ASP-based Approaches for Solving the Nuclear Medicine Scheduling problem

Carmine Dodaro[1] [0000−0002−5617−5286], Giuseppe Galatà[2] [0000−0002−1948−4469], Marco Maratea[1] [0000−0002−9034−2527], Cinzia Marte[1] [0000−0003−3920−8186], and Marco Mochi[3] [0000−0002−5849−3667]

[1] DeMaCS, University of Calabria, Rende, Italy, name.surname@unical.it
[2] SurgiQ srl, Genova, Italy, giuseppe.galata@surgiq.com
[3] University of Genoa, Genova, Italy, marco.mochi@edu.unige.it

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 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 two solutions to the NMS problem based on Answer Set Programming (ASP). The first solution is a direct ASP encoding, then processed by an ASP solver, while the second solution employs a Logic-based Bender Decomposition (LBBD) approach implemented through the usage of multi-shot solving. Experiments employing real data show that the direct encoding provides overall satisfying results in terms of solutions quality in a relatively short time, and that the LBBD approach also helps improving scalability.[4]

Keywords: Answer Set Programming · Logic Programming · Digital Health

1 Introduction

Nuclear Medicine [2,46,52] 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 million nuclear medicine exams have been carried on during 2022 in Italy[5]. The process of treating patients with this technique is complex since it involves multiple hospital resources and requires multiple steps at varying times. 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. In fact, 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 patients in the hospital is vital for increasing the satisfaction of the patients.

The Nuclear Medicine Scheduling (NMS) problem consists of assigning patients to a day, on which the patient will undergo the medical check, the preparation, and the actual image detection process. The schedule of the patients considers the different requirements of the patients and the available resources, e.g.,

[4] This is a revised and extended version of a paper appearing in the CEUR proceedings of the CILC 2024 workshop [24].

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

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[6], leading provider of technological innovation across cancer care and diagnostic imaging in Italy, in collaboration with SurgiQ[7], 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) [13]. The success of ASP is due to different factors, including a simple but rich syntax [15], which includes optimization statements as well as powerful database-inspired constructs like aggregates, an intuitive semantics [35], and the availability of efficient solvers (see, e.g., [31,33,5]). Moreover, ASP has already been successfully employed to solve scheduling applications in a variety of application fields (see, e.g., [16,18,27,29,34]).

In this paper, we present two solutions to the NMS problem based on ASP. The first solution, introduced in [24], is a direct ASP encoding of the problem specifications, then processed by the state-of-the-art system CLINGO [31], and tested on real data provided by Medipass. This approach provides overall satisfying results in terms of solutions quality in a relatively short time, but fails to return satisfying solutions on large inputs. The second solution, instead, employs a Logic-based Bender Decomposition (LBBD) approach taken from the field of Operation Research and already successfully employed in ASP-based scheduling solutions in Healthcare [16]. It consists of defining a master problem and several subproblems that interact with it by passing constraints that "cut" the search space and help converging to an (optimal) solution. The LBDD approach is implemented through the usage of the multi-shot solving variant of CLINGO [32], and helps improving scalability compared to the direct approach. Then, given that the solutions provided may have fairness issues related to equitable utilization of resources and outcomes across a diverse range of patients, we extended (in a modular way) our solutions by adding fairness constraints, thus resulting in more fair solutions with a very low impact on the performance.

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 and LBBD in Section 4, Sections 5 and 6 show the two encodings, whose experimental evaluations are presented in Section 7. The modeling and impact of fairness constraints are, instead, considered in Section 8. The paper ends by discussing related works and conclusions in Section 9 and 10, respectively.

2 Problem Description

The NMS problem consists of assigning patients to a day and a tomograph and/or an 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 consider 11 different protocols. Each protocol can encompass up to four phases, i.e. (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, and if the protocol requires the infusion chair for phase (p_3).

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

[6] `https://ergeagroup.com/it/`
[7] `https://surgiq.com/`

Protocol Number	#TS for p_1	#TS for p_2	#TS for p_3	#TS for p_4	#TS total	Chair
813	3	2	0	8	13	NO
814	3	2	0	8	13	NO
815	2	2	4	6	14	YES
817	2	2	3	7	14	NO
819	2	2	5	7	16	YES
822	2	2	2	7	13	NO
823	2	2	10	7	21	YES
824	2	2	5	8	17	YES
827	2	2	2	7	13	NO
828	3	3	0	7	13	NO
888	2	2	2	9	15	YES

Table 1: Specifications for each protocol, including the number of time slots (TS) needed for each phase, the total time slots for the entire protocol, and if a chair is required (YES) or not (NO).

because of the high cost of the drugs and machines involved in the exams, 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, are 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 consider 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 on 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 anamnesis phase;
- the injection phase must be done in an injection chair or on a tomograph according to the required protocol;
- the image detection phase must be done in a tomograph for all the considered protocols;
- 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 in the same room;
- the protocol identified by the id 815 cannot be assigned on the same day and tomograph for more than one patient.

The solution should also 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, and TS denote the set of time slots. 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 a resource is not required. Let PR be the set collecting the protocol numbers referred to exams and $\widetilde{PR}$ a subset of it containing only the protocols with a limit on the number of exams that can be executed on a tomograph for that protocol. Specifically, each protocol may comprise up to four phases, represented by the set $P = \{p_1, p_2, p_3, p_4\}$.

Moreover, let:

- $\lambda : T \times \widetilde{PR} \to \mathbb{N}$ 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 a chair for the injection phase; 0 if it requires a tomograph;
- $\alpha : S \times R \to \{0,1\}$ 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 existing associations between resources and rooms.

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 $\sigma : \mathsf{T} \times \mathsf{A} \to \{0,1\}$ such that $\sigma((n,d,x,p,y),(s,r)) = 1$ 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 $\mathsf{S} = \sigma^{-1}(1) = \{((n,d,x,p,y),(s,r)) \in \mathsf{T} \times \mathsf{A} \mid \sigma((n,d,x,p,y),(s,r)) = 1\}$ collects all the tuples eligible as scheduling.*

We can now define the NMS problem.

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

(c_1) $\forall d \in D,\ \forall y \in TS,\ \forall (s,r) \in \mathsf{A}\ |\{(n,d,x,p,y,s,r) \in \psi : p \neq p_1\}| \leq 1$;
(c_2) $\forall d \in D,\ \forall y \in TS,\ \forall (s,r) \in \mathsf{A}\ |\{(n,d,x,p,y,s,r) \in \psi : p = p_1\}| \leq 2$;
(c_3) $\forall x \in PR : \beta(x) = 1$, *it holds that* (n,d,x,p_3,y',c,r') *and* (n,d,x,p_4,y'',t,r''), *with* $y' \neq y''$, *belong to* ψ *iff* $r' = r''$;
(c_4) $\forall d \in D,\ \forall x \in \widetilde{PR}, \forall (s,r) \in \mathsf{A}\ |\{(n,d,x,p,y,s,r) \in \psi : s = t\}| \leq \lambda(t,x)$;
(c_5) (n,d,x,p_i,y',s',r) *and* $(n,d,x,p_{i+1},y'',s'',r)$ *belong to* ψ *iff* $y'' \geq y' + \omega(p_i,x)$;
(c_6) $\forall (n,d,x,p,y,s,r) \in \psi$ *it holds that* $y + \omega(p,x) \in TS$;
(c_7) $\forall (n,d,x,p,y,s,r) \in \psi$
- *if* $p = p_1$ *it holds that* $s = \varepsilon$;
- *if* $p \in \{p_2,p_3\}$ *and* $\beta(x) = 1$ *it holds that* $s \in C$;
- *if* $p \in \{p_2,p_3\}$ *and* $\beta(x) = 0$ *it holds that* $s \in T$;
- *if* $p = p_4$ *it holds that* $s \in T$.

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 slot 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 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 time allocated by the solution for that protocol and the actual time required by a protocol, for each registration number n. Additionally, let $\Sigma_\psi = \{(n,d,x,p,y,s,r) \in \psi\}$ represent the set of all elements in a solution ψ. 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

In this section, we introduce the necessary preliminaries on ASP and the LBBD-based approach, organized into two separate subsections.

4.1 Background on ASP

Answer Set Programming (ASP) [13] is a programming paradigm developed in the field of non-monotonic reasoning and logic programming. More detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in [13,15]. Hereafter, we assume the reader is familiar with logic programming conventions.

Syntax. The syntax of ASP is similar to that 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 the 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* [14] ω is of the form:

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

where w and l are the weight and level of ω, respectively. (Intuitively, $[w@l]$ is read as "weight w at level l"). 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 the rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [28] 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) [28].

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 by levels.

Algorithm 1: Algorithm to solve a problem P with the LBBD methodology

Input: A problem P, a set of constraints $C(x,y)$, $C_x(x)$, and $C_y(y)$
Output: A solution to P

split P in P_M and $\mathscr{P}_S = \{P_S^1, \ldots, P_S^n\}$;
while *True* **do**
  $x^* = solve(P_M, C_x(x))$;
  if P_M *is* UNSAT **then**
    return UNSAT
  $Y = \emptyset$;
  for each $P_S^i \in \mathscr{P}_S$ **do**
    $y^* = solve(P_S^i(x^*), C(x^*,y), C_y(y))$; `//` $P_S^i(x^*)$ `denotes that` P_S^i `keeps the input values` x^*
    if P_S^i *is* UNSAT **then**
      $C_x(x) = C_x(x) \cup C^*$
    else
      $Y = Y \cup \{y^*\}$;
  if *all* P_S^i *is* SAT **then**
    return (x^*, Y)

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.

