# Cyclic Nurse Scheduling with ASP: Two Case Studies from Cosenza Hospitals

Carmine Dodaro[1,*], Marco Maratea[1] and Nicola Ramacciati[2,3]

[1]Dipartimento di Matematica e Informatica, Università della Calabria, Via P. Bucci, 87036, Rende (CS)
[2]Dipartimento di Farmacia e Scienze della Salute e della Nutrizione, Università della Calabria, Via P. Bucci, 87036, Rende (CS)
[3]Ospedale Mariano Santo, Contrada da Muoio Piccolo, 87100, Cosenza (CS)

**Abstract**

The Nurse Scheduling Problem (NSP) is a well-known combinatorial optimization problem with significant practical implications in healthcare workforce management. In this paper, we investigate the use of Answer Set Programming (ASP) to model and solve realistic nurse scheduling scenarios in two Italian healthcare institutions: the Mariano Santo and the Annunziata hospitals. We design ASP encodings capable of handling both general and institution-specific constraints, including shift coverage requirements, rotation rules, and personal unavailability. We analyze the impact of solver configurations and optimization strategies, comparing the performance of CLINGO and WASP across multiple solving modes. Our experimental results show that unsatisfiable core-based strategies are able to find optimal solutions for the tested instances within a few seconds.

**Keywords**

Logic Programming, Healthcare, Answer Set Programming, Nurse Scheduling Problem

## 1. Introduction

The organization of nurses' work schedules is a fundamental task in healthcare management, since it has a direct impact on the quality of the service provided to patients and on the satisfaction and well-being of the medical staff [1, 2]. The Nurse Scheduling Problem (NSP) is the problem of assigning shifts to nurses over a planning horizon, typically a week or a month, while respecting a wide range of constraints. These constraints usually involve coverage requirements, labor regulations, workload balance, and personal or institutional preferences [3, 4]. The NSP is characterized by a high degree of variability, as its formulation often depends on the specific organizational rules and labor regulations in place within each country, as well as on local practices adopted by individual hospitals or departments. In Italy, for instance, there are general national-level guidelines that define key aspects such as maximum working hours, mandatory rest periods, and vacation entitlements. However, these general rules are typically complemented by more specific requirements that reflect the internal organization of each healthcare facility. These may include the structural needs of different departments, such as the required number of nurses per shift or the distinction between inpatient and outpatient units, as well as informal practices and long-standing habits developed over time by the personnel. As a result, each instance of the NSP must be modeled to take into account both regulatory constraints and the operational and cultural context of the department for which the schedule is being produced. For this reason, the NSP has been widely studied in the literature, and it is known to be a combinatorial optimization problem of high complexity, especially when applied to real-world scenarios where numerous practical rules and exceptions must be taken into account.

In this paper, we propose a solution based on Answer Set Programming (ASP) [5, 6] to two different variants of the NSP. ASP is a declarative programming paradigm that is particularly well-suited to model problems characterized by complex constraints and multiple interacting components as demonstrated

CILC 2025: 40th Italian Conference on Computational Logic, June 25–27, 2025, Alghero, Italy
*Corresponding author.
carmine.dodaro@unical.it (C. Dodaro); marco.maratea@unical.it (M. Maratea); nicola.ramacciati@unical.it (N. Ramacciati)
0000-0002-5617-5286 (C. Dodaro); 0000-0002-9034-2527 (M. Maratea); 0000-0002-1176-8133 (N. Ramacciati)

by the high number of its applications in both academic and industrial contexts [7, 8, 9, 10, 11]. Its ability to represent non-monotonic reasoning and to express hard and soft constraints in a natural and compact form makes it a powerful tool for addressing scheduling and planning problems [12]. Indeed, ASP has already been successfully applied to various problems in the healthcare domain [13, 14], including the NSP itself, although in previous works the focus has been on different types of constraints compared to those addressed in this study [15, 16].

As for the NSP variants considered in this paper, they were defined by two different hospital departments in the city of Cosenza, Italy. The first department belongs to the Mariano Santo Hospital, while the second is part of the Annunziata Hospital. Although the two departments present similar scheduling requirements from a general point of view, there are several differences in the specific rules and organizational needs that must be respected. In both cases, the objective is to generate cyclic schedules, that is, schedules based on a repeating sequence of working days and rest days, in order to ensure a balanced distribution of shifts over time. However, the internal policies of the two departments differ, and therefore require separate modeling strategies.

The proposed ASP encodings allow us to generate complete monthly schedules that satisfy all the required constraints, both common and department-specific. Thanks to the efficiency of modern ASP solvers, specifically CLINGO [17] and WASP [18], we are able to obtain (optimal) solutions in a few seconds. This confirms the applicability of ASP as a viable approach for supporting nurse scheduling in real healthcare contexts, where flexibility and customizability are essential.

To summarize, the contribution of this work is twofold. First, we provide two concrete case studies of the NSP arising in real-world contexts. Second, we show how ASP can be used to formalize and solve them within a few seconds.

## 2. Nurse Scheduling Problem (NSP)

The NSP consists of assigning nurses to shifts over a predetermined period of time. In this section, we describe the NSP as it arises in the two hospital departments considered in our study: one from Mariano Santo Hospital and the other from Annunziata Hospital, both located in Cosenza, Italy. While the general structure of the scheduling problem is similar in both cases, each department presents specific operational needs and organizational constraints that must be taken into account when generating feasible and acceptable schedules.

We begin by outlining a set of general rules and requirements that are common to both hospitals. These include the definition of shift types, national labor regulations, and standard contractual obligations that apply uniformly to the nursing staff. Then, we introduce the specific rules associated with each department.

### 2.1. General Rules

The scheduling model is based on a set of predefined shifts, each with specific time intervals and durations. The available shifts include: morning (08:00–14:00, 6 hours), afternoon (14:00–20:00, 6 hours), night (20:00–00:00, 4 hours), and post-night (00:00–08:00, 8 hours). The night and post-night shifts are always considered as a combined duty period, representing a full night shift of 12 consecutive hours.

The schedule must comply with several temporal and contractual constraints. A minimum rest period of 11 hours is required between the end of one shift and the start of the next. The maximum number of working hours allowed per day is 13. In some cases, a nurse may be assigned both morning and afternoon shifts on the same day, which is referred to as a *long call*. On a weekly basis, nurses are expected to work approximately 36 hours, a target that is monitored and enforced over the course of the entire month rather than weekly.

Each nurse is entitled to 32 vacation days per year (30 in certain cases), with two weeks selected by the administration and two weeks chosen by the employee. These vacation periods are commonly managed in two-week blocks, known as quindicine.

