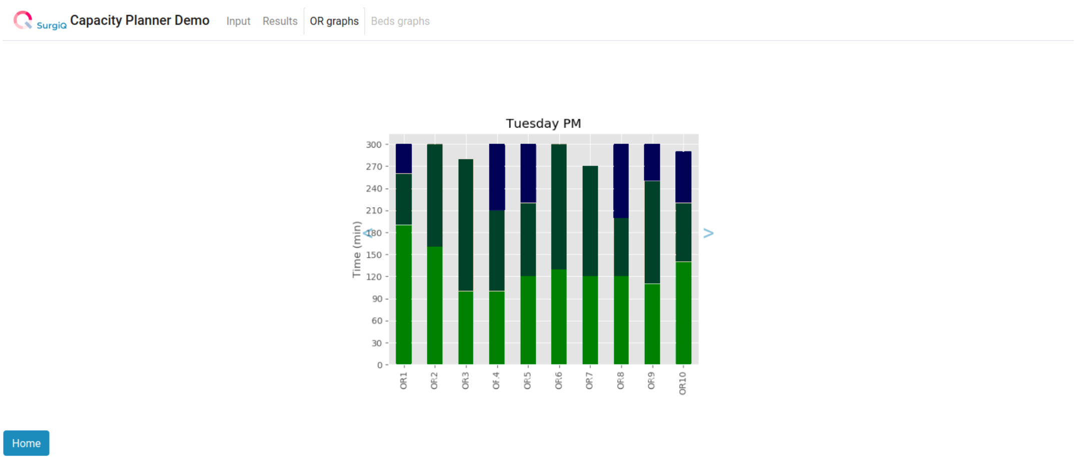
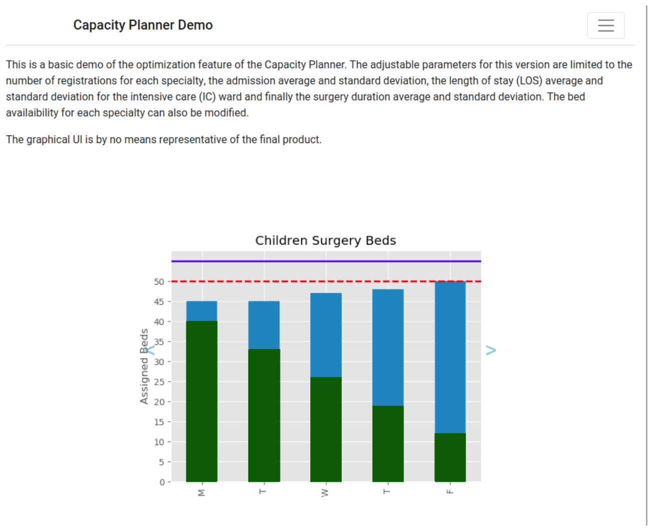


Figure 8.4 Graphical representation of the OR schedules for a single day and shift.

- Finally, we have two links that lead to the graphical representation of the OR scheduling and bed occupancy for each day of the planning period.

The graphical representation of each OR schedule is shown through colored bars, one for each day and shift of the period (see Figure 8.4). In the OR graphs each column displays an OR and each bar inside the column represents a registration. The bed occupancy is also shown through a carousel of stacked bar graphs (see Figure 8.5). This time each graph displays the bed occupancy of a single specialty: each column shows the situation in a day, in particular the green part of the bar denotes the beds already occupied by patients operated previously, while the blue part shows the beds occupied by the patients scheduled in the current period. The solid line shows the total number of beds assigned to the specialty, while the dashed line gives the maximum number of available beds, respecting the quota for emergencies.



Capacity Planner Demo

This is a basic demo of the optimization feature of the Capacity Planner. The adjustable parameters for this version are limited to the number of registrations for each specialty, the admission average and standard deviation, the length of stay (LOS) average and standard deviation for the intensive care (IC) ward and finally the surgery duration average and standard deviation. The bed availaibility for each specialty can also be modified.

