Nuclear Medicine Scheduling via Answer Set Programming

Carmine Dodaro[1], Giuseppe Galatà[2], Cinzia Marte[1], Marco Maratea[1,*] and Marco Mochi[3]

[1]*DeMaCS, University of Calabria, Rende, Italy*
[2]*SurgiQ srl, Genova, Italy*
[3]*DIBRIS, University of Genoa, Genoa, Italy*

Abstract
The Nuclear Medicine Scheduling (NMS) problem consists of assigning patients to a day, in which the patient will undergo the medical check, the preparation, and the actual image detection process. The schedule of the patients should consider the different requirements of the patients and the available resources, e.g., varying time required for different diseases and radiopharmaceuticals used, number of injection chairs, and tomographs available. In this paper we present a solution where modeling and solving are done via Answer Set Programming. Experiments employing real data show that the solution provides satisfying results in a short time.

Keywords
Answer Set Programming, Logic Programming, Digital Health

1. Introduction

Nuclear Medicine is a medical specialty that uses radiopharmaceuticals, a particular kind of drug containing radioactive elements, to treat or diagnose diseases. According to data by the Italian Ministry of Health, almost 2 millions nuclear medicine exams have been carried on during 2022 in Italy[1]. The process of treating patients with this technique is complex since it involves multiple resources of the hospital and requires multiple steps of varying time. Moreover, often these drugs contain radioactive elements characterized by short half-lives, meaning that they decay rapidly after their preparation. Thus, the timing should be as precise as possible in order to obtain images of good quality. Addressing this problem effectively is crucial due to the nature of the diagnosed illnesses and treated through nuclear medicine, alongside the significant costs associated with this kind of technique. An efficient, possibly optimal, solution can reduce the waiting time of the patients and can thus increase the effective utilization of the resources, avoiding waste of time and resources. Nevertheless, reducing the unnecessary time spent by the patients in the hospital is vital for increasing the satisfaction of the patients.

CILC'24: 39th Italian Conference on Computational Logic June 26-28, 2024 - Rome, Italy
carmine.dodaro@unical.it (C. Dodaro); giuseppe.galata@surgiq.com (G. Galatà); cinzia.marte@unical.it (C. Marte); marco.maratea@unical.it (M. Maratea); marco.mochi@edu.unige.it (M. Mochi)
0000-0002-5617-5286 (C. Dodaro); 0000-0002-1948-4469 (G. Galatà); 0000-0003-3920-8186 (C. Marte); 0000-0002-9034-25276 (M. Maratea); 0000-0002-5849-3667 (M. Mochi)

[1] https://www.salute.gov.it/imgs/C_17_pubblicazioni_3425_allegato.pdf

Thus, the Nuclear Medicine Scheduling (NMS) problem consists of assigning patients to a day, in which the patient will undergo the medical check, the preparation, and the actual image detection process. The schedule of the patients consider the different requirements of the patients and the available resources, e.g., varying time required for different procedures and radiopharmaceuticals used, number of injection chairs and tomographs available. We followed the definition of the problem given by Medipass[2], leading provider of technological innovation across cancer care and diagnostic imaging in Italy, in collaboration with SurgiQ[3], an Italian company active in planning and scheduling solutions.

Complex combinatorial problems, possibly involving optimizations, such as the NMS problem, are usually the target applications of AI languages for knowledge representation and reasoning, such as Answer Set Programming (ASP). The success of ASP is due to different factors, including a simple but rich syntax [1], which includes optimization statements as well as powerful database-inspired constructs like aggregates, an intuitive semantics [2], the availability of efficient solvers (see, e.g., [3, 4, 5, 6, 7]), and the fruitful combination with machine learning approaches (see, e.g., [8, 9]). Moreover, ASP has been already successfully employed to solve other scheduling applications (see, e.g., [10, 11, 12, 13, 14, 15]).

Subsequently, we propose a solution leveraging ASP, where we design an encoding that encapsulates the problem specifications. This encoding is then processed using the state-of-the-art system CLINGO [4] and executed on real data provided by Medipass. The results of an experimental analysis conducted on such data demonstrate that our solution achieves satisfactory quality outcomes, even in time-constrained scenarios.

The paper is structured as follows. Sections 2 and 3 present an informal description of the problem and a precise, mathematical formulation, respectively. Then, after the needed background on ASP in Section 4, Section 5 shows our ASP encoding, whose experimental evaluation is presented in Section 6. The paper ends by discussing related works and conclusions in Section 7 and 8, respectively.

2. Problem Description

The NMS problem consists of assigning patients to a day and to a tomograph and/or injection chair if required by the patient or the specific procedure. In our problem, for each day we consider a set of 120 time slots (TS), each representing 5 minutes. Each patient needs an exam and each exam is linked to a protocol defining the phases and the time required for each phase. We considered 11 different protocols. Each protocol can encompass up to four phases, represented as (p_1) anamnesis, (p_2) medical check, (p_3) radiopharmaceuticals injection and bio-distribution time, and (p_4) image detection. Moreover, each phase can require a different amount of time depending on the exam. Table 1 shows the total time needed by each protocol and the partial time required by each phase, expressed in the number of time slots used.