4.2 Background on LBBD

Logic-Based Benders Decomposition (LBBD) [40,42] is an optimization technique that extends the classical Benders Decomposition (BD) approach [12,36,47], offering greater flexibility and applicability to a wider range of complex problems.

In LBBD the problem is divided into a *master problem* (MP) and one or more *subproblems* (SPs), where the MP typically involves high-level decisions, while the SPs address more detailed, often logically complex, constraints. The core idea of LBBD is to iteratively solve the MP and SPs in a coordinated manner. After solving the MP, the resulting solution is passed to the first SP. If the SP finds this solution infeasible or suboptimal, it generates a *Benders cut*—usually in the form of a constraint based on the logical structure of the SP—which is then added to the MP to eliminate the current solution and guide the search towards better or feasible solutions. This process continues until an optimal or satisfactory solution is found.

The key difference between classical BD and LBBD lies in the nature of the SPs, in the method used to generate Benders cuts, and in the properties that the solutions have. In classical BD, the SP is typically a continuous linear or nonlinear programming problem. This allows for a standard, well-established approach to deriving Benders cuts, which are used to iteratively refine the MP. In LBBD, the SP can be any arbitrary optimization problem, often involving logical constraints. Because of this complexity, there is no universal method to generate Benders cuts and a specialized approach must be developed for each specific problem class. This is done by solving what is known as the "inference dual" of the SP, that is a duality of search and inference that captures the logical and combinatorial structure of the SP. The inference duality provides sensitivity analysis and nogood-based search that encapsulates the essential constraints and logical relationships of the original SP. These cuts are then added to the MP, guiding the overall solution process. About

the optimality of the solutions found, while BD guarantees the optimality, this is not the case for LBBD, and this property depends on the specific cuts employed for the problem at hand.

We outline the key steps of the LBBD approach in Algorithm 1. Consider a problem whose objective is to maximize a function $f(x,y)$, subject to a set of constraints $C(x,y)$, $C_x(x)$, and $C_y(y)$, with x and y vectors of variables that belong to the respective domains D_x and D_y. The first step of the LBBD approach is to split P into a MP P_M and a set of SPs $\mathscr{P}_S = \{P_S^1, \ldots, P_S^n\}$ (line 1). Assume to solve P_M and to find an optimal solution x^* (line 3) such that, fixed $y \in D_y$, maximizes the function $f(x,y)$ under the constraints $C_x(x)$. Instead, if P_M is unsatisfiable (line 4), the procedure stops (line 5); otherwise, the value x^* is passed to each SP in $\mathscr{P}_S$ (line 7) in order to solve it. Each $\mathscr{P}_S$ can (i) admit an optimal solution y^* (line 8), or (ii) be unsatisfiable (line 9). In case (i), the solution is stored in a set, initially empty (line 6), which will accumulate all the optimal solutions from the SPs (line 12). In case (ii), it is not possible to extend the optimal solution x^* to the SP P_S. Hence, it is necessary to add a new constraint C^*on x^* from P_S to P_M (line 10) and to repeat the algorithm (starting from line 2). The added constraint C^* is used to limit the MP with a new upper bound on the value of the objective function. The new upper bound value is derived from the last solution of the MP, which led to an unsatisfiable SP. This iterative process continues until either P_M is proven unsatisfiable (which would imply that the entire problem is unsatisfiable) or the optimal solution is found (lines 13 and 14).

5 Direct ASP Encoding

In this section, we first present a direct ASP encoding for the problem presented in Section 2 and formalized in Section 3, followed by a comparison with alternative logic-based formalisms. The encoding consists of three parts: a guess part, where the program assigns patients to time slots for the anamnesis phase, propagates scheduling across subsequent phases, and assigns chairs and tomographs to patients; a check part, where it verifies that all scheduling and resource constraints are satisfied; an optimization part, where it minimizes the number of unassigned registrations and reduces patient appointment durations beyond what is necessary.

5.1 Encoding

The underlying encoding is based on the input language of CLINGO [31]. The program manages the scheduling of patient protocols across the various phases required by the medical procedure. The guessing part involves first assigning patients to an initial anamnesis phase and then propagating their sessions through the subsequent phases. It also includes the assignment of necessary resources, such as chairs and tomographs, to each patient. The checking part verifies that all temporal constraints are satisfied, including start times and phase durations. It also ensures that each resource is allocated to at most one patient per time slot and enforces equipment is used within its limits. Finally, the optimization part aims to minimize the number of unassigned patient registrations and to reduce unnecessary appointment duration.

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`. These instances correspond to the set R as defined in Definition 1.
- instances of `avail(TS,D)` denote that the time slot `TS` is available on day `D`;
- instances of `exam(PrID,P,NumTS)` denote the features of an 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. These instances model the function ω in Section 3.

```
1  0 {x(ID, D, TS, PrID, 0) : avail(TS, D)} 1 :- reg(ID, D, PrID).
2  {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.
3  :- x(ID, _, TS, PrID, 3), exam(PrID, 3, NumTS), TS + NumTS > 120.
4  timeAnamnesis(ID, TS..TS+NumTS-1) :- x(ID, D, TS, PrID, 0), exam(PrID, 0, NumTS).
5  :- #count{ID: timeAnamnesis(ID, TS)} > 2, avail(TS,D).
6  timeOccupation(ID, D, TS, END-1, PrID) :- x(ID, D, TS, PrID, 1), x(ID, D, END, PrID, 3).
7  res(ID, D, TS..END,0) :- timeOccupation(ID, D, TS, END, PrID), required_chair(PrID).
8  res(ID, D, TS..TS+NumTS-1,1) :- x(ID, D, TS, PrID, 3), exam(PrID, 3, NumTS),
       required_chair(PrID).
9  res(ID, D, TS..END+NumTS-1,1) :- timeOccupation(ID, D, TS, END, PrID), exam(PrID, 3,
       NumTS), not required_chair(PrID).
10 :- #count{ID: tomograph(T, ID, D), x(ID, D, _, PrID, _)} > N, limit(PrID, N),
       tomograph(T,_).
11 1 {chair(C, ID, D) : chair(C, _)} 1 :- x(ID, D, _, PrID, _), required_chair(PrID).
12 1 {tomograph(T, ID, D) : tomograph(T, _)} 1 :- x(ID, D, _, PrID, _).
13 :- chair(C, ID, D), tomograph(T, ID, D), chair(C, R1), tomograph(T, R2), R1 != R2.
14 chair(C, ID, D, TS) :- chair(C, ID, D), res(ID, D, TS, 0).
15 tomograph(T, ID, D, TS) :- tomograph(T, ID, D), res(ID, D, TS, 1).
16 :- #count{ID: tomograph(T, ID, D, TS)} > 1, tomograph(T,_), avail(TS,D).
17 :- #count{ID : chair(C, ID, D, TS)} > 1, chair(C,_), avail(TS,D).
18 :~ not x(ID, D, _, _,0), reg(ID, D, _). [1@3, ID, D]
19 :~ x(ID, _, START, PrID, 0), x(ID, _, END, _, 3), cost(PrID, NumTS), END - START - NumTS
       >= 0. [END - START - NumTS@2, ID]
```

Fig. 1: ASP encoding of the problem.

- 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. These instances correspond to the set A in Section 3.
- instances of `required_chair(PrID)` denote the necessity of a chair for the phases 1 and 2 for the protocol `PrID`. These instances model the function β in Section 3.
- 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. These instances model the function λ in Section 3.

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 resource (either the chair `C` or the tomograph `T`, respectively) allocated to the registration with identification number `ID` on the day `D`.These atoms together model the set ψ as defined in Definition 4.

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 end 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, encoding condition c_6. 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, encoding condition c_2. 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 in phases 1 and 2, according to the atom `timeOccupation`. Rule r_{10} ensures that the limit of protocols that can be executed on a single tomograph is respected, encoding condition c_4. 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} behaves as rule r_{11}, just producing the atom `tomograph(T,ID,D)` instead of `chair(C,ID,D)`. Rule r_{13} prevents the patient who moves from the chair to the tomograph from changing room, encoding condition c_3. Rule r_{14} and r_{15} generate the atoms `chair(C,ID,D,TS)` and `tomograph(T,ID,D,TS)` respectively, indicating the time slot `TS` during which the chair `C` and tomograph `T` are utilized by the patients `ID` on the day `D`. Rules r_{16} and r_{17} ensure that at most one patient is assigned to each tomograph and chair in every time slot, respectively, encoding condition c_1. 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.

5.2 Comparison to Alternative Logic-based Formalisms

In the following, we present an empirical comparison of the direct solution presented in Subsection 5.1 to different alternative logic-based formalisms, obtained by applying automatic translations of ASP instances. With this analysis, we want to compare ASP performance to that of other, possibly commercial, solutions. In more detail, we used the ASP solver WASP [6], with the option `--pre=wbo`, which collapses priority levels into a single level and converts ground ASP instances into pseudo-Boolean instances in the wbo format [45]. Then, we used the tool PYPBLIB [10] to encode wbo instances as MaxSAT instances.

For the comparison, we considered two state-of-the-art MaxSAT solvers, namely MAXHS [50] and OPEN-WBO [43], and the industrial ILP tool for solving optimization problems GUROBI [38], which is able to process instances in wbo format. Concerning ASP, we used CLINGO with the option `--restart-on-model`, which was the solver and the option used in Section 7, but now run on ground instances, and the option `--opt-strategy=usc`.