The graphical UI is by no means representative of the final product.

Figure 8.5 Graphical representation of the bed occupancy for a single specialty.

Chapter 9

Related Work

In this section we review related literature, organized into two paragraphs. The first paragraph is devoted to outlining different techniques for solving the ORS problem, while in the second paragraph we report about other scheduling problems where ASP has been employed.

Solving ORS problems. Aringhieri et al. (2015a) addressed the joint OR planning (MSS) and scheduling problem, described as the allocation of OR time blocks to specialties together with the subsets of patients to be scheduled within each time block over a one week planning horizon. They developed a 0-1 linear programming formulation of the problem and used a two-level meta-heuristic to solve it. Its effectiveness was demonstrated through numerical experiments carried out on a set of instances based on real data and resulted, for benchmarks of 80-100 assigned registrations, in a 95-98% average OR utilization rate, for a number of ORs ranging from 4 to 8. The execution times were around 30-40 seconds. In Landa et al. (2016), the same authors introduced a hybrid two-phase optimization algorithm which exploits neighborhood search techniques combined with Monte Carlo simulation, in order to solve the joint advance and allocation scheduling problem, taking into account the inherent uncertainty of surgery durations. In both the previous works, the authors solve the bed management part of the problem limited to weekend beds, while assuming that each specialty has its own post-surgery beds from Monday to Friday with no availability restriction. In Aringhieri et al. (2015b), some of the previous authors face the bed management problem for all the days of the week, with the aim to level the post-surgery ward bed occupancies during the days, using a Variable Neighbourhood Search approach.
Other relevant approaches are: Abedini et al. (2016), that developed a bin packing model with a multi-step approach and a priority-type-duration rule; Molina-Pariente et al. (2015), that tackled the problem of assigning an intervention date and an OR to a set of surgeries on the

waiting list, minimizing the access time for patients with diverse clinical priority values; and Zhang et al. (2017), that addressed the problem of OR planning with different demands from both elective patients and non-elective ones, with priorities in accordance with urgency levels and waiting times. However, bed management is not considered in these three last mentioned approaches. Approaches more related to the treatment of surgical teams are: Meskens et al. (2013) considered the surgical teams in the computation of an OR schedule, and developed a model using Constraint Programming (CP) with multiple constraints such as availability, staff preferences and affinities among surgical teams. They optimize the use of ORs by minimizing makespan and maximizing affinities among surgical team members. The effectiveness of their proposed method for improving surgical cases was evaluated using real data from an Hospital. Hamid et al. (2019) incorporated the decision-making styles (DMS) of the surgical team to improve the compatibility level by considering constraints such as the availability of material resources, priorities of patients, and availability, skills, and competencies of the surgical team. They developed a multi-objective mathematical model to schedule surgeries. Two metaheuristics, namely Non-dominated Sorting Genetic Algorithm and Multi-Objective Particle Swarm Optimization, were developed to find pareto-optimal solutions. Xiang et al. (2015) proposed an Ant Colony Optimization (ACO) approach to surgical scheduling taking into account all resources in the entire process of a surgery. The problem was represented as an extended multi-resource constrained flexible job shop scheduling problem, which was solved using a two-level hierarchical graph to integrate sequencing job and allocating resources. To evaluate the efficiency of ACO, a Discrete Event System (DES) model of an OR system was developed in the simulation platform SIMIO. Monteiro et al. (2015) developed a comprehensive multi-objective mathematical model using epsilon-constraint method coupled to the CPLEX solver. Vijayakumar et al. (2013) used Mixed Integer Programming (MIP) model for multi-day, multi-resource, patient-priority-based surgery scheduling. A First Fit Decreasing algorithm was developed. From a solution time perspective, their model took hours and in most cases was unable to optimally solve the problem. Belkhamsa et al. (2018) proposed two meta heuristics, an Iterative Local Search (ILS) approach and Hybrid Genetic Algorithm (HGA) to solve a daily surgery scheduling problem. Zhou et al. (2016) developed an Integer Programming model for optimal surgery schedule of assigning patients to different resources in any surgical stage. They used Lagrangian Relaxation algorithm and solved the subproblem by using branch and bound. They verified their model using real data instances from an Hospital. A common issue with all such solutions seem to be computation time and scalability.

