An Advanced Answer Set Programming Encoding for Nurse Scheduling

Mario Alviano[1], Carmine Dodaro[2(✉)], and Marco Maratea[2]

[1] DEMACS, University of Calabria, Rende, Italy
alviano@mat.unical.it
[2] DIBRIS, University of Genova, Genoa, Italy
{dodaro,marco}@dibris.unige.it

Abstract. The goal of the Nurse Scheduling Problem (NSP) is to find an assignment of nurses to shifts according to specific requirements. Given its practical relevance, many researchers have developed different strategies for solving several variants of the problem. One of such variants was recently addressed by an approach based on Answer Set Programming (ASP), obtaining promising results. Nonetheless, the original ASP encoding presents some intrinsic weaknesses, which are identified and eventually circumvented in this paper. The new encoding is designed by taking into account both intrinsic properties of NSP and internal details of ASP solvers, such as cardinality and weight constraint propagators. The performance gain of CLINGO and WASP is empirically verified on instances from ASP literature. As an additional contribution, the performance of CLINGO and WASP is compared to other declarative frameworks, namely SAT and ILP; the best performance is obtained by CLINGO running the new ASP encoding.

Keywords: Answer Set Programming · Knowledge representation and reasoning · Nurse Scheduling

1 Introduction

The Nurse Scheduling Problem (NSP) consists of generating a schedule of working and rest days for nurses working in hospital units. The schedule should determine the shift assignments of nurses for a predetermined window of time and must satisfy requirements imposed by the Rules of Procedure of hospitals. A proper solution to NSP is crucial to guarantee the high level of quality of health care, to improve the degree of satisfaction of nurses, and the recruitment of qualified personnel. Given its practical relevance on the quality of hospital structures, NSP has been widely studied in the literature and several variants have been considered [14,18]. Such variants are usually grouped according to several factors, as the planning period, the different types of shifts considered, and requirements on the preferences of hospitals and nurses.

The NSP variant considered in this paper concerns a planning period fixed to one year with three different types of shifts (morning, afternoon and night) and

F. Esposito et al. (Eds.): AI*IA 2017, LNAI 10640, pp. 468–482, 2017.
https://doi.org/10.1007/978-3-319-70169-1_35

requirements on nurses and hospitals provided by an Italian hospital. Specifically, such requirements concern restrictions to the number of working hours per year and to the number of times nurses are assigned to a specific shift.

Recently, the aforementioned variant of NSP has been modeled by means of an Answer Set Programming (ASP) [13] encoding presented in [21]. The encoding resulted to be natural and intuitive, in the sense that it was designed by applying the standard modeling methodology, yet it obtained reasonable performance on solving the analyzed instances.

On the other hand, it turned out that the encoding presented in [21] shows some limitations and intrinsic weaknesses, mainly due to *aggregates* [7], i.e. operations on multi-sets of weighted literals that evaluate to some value. The encoding of [21] presents some aggregates with a quite large number of literals and few different weights, resulting to be counterproductive for the performance of modern ASP solvers [27], since they deteriorate their propagation power. In this paper, we circumvented such limitations by taking into account only combinations of values that can lead to admissible schedules. Interestingly, the new encoding did not require to significantly sacrifice the readability of the encoding, which remains intuitive and clear.

The performance of the ASP solvers executed on the new encoding has been empirically evaluated on the same data and settings used in [21], showing a clear improvement on the performance of state of the art ASP systems CLINGO [26] and WASP [5]. As an additional contribution, the ASP-based approaches have been compared to other declarative frameworks, namely Propositional Logic Satisfiability (SAT) [8] and Integer Linear Programming (ILP) [1]. Results show that CLINGO and WASP executed on the new encoding outperform their counterparts executed on the original encoding. Moreover, CLINGO executed on the new encoding is considerably faster than all other tested approaches.