As a general principle, a balanced shift rotation is encouraged, typically following the sequence: morning → afternoon → night → post-night → rest. In any case, achieving an even distribution of different shift types across the staff is considered beneficial. Regarding night shifts, a maximum of two consecutive nights is allowed, although the preferred pattern is one night followed by a rest day.

### 2.2. Mariano Santo

Operational constraints also stem from the organizational structure of the hospital. The outpatient unit operates, during both morning and afternoon, Monday through Friday and remains closed on holidays, while the inpatient department (or ward) is open 24 hours a day, seven days a week. Some nurses are permanently assigned to specific units due to personal or medical reasons. Others are restricted from working in the inpatient department and can only be assigned to outpatient duties. There are also nurses who are exempt from night shifts, while a portion of the staff is fully flexible and can be scheduled in both units without restrictions.

Each unit has specific requirements concerning staffing levels. The baseline configuration assumes one nurse per shift in the inpatient department (morning, afternoon, and night), and a minimum of seven nurses in the outpatient unit during the morning shift, and two during the afternoon. Additionally, each day two nurses are assigned to an "off-shift" status (not to be confused with rest days) meaning they are available as reserve staff to cover any unforeseen needs. These off-shift assignments are scheduled either for the entire month or on a weekly basis, depending on planning requirements.

### 2.3. Annunziata

The considered department at Annunziata Hospital is an inpatient unit that operates continuously, 24 hours a day, 7 days a week. The scheduling model for this unit is relatively regular, as it involves only the ward, with no outpatient services. Each shift requires the presence of two nurses, which must be ensured for every shift throughout the month.

From a constraint perspective, this scenario does not introduce significant complexity beyond the general rules previously described. However, a particular requirement concerns one of the nurses, who is married to another nurse working in a different department of the same hospital. According to national regulations, spouses employed within the same institution must not be assigned to the same shift or to consecutive shifts on the same day. For instance, if the spouse is assigned to the morning shift, the nurse in question cannot be assigned to the afternoon shift; instead, they must either be on rest or scheduled for the night shift.

An exception to this rule has been agreed upon by the individuals involved and is respected by both departments: both nurses must work the same night shift. This exception simplifies coordination in those cases, but the constraint remains valid for all other shifts. Since this constraint affects two distinct departments, the heads of the respective units coordinate closely to ensure compliance. Typically, the responsibility for adapting the schedule to accommodate this rule alternates monthly between the two departments: one month the schedule is adjusted by the manager of one department, and the next month by the other.

## 3. ASP Solution

In this section, we present the ASP encodings developed to model the nurse scheduling problems for the two case studies considered in this work. The first concerns the Mariano Santo rehabilitation clinic, while the second focuses on the Annunziata hospital. The ASP model for the Mariano Santo clinic is introduced in Section 3.1, while the encoding for the Annunziata hospital is discussed in Section 3.2. In the following, we assume the reader to be familiar with ASP syntax and semantics [6, 19].

## 3.1. Mariano Santo

The ASP model for the Mariano Santo clinic is divided into two parts. In Section 3.1.1, we describe the data model used to represent the structure of the problem, including nurses and shifts. Section 3.1.2 then presents the ASP encoding that formalizes the scheduling logic and optimization criteria, based on the defined data schema.

### 3.1.1. Data Model

The encoding uses several key predicates to model the nurse scheduling domain:

- `shift(ID, TYPE, HOURS)` defines a shift identified by `ID`, with type `TYPE` (e.g., morning, afternoon, etc.) and associated with `HOURS` working hours.
- `day(D)` defines the days of the month under consideration.
- `clinic_open(WEEK, DAY)` indicates that the clinic is open during a given `WEEK` on specified `DAY`.
- `nurse(ID)` defines the set of nurses involved in the scheduling.
- `nurse_rotation(NURSE_ID, ORDER, SHIFT_ID)` encodes the predefined cyclic rotation for certain nurses, mapping their rotation order to specific shifts.
- `rotation_length(N)` defines the length of the rotation cycle.
- `clinic_only(ID)` indicates nurses who can only be assigned to outpatient (clinic) duties.
- `reserve(ID)` marks nurses who are on reserve and used to fill schedule gaps as needed.
- `staff_per_shift(SHIFT_ID, N)` specifies the required number of personnel for a shift type within the outpatient unit.
- `clinic_shifts(ID, TYPE, HOURS)` includes the clinic-compatible shifts (excluding night and postnight), with an additional "long" shift type.
- `assign(NURSE, DAY, SHIFT, ORDER)` models the assignment of a nurse to a specific shift on a given day, according to their position in the cyclic rotation.
- `result(NURSE, DAY, SHIFT)` is the output predicate, collecting the assignments made for the final schedule.

The following is the set of input facts provided to the problem.

```
shift(1, morning, 6). shift(2, afternoon, 6). shift(3, night, 4).
shift(4, postnight, 8). shift(5, vacation, 0). shift(6, rest, 0).
day(1..30). nurse(1..14). reserve(13..14). clinic_only(6..14).
clinic_open(1,1..4). clinic_open(2,7..11). clinic_open(3,14..18).
clinic_open(4,22..24). clinic_open(5,28..30). rotation_length(5).
nurse_rotation(1, 0, 1). nurse_rotation(2, 1, 2). nurse_rotation(3, 2, 3).
nurse_rotation(4, 3, 4). nurse_rotation(5, 4, 6).
staff_per_shift(1, 7). staff_per_shift(2, 2).
```

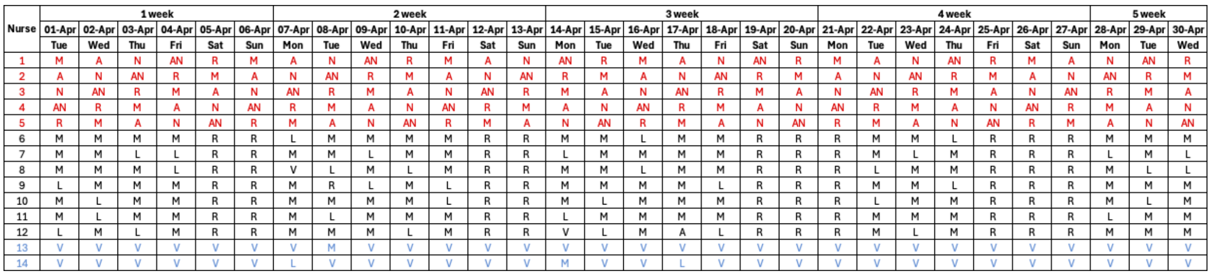
An example of the computed schedule is reported in Figure 1, where `M`, `A`, `N`, `AN`, `V`, `R`, `L` stand for `morning`, `afternoon`, `night`, `postnight`, `vacation`, `rest`, and `long`, respectively.