ASP in scheduling problems. We already mentioned in the introduction that ASP has been already successfully used for solving hard combinatorial and application problems in several research areas, and in particular in scheduling. Concerning ORS, the problem has been already addressed in Dodaro et al. (2018, 2019), but without taking into account beds and surgical teams, which instead are a fundamental resource to be considered. Concerning scheduling problems other than ORS, ASP encodings were proposed for the following problems: *Incremental Scheduling Problem* (Balduccini (2011); Calimeri et al. (2016); Gebser et al. (2017a,b)), where the goal is to assign jobs to devices such that their executions do not overlap one another; *Team Building Problem* (Ricca et al. (2012)), where the goal is to allocate the available personnel of a seaport for serving the incoming ships; *Nurse Scheduling Problem* (Alviano et al. (2017, 2018); Dodaro and Maratea (2017)), where the goal is to create a scheduling for nurses working in hospital units; *Interdependent Scheduling Games* (Amendola (2018)), which requires interdependent services among players, that control only a limited number of services and schedule independently, and the *Conference Paper Assignment Problem* (Amendola et al. (2016)), which deals with the problem of assigning reviewers in the Program Committee to submitted conference papers. More recent problems, still in the Healthcare domain, include the *Chemotherepy Treatment Scheduling* problem (Dodaro et al. (2021)), in which patients are assigned a chair or a bed for their treatments, and the *Rehabilitation Scheduling Problem* (Cardellini et al. (2021)), which assigns patients to operators in rehabilitation sessions. Other relevant papers are (Gebser et al. (2018)), where, in the context of routing driverless transport vehicles, the setup problem of routes such that a collection of transport tasks is accomplished in case of multiple vehicles sharing the same operation area is solved via ASP, in the context of car assembly at Mercedes-Benz Ludwigsfelde GmbH, and the recent survey paper by Falkner et al. (2018), where industrial applications dealt with ASP are presented, including those involving scheduling problems.

Chapter 10

Conclusions and future research

In this research, we have employed ASP for solving the ORS problem with several enhancements, i.e., bed management and the inclusion of surgical teams, given ASP has already proved to be a viable tool for solving scheduling problems due to the readability of the encoding and availability of efficient solvers. The specifications of the problem are modularly expressed as rules in the ASP encoding, and CLINGO has been used. We have then presented the results of experimental and scalability analysis on ORS benchmarks with realistic sizes and parameters on three scenarios, that reveal that our solution is able to utilize efficiently whichever resource is more constrained, being either the OR time or the beds. Moreover, for the planning length of 5 days usually used in small-medium Italian hospitals, this is obtained in short timings in line with the needs of the application. We also developed and tested a rescheduling procedure to be used if the original schedule could not be fully executed for some reason. We finally also presented a web framework that supports the online execution of our scheduling solution. While we do not directly manage emergencies, the flexibility of our algorithm can be exploited even in those cases. For example, if a part of the hospital resources (being OR time, ICU, or ward beds) must be suddenly redirected to serve other purposes, as for example, happened in the Covid-19 emergency, by simply adjusting the numbers, our solution can immediately be utilized to manage the remaining resources at the best of their capacity.

Future work includes both short-term and medium-term objectives. Short-term objectives includes: the usage of other solving paradigms such as, e.g., SMT or CSP. In terms of efficiency, we plan to evaluate both heuristics and optimization techniques (see, e.g. Alviano et al. (2020); Giunchiglia et al. (2002, 2003); Rosa et al. (2008)), as well as further clingo options, and improvement to the current encoding. Medium-term objectives, instead aim at completing the picture that is somehow fragmented in some respects and includes: (i) testing

the proposed solutions on real-world data for hospitals, (ii) the inclusion of bed management in the alternative encoding, which showed superior performance; (iii) the extension of re-scheduling solutions to the treatment of surgical teams; (iv) the extension of the web application to deal with both surgical teams and re-scheduling, and (v) a further extension of all our contributions to also include nurses in the problem definition and modeling.(vi) the further extension of the web Application also for surgical teams.
All materials presented in this work, including benchmarks and encodings, can be found at: http://www.star.dist.unige.it/~marco/RuleMLRR2TPLP/material.zip.