The contributions of the paper can be summarized as follows:

1. We formalize the variant of NSP considered in this paper and in [21] (Sect. 2.2).
2. We propose a new ASP-based solution to NSP overcoming some limitations of the encoding presented in [21] (Sect. 3.3).
3. We present an experimental analysis comparing the ASP solution proposed in this paper with the previous one as well as with SAT and ILP based solutions (Sect. 4). Results show a significant improvement of the performance of ASP solvers and, specifically, CLINGO performs better than all other alternatives.

2 Description and Formalization

We start this section by providing a description of the problem as posed by an Italian hospital (Sect. 2.1). In the description we identify parameters that allow to reuse the proposed solution even if part of the specification given by the hospital will change. The formalization presented in Sect. 2.2 in fact considers these parameters as part of the input.

2.1 Nurse Scheduling Problem

NSP amounts to the totalization of partial schedules assigning nurses to working and rest days over a predetermined period of time, which is fixed to one year in this paper. Usually, partial schedules to be totalized involve few data concerning already authorized vacations. Admissible schedules must satisfy a set of requirements dictated by the rules of the hospital units. In the following, we report the requirements specified by an Italian hospital.

Hospital requirements. For every working day, nurses can be assigned to exactly one of the following shifts: *morning* (7 A.M.–2 P.M.), *afternoon* (2 P.M.–9 P.M.), *night* (9 P.M.–7 A.M.). Thus, the morning and the afternoon shifts last 7 h, whereas the night shift lasts 10 h. In order to ensure the best assistance program for patients, the number of nurses in every shift $x \in \{morning, afternoon, night\}$ must range from x^{nurse}_{min} to x^{nurse}_{max}.

Nurses requirements. In order to guarantee a fair workload, each nurse must work a number of hours ranging from $work_{min}$ to $work_{max}$. Additional requirements are also imposed to ensure an adequate rest period to each nurse: *(a)* nurses are legally guaranteed 30 days of paid vacation; *(b)* the starting time of a shift must be at least 24 h later than the starting time of the previous shift; and *(c)* each nurse has at least two ordinary rest days for every window of fourteen days. In addition, nurses working on two consecutive nights deserve one special rest day in addition to the ordinary rest days.

Balance requirements. The number of morning, afternoon and night shifts assigned to every nurse should range over a set of acceptable values, that is, from x^{day}_{min} to x^{day}_{max} for each $x \in \{morning, afternoon, night\}$.

2.2 Formalization

According to the above requirements, we define the following decisional problem NSP^d: Given a set N of nurses, a partial schedule

$$s' : N \times [1..365] \nrightarrow \{morning, afternoon, night, special\text{-}rest, rest, vacation\} \tag{1}$$

natural numbers $work_{min}$, $work_{max}$, and x^{nurse}_{min}, x^{nurse}_{max}, x^{day}_{min}, x^{day}_{max} for $x \in \{morning, afternoon, night\}$, check the existence of a schedule

$$s : N \times [1..365] \to \{morning, afternoon, night, special\text{-}rest, rest, vacation\} \tag{2}$$

extending s' and satisfying the following conditions:

$$x^{nurse}_{min} \leq \mid \{n \in N : s(n, d) = x\} \mid \leq x^{nurse}_{max} \tag{3}$$

for all $x \in \{morning, afternoon, night\}$, and all $d \in [1..365]$;

$$\begin{aligned} work_{min} \leq \ & 7 \cdot \mid \{d \in [1..365] : s(n, d) \in \{morning, afternoon\},\ n \in N\} \mid \\ & + 10 \cdot \mid \{d \in [1..365] : s(n, d) = night,\ n \in N\} \mid \leq work_{max}; \end{aligned} \tag{4}$$

$$| \{d \in [1..365] : s(n,d) = vacation\} | = 30 \tag{5}$$

$$| \{d \in [2..365] : s(n,d) = morning,\ s(n,d-1) \in \{afternoon, night\}\} | = 0$$