### 3.1.2. ASP Encoding

In the following, we describe the meaning of each rule of the ASP encoding reported in Figure 2, using the notation $r_i$ to refer to the rule appearing at line $i$ in the logic program.

Rule $r_1$ initializes the rotation-based schedule for each nurse. Specifically, for any nurse `X`, this rule assigns them to the shift `S` that corresponds to their position `Ord` in the predefined cyclic rotation. This assignment is made for the first day of the planning period.

Rule $r_2$ extends the cyclic schedule over the rest of the planning horizon. Given that a nurse `X` was assigned to shift `S` with order `Ord` on day `D`, the rule computes the next order `Ord2` (modulo the rotation length `Z`) and assigns the corresponding shift `NS` on the following day `D+1`.



| Nurse | 1 week | | | | | | 2 week | | | | | | | 3 week | | | | | | | 4 week | | | | | | | 5 week | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 01-Apr | 02-Apr | 03-Apr | 04-Apr | 05-Apr | 06-Apr | 07-Apr | 08-Apr | 09-Apr | 10-Apr | 11-Apr | 12-Apr | 13-Apr | 14-Apr | 15-Apr | 16-Apr | 17-Apr | 18-Apr | 19-Apr | 20-Apr | 21-Apr | 22-Apr | 23-Apr | 24-Apr | 25-Apr | 26-Apr | 27-Apr | 28-Apr | 29-Apr | 30-Apr |
| | Tue | Wed | Thu | Fri | Sat | Sun | Mon | Tue | Wed | Thu | Fri | Sat | Sun | Mon | Tue | Wed | Thu | Fri | Sat | Sun | Mon | Tue | Wed | Thu | Fri | Sat | Sun | Mon | Tue | Wed |
| 1 | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R |
| 2 | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M |
| 3 | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A |
| 4 | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N |
| 5 | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN |
| 6 | M | M | M | M | R | R | L | M | M | M | M | R | R | M | M | L | M | M | R | R | R | M | M | L | R | R | R | M | M | M |
| 7 | M | M | L | L | R | R | M | M | L | M | M | R | R | L | M | M | M | M | R | R | R | M | L | M | R | R | R | L | M | L |
| 8 | M | M | M | L | R | R | V | L | M | L | M | R | R | M | M | L | M | M | R | R | R | L | M | M | R | R | R | M | L | L |
| 9 | L | M | M | M | R | R | M | R | L | M | L | R | R | M | M | M | M | L | R | R | R | M | M | L | R | R | R | M | M | M |
| 10 | M | L | M | M | R | R | M | M | M | M | L | R | R | M | L | M | M | M | R | R | R | L | M | M | R | R | R | M | L | M |
| 11 | M | L | M | M | R | R | M | L | M | M | M | R | R | L | M | M | M | M | R | R | R | M | M | M | R | R | R | L | M | M |
| 12 | L | M | L | M | R | R | M | M | M | L | M | R | R | V | L | M | A | L | R | R | R | M | L | M | R | R | R | M | M | M |
| 13 | V | V | V | V | V | V | V | M | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V |
| 14 | V | V | V | V | V | V | L | V | V | V | V | V | V | M | V | V | L | V | V | V | V | V | V | V | V | V | V | V | V | V |

**Figure 1:** Example of a possible solution for the NSP variant considered in Mariano Santo hospital.

```
1  assign(X,1,S,Ord) :- nurse(X), nurse_rotation(X,Ord,S).

2  assign(X,D+1,NS,Ord2) :- assign(X,D,S,Ord), Ord2 = (Ord+1)\Z, rotation_length(Z),
       nurse_rotation(_,Ord2,NS), day(D+1).

3  clinic_shifts(X, N, H) :- shift(X, N, H), X != 3, X != 4.
4  clinic_shifts(7, long, 12).
5  {assign(X,D,S) : clinic_shifts(S,_,_)} = 1 :- clinic_only(X), clinic_open(_,D).

6  :- #count{X: assign(X,D,S); X: assign(X,D,7)} != N, clinic_open(_,D), staff_per_shift(S,
       N).

7  :- clinic_only(X), not reserve(X), clinic_open(W,_), #sum{H,D: assign(X,D,S),
       clinic_open(W,D), clinic_shifts(S,_,H)} > 36.

8  :~ reserve(N), day(D), not assign(N,D,5). [1@1,D,X]

9  result(X,D,N) :- assign(X,D,SID,_), shift(SID,N,_).
10 result(X,D,N) :- assign(X,D,SID), shift(SID,N,_).
11 #show result/3.
```

**Figure 2:** ASP Encoding for the Mariano Santo Hospital.

Rule $r_3$ defines which shifts are considered valid for outpatient (clinic-only) nurses. Specifically, it excludes night (`X` = 3) and postnight (`X` = 4) shifts from this category.

Rule $r_4$ introduces a synthetic "long shift" of 12 hours, identified by `ID` 7 and labeled as `long`, which is also available to outpatient-only nurses. This shift models a combined morning and afternoon shift.

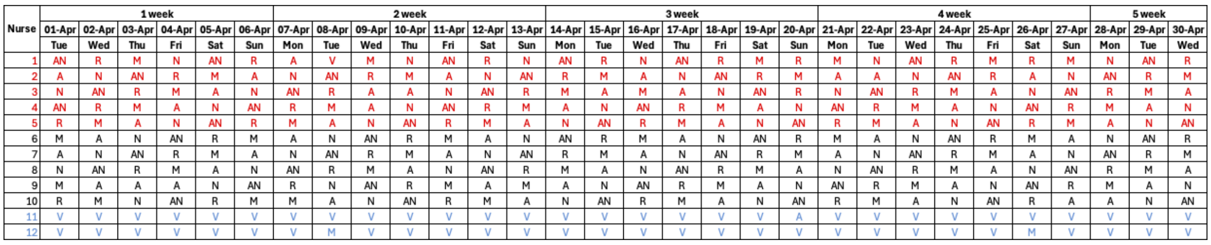
Rule $r_5$ states that, for each clinic-only nurse `X`, exactly one clinic-compatible shift must be assigned on each day `D` when the clinic is open.

Rule $r_6$ enforces that, for each clinic-open day `D`, the number of assigned nurses (counting both normal and long shifts) must exactly match the required number `N` specified for shift type `T`. This is a global cardinality constraint for staff coverage.