We tested the different logics in 31 instances randomly selected from the whole instances used for the empirical analysis will be made in Section 7.

Results are presented in Table 2, where for each solver we report the number of instances for which the different solvers were able to find an optimal solution, a satisfiable solution, or were not able to find any solution. The results obtained show that none of the employed solvers is able to find an optimal solution in all the instances; however, CLINGO-ROM is able to find the highest number of optimal solutions, as CLINGO-OPT, but is the only solver that is able to find at least a satisfiable solution for all the instances. Indeed,

Table 2: Comparison of the direct ASP solution using CLINGO with the option `restart-on-model` and `opt-strategy=usc` (CLINGO-ROM and CLINGO-OPT in the table, respectively) and the alternative logic-based solutions GUROBI, on wbo instances, MAXHS and OPEN-WBO. For every logic there is the number of instances for which an optimal solution is found, a satisfiable solution is found, and a solution is not found, respectively.

Instance	CLINGO-ROM	CLINGO-OPT	GUROBI	MAXHS	OPEN-WBO
Optimal Solutions	17	17	0	15	8
Satisfiable Solutions	14	0	0	0	0
Unknown Solutions	0	14	31	16	23

while CLINGO-OPT and MAXHS are able to find a high number of optimal solutions, they are not able to find any less than optimal solutions. Concerning OPEN-WBO, it has the same problem as CLINGO-OPT and MAXHS to find satisfiable solutions and it can find less optimal solutions than the other solvers. To sum up, it is possible to state that CLINGO-ROM is the solver with the best performances overall, while, if someone is interested in just the optimal solutions, discarding the ability to find sub-optimal solutions, CLINGO-OPT and MAXHS have similar performances. Finally, GUROBI is not able to find a solution in any of the instances. This seems to be due to the size of the problems, since the solving process is stuck for the majority of the time in the reading of the input file.

6 LBBD Encoding

In this section, we introduce the encoding developed to model the LBBD approach for the NMS problem, detailing the proposed solving process.

In our settings, the NMS problem is split into a MP P_M and a number of SPs equal to the number of available rooms. Specifically, P_M addresses the assignment of patients to the rooms without considering the allocation of other resources or treatment starting times. Conversely, each SP focuses on assigning starting times and resources to the patients previously assigned to each room. At the beginning of the process, P_M is grounded and solved, determining the room assignments for each patient. This solution, representing x^* in Algorithm 1, is then parsed, grouping patients by their assigned rooms $i \in \{1, \ldots, |R|\}$, and used as input for the corresponding SPs $P_{S_1}, \ldots, P_{S_{|R|}}$. The SPs are solved sequentially in ascending order. If any SP has no feasible solution or fails to solve within a given time limit, a no-good cut based on the number of patients assigned to that room is generated. This step corresponds to the addition of C^* in Algorithm 1, which is used to refine P_M. In this case, the second iteration of Algorithm 1 solves a more constrained version of P_M, preventing a solution where the same number of patients is assigned to the room. Finally, if all SPs find a feasible solution, the process terminates, and the final solution is the union of the room-wise assignments.

We present the ASP encoding for the MP and the SPs in two separate subsections, respectively, where all encodings are still based on the input language of CLINGO [31].

6.1 ASP encoding for the MP

Data Model. The data model of the MP P_M is the same as presented in Section 5. The output consists of the assignments represented by atoms of the form:

- `x(ID,D,PrID,0)`, where the intuitive meaning is the same as the one presented in Section 5 but without assigning the starting time of the first phase represented by `TS`;

```
1  0 {x(ID, D, PrID, 0) : avail(_, D)} 1 :- reg(ID, D, PrID).
2  {room(ID, RoomID) : tomograph(_, RoomID)} = 1 :- x(ID, D, _, _).
3  timeOccupation(ID, L1+L2, PrID) :- reg(ID, _, PrID), exam(PrID, 1, L1), exam(PrID, 2, L2).
4  dur(ID, L, 0) :- timeOccupation(ID, L, PrID), required_chair(PrID).
5  dur(ID, L, 1) :- reg(ID, _, PrID), exam(PrID, 3, L), required_chair(PrID).
6  dur(ID, L0+L1,1) :- timeOccupation(ID, L0, PrID), exam(PrID, 3, L1),
                       not required_chair(PrID).
7  :- #sum{L, ID: dur(ID, L, 0), room(ID,RoomID)} > 360, room(_, RoomID).
8  :- #sum{L, ID: dur(ID, L, 1), room(ID,RoomID)} > 120, room(_, RoomID).
9  :- #count{ID: room(ID, RoomID), x(ID, _, X, _)} > N, limit(X, N), room(_,RoomID).
10 :~ not x(ID, D, _,0), reg(ID, D, _). [1@1, ID, D]
```

Fig. 2: ASP encoding for the MP.

– `room(ID,RoomID)`, where the intuitive meaning is that the registration with the identification number `ID` is assigned to the room identified by the id `RoomID`.

Encoding. The ASP-based formulation of the MP P_M is shown in Figure 2. We denote with r_i the rule appearing at line i of Figure 2. Rule r_1 may assign or not the registration with identifier `ID` to a possible day `D`. Rule r_2 assigns the patients previously assigned to a day to a room `RoomID`. Rule r_3 derives the time required by the patients to complete phases 1 and 2. Rules from r_4 to r_6 use the atom derived in rule r_3 to compute the total durations required in the chair and in the tomograph by the patients. Rules r_7 and r_8 ensure that the total sum of the time required by each registration assigned to a room is less than the total available time in the chairs and in the tomographs, respectively. Rule r_9 ensures that the registrations with a certain protocol `X`, identified through the atom `limit(X,N)` and assigned to a room, do not exceed the value `N`. Rule r_{10} minimizes the number of registrations that are not assigned.

6.2 ASP encoding for the SPs

Data Model. The input data is defined through the atoms `reg`, `exam`, and `tomograph`, introduced in Section 5, along with the following additional atoms:

– the constant `max_time` represents the number of time slots available;
– instances of `room(ID,RoomID)` and `x(ID,D,PrID,0)` output of the MP problem;
– instances of `delay(X)` represent the maximum unnecessary waiting time, fixed to 6 TS as for the direct encoding.

The output consists of the atoms `delay(ID,N)`, where the intuitive meaning is that to the registration with the identification number `ID` is assigned a waiting time of `N` time slots.

Encoding. The encoding shown in Figure 3 is used for all the SPs, differing only in their input. We refer to the rule appearing at line i of Figure 3 as r_i. Rule r_1 creates an auxiliary atom that derives all the patients assigned to the same room and the time they need to spend in the tomograph. Rules r_2 and r_3 decide an order between the patients sharing the same room. Rules from r_4 to r_7, starting from the assigned order, derive all the possible starting and ending times in which the patients stay in a tomograph. Rule r_8 derives the correct starting and ending times in which a patient stays in a tomography, while rule r_9 ensures that the ending

```
overlap(ID1, ID2, L1, L2) :- room(ID1, _), dur_tom(ID1,L1), x(ID2, _, _, _), room(ID2, _),
    dur_tom(ID2,L2), ID1 < ID2.
{order(ID1, ID2, L2)} :- overlap(ID1,ID2,_,L2).
order(ID2, ID1, L1) :- not order(ID1, ID2, _), overlap(ID1, ID2, L1, _).
start(ID, L+L0) :- room(ID, _), reg(ID, _, PrID), exam(PrID,0,L0), dur_chair(ID, L).
start(ID, L) :- room(ID, _), reg(ID, _, PrID), exam(PrID,0,L), not dur_chair(ID, _).
start(ID, L) :- order(ID2, ID, _), end(ID2, L).
end(ID, S+L) :- start(ID, S), dur_tom(ID, L), S < max_time.
last_end(ID,T) :- #max{Q : end(ID,Q)} = T, end(ID,_).
:- last_end(ID, T), T > max_time.
:- start(ID, S), S >= max_time.
:- start(ID, S), dur_tom(ID, L), S+L > max_time.
start_phase_0(ID, L0-1, E-LT-LC-L0) :- room(ID, _), last_end(ID, E), exam(PrID,0,L0),
    dur_chair(ID, LC), dur_tom(ID, LT).
start_phase_0(ID, L0-1, E-LT-L0) :- room(ID, _), last_end(ID, E), exam(PrID,0,L0),
                            not dur_chair(ID, _), dur_tom(ID, LT).
start_phase_1(ID, L-1, E+L0) :- room(ID, _), start_phase_0(ID, L0, E), dur_chair(ID, L).
res_chair(ID, T..T+L) :- start_phase_1(ID, L, T).
{chair(RoomID, ID, N) : chair(N, RoomID)} = 1:- room(ID, RoomID), reg(ID,_,PrID),
    required_chair(PrID).
overlap_chair(ID1, ID2) :- chair(RoomID, ID1, N), chair(RoomID, ID2, N), ID1 < ID2.
:- res_chair(ID1, T), res_chair(ID2, T), overlap_chair(ID1, ID2).
res_adm(ID, T..T+L-1) :- start_phase_0(ID, L, T).
:- #count{1,R: res_adm(R, T)} > 1, res_adm(R1, T), res_adm(R2, T), R1 != R2.
timeOccupation(ID, L1+L2, PrID) :- reg(ID, _, PrID), exam(PrID, 1, L1), exam(PrID, 2, L2).
{delay(ID,N) : N=0..X, delay(X)} = 1 :- room(ID,_).
dur_chair(ID, L+N) :- timeOccupation(ID, L, PrID), delay(ID,N), required_chair(PrID).
dur_tom(ID, L) :- reg(ID, _, PrID), exam(PrID, 3, L), required_chair(PrID).
dur_tom(ID, L0+L1+N) :- timeOccupation(ID, L0, PrID), exam(PrID, 3, L1), delay(ID,N),
                            not required_chair(PrID).
:~ delay(ID,N). [N@1,ID]
```