Bibliography

Abedini, A., Ye, H., and Li, W. (2016). Operating Room Planning under Surgery Type and Priority Constraints. *Procedia Manufacturing*, 5:15–25.

Alviano, M., Dodaro, C., and Maratea, M. (2017). An advanced answer set programming encoding for nurse scheduling. In Esposito, F., Basili, R., Ferilli, S., and Lisi, F. A., editors, *Advances in Artificial Intelligence - Proceedings of the 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017)*, volume 10640 of *Lecture Notes in Computer Science*, pages 468–482. Springer.

Alviano, M., Dodaro, C., and Maratea, M. (2018). Nurse (re)scheduling via answer set programming. *Intelligenza Artificiale*, 12(2):109–124.

Alviano, M., Dodaro, C., Marques-Silva, J., and Ricca, F. (2020). Optimum stable model search: Algorithms and implementation. *Journal of Logic and Computation*, 30(4). In press.

Alviano, M., Faber, W., and Gebser, M. (2015). Rewriting recursive aggregates in answer set programming: back to monotonicity. *Theory and Practice of Logic Programming*, 15(4-5):559–573.

Amendola, G. (2018). Preliminary results on modeling interdependent scheduling games via answer set programming. In *RiCeRcA@AI*IA*, volume 2272 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Amendola, G., Dodaro, C., Leone, N., and Ricca, F. (2016). On the application of answer set programming to the conference paper assignment problem. In Adorni, G., Cagnoni, S., Gori, M., and Maratea, M., editors, *Advances in Artificial Intelligence - Proceedings of the 15th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2016)*, volume 10037 of *Lecture Notes in Computer Science*, pages 164–178. Springer.

Aringhieri, R., Landa, P., Soriano, P., Tànfani, E., and Testi, A. (2015a). A two level metaheuristic for the operating room scheduling and assignment problem. *Computers & Operations Research*, 54:21–34.

Aringhieri, R., Landa, P., and Tànfani, E. (2015b). Assigning surgery cases to operating rooms: A vns approach for leveling ward beds occupancies. *Proceedings of the 3rd International Conference on Variable Neighborhood Search (VNS'14). Electronic Notes in Discrete Mathematics*, 47:173 – 180.

Balduccini, M. (2011). Industrial-size scheduling with ASP+CP. In *Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings*, volume 6645 of *Lecture Notes in Computer Science*, pages 284–296. Springer.

Belkhamsa, M., Jarboui, B., and Masmoudi, M. (2018). Two metaheuristics for solving no-wait operating room surgery scheduling problem under various resource constraints. *Comput. Ind. Eng.*, 126:494–506.

Brewka, G., Eiter, T., and Truszczynski, M. (2011). Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103.

Buccafurri, F., Leone, N., and Rullo, P. (2000). Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):845–860.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., and Schaub, T. (2020). ASP-Core-2 input language format. *Theory and Practice of Logic Programming*, 20(2):294–309.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., and Schaub, T. (2013). ASP-Core-2 Input Language Format.

Calimeri, F., Gebser, M., Maratea, M., and Ricca, F. (2016). Design and results of the Fifth Answer Set Programming Competition. *Artificial Intelligence*, 231:151–181.

Cardellini, M., Nardi, P. D., Dodaro, C., Galatà, G., Giardini, A., Maratea, M., and Porro, I. (2021). A two-phase ASP encoding for solving rehabilitation scheduling. In Moschoyiannis, S., Peñaloza, R., Vanthienen, J., Soylu, A., and Roman, D., editors, *Proceedings of the 5th International Joint Conference on Rules and Reasoning (RuleML+RR 2021)*, volume 12851 of *Lecture Notes in Computer Science*, pages 111–125. Springer.