Due to the high number of phases required by each patient and the variety of the considered protocols, in many clinics the schedule of the patients is sub-optimal. A sub-optimal schedule is problematic not only because of the high cost of the drugs and machines involved in the exams,

[2]https://ergeagroup.com/it/.
[3]https://surgiq.com/.

Protocol Number	#TS for p_1	#TS for p_2	#TS for p_3	#TS for p_4	#TS total
813	3	2	0	8	13
814	3	2	0	8	13
815	2	2	4	6	14
817	2	2	3	7	14
819	2	2	5	7	16
822	2	2	2	7	13
823	2	2	10	7	21
824	2	2	5	8	17
827	2	2	2	7	13
828	3	3	0	7	13
888	2	2	2	9	15

Table 1
Partial and total time required by each protocol in terms of number of time slots.

but is particularly detrimental for the patients since the order and the time required by each phase, in particular the injection and the bio-distribution time, is fundamental for a proper image detection.

Different clinics have different resource availability and may have different requirements in defining a proper solution. Here we present the criteria followed in the clinic that provided us with the real data of the patients and that we use to define the problem. We considered a clinic with two rooms, each with one tomograph and three injection chairs. We started from a list of patients, each requiring a specific protocol, to be assigned in a day. A proper solution must satisfy the following conditions:

- a starting and an ending time should be assigned to every scheduled patient for each required phase;
- there must be at most two patients concurrently in the medical check phase;
- the injection phase must be done in an injection chair or on a tomograph according to the required protocol;
- each injection chair and tomograph can be used by just one patient at the same time;
- patients requiring an injection chair must be assigned to the tomograph of the same room;
- protocol identified by the id 815 cannot be assigned on the same day and tomograph for more than one patient.

The solution should maximize the number of scheduled patients in the considered days and, to increase the satisfaction of the patients and the effectiveness of the exams, the solution should also try to minimize the unnecessary time spent in the clinic by the patients.

3. Formalization of the NMS problem

Let N be the set of reservation numbers, D be the set of days, TS denote the set of time slots, and PR be the set collecting the protocol numbers referred to exams. Each protocol may comprise a maximum of four phases, represented by the set $P = \{p_1, p_2, p_3, p_4\}$. Let R denote the set of rooms and $S = T \cup C \cup \{\varepsilon\}$ be the set representing the available resources, given by the union of the set T of tomographs, the set C of chairs and the element ε denoting that it is no required a resource.

Moreover, let:

- $\lambda : T \times PR \to \mathbb{N}$ be the function that returns the maximum number of exams that can be executed on a tomograph of a specific protocol, for all the protocols that require a limitation;
- $\omega : P \times PR \to \mathbb{N}$ be the function that returns the number of time slots required for each phase of a specific protocol;
- $\beta : PR \to \{0,1\}$ be the function that assigns the value 1 if the protocol requires both a chair and a tomograph, 0 otherwise;
- $\alpha : S \times R \to \{0,1\}$ be the function that assigns the value 1 if there is an association between the resource and the room, 0 otherwise. Furthermore, let $\mathsf{A} = \alpha^{-1}(1)$ be the set collecting all the good associations between resource and room.

To associate a reservation number, a day, and a protocol, we now introduce the notion of *registration*.

Definition 1. *A registration ρ is a function of the form $\rho : N \times D \times PR \to \{0,1\}$ such that $\rho(n,d,x) = 1$ if there exists a reservation n on a day d for the protocol x,* 0 *otherwise. Let* $\mathsf{R} = \rho^{-1}(1) = \{(n,d,x) \in N \times D \times PR \mid \rho(n,d,x) = 1\}$ *be the set collecting all the registrations.*

Consequently, we define the notion of *assignment* to link together a registration with a specific phase of the protocol under consideration and a time slot.

Definition 2. *An assignment τ is a function of the form $\tau : \mathsf{R} \times P \times TS \to \{0,1\}$ such that $\tau(n,d,x,p,y) = 1$ if a phase p and an initial time slot y are assigned to a registration. Let* $\mathsf{T} = \tau^{-1}(1) = \{(n,d,x,p,y) \in R \times P \times TS \mid \tau((n,d,x),p,y) = 1\}$ *be the set collecting all the assignments.*

Before presenting the main problem of this paper, we define the concept of *scheduling*, which connects an assignment with a resource and its allocation.

Definition 3. *A scheduling σ is a function of the form $\sigma : \mathsf{T} \times \mathsf{A} \to \{0,1\}$ that assigns the value* 1 *if there exists a reservation number n on day d for the protocol x, referred to the phase p on time slot y, using the resource s in the room r. The set* $\mathsf{S} = \sigma^{-1}(1) = \{\mathbf{t} = (n,d,x,p,y,s,r) \mid \sigma(\mathbf{t}) = 1\}$ *collects all the tuples eligible as scheduling.*

We can now define the *Nuclear Medicine Scheduling* (NMS) problem.