Rule $r_7$ limits the total number of hours worked by non-reserve outpatient-only nurses in each open clinic week `W`. The rule ensures that the sum of the durations `H` of assigned shifts does not exceed 36 hours. Note that this requirement is automatically enforced for nurses assigned to the inpatient department, due to the use of cyclic rotation. Moreover, for nurses on reserve, this constraint is relaxed, as they are deployed to fill staffing gaps as needed. However, in practice, it is unlikely that they will exceed the allowed number of working hours.

Rule $r_8$ is a weak constraint that penalizes the use of reserve nurses for any shift other than vacation (shift 5). The solver attempts to minimize such occurrences, preferring solutions where reserves are used only when strictly necessary.

Rule $r_9$ and rule $r_{10}$ are used to project the final output of the program. They collect all assignments, whether produced by the rotation-based logic or not, under the atoms of the form `result`$(\cdot, \cdot, \cdot)$.



| Nurse | 1 week | | | | | | 2 week | | | | | | | 3 week | | | | | | | 4 week | | | | | | | 5 week | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 01-Apr | 02-Apr | 03-Apr | 04-Apr | 05-Apr | 06-Apr | 07-Apr | 08-Apr | 09-Apr | 10-Apr | 11-Apr | 12-Apr | 13-Apr | 14-Apr | 15-Apr | 16-Apr | 17-Apr | 18-Apr | 19-Apr | 20-Apr | 21-Apr | 22-Apr | 23-Apr | 24-Apr | 25-Apr | 26-Apr | 27-Apr | 28-Apr | 29-Apr | 30-Apr |
| | Tue | Wed | Thu | Fri | Sat | Sun | Mon | Tue | Wed | Thu | Fri | Sat | Sun | Mon | Tue | Wed | Thu | Fri | Sat | Sun | Mon | Tue | Wed | Thu | Fri | Sat | Sun | Mon | Tue | Wed |
| 1 | AN | R | M | N | AN | R | A | V | M | N | AN | R | N | AN | R | N | AN | R | M | R | M | N | AN | R | M | R | M | N | AN | R |
| 2 | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | A | N | AN | R | A | N | AN | R | M |
| 3 | N | AN | R | M | A | N | AN | R | A | A | N | AN | R | M | A | M | A | N | AN | R | N | AN | R | M | A | N | AN | R | M | A |
| 4 | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N |
| 5 | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN |
| 6 | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R |
| 7 | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M |
| 8 | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A |
| 9 | M | A | A | A | N | AN | R | N | AN | R | M | A | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N |
| 10 | R | M | N | AN | R | M | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | M | A | N | AN | R | A | A | N | AN |
| 11 | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | A | V | V | V | V | V | V | V | V | V | V |
| 12 | V | V | V | V | V | V | V | M | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | V | M | V | V | V | V |

**Figure 3:** Example of a possible solution for the NSP variant considered in Annunziata hospital.

Finally, the directive at line 11 instructs the ASP solver to display only the `result` atoms in the output, which represent the final schedule.

## 3.2. Annunziata

Concerning the Annunziata hospital, the data model is similar of the one for the Mariano Santo Hospital, but we consider the additional atoms of the form `unavailable(N,D,S)`, meaning that the nurse `N` cannot be assigned to the shift `S` during the day `D`. The following is a set of the input facts provided to the problem.

```
shift(1, morning, 6). shift(2, afternoon, 6). shift(3, night, 4).
shift(4, post-night, 8). shift(5, vacation, 0). shift(6, rest, 0).
day(1..30). nurse(1..12). reserve(11..12). cycle_length(5).
week(1,1..6). week(2,7..13). week(3,14..20). week(4,21..27). week(5,28..30).
nurse_rotation(1, 0, 1). nurse_rotation(2, 1, 2). nurse_rotation(3, 2, 3).
nurse_rotation(4, 3, 4). nurse_rotation(5, 4, 6).
nurse_rotation(6, 0, 1). nurse_rotation(7, 1, 2). nurse_rotation(8, 2, 3).
nurse_rotation(9, 3, 4). nurse_rotation(10, 4, 6).
required_staff_per_shift(1..4, 2).
```

Note that atoms of the form `unavailable(N,D,S)` are optional and depend on the work schedule of the nurse's spouse.

An example of the computed schedule is reported in Figure 3. As in the previous case, `M`, `A`, `N`, `AN`, `V`, `R`, `L` stand for `morning`, `afternoon`, `night`, `postnight`, `vacation`, `rest`, and `long`, respectively.

The ASP encoding used in this hospital is reported in Figure 4 and described in the following.

Rule $r_1$ is used to assign each nurse to a shift on day 1 based on their initial position in the predefined rotation cycle, as specified by the predicate `nurse_rotation/3`.

Rule $r_2$ ensures the daily progression of nurse assignments according to the rotation: after working shift `S` at day `D` with order `Ord`, a nurse moves to the next shift `NS` corresponding to the incremented order `Ord2`, cycling back according to the predefined cycle length.

Rule $r_3$ defines a derived predicate `result/3`, which records the assignment of nurse `N` to shift `SID` on day `D`, abstracting away the rotation order information.

Rule $r_4$ imposes that exactly one shift must be assigned to each nurse on each day, ensuring that each nurse has precisely one work assignment per day.

Rule $r_5$ enforces the constraint that if a nurse is assigned a night shift (shift 3) on day `D`, they must be assigned a post-night recovery shift (shift 4) on day `D+1`.

Rule $r_6$ ensures that if a nurse is assigned a post-night shift (shift 4) on day `D`, then the previous day `D-1` must have been a night shift (shift 3).

Rule $r_7$ requires that after a post-night shift (shift 4) on day `D`, the nurse must be assigned a rest day (shift 6) on day `D+1`, enforcing proper recovery after night work.

Rule $r_8$ verifies that for every day `D` and for every shift `S`, the number of nurses assigned matches the required staffing levels, as specified by `required_staff_per_shift/2`.