Dodaro, C., Galatà, G., Grioni, A., Maratea, M., Mochi, M., and Porro, I. (2021). An ASP-based solution to the chemotherapy treatment scheduling problem. *Theory and Practice of Logic Programming*, 21(6):835–851.

Dodaro, C., Galatà, G., Maratea, M., and Porro, I. (2018). Operating room scheduling via answer set programming. In Ghidini, C., Magnini, B., Passerini, A., and Traverso, P., editors, *Advances in Artificial Intelligence - Proceedings of the 17th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2018)*, volume 11298 of *Lecture Notes in Computer Science*, pages 445–459. Springer.

Dodaro, C., Galatà, G., Maratea, M., and Porro, I. (2019). An ASP-based framework for operating room scheduling. *Intelligenza Artificiale*, 13(1):63–77.

Dodaro, C. and Maratea, M. (2017). Nurse scheduling via answer set programming. In Balduccini, M. and Janhunen, T., editors, *Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2017)*, volume 10377 of *Lecture Notes in Computer Science*, pages 301–307. Springer.

Faber, W., Pfeifer, G., and Leone, N. (2011). Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298.

Falkner, A. A., Friedrich, G., Schekotihin, K., Taupe, R., and Teppan, E. C. (2018). Industrial applications of answer set programming. *Künstliche Intelligenz*, 32(2-3):165–176.

Ferraris, P. (2011). Logic programs with propositional connectives and aggregates. *ACM Trans. Comput. Log.*, 12(4):25.

Gebser, M., Harrison, A., Kaminski, R., Lifschitz, V., and Schaub, T. (2015). Abstract gringo. *Theory Practice of Logic Programming*, 15(4-5):449–463.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Wanko, P. (2016). Theory solving made easy with clingo 5. In Carro, M., King, A., Saeedloei, N., and Vos, M. D., editors, *Proceedings of ICLP (Technical Communications)*, volume 52 of *OASICS*, pages 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Gebser, M., Maratea, M., and Ricca, F. (2017a). The design of the seventh answer set programming competition. In Balduccini, M. and Janhunen, T., editors, *LPNMR*, volume 10377 of *Lecture Notes in Computer Science*, pages 3–9. Springer.

Gebser, M., Maratea, M., and Ricca, F. (2017b). The sixth answer set programming competition. *Journal of Artificial Intelligence Research*, 60:41–95.

Gebser, M., Obermeier, P., Schaub, T., Ratsch-Heitmann, M., and Runge, M. (2018). Routing driverless transport vehicles in car assembly with answer set programming. *Theory Practice of Logic Programming*, 18(3-4):520–534.

Giunchiglia, E., Maratea, M., and Tacchella, A. (2002). Dependent and independent variables in propositional satisfiability. In Flesca, S., Greco, S., Leone, N., and Ianni, G., editors, *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA 2002)*, volume 2424 of *Lecture Notes in Computer Science*, pages 296–307. Springer.

Giunchiglia, E., Maratea, M., and Tacchella, A. (2003). (In)Effectiveness of look-ahead techniques in a modern SAT solver. In Rossi, F., editor, *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of *Lecture Notes in Computer Science*, pages 842–846. Springer.

Hamid, M., Nasiri, M. M., Werner, F., Sheikhahmadi, F., and Zhalechian, M. (2019). Operating room scheduling by considering the decision-making styles of surgical team members: A comprehensive approach. *Computers & Operation Research*, 108:166–181.

Landa, P., Aringhieri, R., Soriano, P., Tànfani, E., and Testi, A. (2016). A hybrid optimization algorithm for surgeries scheduling. *Operations Research for Health Care*, 8:103–114.

Macario, A. (2010). What does one minute of operating room time cost? *Journal of Clinical Anesthesia*, 22(4):233–236.