Fig. 3: ASP encoding for the SPs.

time occurs before the final time slot. Rules r_{10} and r_{11} ensure that each patient's starting time is before the final time slot and that the starting time, plus the required duration in the tomograph, does not exceed the final time slot, respectively. Rules r_{12} and r_{13} create auxiliary atoms representing the real starting time of phase 0. In particular, rules r_{12} and r_{13} derive the starting time of phase 0 of the registrations requiring and not requiring a chair for the injection phase, respectively. While rule r_{14} create auxiliary atoms representing the real starting time of the phase 1. Rule r_{15} derives all the time slots in which the patients are in phase 1. Rules r_{16} and r_{17} assign a chair to the patients requiring it and derive the patients sharing the same chair, respectively. Rule r_{18} ensures that the patients sharing the same chair do not require it concurrently. Rules r_{19} and r_{20} derive the time slots in which the patients are in phase 0 and ensure that there is no more than 1 patient in phase 0 in every time slot, respectively. Rule r_{21} derives the time required by the patients to complete phase 1 and phase 2. Rule r_{22} assigns a value to the delay of each patient. The value is in a range between 0 and X, where X is the maximum waiting time assignable to the patients. Rules from r_{23} to r_{25}

use the atom derived in rule r_{21} to derive the total duration required in the chair and the tomograph by the registration. Finally, rule r_{26} minimizes the unnecessary waiting times assigned to the patients.
In case any of the SP P_i is infeasible, a no-good cut of the following form is generated:

```
:- #sum{L, P: dur(P, L, 1), room(P, room)} >= n, bound(n, room, iter).
```

where the meaning is that in the iteration `iter` for the room `room` is added a constraint that limits the total sum of the durations of the treatments required in each room to a value `n`, identified by the new atom `bound` that is added as input to the MP. Differently from classical Bender's Decomposition, adding these rules at each step does not guarantee an optimal solution, as feasible solutions might be excluded. For example, two groups could require the same amount of time in the tomograph but differ in their need for other resources, like a chair. If the master problem assigns patients needing the chair, the subproblem might find this infeasible, generating a no-good cut that excludes all treatments exceeding a fixed duration. This also cuts feasible solutions, such as those for patients who do not need the chair and actually fit within time limits.

7 Experimental Results

In this section, we report the results of our empirical analysis of the NMS problem solved via ASP. We employ 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 [31] 5.6.2, using parameters *--restart-on-model* for faster optimization (as a result of a preliminary analysis with several configurations). The choice of time limits is based on an empirical analysis, as it represents a good compromise between solution quality and the need to have solutions quickly. Specifically, for the direct encoding, we set a time limit of 120 seconds. This is a balance between providing the hospital with timely results–given that we are scheduling a full day–and trying to obtain as many optimal results as possible. For the LBBD methodology, we maintain the overall time limit of 120 seconds but impose a time limit of 5 seconds for solving the main problem and 3 seconds for each subproblem. Moreover, to reduce the time spent during the grounding phase, we decide to use the multi-shot solving technique [32] through the usage of the Clingo API in conjunction with the LBBD methodology. Indeed, the multi-shot approach enables iterative enrichment of the ASP model with non-ground rules, avoiding the need to repeat the entire grounding process. This feature is particularly advantageous in the context of LBBD since it allows the MP to be incrementally refined by adding new constraints derived from the SPs without requiring full re-grounding at each iteration. For the specific NMS problem, the rule composing the cut is invoked as shown at the end of the previous section using the multi-shot methodology and passing to the Clingo API the values to be associated with the parameters of `bound(n,room,iter)`, where the meaning is that in the iteration `iter` for the room `room` is added a constraint that impose that the total sum of time required in the tomograph is less than the value `n`. Finally, a third subsection is instead devoted to evaluate whether the LBDD approach helps alleviating the performance issues of GUROBI on ILP instances produced by automatic translation of the ASP encoding. The encoding and benchmarks employed in this section can be found at: `https://github.com/MarcoMochi/JLC2024NSP`.

7.1 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 provides a schedule of patients throughout the day in a range of 10 hours, split into 120 time slots of 5 minutes each.

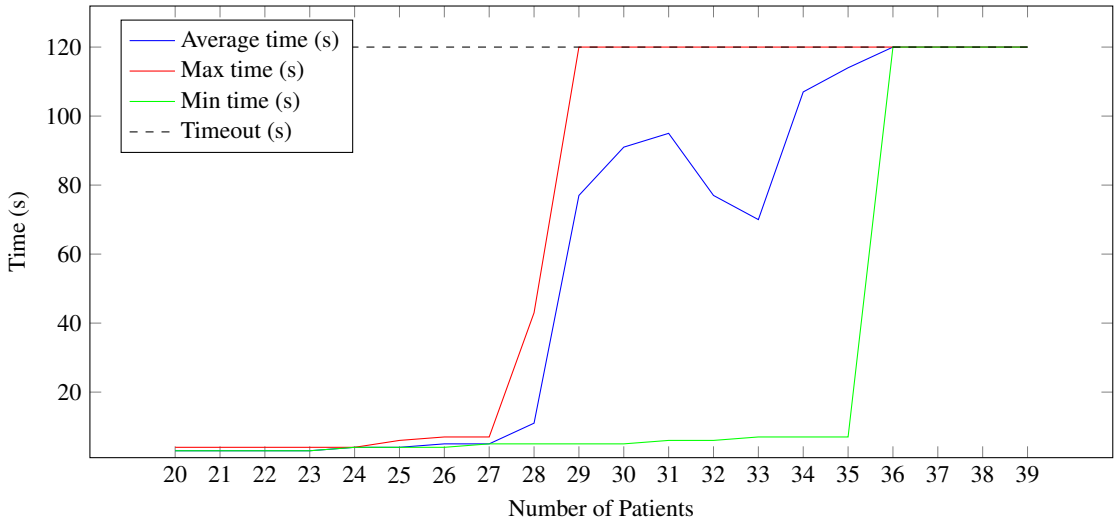
Each patient is linked to one of the possible exams. By analyzing the real data used, the vast majority of patients, about 85%, require the same type of examination, which spends 2 time slots for the anamnesis phase, 2 time slots for the medical preparation, 10 time slots for the drug injection and the bio-distribution time, and, finally, at least 7 time slots for the image detection. The other patients can be associated with one of the other 10 possible protocols. The available resources considered for the schedule include two rooms for the radiotherapy, each equipped with a tomograph and three chairs. The number of patients requiring exam changes daily but, on average, there are 29 patients to be scheduled, with a maximum of 37 patients.

7.2 Results

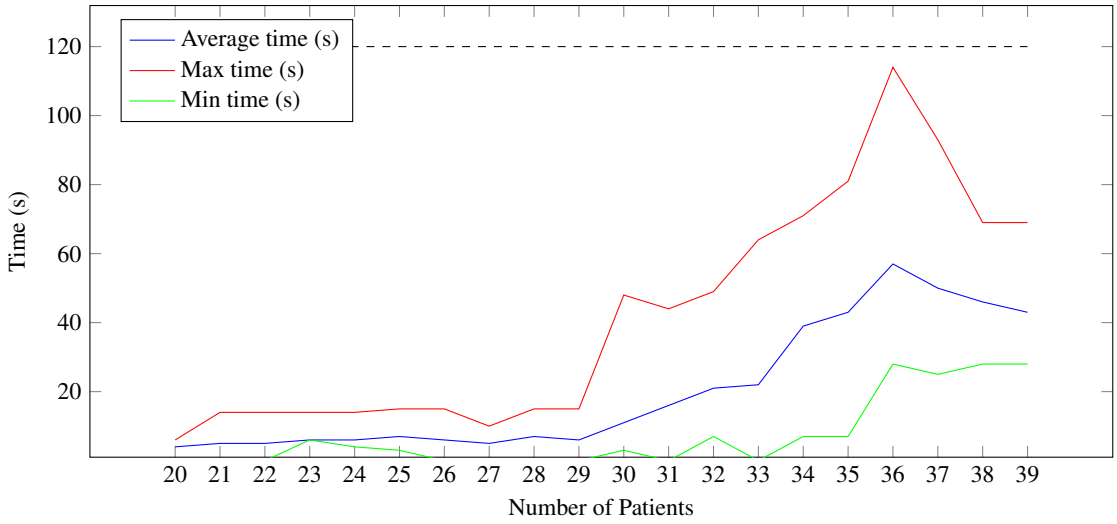
We now present and compare the results obtained by two approaches introduced in this paper, detailed in Sections 5 and 6, using real data. As discussed above, the encoding employing the LBBD methodology does not always guarantee to reach an optimal solution; consequently, just looking at the time required to compute the solution is not enough to fully evaluate its performance. For this reason, we perform the comparison along (i) the time required to compute the solution, and (ii) the quality of the solution depending on the two levels of optimization considered, comprising both optimal and non-optimal solutions. Before delving into the details of the results, we report here some statistics regarding the number of iterations of the LBDD approach: the minimum is 1, the maximum is 10, while the mean number is 3.38.