Rule $r_9$ introduces a weekly workload constraint: for each nurse `N` and each week `W`, the sum of the

```
1  assign(N,1,S,Ord) :- nurse(N), nurse_rotation(N,Ord,S).

2  assign(N,D+1,NS,Ord2) :- assign(N,D,S,Ord), Ord2 = (Ord+1)\Z, cycle_length(Z),
       nurse_rotation(_,Ord2,NS), day(D+1).

3  result(N,D,SID) :- assign(N,D,SID,_).

4  {assign(N,D,S) : shift(S,_,_)} = 1 :- nurse(N), day(D).

5  :- assign(N,D,3), not assign(N,D+1, 4), day(D+1).
6  :- assign(N,D,4), not assign(N,D-1, 3), day(D-1).
7  :- assign(N,D,4), not assign(N,D+1, 6), day(D+1).

8  :- day(D), shift(S,_,_), required_staff_per_shift(S, V), #count{N : assign(N,D,S)} != V.
9  :- nurse(N), not reserve(N), week(W,_), #sum{H,D : assign(N,D,S), week(W,D),
       shift(S,_,H)} > 36.

10 :- unavailable(N,D,S), assign(N,D,S).

11 :- nurse(N), #count{D: assign(N,D,3)} > 6.

12 :~ not assign(N,D,S), result(N,D,S). [1@1,N,D]
13 :~ reserve(N), day(D), not assign(N,D,5). [1@2,N,D]

14 #show assign/3.
```

**Figure 4:** ASP Encoding for the Annunziata Hospital.

hourly workload associated with their assigned shifts must not exceed 36 hours. As in the Mariano Santo requirements, for nurses on reserve, this constraint is relaxed, as they are deployed to fill staffing gaps as needed.

Rule $r_{10}$ prohibits assignments that conflict with individual unavailabilities.

Rule $r_{11}$ enforces a limit on the number of night shifts assigned to each nurse. Specifically, it prohibits any answer set in which a nurse `N` is assigned to the night shift (identified as shift 3) on more than 6 days.

Rule $r_{12}$ defines a weak constraint that penalizes missing assignments predicted by the previous solution (captured by `result/3`), guiding the solver towards maintaining continuity with the cyclic plan.

Rule $r_{13}$ defines another weak constraint penalizing the failure to assign a reserve nurse to vacation (shift 5) on a day, encouraging the use of reserves mainly for covering absences.

Finally, the directive `#show assign/3` specifies that only the nurse-day-shift assignments should be displayed in the output.

## 4. Experiments

In this section, we present the results of the experiments conducted on real-world data from the two hospitals. The experimental analysis follows three main directions. The first one concerns changing the heuristic options of the considered solvers: `auto`, `frumpy`, `jumpy`, `handy`, `crafty`, `trendy`, `many` for CLINGO v.5.7.1 [17]; and `moms`, `binary`, `watches`, `combination` for WASP v.3.0.3 [20]. These heuristic parameters influence how the solver performs the different operations during the computation of an answer set, as variable selection, branching decisions, and restart policies. In general, no single heuristic configuration is expected to perform optimally across all problem domains. Instead, certain configurations tend to be more effective for specific classes of problems, making the choice of heuristics

| **Configuration** | **Time (s)** |
|---|---|
| CLINGO | 69.161 |
| CLINGO `--configuration=auto` | 69.085 |
| CLINGO `--configuration=frumpy` | >300 |
| CLINGO `--configuration=jumpy` | 90.373 |
| CLINGO `--configuration=handy` | 3.612 |
| CLINGO `--configuration=crafty` | >300 |
| CLINGO `--configuration=trendy` | 6.845 |
| CLINGO `--configuration=many` | 69.189 |
| CLINGO `--opt-strategy=usc` | 3.316 |
| CLINGO `--opt-strategy=usc --configuration=auto` | 3.299 |
| CLINGO `--opt-strategy=usc --configuration=frumpy` | **1.904** |
| CLINGO `--opt-strategy=usc --configuration=jumpy` | 3.956 |
| CLINGO `--opt-strategy=usc --configuration=handy` | 4.927 |
| CLINGO `--opt-strategy=usc --configuration=crafty` | 2.053 |
| CLINGO `--opt-strategy=usc --configuration=trendy` | 2.554 |
| CLINGO `--opt-strategy=usc --configuration=many` | 3.315 |
| CLINGO `--parallel-mode=2` | 3.263 |
| CLINGO `--parallel-mode=4` | 4.604 |
| CLINGO `--parallel-mode=8` | 5.944 |
| WASP | 10.393 |
| WASP `--init-strategy=moms` | 8.757 |
| WASP `--init-strategy=binary` | **8.744** |
| WASP `--init-strategy=watches` | 10.560 |
| WASP `--init-strategy=combination` | 12.324 |
| WASP `--weakconstraints-algorithm=basic` | 17.856 |
| WASP `--weakconstraints-algorithm=basic --init-strategy=moms` | 14.591 |
| WASP `--weakconstraints-algorithm=basic --init-strategy=binary` | 14.288 |
| WASP `--weakconstraints-algorithm=basic --init-strategy=watches` | 15.540 |
| WASP `--weakconstraints-algorithm=basic --init-strategy=combination` | 16.736 |

**Table 1**
Execution times (in seconds) for various CLINGO and WASP configurations on the instance from Mariano Santo hospital. Best results are highlighted in bold.

a critical factor in solver performance [21].

The second one consists of modifying the algorithms used to process weak constraints. Before describe the main configurations, it is important to briefly recall the main algorithms used by the solvers to compute optimal stable models. Indeed, weak constraints in ASP are typically handled using either model-guided or unsatisfiable core-guided optimization algorithms. The two approaches differ fundamentally in how they search for optimal solutions.

Model-guided algorithms incrementally explore the space of stable models, attempting to find successively better solutions by enforcing stricter bounds on the cost. These algorithms are conceptually simple and can quickly find feasible (suboptimal) models, but they often struggle to prove optimality, especially in instances with a large or complex cost structure. In WASP, this approach is activated by selecting the basic weak constraint algorithm using the option `--weakconstraints-algorithm=basic`. In CLINGO, model-guided optimization is used by default when the `--opt-strategy` option is not explicitly specified.

Unsatisfiable core-guided algorithms, on the other hand, operate by iteratively identifying unsatisfiable subsets (cores) of the problem that prevent further cost improvements. Each core represents a conflict that must be resolved to achieve a lower cost, and by systematically eliminating such cores, the solver converges toward the optimal solution. This strategy tends to be significantly more effective in proving optimality. In CLINGO, core-guided optimization is enabled by setting `--opt-strategy=usc`. In WASP, this is the default optimization strategy and is used whenever `--weakconstraints-algorithm=basic` is not specified. We refer the reader to [22] for an explanation and a comparison of the different algorithms.