Meskens, N., Duvivier, D., and Hanset, A. (2013). Multi-objective operating room scheduling considering desiderata of the surgical team. *Decis. Support Syst.*, 55(2):650–659.

Molina-Pariente, J. M., Hans, E. W., Framinan, J. M., and Gomez-Cia, T. (2015). New heuristics for planning operating rooms. *Computers & Industrial Engineering*, 90:429–443.

Monteiro, T., Meskens, N., and Wang, T. (2015). Surgical scheduling with antagonistic human resource objectives. *International Journal of Production Research*, 53(24):7434–7449.

Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., and Leone, N. (2012). Team-building with answer set programming in the Gioia-Tauro seaport. *Theory and Practice of Logic Programming*, 12(3):361–381.

Rosa, E. D., Giunchiglia, E., and Maratea, M. (2008). A new approach for solving satisfiability problems with qualitative preferences. In Ghallab, M., Spyropoulos, C. D., Fakotakis, N., and Avouris, N. M., editors, *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 510–514. IOS Press.

Vijayakumar, B., Parikh, P. J., Scott, R., Barnes, A., and Gallimore, J. (2013). A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operation Research*, 224(3):583–591.

Xiang, W., Yin, J., and Lim, G. (2015). An ant colony optimization approach for solving an operating room surgery scheduling problem. *Comput. Ind. Eng.*, 85:335–345.

Zhang, J., Dridi, M., and Moudni, A. E. (2017). A stochastic shortest-path MDP model with dead ends for operating rooms planning. In *Proceedings of the 23rd International Conference on Automation and Computing (ICAC 2017)*, pages 1–6. IEEE.

Zhou, B., Yin, M., and Lu, Z. (2016). An improved lagrangian relaxation heuristic for the scheduling problem of operating theatres. *Comput. Ind. Eng.*, 101:490–503.

Appendix A

Detailed results

In order for further investigations this part reports metrics like CPU time, answer sets, optimization and assigned registrations for the experiments performed on both encoding 1 and 2, for scenario A in Table A.1 , for scenario B in Table A.2, for scenario C in Table A.3 and for scenario D in Table A.4. Each table consists of eight column, with first column showing the encoding, the second column shows the benchmark, third column reports the time-limit in second for the solver, the forth column shows the cpu time, the fifth column shows the optimization value, the sixth column shows the generated answer sets and the last two columns reports the total assigned and total generated registrations for each benchmark. For comparison purposes in each Scenario (A,B,C,D) a single instance from each benchmark tested by both encodings are reported. It can be seen the first two column shows the name of the encoding and the benchmark for which the experiments are performed. The next four columns shows the time-limit for solver, CPU time, optimization and generated answer sets while the last two column mention the total assigned registrations and total generated registrations for each benchmark. The further details of these experiments for all the instances can be found in part3.
As we can see below for scenario A in Table A.1 for 1-day benchmark and for both the encodings the total assigned registrations are 47 but encoding-2 took less CPU time 8.2 seconds than encoding 1 for 299.5 seconds. Similarly in other scenarios (B,C and D) for 1-day benchmark, encoding 2 took less CPU time.
Furthermore, for scenario B in Table A.2, for scenario C in Table A.3 and scenario D in Table A.4 the CPU time or computation time for 2,3 and 5-days benchmarks for both the encodings 1 and 2 remains almost the same with equal number of total number of assigned registrations.

Additionally, Table A.5, Table A.6, Table A.7 and Table A.8 reports the CPU time, Optimum , Answer Set and Total assigned registration for all the 10 instances for 1, 2, 3 and 5 day benchmark executed for both encoding 1 and 2.

Overall results shows that the two encodings have the same results on easy instances (1 day) as far as optimization is concerned of course the assignment may be different, but the cost is the same. Also the new encoding can perform better, also significantly. The answer sets "11 vs 20" reported below in Table A.1 for 1 day benchmark in scenario A is because we have a different set of predicates, and thus the heuristic of Clingo can take different paths towards solution outputting different (partial) solutions. The same can be observed in Table A.2, A.3 and A.4.