Definition 4 (NMS). *The Nuclear Medicine Scheduling (NMS) problem is defined as the problem of finding a set ψ of tuples $\mathbf{t} = (n,d,x,p,y,s,r) \in \mathsf{S}$ that satisfies the following conditions:*

(c_1) $\forall (y,s) \in TS \times S \; |\{(n,d,x,p,y',s,r) \in \psi \mid p \neq p_1 \wedge y \in \{y', \dots, y' + w(p,x)\}\}| \leq 1;$

(c_2) $\forall y \in TS \; |\{(n,d,x,p,y',s,r) \in \psi : p = p_1 \land y \in \{y', \ldots y' + w(p,x)\}\}| \leq 2$;
(c_3) $\forall x \in PR : \beta(x) = 1$, *it holds that* $\mathbf{t}' = (n,d,x,p',y',c,r')$ *and* $\mathbf{t}'' = (n,d,x,p'',y'',t,r'')$ *belong to* ψ *iff* $r' = r''$;
(c_4) $\forall t \in T \; |\{x \mid (n,d,x,p,y,s,r) \in \psi\}| \leq \lambda(t,x)$;
(c_5) $\mathbf{t}' = (n,d,x,p_i,y',s',r)$ *and* $\mathbf{t}'' = (n,d,x,p_{i+1},y'',s'',r)$ *belong to* ψ *iff* $y'' \geq y' + \omega(p_i,x)$
(c_6) $\forall \mathbf{t} = (n,d,x,p,y,s,r) \in \psi$ *it holds that* $y + \omega(p,x) \in TS$
(c_7) $\forall \mathbf{t} = (n,d,x,p,y,s,r) \in \psi$

- *if* $p = p_1$ *it holds that* $s = \varepsilon$,
- *if* $p = p_2$ *or* $p = p_3$ *and* $\beta(x) = 1$ *it holds that* $s = c$, *with* $c \in C$,
- *if* $p = p_2$ *or* $p = p_3$ *and* $\beta(x) = 0$ *it holds that* $s = t$, *with* $t \in T$,
- *if* $p = p_4$ *it holds that* $s = t$, *with* $t \in T$.

The specified conditions are necessary to enforce the following constraints: (c_1) each resource (chair or tomograph) can be used by at most one patient at a time; (c_2) at most two patients at a time are allowed during the anamnesis phase; (c_3) patients requiring an injection chair must be assigned to the tomograph of the same room; (c_4) the number of protocols executed on a single tomograph is limited; (c_5) given two consecutive phases p_i and p_{i+1} for a patient, the initial time slot of p_{i+1} must be consistent with respect to p_i, i.e. p_{i+1} must start after that p_i has terminated; (c_6) each schedule must not exceed the available time slot; (c_7) the resources must be well distributed, i.e. the phase anamnesis does not include any resource, the phases 2 and 3 may require a chair or a tomograph depending on the protocol, and the last phase requires the usage of a tomograph.

As previously explained, an optimal solution aims to maximize the number of scheduled patients on the considered days while minimizing the idle time spent in the clinic by patients. In order to define the notion of the optimal solution, we want to evaluate the idle time spent by a patient. Accordingly, given (n,d,x,p_1,y',s',r) and $(n,d,x,p_4,y'',s'',r) \in \psi$ let $Htime(n) = (y'' + \omega(p_4,x)) - y')$ be the time spent by the patient in the hospital deriving from the scheduling and let $Rtime(n) = \sum_{p \in P} \omega(p,x)$ be the minimum time required to execute the protocol.

Definition 5 (Dominating Solution). *Let* $\delta_\psi = \sum_{n \in N} |Htime(n) - Rtime(n)|$ *be the sum of the differences between the actual time required by a protocol and the time allocated by the solution for that protocol for each patient n, and let* $\Sigma_\psi = \{(n,d,x,p,y,s,r) \in \psi\}$. *A solution* ψ *dominates a solution* ψ' *if*

- $|\Sigma_{\psi'}| < |\Sigma_\psi|$, *or if*
- $|\Sigma_{\psi'}| = |\Sigma_\psi| \Rightarrow \delta_\psi < \delta_{\psi'}$.

Finally, we define the notion of *maximal scheduling solution*.

Definition 6 (Maximal Scheduling Solution). *A scheduling solution is maximal if any other scheduling solution does not dominate it.*

4. Background on ASP

Answer Set Programming (ASP) [2] is a programming paradigm developed in the field of non-monotonic reasoning and logic programming. In this section, we 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 [2, 1]. 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 an uppercase letter, and constants are non-negative integers or strings starting with lowercase letters. A *term* is either a variable or a constant. A *standard atom* is an expression $p(t_1,\ldots,t_n)$, where p is a *predicate* of arity n and $t_1,\ldots,t_n$ are terms. An atom $p(t_1,\ldots,t_n)$ is ground if $t_1,\ldots,t_n$ are constants. A *ground set* is a set of pairs of the form $\langle consts\!:\!conj\rangle$, where *consts* is a list of constants and *conj* is a conjunction of ground standard atoms. A *symbolic set* is a set specified syntactically as $\{Terms_1 : Conj_1;\cdots;Terms_t : Conj_t\}$, where $t > 0$, and for all $i \in [1,t]$, each $Terms_i$ is a list of terms such that $|Terms_i| = k > 0$, and each $Conj_i$ is a conjunction of standard atoms. A *set term* is either a symbolic set or a ground set. Intuitively, a set term $\{X\!:\!a(X,c),p(X);Y\!:\!b(Y,m)\}$ stands for the union of two sets: the first one contains the X-values making the conjunction $a(X,c),p(X)$ true, and the second one contains the Y-values making the conjunction $b(Y,m)$ true. An *aggregate function* is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Basically, aggregate functions map multisets of constants to a constant, e.g., the function *#count* computes the number of terms.