Time efficiency. The first analysis of the obtained results, reported in Figure 4, includes an examination of the time required to achieve an optimal solution according to the number of patients to be scheduled. Specifically, Figure 4a presents the minimum, maximum, and average times (for instances not solved optimally, we assign the time limit) needed to achieve an optimal solution using the direct encoding, while Figure 4b, provides the corresponding data for the LBBD approach. In this way, it is possible to evaluate the waiting time for the hospital to obtain an optimal solution, depending on the number of patients to be assigned. From the comparison, it can be seen that using a direct encoding, all instances with up to 29 patients are optimally solved within the given time limit. Further, on average, instances with less than 36 patients are overall solved optimally in less than 120 seconds, meaning that at least one instance is optimally solved for each number of patients in this time range. In the other cases, only a few days involve more than 36 patients and, in such cases, the encoding is not able to find an optimal schedule. Overall, the direct encoding successfully provides an optimal solution for more than 60% of the instances. Instead, Figure 4b shows that the LBBD approach consistently leads to an optimal solution within the time limit. Moreover, starting from 28 patients, all computed minimum, maximum and average times are lower than those of the direct encoding, sometimes significantly. In particular, the optimal solution is obtained, on average, in less than 60 seconds for all instances. This represents a significant improvement, as this time threshold was only achieved using the direct encoding for instances with up to 29 patients.

Figure 5 displays the comparison of the time (in seconds) required to solve each instance using the two proposed approaches across various instances. The x-axis represents the time employed by the LBBD approach, while the y-axis shows the time taken by the direct encoding. The diagonal line serves as a reference point to appreciate the instances in which one of the methodologies outperformed the other: instances that fall above this line indicate that the direct encoding requires more time to compute the solution than the LBBD approach, whereas instances below the line show that the direct encoding is more efficient for those cases. Overall, the graph confirms the good performances of both approaches. The vast majority of the instances are solved in less than 10 seconds by the LBBD approach, while many instances are solved in less than 20 seconds using the direct encoding. Notably, all the instances that reach the time limit with the direct encoding are instead solved (optimally) by the LBBD approach. In particular, most of these instances remain



(a) Direct encoding



(b) LBBD approach

Fig. 4: Average, maximum, and minimum times (seconds) required to solve instances with different numbers of patients using the (a) direct encoding, and (b) the LBBD approach.

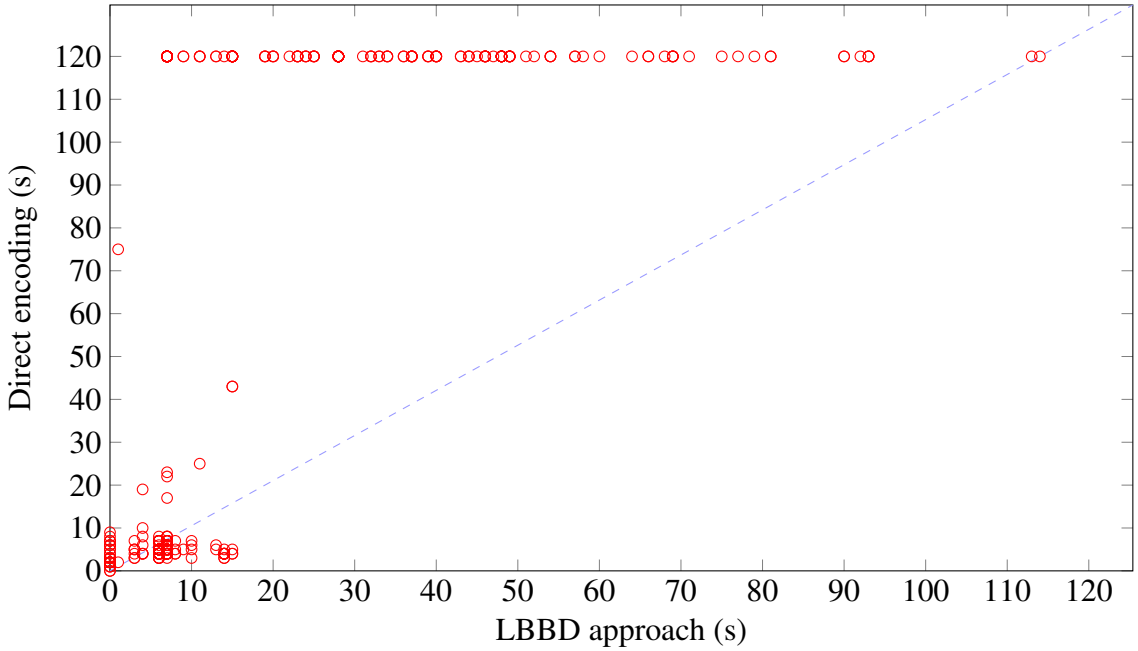


Fig. 5: Comparison of the time (in seconds) required to solve each instance using the LBBD approach (X-axis) and the direct encoding (Y-axis).

in the first half of the graph, meaning they are solved in less than 60 seconds. Additionally, the graph clearly shows that few instances require more than 90 seconds when using the LBBD approach, further confirming the performance improvement.

Solutions quality. The second analysis is focused on the quality of the produced solutions, starting from the first optimization criterion. Overall, out of the 210 instances in which the direct encoding produces the optimal solution, the LBBD approach is able to return the same result in 209 instances. In all instances, it matches or improves the solutions obtained with the direct encoding in 98.6% of the tested instances. Specifically, 35.2% of these instances admit better solutions than those found by the direct encoding, meaning that at least one additional patient is successfully assigned. In the 1.4% of instances where the LBBD approach produced worse results, this is likely due not only to the constraints we used but also to the settings we chose. We set a very short time limit for the SPs, which works well overall but may have led to lower-quality solutions in a few cases. However, just one patient is not assigned in these cases in comparison to the direct encoding.

Further, we observe that, due to the restrictions imposed by the data, the optimal solution may not always provide a high number of scheduled patients. Therefore, we also want to deeply analyze the obtained solutions to assess the quality of the schedule. The analysis of the results shows that, despite not being optimal, more than 80% of the non-optimal solutions manage to schedule over 80% of the patients. In the remaining cases, the solution still manages to assign over 70% of the total patients in all instances. These findings indicate that, even under strict time limits, the results remain of satisfactory quality, which is a crucial aspect of operational scenarios.

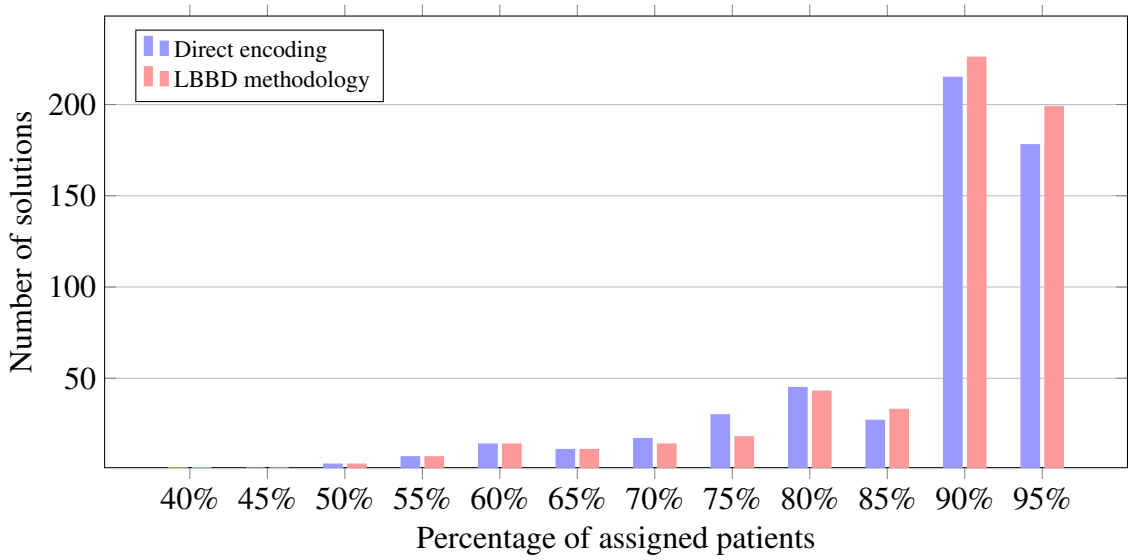


Fig. 6: Number of solutions corresponding to various percentages of assigned patients, comparing the direct encoding (in blue) and the LBBD approach (in red).

Having thoroughly analyzed this aspect, we can now proceed to summarize the results obtained across all instances. Figure 6 shows the number of solutions (y-axis) based on the percentage of assigned patients (x-axis), comparing the direct encoding (blue) and the LBBD approach (red). The sum of the columns for each color amounts to 366, which represents the total number of solutions analyzed. Overall, the majority of solutions have a patients assignment rate exceeding 80%. Moreover, concerning the comparison of the two methodologies, it is possible to note that, when considering the solutions with more than 80% of patients assigned, the LBBD methodology has always more solutions found. This confirms that this methodology is not only better in terms of time required to get a solution, but is almost always able to get a solution of higher quality. We observe that, unlike the non-optimal solutions, optimally solved instances occasionally exhibit a percentage of assigned patients below 60%, attributed to resource limitations. We analyze this scenario in more detail to explain why this is the case. 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. Following a thorough evaluation, we find that among the 33 patients, 14 require an exam with protocol number 823, while the remaining 19 require an exam with protocol number 815. The solution successfully assigns exams to all 14 patients with protocol number 823. However, due to the nature of the exam with protocol number 815, the solution can only assign one patient with this protocol to each tomography, that is it assigns exams to just 2 patients. This limitation explains the low percentage of assigned patients in this instance.