Table A.1 Solver time-limit, cpu time, answer sets and the assigned registrations of a single instance for Scenario A

Encoding	Bench.	Time-limit (sec)	CPU Time (sec)	Optimization	Answer Set	Total Assigned Regs’	Total Generated Regs’
1	5 days	300.0	-	-	-	-	350
	3 days	300.0	300.0	[31,56]	82	123	210
	2 days	300.0	300.0	[21,37]	58	82	140
	1 day	300.0	300.0	[3,20]	41	47	70
2	5 days	300.0	300.0	[53,86]	209	211	350
	3 days	300.0	300.0	[18,67]	142	125	210
	2 days	300.0	300.0	[13,46]	65	81	140
	1 day	300.0	8.2	[3,20]	11	47	70

Table A.2 Solver time-limit, cpu time, answer sets and the assigned registrations of a single instance for Scenario B

Encoding	Bench.	Time-limit (sec)	CPU Time (sec)	Optimization	Answer Set	Total Assigned Regs’	Total Generated Regs’
1	5 days	300.0	300.0	[27,107]	148	216	350
	3 days	300.0	300.0	[26,62]	104	122	210
	2 days	300.0	300.0	[18,44]	70	78	140
	1 day	300.0	300.0	[6,21]	44	43	70
2	5 days	300.0	300.0	[48,82]	218	220	350
	3 days	300.0	300.0	[11,73]	193	126	210
	2 days	300.0	300.0	[13,49]	80	78	140
	1 day	300.0	300.0	[6,21]	32	43	70

Table A.3 Solver time-limit, cpu time, answer sets and the assigned registrations of a single instance for Scenario C

Encoding	Bench.	Time-limit (sec)	CPU Time (sec)	Optimization	Answer Set	Total Assigned Regs'	Total Generated Regs'
1	5 days	300.0	300.0	[19,121]	140	210	350
	3 days	300.0	300.0	[24,52]	113	134	210
	2 days	300.0	300.0	[12,39]	68	89	140
	1 day	300.0	300.0	[3,26]	44	41	70
2	5 days	300.0	300.0	[41,97]	233	212	350
	3 days	300.0	300.0	[18,60]	193	132	210
	2 days	300.0	300.0	[12,39]	97	89	140
	1 day	300.0	3.9	[3,26]	41	41	70

Table A.4 Solver time-limit, cpu time, answer sets and the assigned registrations of a single instance for Scenario D

Encoding	Bench.	Time-limit (sec)	CPU Time (sec)	Optimization	Answer Set	Total Assigned Regs'	Total Generated Regs'
1	5 days	300.0	300.0	[20,107]	205	223	350
	3 days	300.0	300.0	[24,52]	113	134	210
	2 days	300.0	300.0	[7,46]	68	87	140
	1 day	300.0	300.0	[2,25]	49	43	70
2	5 days	300.0	300.0	[35,91]	230	224	350
	3 days	300.0	300.0	[18,60]	193	132	210
	2 days	300.0	300.0	[7,46]	124	87	140
	1 day	300.0	32.6	[2,25]	43	43	70

Table A.5 Solver time, cpu time, answer set and the assigned registrations for 1-day in scenario A with encoding 1 and 2

Solver Time (sec)	**Encoding 1**				**Encoding 2**				Total Regs'
	CPU Time (sec)	Optimum	Answer Set	Total Assigned Regs'	CPU Time (sec)	Optimum	Answer Set	Total Assigned Regs'	
300.0	300.0	[3,20]	41	47	8.2	[3,20]	11	47	70
300.0	300.0	[9,20]	44	41	11.3	[9,20]	13	41	
300.0	300.0	[3,25]	31	42	7.7	[3,25]	9	42	
300.0	300.0	[7,22]	38	41	13.2	[7,22]	22	41	
300.0	300.0	[0,26]	38	44	15.9	[0,26]	17	44	
300.0	300.0	[3,23]	42	44	10.5	[3,23]	22	44	
300.0	300.0	[7,18]	49	45	14.6	[7,18]	16	45	
300.0	300.0	[5,19]	49	46	9.4	[4,21]	14	45	
300.0	300.0	[12,14]	45	44	55.3	[12,13]	16	45	
300.0	144.6	[3,20]	42	47	3.2	[3,20]	18	47	