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 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:

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

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 $a_1 \mid \ldots \mid a_n$ 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* [16] 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 the weight is the “cost” of violating the condition in the body of w, whereas levels can be specified for defining a priority among preference criteria). An ASP program with weak constraints is $\Pi = \langle P, W \rangle$, where P is a program and W is a set of weak constraints.

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

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

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

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

A model is an interpretation that satisfies all rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [17] 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) [17].

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 $p \mid p'$, where p' is a fresh new atom not appearing elsewhere in the program, meaning that the atom p can be chosen as true.

5. ASP Encoding

In this section, we present the ASP encoding for the problem presented in Section 2 and formalized in Section 3. The underlying encoding is based on the input language of CLINGO [4].

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

- instances of `reg(ID,D,PrID)` represent a registration with identification number `ID`, on day `D`, for a specific exam with protocol number `PrID`;
- instances of `avail(TS,D)` denote that the time slots `TS` is available on day `D`;
- instances of `exam(PrID,P,NumTS)` denote the features of a exam, where `PrID` denotes the exam protocol number, `P` indicates the phase, and `NumTS` specifies the time required for that phase in terms of the number of time slots;

- instances of `tomograph(T,R)` and `chair(C,R)` denote the allocation of the tomograph `T` and the chair `C` to the room `R`, respectively;
- instances of `required_chair(PrID)` denote the necessity of a chair for the phases 1 and 2 for the protocol `PrID`;
- instances of `cost(PrID, NumTS)` represent the total duration in terms of time slots, denoted as `NumTS`, of the phases within the protocol identified by `PrID`;
- instances of `limit(PrID,N)` denote the maximum number `N` of exams with protocol number `PrID` that can be executed on a fixed tomograph in a day.

The output consists of assignments represented by the atom `x(ID,D,TS,PrID,P)` where the intuitive meaning is that the registration with identification number `ID` for the exam with protocol number `PrID`, regarding the phase `P`, has been scheduled for the day `D` during the time slot `TS`. Additionally, it includes atoms `chair(C,ID,D)` and `tomograph(T,ID,D)`, denoting the

```
0 {x(ID, D, TS, PrID, 0) : avail(TS, D)} 1 :- reg(ID, D, PrID).
{x(ID, D, START, PrID, P+1) : avail(START,D), START >= TS+NumTS, START <
    TS+NumTS+6} = 1 :- x(ID, D, TS, PrID, P), exam(PrID, P, NumTS), P >= 0,
    P < 3.
:- x(ID, _, TS, PrID, 3), exam(PrID, 3, NumTS), TS + NumTS > 120.
timeAnamnesis(ID, TS..TS+NumTS-1) :- x(ID, D, TS, PrID, 0), exam(PrID, 0,
    NumTS).
:- #count{ID: timeAnamnesis(ID, TS)} > 2, avail(TS,D).
timeOccupation(ID, D, TS, END-1, PrID) :- x(ID, D, TS, PrID, 1), x(ID, D,
    END, PrID, 3).
res(ID, D, TS..END,0) :- timeOccupation(ID, D, TS, END, PrID),
    required_chair(PrID).
res(ID, D, TS..TS+NumTS-1,1) :- x(ID, D, TS, PrID, 3), exam(PrID, 3,
    NumTS), required_chair(PrID).
res(ID, D, TS..END+NumTS-1,1) :- timeOccupation(ID, D, TS, END, PrID),
    exam(PrID, 3, NumTS), not required_chair(PrID).
:- #count{ID: tomograph(T, ID, D), x(ID, D, _, PrID, _)} > N, limit(PrID,
    N), tomograph(T,_).
1 {chair(C, ID, D) : chair(C, _)} 1 :- x(ID, D, _, PrID, _),
    required_chair(PrID).
1 {tomograph(T, ID, D) : tomograph(T, _)} 1 :- x(ID, D, _, PrID, _).
:- chair(C, ID, D), tomograph(T, ID, D), chair(C, R1), tomograph(T, R2), R1
    != R2.
chair(C, ID, D, TS) :- chair(C, ID, D), res(ID, D, TS, 0).
tomograph(T, ID, D, TS) :- tomograph(T, ID, D), res(ID, D, TS, 1).
:- #count{ID: tomograph(T, ID, D, TS)} > 1, tomograph(T,_), avail(TS,D).
:- #count{ID : chair(C, ID, D, TS)} > 1, chair(C,_), avail(TS,D).
:~ not x(ID, D, _, _,0), reg(ID, D, _). [1@2, ID, D]
:~ x(ID, _, START, PrID, 0), x(ID, _, END, _, 3), cost(PrID, NumTS), END -
    START - NumTS >= 0. [END - START - NumTS@1, ID]
```

Figure 1: ASP encoding of the problem.

resource (either the chair `C` or the tomograph `T`, respectively) allocated to the patient `ID` on the day `D`.