We now proceed to analyze the second optimization criterion. Specifically, we focus on the average waiting time for each patient, quantified as the unnecessary number of time slots spent in the hospital. We start by analyzing the original schedule of the hospital. In particular, only 19% of the considered days had zero waiting times for each patient. In contrast, our solutions utilizing the direct encoding achieved zero waiting times for patients in 70% of instances. This indicates that all the patients can comply with their protocols optimally in these solutions. These results further improved when using the LBBD approach, which resulted

in 88% of instances having no delays. Furthermore, in 90% of the instances, the average waiting time per patient, measured in time slots, is less than 2. These results significantly enhance those obtained by the hospital, reducing the unnecessary time patients spend in the hospital. Moreover, the original hospital schedules achieved only 75% of instances with less than 5 time slots of waiting time. In contrast, the ASP-based solutions produced significantly better results, improving this percentage to 98% and 100% for the direct encoding and the LBBD approach, respectively. In a real-world application, even small improvements in terms of waiting times are important for the satisfaction of patients. When it comes to higher saving, as it is the case of our solutions on some instances when compared to the original hospital schedules, this may pave the way to schedule more patients in the time horizon, as shown in previous analysis.

7.3 Application of the LBDD approach to Alternative Logic-Based Formalisms

In Subsection 5.2, we observed that GUROBI was unable to solve any of the instances obtained through the automatic translation of ASP programs. This result was rather unexpected, given that GUROBI is a powerful solver and has often demonstrated efficiency on real-world instances derived from automatic translations. This raises a natural research question: can the LBBD approach enhance GUROBI's performance on such translated instances? To investigate this, we applied the LBBD encoding to ASP instances, and we translated the resulting master problem (MP), subproblems (SPs), and generated cuts into ILP formulations, allowing GUROBI to attempt solving the corresponding problems. Using the same experimental setting as in Subsection 5.2, i.e. a 120 seconds timeout for the overall execution and a 5 seconds timeout per subproblem, the ASP-based LBBD approach solved all 31 instances, including 24 optimally. In contrast, the ILP encoding executed by GUROBI failed to solve any instance under these constraints. However, when increasing the total timeout to 10 minutes and allowing up to 60 seconds per subproblem, GUROBI was able to solve all 31 instances, 23 of them optimally. It is worth noting that even with this extended timeout, the direct ILP encoding (i.e. without LBBD) still failed to solve any instance. These findings suggest that the LBBD approach provides substantial benefits not only for ASP but also for other logic-based formalisms such as ILP.

8 Handling Fairness Rules

In this section, we state why fairness rules are important, present the encoding we implemented to solve the fairness problem, and analyze the obtained results.

8.1 Motivation and Fairness Problems in the NMS

In the context of AI and in particular in Digital Health, it is important not only to present a working solution but also be aware of the possible problems that a solution could create. As an example, even an optimal solution could lead to financial waste due to, e.g., some machines remaining underutilized, leading to unnecessary costs. Further, this can have serious ethical and legal implications, can compromise the quality of care and could result in unequal treatment of patients. Fairness rules, not required to solve a problem, help to eliminate unwanted biases towards patients and diseases.

In the NMS, considering the data we have, fairness issues may correspond to the underutilization of the tomographs, which are the machines that have a high cost and should be used as much as possible during the day, and the tendency of the solutions to schedule patients with protocols requiring low times. Indeed, being the minimization of unassigned patients the weak constraints with the highest priority, the solution assigns as many patients as possible, i.e. it prefers to assign two patients requiring low time in the tomograph than a single patient that requires long time in the tomograph.

```
1 to_check(ID1, PrID1, ID2, PrID2) :- not x(ID1, DAY, _, PrID1,0), reg(ID1, DAY, PrID1),
      x(ID2, DAY, _, PrID2,0), PrID1 != PrID2.
2 :~ to_check(ID1, PrID1, ID2, PrID2), cost(PrID1, N), exam(PrID1,3,M), exam(PrID2,3,M1),
      cost(PrID2, N1), (N+M) > (N1+M1). [1@1, ID1]
```

Fig. 7: ASP rules for mitigating fairness problems.

Before presenting the rules to handle these problems, it is important to analyze the results we have obtained in order to decide if tailored rules are required and to determine if they work. We decided to use the results obtained with the direct encoding presented in Section 5 to analyze the usage of the tomographs (since the direct encoding and the LBBD one minimize the same objectives, the solutions share the same effects and biases). By analyzing the results, it is possible to see that in more than 60% of the tested instances the tomographs are used more than 80% of the time. Further, considering only instances that are not optimally solved, the tomographs are used on average 89% of the time. Considering that all the protocols require some time before passing through the tomographs and then some unused time is unavoidable, we can state that in our solution the machines are already deeply utilized as a consequence of the minimization of the unassigned patients.

Concerning the biases towards registrations with protocols requiring a low time in the tomograph and on the phases before it, we again analyzed the results obtained by the direct encoding. In particular, we focused on two protocols that could get penalized from the schedule: protocol number 888, being the protocol requiring the longest time on the tomograph, and protocol number 819, being among the protocols requiring the larger time between all the phases. We excluded from this analysis protocol number 823 even if it is the protocol requiring the largest time in total because it is by far the most required protocol and the most assigned, occurring on some days as the only protocol requested. In the analysis, we found that, in general, 85% of the patients with any protocol but the 888 or the 819 are assigned. However, just 76% and 71% of the patients with protocols 888 and 819, respectively, are assigned. This highlights a bias towards the protocols that require less time, thus, in the following, exploiting the modularity of ASP, we will present a set of rules that can be added to the original encoding to solve this issue.

8.2 Rules for Handling Fairness and Results

The rules added to improve the fairness of the solutions are reported in Figure 7. In particular, they try to force the schedule to assign the treatment to as many registrations with long therapy as possible, without reducing the number of patients assigned by the schedule, being the added weak constraints with lowest priority. Indeed, since we want to ensure that registrations requiring protocols having longer treatments are assigned as the others, the atom `to_check` is used to derive all the registrations that are not assigned and have a protocol that is different from all the protocols required by the assigned registrations. Finally, the weak constraint minimizes the number of registrations that were not assigned and required a longer time in the hospital than the assigned ones.

We used the rules in Figure 7 in combination with the encoding in Figure 1 in all the instances to test the effectiveness of the solution to overcome the bias towards some registrations (the encoding employed in this section can be found at: `https://github.com/MarcoMochi/JLC2024NSP/tree/main/Fair`). In particular, we used the same parameters and time limit as presented in Section 7 to compute a solution to all the instances and analyze the results. To determine if the new rules had a positive impact on the biases, we evaluated again the percentage of patients assigned according to their protocols and then we determined the

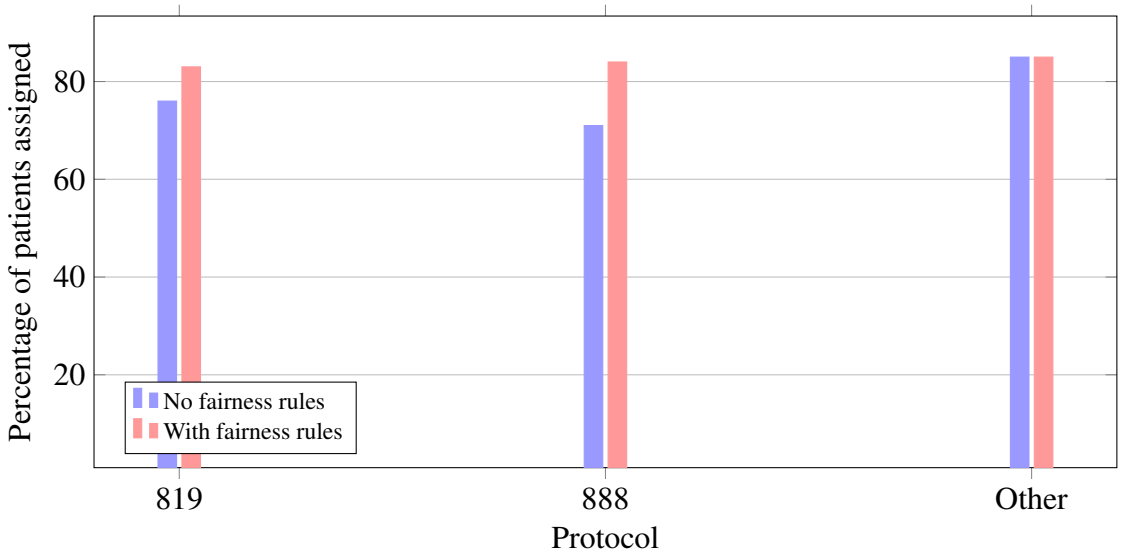


Fig. 8: Percentage of patients having protocol 819 and 888 assigned when using the direct encoding without and with the fairness rules.

cost of these new rules, in terms of the quality of the solutions. As can be seen from Figure 8, we obtained that patients with protocol 888 went from being assigned 76% of the time to 83%, while patients with protocol 819, in the new solution are assigned 84% of time, while it was 71% previously. These values are very close to the percentage of assigned patients having other protocols, that is 85%. This analysis confirms that the added rules are effective in reducing the bias towards the patients under certain protocols. However, it is important to understand what is the cost of these added rules. To analyze the impact of the added rules we consider as a benchmark the results obtained by the direct encoding. First of all, we found that 95% of the results obtained with the added rules have the same or fewer number of unassigned patients, which means that adding the rules does not decrease the quality of the solutions. Thus, just 5% of the tested instances obtained a worse solution with the added fairness rules. However, in all of these worsened solutions, the number of assigned patients decreased by just one patient, meaning that the added rules have a weak impact on decreasing the quality of the encoding. It is important to note that these results are obtained by maintaining the same timeout and that all the previously optimally solved solutions are still solved optimally introducing the fairness rules. Moreover, when using the encoding with the fairness rules, the solver requires at most 5 seconds more to find an optimal solution.