Finally, the third direction is about the effect of parallelism in CLINGO (as WASP does not support parallel solving). When executed in parallel mode, CLINGO assigns a distinct configuration to each thread, both in terms of search heuristics and weak constraint optimization strategies. This means that each solver thread explores the search space using a different combination of parameters, such as decision heuristics, restart policies, and optimization techniques (e.g., model-guided vs. unsatisfiable core-guided). This design increases the diversity of the search and enhances the likelihood of quickly identifying either high-quality feasible solutions or proving optimality. The parallel mode can be activated via the option `--parallel-mode=n`, where `n` specifies the number of threads to run concurrently. In our experiments, we executed CLINGO with 2, 4, and 8 threads (`--parallel-mode=2`, `--parallel-mode=4`, and `--parallel-mode=8`, respectively).

All experiments were executed on a MacBook Air equipped with an Apple M2 processor and 24 GB of RAM. A timeout of 5 minutes was imposed for each run. Configurations that exceeded the timeout are marked with a >300 indication in the table.

## 4.1. Mariano Santo

To evaluate the computational performance of our ASP encoding, we conducted an experiment on a single instance related to the scheduling problem for the month of April at the Mariano Santo clinic. The specific instance considered is reported in Section 3.1.1. The instance models a planning horizon of 30 days, corresponding to the month of April, with the clinic being closed during weekends, on Easter Monday (April 21), and on April 25, which is a national holiday in Italy.

Table 1 reports the execution time required to find an optimal solution using various configurations of the solvers CLINGO and WASP. It is important to note that a feasible solution was found within a few seconds across all configurations; the reported times concern the effort necessary to guarantee optimality.

Execution times vary significantly across configurations. As a general observation, algorithms based on unsatisfiable cores achieve the best performance in both CLINGO and WASP, with CLINGO configured with `frumpy` is able to find the optimal solution in less than 2 seconds, and WASP configured with the init strategy set to `binary` is able to solve the instance in approximately 9 seconds. For CLINGO, all the configurations are able to terminate within 5 seconds when the algorithm based on unsatisfiable cores is enabled. Instead, the default algorithm obtains mixed results, as there are two cases (configurations FRUMPY and CRAFTY) where it is not able to terminate within 5 minutes. The best configurations in this case are HANDY and TRENDY. Moreover, parallel solving in CLINGO is also quite efficient, particularly with 2 threads.

For WASP, solving times were generally below 20 seconds across all configurations. The `binary` and `moms` initialization strategies resulted in slightly faster solving times compared to the default, while the use of the `basic` weak constraint algorithm increased the solving time, as seen in CLINGO.

## 4.2. Annunziata

Table 2 reports the execution times for the Annunziata hospital instance, evaluating the same solver configurations as in the previous experiment. This instance corresponds to the dataset described in Section 3.2, where we simulate the unavailability of nurse 1 using atoms of the form `unavailable(N,D,S)`. Specifically, the unavailability data were derived from the real work schedule of nurse 1's spouse, ensuring realistic constraints in the generated instance.

An interesting trend emerges from the results: configurations based on unsatisfiable core algorithms (i.e., those using the `--opt-strategy=usc` flag in CLINGO or the default algorithm in WASP) consistently solve the instance in less than two seconds. These include all CLINGO configurations combined with `usc` as well as WASP with or without alternative initialization strategies. In particular, the fastest CLINGO configuration was `usc` with `crafty` (0.703s), while the versions of wasp perform approximately the same, with a slight advantage when the init strategy is set to `watches` (1.127s).

| **Configuration** | **Time (s)** |
|---|---|
| CLINGO | >300 |
| CLINGO `--configuration=auto` | >300 |
| CLINGO `--configuration=frumpy` | >300 |
| CLINGO `--configuration=jumpy` | >300 |
| CLINGO `--configuration=handy` | >300 |
| CLINGO `--configuration=crafty` | >300 |
| CLINGO `--configuration=trendy` | >300 |
| CLINGO `--configuration=many` | >300 |
| CLINGO `--opt-strategy=usc` | 1.274 |
| CLINGO `--opt-strategy=usc --configuration=auto` | 1.274 |
| CLINGO `--opt-strategy=usc --configuration=frumpy` | 1.772 |
| CLINGO `--opt-strategy=usc --configuration=jumpy` | 0.861 |
| CLINGO `--opt-strategy=usc --configuration=handy` | 1.237 |
| CLINGO `--opt-strategy=usc --configuration=crafty` | 0.703 |
| CLINGO `--opt-strategy=usc --configuration=trendy` | 1.299 |
| CLINGO `--opt-strategy=usc --configuration=many` | 1.274 |
| CLINGO `--parallel-mode=2` | 0.526 |
| CLINGO `--parallel-mode=4` | 0.293 |
| CLINGO `--parallel-mode=8` | **0.248** |
| WASP | 1.463 |
| WASP `--init-strategy=moms` | 1.147 |
| WASP `--init-strategy=binary` | 1.154 |
| WASP `--init-strategy=watches` | **1.127** |
| WASP `--init-strategy=combination` | 1.193 |
| WASP `--weakconstraints-algorithm=basic` | >300 |
| WASP `--weakconstraints-algorithm=basic --init-strategy=moms` | >300 |
| WASP `--weakconstraints-algorithm=basic --init-strategy=binary` | >300 |
| WASP `--weakconstraints-algorithm=basic --init-strategy=watches` | >300 |
| WASP `--weakconstraints-algorithm=basic --init-strategy=combination` | >300 |

**Table 2**
Execution times (in seconds) for various CLINGO and WASP configurations on the instance from Mariano Santo hospital. Best results are highlighted in bold.

In contrast, all model-guided optimization approaches, such as those enabled by the default configurations of `usc` or WASP using the `basic` algorithm for weak constraints, failed to find the optimal solution within the timeout. However, like in the previous experiment, these configurations were still able to find a feasible (but not optimal) solution relatively quickly.

Additionally, we observe that enabling parallel solving in CLINGO also lead to good performance. The best result was achieved with 4 threads (`--parallel-mode=8`), reaching optimality in 0.248 seconds, confirming the benefit of concurrent search in core-based optimization.

Overall, this experiment confirms that for complex, highly constrained instances such as the Annunziata scheduling problem, unsatisfiable core-based algorithms dominate in performance, both in CLINGO and WASP. In contrast, model-guided techniques appear unsuitable for this instance when optimality is required within tight time constraints.

## 5. Related Work

In recent years, a wide range of approaches have been proposed to tackle the NSP, each characterized by different assumptions and constraints. The primary distinctions among these works concern *(i)* the length of the planning horizon, *(ii)* the modeling of shifts, *(iii)* the staffing requirements per shift, and (iv) the application of institutional or contractual constraints on nurses' schedules. A detailed review of these dimensions is provided in [3].