Encoding. The related encoding is shown in Figure 1 and is described next. To simplify the description, we denote as r_i the rule appearing at line i of Figure 1.
Rule r_1 may assign or not the registration with identifier `ID` to a specific time slot `TS` for the day `D` in phase 0, i.e. the phase of the anamnesis. Rule r_2 assigns an already scheduled session for a given phase `P` to the subsequent planned phases, under the condition that the start of the phase does not extend beyond the latest available time slot for a session on that day. Furthermore, it ensures that the subsequent phase starts at most 5 time slots after the ending of the previous phase. Rule r_3 ensures that the duration of the final phase is also consistent with the time slots, ensuring that all phases are completed within the specified limit. Rule r_4 keeps track of the time slots allocated to a patient during phase 0 via the auxiliary atom `timeAnamnesis(ID, TS)`. Rule r_5 restricts the number of patients during the anamnesis phase to a maximum of two. Rule r_6 produces the auxiliary atom `timeOccupation(ID,D,TS,END,PrID)`, representing the duration needed for each patient `ID` from the initial time slot `TS` of phase 1 to the final one `END` of phase 2, concerning the protocol `PrID` on the day `D`. Rule r_7 produces the auxiliary atom `res(ID, D, TS, 0)` for each time slot derived from the previous rule. Specifically, the constant `0` denotes that a chair is required for each of these time slots. Rules r_8 and r_9 produce the atom `res(ID,D,TS,1)`, which differs from the previous one for the constant `1`, indicating the use of a tomograph. From rule r_8, it is inferred that a tomograph is employed during phase 3, whereas rule r_9 indicates the tomograph's usage from phase 1 to 3, according to the atom `timeOccupation`. Rule r_{10} ensures that the limit of protocols that can be executed on a single tomograph is respected. Rule r_{11} produces the atom `chair(C,ID,D)` representing the assignment of a chair `C` on the day `D` to the patient `ID` when the protocol `PrID` requires a chair. Rule r_{12} produces the atom `tomograph(T,ID,D)` representing the assignment of a tomograph `T` on the day `D` to the patient `ID`. Rule r_{13} prevents the patient who moves from the chair to the tomograph from changing room. Rule r_{14} and r_{15} generate the atoms `chair(C,ID,D,TS)` and `tomograph(T,ID,D,TS)` respectively, indicating the time slots `TS` during which the chair `C` and tomograph `T` are utilized by the patients `ID` on the day `D`. Rule r_{16} and r_{17} ensure that at most one patient is assigned to each tomograph and chair in every time slot, respectively. Finally, the optimal solution is achieved through the application of rule r_{18}, which minimizes (with the highest priority) the number of registrations not assigned to a schedule, and rule r_{19}, which minimizes the duration of patient appointments beyond the time necessary to perform the test.

6. Experimental Results

In this section, we report the results of an empirical analysis of the NMS problem via ASP. The data used for the empirical analysis are real data coming from a medium size hospital provided by Medipass. We performed experiments on an Apple M1 CPU @ 3.22 GHz machine with 8 GB of physical RAM. The ASP system used was CLINGO [4] 5.6.2, using parameters *--restart-on-model* for faster optimization and *--parallel-mode 2* for parallel execution: we conducted a preliminary analysis with various options and found these parameters to be the most effective.

6.1. NMS benchmarks

We tested instances of more than a year of daily exams. In particular, we tested 366 instances, each corresponding to a weekday excluding weekends, resulting in a total of 72 weeks. The solution schedules the patients in a day in a range of 10 hours, split into 120 time slots of 5 minutes. Each patient is linked to one of the possible exams. In particular, one exam protocol is required by more than 85% of the patients, thus, the majority of the patients need a exam protocol that requires 2 time slots for the anamnesis and other 2 time slots for medical preparation, 10 time slots for the drug injection and the bio-distribution time and, at last, the image detection requires 7 time slots. The other patients can be associated with one of the other 11 possible protocols. The schedule is done having as available resources two rooms for the radiotherapy. In each room, there is a tomograph and 3 chairs. The number of patients requiring exam changes every day but, on average, there are 29 patients to be scheduled, with a maximum of 37 patients.

6.2. Results

The results obtained by testing the encoding presented in Section 5 with the real data are reported in the following.

We decided to schedule the patients with a time limit of 60 seconds. This allows the hospital to get a result in a very short time and to get an estimation of the usefulness of our solution in a time-constrained environment. Figure 2 presents the average, the maximum, and the minimum times required to obtain a solution according to the number of patients to be scheduled. In this way, it is possible to analyze the waiting time a hospital should expect to get a solution, depending on the number of patients to be assigned. From the graph it can be seen that up to 29 patients all the instances are optimally solved before the time limit. While on average the instances are solved optimally in less than 60 seconds in the instances with less than 36 patients. We were able to test the encoding in just a few days having 36 and 37 patients and, in that case, the encoding is not able to optimally schedule the patients. In general, we are able to find the optimal solution for more than 60% of the instances.

After analyzing the time required to generate schedules on different days, we proceed to evaluate the quality of the results obtained. Specifically, we first examine the number of registrations that the solution fails to assign during the day, as this was the primary optimization criterion. Subsequently, we assess the second optimization criterion.

To gain a better understanding of the schedule's quality, we present the results considering all schedules and we specifically focus on non-optimal solutions. Even when the solution is optimal, some days still have a high number of unassigned patients due to constraints imposed by the data.