Table A.6 Solver time, cpu time, answer set and the assigned registrations for 2-days in scenario A with encoding 1 and 2

Solver Time (sec)	**Encoding 1**				**Encoding 2**				Total Regs’
	CPU Time (sec)	Optimum	Answer Set	Total Assigned Regs’	CPU Time (sec)	Optimum	Answer Set	Total Assigned Regs’	
300.0	300.0	[21,37]	58	82	300.0	[13,46]	65	81	140
300.0	300.0	[23,35]	71	82	300.0	[17,41]	94	82	
300.0	300.0	[16,34]	66	90	300.0	[13,38]	58	89	
300.0	300.0	[23,31]	72	86	300.0	[12,41]	84	87	
300.0	300.0	[19,34]	61	87	300.0	[12,40]	64	88	
300.0	300.0	[11,43]	66	86	300.0	[6,49]	74	85	
300.0	300.0	[16,43]	69	81	300.0	[12,48]	62	80	
300.0	300.0	[19,38]	59	83	300.0	[11,48]	64	81	
300.0	300.0	[15,40]	76	85	300.0	[8,45]	80	87	
300.0	144.6	[14,42]	57	84	300.0	[7,49]	76	84	

Table A.7 Solver time, cpu time, answer set and the assigned registrations for 3-days in scenario A with encoding 1 and 2

Solver Time (sec)	**Encoding 1**				**Encoding 2**				Total Regs’
	CPU Time (sec)	Optimum	Answer Set	Total Assigned Regs’	CPU Time (sec)	Optimum	Answer Set	Total Assigned Regs’	
300.0	300.0	[31,56]	82	123	300.0	[18,67]	142	125	210
300.0	300.0	[21,61]	86	128	300.0	[24,54]	139	132	
300.0	300.0	[39,50]	99	121	300.0	[23,66]	146	121	
300.0	300.0	[36,52]	100	122	300.0	[23,60]	126	127	
300.0	300.0	[30,57]	92	123	300.0	[24,60]	132	126	
300.0	300.0	[26,59]	78	125	300.0	[20,64]	97	126	
300.0	300.0	[20,72]	78	118	300.0	[26,62]	113	122	
300.0	300.0	[30,59]	109	121	300.0	[22,59]	132	129	
300.0	300.0	[17,74]	90	119	300.0	[9,78]	148	123	
300.0	300.0	[36,51]	88	123	300.0	[22,62]	142	126	

Table A.8 Solver time, cpu time, answer set and the assigned registrations for 5-day in scenario A with encoding 1 and 2

Solver Time (sec)	**Encoding 1**				**Encoding 2**				Total Regs'
	CPU Time (sec)	Optimum	Answer Set	Total Assigned Regs'	CPU Time (sec)	Optimum	Answer Set	Total Assigned Regs'	
300.0	300.0	-	-	-	300.0	[53,86]	209	211	350
300.0	300.0	-	-	-	300.0	[47,92]	198	211	
300.0	300.0	-	-	-	300.0	[49,88]	184	213	
300.0	300.0	-	-	-	300.0	[52,84]	220	214	
300.0	300.0	-	-	-	300.0	[57,88]	197	205	
300.0	300.0	-	-	-	300.0	[57,84]	209	209	
300.0	300.0	-	-	-	300.0	[51,92]	184	207	
300.0	300.0	-	-	-	300.0	[47,83]	220	220	
300.0	300.0	-	-	-	300.0	[56,86]	197	208	
300.0	300.0	-	-	-	300.0	[56,87]	170	208	