The variant of NSP presented in this work considers a one-month scheduling window, which is aligned with the common planning periods adopted in operational practice, although some previous studies, such as [23], have focused on longer horizons, including annual schedules. Unlike some models that consider homogeneous or overlapping shifts, our approach explicitly distinguishes between morning, afternoon, and night shifts, with strict non-overlapping constraints. In contrast, early formulations of the problem often assumed a single type of shift or less granular temporal divisions [24].

Several computational approaches have been investigated for NSP, ranging from exact methods to heuristic and metaheuristic techniques. Integer programming models remain a foundational strategy, as seen in works such as [25, 26, 27]. Metaheuristic approaches have also proven effective, including genetic algorithms [28], fuzzy logic-based formulations [29], and ant colony optimization [30].

More recently, attention has also been directed toward the use of declarative and hybrid methods for nurse scheduling. For example, [31] explores a model of the Hybrid Salp Swarm Algorithm and Genetic Algorithm (HSSAGA). In the context of workforce well-being, studies such as [1], [2], [32], and [33] emphasize the impact of schedule quality on nurse satisfaction, performance and retention, highlighting the importance of fairness and compatibility with personal life, including family constraints.

With respect to the use of ASP for solving the NSP, [15] proposed an ASP encoding tailored to a real-world scenario described by an Italian hospital. Some of the constraints addressed in their work, such as the number of nurses required per shift and the general organization of shift types, are similar to those considered in this paper. However, their approach targets a one-year planning horizon and does not incorporate cyclic rotation constraints as we do here.

Building on that work, [16] introduced an enhanced version of the encoding and developed a rescheduling framework capable of adapting to unexpected changes in staff availability, thus enabling more flexible shift management in dynamic environments.

For a comprehensive overview of the field, we refer the reader to the surveys [3, 4, 34, 35].

## 6. Conclusions

In this paper, we presented an ASP-based approach to solve two variants of the NSP, focusing on real-case scenarios from Italian healthcare facilities. Our encodings capture both common scheduling requirements and domain-specific constraints, demonstrating how ASP can be adapted to reflect diverse organizational practices. The experimental analysis highlights a clear advantage of unsatisfiable core-guided optimization strategies, which achieved optimal solutions in a fraction of the time required by traditional model-guided approaches. In particular, CLINGO and WASP showed great performance when appropriately configured, solving complex instances in a few seconds. These results suggest that ASP is a viable and efficient solution for practical nurse rostering tasks, provided that appropriate encoding techniques and solver configurations are adopted. Future work includes extending the approach to additional departments within the two hospitals, where different scheduling preferences may apply and the number of nurses to manage would significantly increase. Another promising direction is the development of a web-based application to support scheduling managers in interactively selecting the constraints to enforce. Moreover, the experimental evaluation could be expanded to cover longer time horizons, such as multiple months, to better assess scalability and robustness. Finally, we aim to explore approaches for recomputing the plan when it cannot be followed due to unexpected events, such as last-minute absences or emergency reassignments.

## Acknowledgments

Carmine Dodaro and Marco Maratea were supported by the European Union - NextGenerationEU and by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.5, project “RAISE - Robotics and AI for Socio-economic Empowerment” (ECS00000035) under the project “Gestione e Ottimizzazione di Risorse Ospedaliere attraverso Analisi Dati, Logic Programming e Digital Twin (GOLD)”, CUP H53C24000400006.

## Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-4o in order to: Grammar and spelling check. After using these tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

## References

[1] L. Yu, H. Zhou, J. Li, X. Yu, Shift work sleep disorder in nurses: a concept analysis, BMC Nursing 24 (2025). URL: https://doi.org/10.1186/s12912-024-02651-z. doi:10.1186/s12912-024-02651-z.

[2] L. Lessi, I. de Barbieri, M. Danielis, Addressing nursing resignation: Insights from qualitative studies on nurses leaving healthcare organisations and the profession, Journal of Advanced Nursing 81 (2025) 2290 – 2315. URL: https://doi.org/10.1111/jan.16546. doi:10.1111/jan.16546.

[3] E. K. Burke, P. D. Causmaecker, G. V. Berghe, H. V. Landeghem, The state of the art of nurse rostering, J. Scheduling 7 (2004) 441–499. URL: https://doi.org/10.1023/B:JOSH.0000046076.75950.0b. doi:10.1023/B:JOSH.0000046076.75950.0b.

[4] B. Cheang, H. Li, A. Lim, B. Rodrigues, Nurse rostering problems - a bibliographic survey, European Journal of Operational Research 151 (2003) 447–460. URL: https://doi.org/10.1016/S0377-2217(03)00021-3. doi:10.1016/S0377-2217(03)00021-3.

[5] M. Gelfond, V. Lifschitz, Classical Negation in Logic Programs and Disjunctive Databases, New Generation Comput. 9 (1991) 365–386.

[6] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (2011) 92–103. URL: http://doi.acm.org/10.1145/2043174.2043195. doi:10.1145/2043174.2043195.

[7] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Mag. 37 (2016) 53–68. URL: https://doi.org/10.1609/aimag.v37i3.2678. doi:10.1609/AIMAG.V37I3.2678.

[8] P. Schüller, Answer set programming in linguistics, Künstliche Intell. 32 (2018) 151–155. URL: https://doi.org/10.1007/s13218-018-0542-z. doi:10.1007/S13218-018-0542-Z.

[9] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, Künstliche Intell. 32 (2018) 165–176. URL: https://doi.org/10.1007/s13218-018-0548-6. doi:10.1007/S13218-018-0548-6.

[10] A. D. Palù, A. Dovier, A. Formisano, E. Pontelli, ASP applications in bio-informatics: A short tour, Künstliche Intell. 32 (2018) 157–164. URL: https://doi.org/10.1007/s13218-018-0551-y. doi:10.1007/S13218-018-0551-Y.

[11] E. Erdem, V. Patoglu, Applications of ASP in robotics, Künstliche Intell. 32 (2018) 143–149. URL: https://doi.org/10.1007/s13218-018-0544-x. doi:10.1007/S13218-018-0544-X.

[12] T. C. Son, M. Balduccini, Answer set planning in single- and multi-agent environments, Künstliche Intell. 32 (2018) 133–141. URL: https://doi.org/10.1007/s13218-018-0546-8. doi:10.1007/S13218-018-0546-8.

[13] P. Cappanera, S. Caruso, C. Dodaro, G. Galatà, M. Gavanelli, M. Maratea, C. Marte, M. Mochi, M. Nonato, M. Roma, Recent answer set programming applications to scheduling problems in digital health, in: AI4CC-IPS-RCRA-SPIRIT, volume 3883 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3883/paper3_RCRA8.pdf.