When considering instances where the encoding failed to find an optimal solution, we observed that in the majority of cases (more than 80%), the solution assigns the exam to over 80% of the possible patients. In the remaining instances, the solution still manages to assign over 70% of the total patients in all cases. These findings indicate that even under strict time limits, the results remain of satisfactory quality, a crucial aspect for operational scenarios.

Next, we summarize the results obtained across all instances, whether optimally or non-optimally solved. Figure 3 illustrates the number of schedules with at least a certain percentage of assigned patients. Unlike the non-optimal solutions, optimally solved instances occasionally

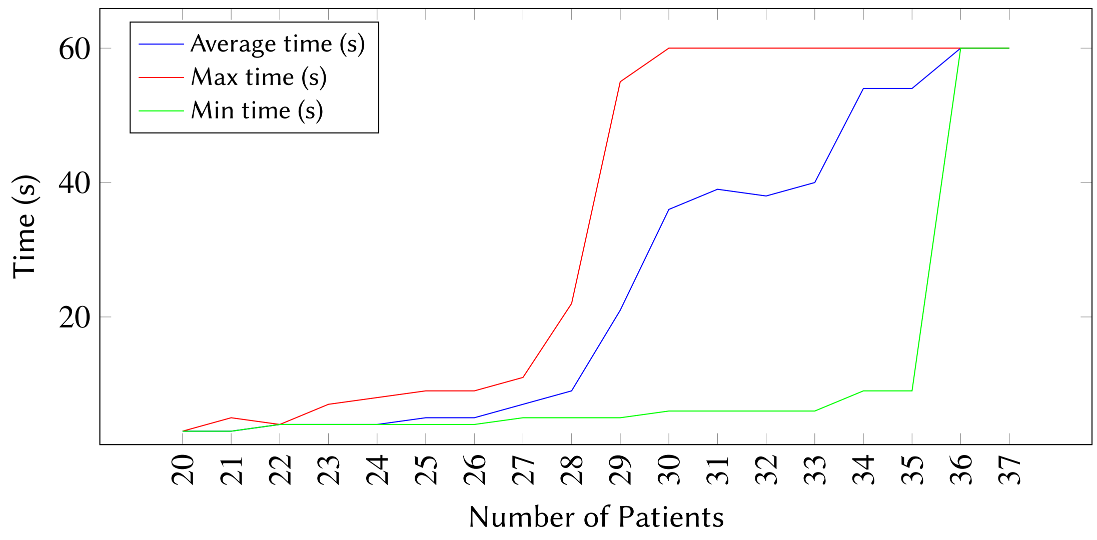

Figure 2: Average time (seconds) required to solve the instances with the different number of patients.

exhibit a percentage of assigned patients below 60%, attributed to resource limitations, which we further investigate. However, overall, the majority of solutions have a patient assignment rate exceeding 80%.

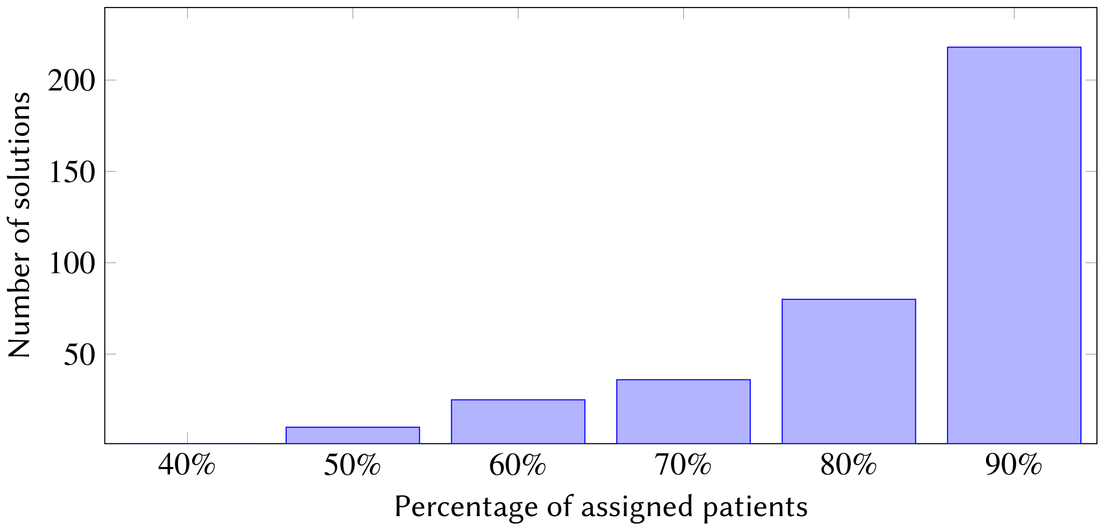

Figure 3: Number of solutions with a percentage of assigned patients at least equal to the corresponding x-axis value and smaller than the next value.