9 Related Work

This paper is an extended and revised version of [24], whose main additions are: (i) a new LBBD solving approach (Section 6), which shows in Section 7 to improve scalability, (ii) the explicit handling of fairness issues in our solution (Section 8), and (iii) an improved related work (this section), which now includes paragraphs dealing with additions (i) and (ii).

The rest of the section is organized in four paragraphs: the first is focused on alternative methods for solving the NMS problem and employing real data, the second mentions works in which the LBBD ap-

proaches in scheduling problems is employed, while the last two paragraphs refer to works in healthcare in which fairness has been considered and to other scheduling problems solved with ASP, respectively.

Solving the NMS problem. The work in [46] 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 in a simulated environment. Another work using real data to validate the solution is [52]. 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. The authors of [2] 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. Another approach is followed by the authors of [52], which employs a genetic algorithm in their solving process. The solution is tested on data from a Chinese Hospital. The current paper provides an alternative, logic-based solution for solving the NMS problem.

LBBD in scheduling problems. The LBBD approach has been widely used in combinatorial optimization problems, especially in scenarios where the problem can be divided into multiple parts. Its versatility and effectiveness have been demonstrated across various domains, most notably in scheduling problems. In this context, LBBD has shown the ability to optimize complex, large-scale schedules while improving computational efficiency. For instance, in [16], the authors present the first LBBD approach in ASP for a scheduling problem, implemented through the multi-shot variant of CLINGO, involving chronic outpatients with non-communicable diseases who require ongoing hospital services. In particular, the authors split the problem into MP and SPs to schedule different packages of treatments for the patients in different days. The MP was used to assign as many packages as possible among the available days, considering relaxed constraints concerning the availability of the resources. Then, the result of the MP is used to generate different SPs, one per day, to check if these assignments satisfy the real constraint concerning the availability of the resources on all the days. Consequently, if one of the SPs is infeasible, they provide a cut indicating that the package assigned in the MP for the day cannot be feasibly provided. Overall, they demonstrate that LBBD expands the applicability of ASP to larger instances while maintaining the optimality of the solutions. Similarly, in [39], the LBBD method is applied to the problem of home hospice care staffing and scheduling using mixed integer programming. The objective is to effectively match hospice care aides with patients and schedule visits to their homes. The authors leverage the versatility of LBBD to recompute staff assignments and visitation schedules as needed efficiently. In another study, [49], the authors develop three novel LBBD methods and a cut propagation mechanism to tackle the distributed operating room scheduling (ORS) problem. In the same domain, [37] tackles a stochastic variant of the problem, incorporating uncertain surgery durations by exploiting the LBBD optimality cut to maintain efficiency and robustness. The recursive LBBD approach is explored in [48], where the authors consider a class of multi-mode appointment scheduling problems involving variable resource availability and setup times. They show the efficiency of this recursive LBBD method using real-life data from a gastroenterology clinic at the University Hospital of Northern Norway. Finally, in [20], the authors propose a heuristic adaptation of LBBD to the home health care problem, focusing on

designing efficient routes and schedules for health care workers who visit patient homes. These are some of the existing works (see [41] for more detail) that showcase the applicability of LBBD across different scheduling scenarios, underscoring its capability to manage complex, large-scale problems while enhancing both solutions quality and computational efficiency. The current paper is one of the very few, together with [16], employing LBBD in ASP-based scheduling, both in Healthcare.

Handling fairness rules. The approach in [17] exploits ASP to address the problem of fairness and preference satisfaction in the mid-term scheduling of medical staff within a hospital network. In [51] the authors examine the challenges of fairness in the clinical integration of AI in medicine. They provide a comprehensive review of concerns related to AI fairness and discuss various strategies to mitigate biases in AI-based healthcare systems. The paper [3] addresses the appointment scheduling problem in hospitals, utilizing a fairness-oriented approach to optimize resource allocation and ensure equitable access to medical services. In [4], the authors propose a mixed-integer linear programming model for patient appointment scheduling in hospitals. Their model prioritizes patients based on their overall health status and incorporates fairness as a key consideration to minimize patient waiting times and improve healthcare service equity. The authors of [44] explore real-world patient arrival patterns and multiple allocation techniques to assess the impact of an optimized allocation on ensuring fair treatment for patients. The authors also investigate how tuning machine learning hyperparameters can balance optimization with fairness considerations. Finally, the work [30] provides a comprehensive survey on fairness and bias in AI, addressing their sources, impacts, and mitigation strategies. The current paper provides another instance of applying fairness for a particular scheduling problem in Healthcare.

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., [7] for a recent survey), the first solved problems were the *Nurse Scheduling Problem* [8,25,9], where the goal is to create a scheduling for nurses working in hospital units, and the ORS problem [23,22], whose goal is to assign operating rooms to patients. More recent problems include the *Chemotherapy Treatment Scheduling* problem [21], in which patients are assigned a chair or a bed for their treatments, the *Rehabilitation Scheduling Problem* [18], which assigns patients to operators in rehabilitation sessions, and the *Pre-Operative Assessment Clinic* problem [19], which schedules patients for pre-operative operations. The Rehabilitation Scheduling Problem and the Pre-Operative Assessment Clinic problem are solved by decomposition, but their approaches do not fall into LBDD, given that they rely on the definition of two problems to be solved sequentially. Other differences of the NMS problem in comparison to these solutions include (a) the fact that differently to, e.g., the ORS problem, the NMS problem introduces distinct phases with varying times to be considered, but does not consider patients prioritization (mainly because it is not included in the real data employed); (b) in comparison to, e.g., the Chemotherapy Treatment Scheduling problem, the NMS problem handles different resources assigned to patients (tomographs); (c) differently from the Pre-Operative Accessment Problem, here we do not address the pre-operative phase; and (d) the implementation of none of the approaches mentioned above employs the multi-shot variant of CLINGO. Concerning scheduling problems beyond the healthcare domain, ASP encodings were proposed in different contexts. The authos of [11] 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. In [1], the authors proposed a hybrid approach, using both pure ASP and difference constraints, to solve the problem of routing, scheduling, and optimizing real-world training scheduling. The approach in [34] used ASP for routing driverless transport vehicles in the context of car assembly at Mercedes-Benz Ludwigsfelde GmbH. About [26], the authors proposed a solution to the Job-shop Scheduling Problem, 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 [29], where industrial applications dealt with ASP are presented, including those involving scheduling problems. Overall, the current paper deals with a novel problem in Healthcare for ASP.

10 Conclusion

In this paper, we have presented an analysis of the Nuclear Medicine Scheduling (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 solutions. Specifically, we proposed a direct encoding and a LBBD approach implemented through the usage of multi-shot solving. The results obtained from real data show satisfying outcomes in terms of quality, and that the LBBD approach also helps improving scalability. Moreover, we extended the direct encoding taking into account the fairness of the solutions ensuring that patients are assigned fairly independently from the protocols they follow. Future works include the implementation of a web application for easy usage of our solution.

References

1. Abels, D., Jordi, J., Ostrowski, M., Schaub, T., Toletti, A., Wanko, P.: Train scheduling with hybrid asp. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) Logic Programming and Nonmonotonic Reasoning. pp. 3–17. Springer International Publishing, Cham (2019)
2. Akhavizadegan, F., Ansarifar, J., Jolai, F.: A novel approach to determine a tactical and operational decision for dynamic appointment scheduling at nuclear medical center. Computers & Operations Research **78**, 267–277 (2017)
3. Ala, A., Alsaadi, F.E., Ahmadi, M., Mirjalili, S.: Optimization of an appointment scheduling problem for healthcare systems based on the quality of fairness service using whale optimization algorithm and nsga-ii. Scientific Reports **11**(1), 19816 (2021)
4. Ala, A., Simic, V., Pamucar, D., Tirkolaee, E.B.: Appointment scheduling problem under fairness policy in healthcare services: fuzzy ant lion optimizer. Expert systems with applications **207**, 117949 (2022)
5. Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F.: Evaluation of disjunctive programs in WASP. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) LPNMR. LNCS, vol. 11481, pp. 241–255. Springer (2019)
6. Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F.: Evaluation of disjunctive programs in WASP. In: LPNMR 2019. LNCS, vol. 11481, pp. 241–255. Springer (2019). `https://doi.org/10.1007/978-3-030-20528-7_18`, `https://doi.org/10.1007/978-3-030-20528-7_18`
7. Alviano, M., Bertolucci, R., Cardellini, M., Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Mochi, M., Morozan, V., Porro, I., Schouten, M.: Answer set programming in healthcare: Extended overview. In: IPS and RCRA 2020. CEUR Workshop Proceedings, vol. 2745. CEUR-WS.org (2020), `http://ceur-ws.org/Vol-2745/paper7.pdf`
8. Alviano, M., Dodaro, C., Maratea, M.: An advanced answer set programming encoding for nurse scheduling. In: AI*IA. LNCS, vol. 10640, pp. 468–482. Springer (2017)
9. Alviano, M., Dodaro, C., Maratea, M.: Nurse (re)scheduling via answer set programming. Intelligenza Artificiale **12**(2), 109–124 (2018)
10. Ansótegui, C., Pacheco, T., Pon, J.: Pypblib (2019), `https://pypi.org/project/pypblib/`
11. Balduccini, M.: Industrial-size scheduling with ASP+CP. In: Delgrande, J.P., Faber, W. (eds.) Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6645, pp. 284–296. Springer (2011)
12. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Computational Management Science **2**(1) (2005)

13. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011)
14. Buccafurri, F., Leone, N., Rullo, P.: Enhancing Disjunctive Datalog by Constraints. IEEE Transactions on Knowledge and Data Engineering **12**(5), 845–860 (2000)
15. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., Schaub, T.: ASP-Core-2 input language format. Theory and Practice of Logic Programming **20**(2), 294–309 (2020)
16. Cappanera, P., Gavanelli, M., Nonato, M., Roma, M.: Logic-based Benders decomposition in answer set programming for chronic outpatients scheduling. Theory and Practice of Logic Programming **23**(4), 848–864 (2023). https://doi.org/10.1017/S147106842300025X, https://doi.org/10.1017/s147106842300025x
17. Cappanera, P., Gavanelli, M., Nonato, M., Roma, M., Fabbri, N., Feo, C., Soverini, R.: Fairness-driven midterm scheduling of medical staff at a hospital network (11 2023). https://doi.org/10.13140/RG.2.2.19565.36326
18. Cardellini, M., Nardi, P.D., Dodaro, C., Galatà, G., Giardini, A., Maratea, M., Porro, I.: A two-phase ASP encoding for solving rehabilitation scheduling. In: Moschoyiannis, S., Peñaloza, R., Vanthienen, J., Soylu, A., Roman, D. (eds.) Proceedings of the 5th International Joint Conference on Rules and Reasoning (RuleML+RR 2021). Lecture Notes in Computer Science, vol. 12851, pp. 111–125. Springer (2021)
19. Caruso, S., Galatà, G., Maratea, M., Mochi, M., Porro, I.: Scheduling pre-operative assessment clinic with answer set programming. Journal of Logic and Computation **34**(3), 465–493 (04 2023). https://doi.org/10.1093/logcom/exad017, https://doi.org/10.1093/logcom/exad017
20. Ciré, A., Hooker, J.N.: A heuristic logic-based benders method for the home health care problem. manuscript, presented at Matheuristics (2012)
21. Dodaro, C., Galatà, G., Grioni, A., Maratea, M., Mochi, M., Porro, I.: An ASP-based solution to the chemotherapy treatment scheduling problem. Theory and Practice of Logic Programming **21**(6), 835–851 (2021)
22. Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Porro, I.: An ASP-based solution for operating room scheduling with beds management. In: Fodor, P., Montali, M., Calvanese, D., Roman, D. (eds.) Proceedings of the Third International Joint Conference on Rules and Reasoning (RuleML+RR 2019). Lecture Notes in Computer Science, vol. 11784, pp. 67–81. Springer (2019)
23. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: Operating room scheduling via answer set programming. In: AI*IA. LNCS, vol. 11298, pp. 445–459. Springer (2018)
24. Dodaro, C., Galatà, G., Marte, C., Maratea, M., Mochi, M.: Nuclear medicine scheduling via answer set programming. In: Angelis, E.D., Proietti, M. (eds.) Proceedings of the 39th Italian Conference on Computational Logic (CILC 2024). CEUR Workshop Proceedings, vol. 3733. CEUR-WS.org (2024)
25. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: LPNMR. LNCS, vol. 10377, pp. 301–307. Springer (2017)
26. El-Kholany, M.M.S., Gebser, M., Schekotihin, K.: Problem decomposition and multi-shot ASP solving for job-shop scheduling. Theory and Practice of Logic Programming **22**(4), 623–639 (2022)
27. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine **37**(3), 53–68 (2016)
28. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1), 278–298 (2011)
29. Falkner, A.A., Friedrich, G., Schekotihin, K., Taupe, R., Teppan, E.C.: Industrial applications of answer set programming. Künstliche Intelligenz **32**(2-3), 165–176 (2018)
30. Ferrara, E.: Fairness and bias in artificial intelligence: A brief survey of sources, impacts, and mitigation strategies. Sci **6**(1), 3 (2023)
31. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: ICLP (Technical Communications). OASICS, vol. 52, pp. 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
32. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot ASP solving with clingo. Theory and Practice of Logic Programming **19**(1), 27–82 (2019)
33. Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., Schaub, T.: Evaluation techniques and systems for answer set programming: a survey. In: Lang, J. (ed.) IJCAI. pp. 5450–5456. ijcai.org (2018)

34. Gebser, M., Obermeier, P., Schaub, T., Ratsch-Heitmann, M., Runge, M.: Routing driverless transport vehicles in car assembly with answer set programming. Theory and Practice of Logic Programming **18**(3-4), 520–534 (2018). https://doi.org/10.1017/S1471068418000182, https://doi.org/10.1017/S1471068418000182
35. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the Fifth International Conference and Symposium , Seattle, Washington, August 15-19, 1988 (2 Volumes). pp. 1070–1080. MIT Press (1988)
36. Geoffrion, A.M.: Generalized benders decomposition. Journal of optimization theory and applications **10**, 237–260 (1972)
37. Guo, C., Bodur, M., Aleman, D.M., Urbach, D.R.: Logic-based benders decomposition and binary decision diagram based approaches for stochastic distributed operating room scheduling. INFORMS Journal on Computing **33**(4), 1551–1569 (2021)
38. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2021), https://www.gurobi.com
39. Heching, A., Hooker, J.: Scheduling home hospice care with logic-based benders decomposition. In: Integration of AI and OR Techniques in Constraint Programming: 13th International Conference, CPAIOR 2016, Banff, AB, Canada, May 29-June 1, 2016, Proceedings 13. pp. 187–197. Springer (2016)
40. Hooker, J.: A hybrid method for planning and scheduling. In: International Conference on Principles and Practice of Constraint Programming. pp. 305–316. Springer (2004)
41. Hooker, J.: Logic-based benders decomposition: theory and applications (2023)
42. Hooker, J., Ottosson, G.: Logic-based benders decomposition. Mathematical Programming **96**(1), 33–60 (2003)
43. Martins, R., Manquinho, V.M., Lynce, I.: Open-wbo: A modular maxsat solver,. In: SAT 2014. LNCS, vol. 8561, pp. 438–445. Springer (2014). https://doi.org/10.1007/978-3-319-09284-3_33, https://doi.org/10.1007/978-3-319-09284-3_33
44. Masroor, F., Gopalakrishnan, A., Goveas, N.: Machine learning-driven patient scheduling in healthcare: A fairness-centric approach for optimized resource allocation. In: 2024 IEEE Wireless Communications and Networking Conference (WCNC). pp. 01–06. IEEE (2024)
45. Olivier Roussel and Vasco Manquinho: Input/Output Format and Solver Requirements for the Competitions of Pseudo-Boolean Solvers (2012), https://www.cril.univ-artois.fr/PB12/format.pdf
46. Pérez, E., Ntaimo, L., Wilhelm, W.E., Bailey, C., McCormack, P.: Patient and resource scheduling of multi-step medical procedures in nuclear medicine. IIE Transactions on Healthcare Systems Engineering **1**(3), 168–184 (2011). https://doi.org/10.1080/19488300.2011.617718, https://doi.org/10.1080/19488300.2011.617718
47. Rahmaniani, R., Crainic, T.G., Gendreau, M., Rei, W.: The benders decomposition algorithm: A literature review. European Journal of Operational Research **259**(3), 801–817 (2017)
48. Riise, A., Mannino, C., Lamorgese, L.: Recursive logic-based benders' decomposition for multi-mode outpatient scheduling. European Journal of Operational Research **255**(3), 719–728 (2016)
49. Roshanaei, V., Luong, C., Aleman, D.M., Urbach, D.: Propagating logic-based benders' decomposition approaches for distributed operating room scheduling. European Journal of Operational Research **257**(2), 439–455 (2017)
50. Saikko, P., Berg, J., Järvisalo, M.: LMHS: A SAT-IP hybrid maxsat solver. In: SAT 2016. LNCS, vol. 9710, pp. 539–546. Springer (2016). https://doi.org/10.1007/978-3-319-40970-2_34, https://doi.org/10.1007/978-3-319-40970-2_34
51. Ueda, D., Kakinuma, T., Fujita, S., Kamagata, K., Fushimi, Y., Ito, R., Matsui, Y., Nozaki, T., Nakaura, T., Fujima, N., et al.: Fairness of artificial intelligence in healthcare: review and recommendations. Japanese Journal of Radiology **42**(1), 3–15 (2024)
52. Xiao, Q., Luo, L., Zhao, S., Ran, X., Feng, Y.: Online appointment scheduling for a nuclear medicine department in a chinese hospital. Computational and Mathematical Methods in Medicine **2018**, 5148215:1–5148215:13 (2018)