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

[15] C. Dodaro, M. Maratea, Nurse scheduling via answer set programming, in: LPNMR, volume 10377 of *LNCS*, Springer, 2017, pp. 301–307. URL: https://doi.org/10.1007/978-3-319-61660-5_27. doi:10.1007/978-3-319-61660-5_27.

[16] M. Alviano, C. Dodaro, M. Maratea, Nurse (re)scheduling via answer set programming, Intelligenza Artificiale 12 (2018) 109–124. URL: https://doi.org/10.3233/IA-170030. doi:10.3233/IA-170030.

[17] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: ICLP TCs, volume 52 of *OASICS*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 2:1–2:15. URL: https://doi.org/10.4230/OASIcs.ICLP.2016.2. doi:10.4230/OASIcs.ICLP.2016.2.

[18] M. Alviano, C. Dodaro, N. Leone, F. Ricca, Advances in WASP, in: LPNMR, volume 9345 of *LNCS*, Springer, 2015, pp. 40–54. URL: https://doi.org/10.1007/978-3-319-23264-5_5. doi:10.1007/978-3-319-23264-5_5.

[19] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, T. Schaub, Asp-core-2 input language format, Theory Pract. Log. Program. 20 (2020) 294–309. URL: https://doi.org/10.1017/S1471068419000450. doi:10.1017/S1471068419000450.

[20] M. Alviano, G. Amendola, C. Dodaro, N. Leone, M. Maratea, F. Ricca, Evaluation of disjunctive programs in WASP, in: LPNMR, volume 11481 of *LNCS*, Springer, 2019, pp. 241–255. URL: https://doi.org/10.1007/978-3-030-20528-7_18. doi:10.1007/978-3-030-20528-7_18.

[21] C. Dodaro, Design and implementation of modern CDCL ASP solvers, Intelligenza Artificiale 18 (2024) 239–259. URL: https://doi.org/10.3233/IA-240019. doi:10.3233/IA-240019.

[22] M. Alviano, C. Dodaro, J. Marques-Silva, F. Ricca, Optimum stable model search: algorithms and implementation, J. Log. Comput. 30 (2020) 863–897. URL: https://doi.org/10.1093/logcom/exv061. doi:10.1093/LOGCOM/EXV061.

[23] P. Chan, G. Weil, Cyclical staff scheduling using constraint logic programming, in: PATAT, volume 2079 of *LNCS*, Springer, 2000, pp. 159–175. URL: https://doi.org/10.1007/3-540-44629-X_10. doi:10.1007/3-540-44629-X_10.

[24] H. E. Miller, W. P. Pierskalla, G. J. Rath, Nurse scheduling using mathematical programming, Operations Research 24 (1976) 857–870. URL: https://doi.org/10.1287/opre.24.5.857. doi:10.1287/opre.24.5.857.

[25] M. N. Azaiez, S. S. A. Sharif, A 0-1 goal programming model for nurse scheduling, Computers & OR 32 (2005) 491–507. URL: https://doi.org/10.1016/S0305-0548(03)00249-1. doi:10.1016/S0305-0548(03)00249-1.

[26] J. F. Bard, H. W. Purnomo, Preference scheduling for nurses using column generation, European Journal of Operational Research 164 (2005) 510–534. URL: https://doi.org/10.1016/j.ejor.2003.06.046. doi:10.1016/j.ejor.2003.06.046.

[27] M. Narlı, O. Derse, Optimal crew scheduling in an intensive care unit: A case study in a university hospital, Applied Sciences (Switzerland) 15 (2025). URL: https://doi.org/10.3390/app15073610. doi:10.3390/app15073610.

[28] U. Aickelin, K. A. Dowsland, An indirect genetic algorithm for a nurse-scheduling problem, Computers & OR 31 (2004) 761–778. URL: https://doi.org/10.1016/S0305-0548(03)00034-0. doi:10.1016/S0305-0548(03)00034-0.

[29] S. Topaloglu, H. Selim, Nurse scheduling using fuzzy modeling approach, Fuzzy Sets and Systems 161 (2010) 1543–1563. URL: http://dx.doi.org/10.1016/j.fss.2009.10.003. doi:10.1016/j.fss.2009.10.003.

[30] W. J. Gutjahr, M. S. Rauner, An ACO algorithm for a dynamic regional nurse-scheduling problem in austria, Computers & OR 34 (2007) 642–666. URL: https://doi.org/10.1016/j.cor.2005.03.018. doi:10.1016/j.cor.2005.03.018.

[31] M. Q. H. Abadi, S. Rahmati, A. Sharifi, M. Ahmadi, HSSAGA: designation and scheduling of nurses for taking care of COVID-19 patients using novel method of hybrid salp swarm algorithm and genetic algorithm, Appl. Soft Comput. 108 (2021) 107449. URL: https://doi.org/10.1016/j.asoc.2021.107449. doi:10.1016/J.ASOC.2021.107449.

[32] A. Mystakidis, C. Koukaras, P. Koukaras, K. Kaparis, S. G. Stavrinides, C. Tjortjis, Optimizing nurse rostering: A case study using integer programming to enhance operational efficiency and care quality, Healthcare (Switzerland) 12 (2024). URL: https://doi.org/10.3390/healthcare12242545. doi:10.3390/healthcare12242545.

[33] H. Wynendaele, E. Peeters, P. Gemmel, D. Myny, J. Trybou, Unravelling the ideal roster: A cross-sectional study of nurse shift preferences using multivariate analysis, Journal of Advanced

Nursing 81 (2025) 1829 – 1844. URL: https://doi.org/10.1111/jan.16373. doi:10.1111/jan.16373.

[34] C. M. Ngoo, S. L. Goh, S. Sze, N. R. Sabar, S. Abdullah, G. Kendall, A survey of the nurse rostering solution methodologies: The state-of-the-art and emerging trends, IEEE Access 10 (2022) 56504–56524. URL: https://doi.org/10.1109/ACCESS.2022.3177280. doi:10.1109/ACCESS.2022.3177280.

[35] D. Allen, H. Strange, N. Jacob, A. M. Rafferty, How can we optimise nurse staffing systems? insights from a comparative document analysis of 10 widely used models and focused interpretative review of implementation experiences, International Journal of Nursing Studies 167 (2025). URL: https://doi.org/10.1016/j.ijnurstu.2025.105056. doi:10.1016/j.ijnurstu.2025.105056.