As previously discussed, some optimal solutions exhibit a low percentage of assigned patients. Here, we aim to analyze in more details this scenario to ensure that no better solution could have been identified. Specifically, we focus on the only instance where the schedule fails to assign an exam to more than 50% of the patients. In this case, the schedule assigns exams to

16 out of the total 33 patients. Upon closer examination, we find that among the 33 patients, 14 require protocol with id 823, while the remaining 19 require protocol with id 815. The solution successfully assigns exams to all 14 patients with protocol id 823. However, due to the nature of the protocol with id 815, the solution can only assign one patient with this protocol to each tomograph. Consequently, it manages to assign exams to just 2 patients. This limitation explains the low percentage of assigned patients in this instance.

Finally, after examining the results related to the first optimization criterion, we proceed to analyze the second one. Specifically, we focus on the average waiting time for each patient, quantified as the unnecessary number of time slots spent in the hospital. In over 70% of instances, the encoding successfully assigns patients with zero waiting times. This indicates that in these solutions, patients can adhere to their protocols optimally. These results greatly improve those obtained by the hospital in terms of unnecessary time spent in the hospital by the patients. Indeed, in the original schedule of the hospital, just 19% of the considered days had zero waiting times for each patient. Moreover, while in the ASP-based solution we have 95% of days with less than 5 time slots of waiting time, in the original schedule there were just 75% of days with this average waiting time.

7. Related Work

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

Solving the NMS problem. Pérez et al. [18] proposed four different scheduling solutions to the NMS, each giving priority to a different aspect. Specifically, the first solution focuses on the preferences of the patients. The second one assigns the patients as soon as possible. The third one is a combination of the first two, while the fourth one fixes the technologists to the machines and assigns the patients based on the machine availability. The proposed solutions were tested on the data of one of the biggest hospitals in Texas and tested in a simulated environment. Another work using real data to validate the solution is from Xiao et al. [19]. The authors got the data from the West China Hospital and proposed a two-step solution. The first step is the development of a nonlinear integer programming model considering the settings of the hospital and the drugs. After obtaining the solution of the first step, they developed a stochastic online algorithm following different scheduling strategies to adapt the solution to the real requests of patients. Akhavizadegan et al. [20] addressed the NMS problem using a Markov decision process (MDP) to decide how many patients to schedule in a day from a tactical perspective and which patient can move on to the next phase from an operational perspective. The MDP is then solved using two heuristic algorithms and a mathematical programming model. This system is evaluated against historical data of patients scheduled following a First-Come-First-Served strategy. The historical data comes from the public Hospital in Tehran-Iran.

Solving scheduling problems with ASP. ASP has been successfully used for solving hard combinatorial and application scheduling problems in several research areas. In the Healthcare domain (see, e.g., [21] for a recent survey), the first solved problems were the *Nurse Scheduling Problem* [22, 23, 24], where the goal is to create a scheduling for nurses working in hospital units, and the *Operating Room Scheduling* [25, 26], whose goal is to assign operating rooms to patients. More recent problems include the *Chemotherapy Treatment Scheduling* problem [11], in which patients are assigned a chair or a bed for their treatments, the *Rehabilitation Scheduling Problem* [10], which assigns patients to operators in rehabilitation sessions. In [13] and [27] the problems are split into two subproblems and solved using ASP. The former deals with the problem of assigning a date to visit or therapy for multiple recurrent exams to chronic patients, the latter, schedule patients for the Pre-Operative Assessment Clinic problem. Concerning scheduling problems beyond the healthcare domain, ASP encodings were proposed in different contexts. Balduccini [28] used ASP for the *Incremental Scheduling Problem*, where the goal is to assign jobs to devices such that their executions do not overlap one another. Abels et al. [29] proposed a hybrid approach, using both pure ASP and difference constraints, to solve the problem of routing, scheduling, and optimizing real-world training scheduling. Gebser et al. [14] used ASP for routing driverless transport vehicles in the context of car assembly at Mercedes-Benz Ludwigsfelde GmbH. El-Kholany et al. [30] proposed a solution to the Job-shop Scheduling Problem (JSP), where tasks sharing a machine are to be scheduled in a sequence in order to complete jobs as early as possible. To solve it they decomposed the problem into time windows whose operations can be successively scheduled and optimized by means of multi-shot ASP solving. Finally, we refer the reader to the survey paper by Falkner et al. [15], where industrial applications dealt with ASP are presented, including those involving scheduling problems.

8. Conclusion

In this paper, we have presented an analysis of the NMS problem, modeled and solved with ASP. We started from a mathematical formulation of the problem, whose specifications come from a real scenario, and then presented our ASP solution. Results on real data show satisfying outcomes in terms of quality in a relatively short time.

Future works include improving the scalability of the encoding and providing the solution of the NMS rescheduling problem, which comes into play when the computed schedule can not be implemented due to some unavailability, and the implementation of a web application for easy usage of our solution. Additionally, exploiting the modularity of ASP, we plan to extend the program taking into account the fairness of the scheduling, ensuring that all the protocols are assigned equally, and a level of priority among patients.

References

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

[2] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Communications of the ACM 54 (2011) 92–103.

[3] M. Alviano, C. Dodaro, Anytime answer set optimization via unsatisfiable core shrinking, Theory and Practice of Logic Programming 16 (2016) 533–551.

[4] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: ICLP (Technical Communications), volume 52 of *OASICS*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 2:1–2:15.

[5] M. Maratea, L. Pulina, F. Ricca, A multi-engine approach to answer-set programming, Theory and Practice of Logic Programming 14 (2014) 841–868.

[6] M. Gebser, N. Leone, M. Maratea, S. Perri, F. Ricca, T. Schaub, Evaluation techniques and systems for answer set programming: a survey, in: J. Lang (Ed.), IJCAI, ijcai.org, 2018, pp. 5450–5456.

[7] M. Alviano, G. Amendola, C. Dodaro, N. Leone, M. Maratea, F. Ricca, Evaluation of disjunctive programs in WASP, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), LPNMR, volume 11481 of *LNCS*, Springer, 2019, pp. 241–255.

[8] P. Bruno, F. Calimeri, C. Marte, M. Manna, Combining deep learning and asp-based models for the semantic segmentation of medical images, in: S. Moschoyiannis, R. Peñaloza, J. Vanthienen, A. Soylu, D. Roman (Eds.), Proceedings of the 5th International Joint Conference on Rules and Reasoning (RuleML+RR 2021), volume 12851 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 95–110.

[9] P. Bruno, F. Calimeri, C. Marte, Dedudeep: An extensible framework for combining deep learning and asp-based models, in: G. Gottlob, D. Inclezan, M. Maratea (Eds.), Proceedings of the 16th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2022), volume 13416 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 505–510.

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

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

[12] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Magazine 37 (2016) 53–68.

[13] P. Cappanera, M. Gavanelli, M. Nonato, M. Roma, Logic-based Benders decomposition in answer set programming for chronic outpatients scheduling, Theory and Practice of Logic Programming 23 (2023) 848–864. URL: https://doi.org/10.1017/s147106842300025x. doi:`10.1017/S147106842300025X`.

[14] M. Gebser, P. Obermeier, T. Schaub, M. Ratsch-Heitmann, M. Runge, Routing driverless transport vehicles in car assembly with answer set programming, Theory and Practice of Logic Programming 18 (2018) 520–534. URL: https://doi.org/10.1017/S1471068418000182. doi:`10.1017/S1471068418000182`.

[15] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, Künstliche Intelligenz 32 (2018) 165–176.

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

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

[18] E. Pérez, L. Ntaimo, W. E. Wilhelm, C. Bailey, P. McCormack, Patient and resource scheduling of multi-step medical procedures in nuclear medicine, IIE Transactions on Healthcare Systems Engineering 1 (2011) 168–184. URL: https://doi.org/10.1080/19488300.2011.617718. doi:10.1080/19488300.2011.617718.

[19] Q. Xiao, L. Luo, S. Zhao, X. bin Ran, Y. bing Feng, Online appointment scheduling for a nuclear medicine department in a chinese hospital, Computational and Mathematical Methods in Medicine 2018 (2018). URL: https://api.semanticscholar.org/CorpusID:5061754.

[20] F. Akhavizadegan, J. Ansarifar, F. Jolai, A novel approach to determine a tactical and operational decision for dynamic appointment scheduling at nuclear medical center, Computers & Operations Research 78 (2017) 267–277.

[21] M. Alviano, R. Bertolucci, M. Cardellini, C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, M. Mochi, V. Morozan, I. Porro, M. Schouten, Answer set programming in healthcare: Extended overview, in: IPS and RCRA 2020, volume 2745 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2745/paper7.pdf.

[22] M. Alviano, C. Dodaro, M. Maratea, An advanced answer set programming encoding for nurse scheduling, in: AI*IA, volume 10640 of *LNCS*, Springer, 2017, pp. 468–482.

[23] C. Dodaro, M. Maratea, Nurse scheduling via answer set programming, in: LPNMR, volume 10377 of *LNCS*, Springer, 2017, pp. 301–307.

[24] M. Alviano, C. Dodaro, M. Maratea, Nurse (re)scheduling via answer set programming, Intelligenza Artificiale 12 (2018) 109–124.

[25] C. Dodaro, G. Galatà, M. Maratea, I. Porro, Operating room scheduling via answer set programming, in: AI*IA, volume 11298 of *LNCS*, Springer, 2018, pp. 445–459.

[26] C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, I. Porro, An ASP-based solution for operating room scheduling with beds management, in: P. Fodor, M. Montali, D. Calvanese, D. Roman (Eds.), Proceedings of the Third International Joint Conference on Rules and Reasoning (RuleML+RR 2019), volume 11784 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 67–81.

[27] S. Caruso, G. Galatà, M. Maratea, M. Mochi, I. Porro, Scheduling pre-operative assessment clinic with answer set programming, Journal of Logic and Computation 34 (2023) 465–493. URL: https://doi.org/10.1093/logcom/exad017. doi:10.1093/logcom/exad017.

[28] M. Balduccini, Industrial-size scheduling with ASP+CP, in: J. P. Delgrande, W. Faber (Eds.), 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*, Springer, 2011, pp. 284–296.

[29] D. Abels, J. Jordi, M. Ostrowski, T. Schaub, A. Toletti, P. Wanko, Train scheduling with hybrid asp, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), Logic Programming and Nonmonotonic Reasoning, Springer International Publishing, Cham, 2019, pp. 3–17.

[30] M. M. S. El-Kholany, M. Gebser, K. Schekotihin, Problem decomposition and multi-shot asp solving for job-shop scheduling, 2022. arXiv:2205.07537.