$$| \{d \in [2..365] : s(n,d) = afternoon,\ s(n,d-1) = night\} | = 0 \tag{6}$$

for all $n \in N$;

$$| \{d' \in [d..d+13] : s(n,d') = rest\} | \geq 2 \tag{7}$$

for all $n \in N$, and all $d \in [1..352]$;

$$s(n,d) = special\text{-}rest \text{ if and only if } s(n,d-1) = night \text{ and } s(n,d-2) = night \tag{8}$$

for all $n \in N$, and all $d \in [3..365]$;

$$x_{min}^{day} \leq | \{d \in [1..365] : s(n,d) = x\} | \leq x_{max}^{day} \tag{9}$$

for all $n \in N$, and $x \in \{morning, afternoon, night\}$.

Optimal balance requirements. In addition to the above requirements, the hospital reported some further requirements to guarantee a balance in the assignment of shifts. Indeed, the number of morning, afternoon and night shifts assigned to every nurse should be *preferably* fixed to some desired values, that is, x^{day} for each $x \in \{morning, afternoon, night\}$.

According to the above additional requirement, we define the following *optimization* problem NSP^o: Given a set N of nurses, natural numbers $work_{min}, work_{max}$, and $x_{min}^{nurse}, x_{max}^{nurse}$, $x^{day}, x_{min}^{day}, x_{max}^{day}$ for $x \in \{morning, afternoon, night\}$, check the existence of a schedule s of the form (2) satisfying (3)–(9), and minimizing

$$\sum_{x \in \{morning, afternoon, night\},\ n \in N} abs(x^{day} - | \{d \in [1..365] : s(n,d) = x\} |). \tag{10}$$

3 ASP Encodings

In Sect. 3.3 we present the new advanced encoding, improving on the existing one introduced in [21] and briefly recalled in Sect. 3.2. We assume that the reader is familiar with basic knowledges of Answer Set Programming and ASP-Core-2 input language specification [15] (some minimal notions are given in Sect. 3.1).

3.1 ASP Evaluation Strategies

In the following a summary of the evaluation strategies of ASP programs are reported in order to provide a better insight on the properties of the new encoding. The evaluation of an ASP program is usually made in two steps, called *grounding* and *solving*. First, the ASP program with variables is evaluated by the grounder, which is responsible to produce its variable-free (propositional) counterpart.

Example 1 (Grounding). Consider as example the following rules:

```
a(1..5). b(1..10). c(1..3).
{output(X,Z,Y) : c(Z)} = 1 :- a(X), b(Y).
```

The grounder produces 50 (i.e. 5×10) propositional rules of the following form:

```
p1 :  {output(1,1,1); output(1,2,1); output(1,3,1)} = 1.
p2 :  {output(1,1,2); output(1,2,2); output(1,3,2)} = 1.
p3 :  {output(1,1,3); output(1,2,3); output(1,3,3)} = 1.
                          ⋮
```

Intuitively, the choice rule p_1 enforces that exactly one atom between `output(1,1,1)`, `output(1,2,1)` and `output(1,3,1)` must be true in an answer set. Similar considerations hold for other ground rules generated. ◁

The resulting propositional program is evaluated by the solver, whose role is to produce an answer set. Modern ASP solvers implement the algorithm CDCL [27], which is based on the pattern *choose-propagate-learn*. Intuitively, the idea is to build an answer set step-by-step by starting from an empty interpretation, i.e. all atoms are initially undefined. Then, the algorithm heuristically *chooses* an undefined atom to be true in the answer set, and the deterministic consequences of this choice are *propagated*, i.e. new atoms are derived true or false in the answer set candidate. The propagation may lead to a *conflict*, i.e. an atom is true and false at the same time. In this case, the conflict is analyzed and a new constraint is added to the propositional program (*learning*). The conflict is then repaired, i.e. choices leading to the conflict are retracted and a new undefined atom is heuristically selected. The algorithm then iterates until no undefined atoms are left, i.e. an answer set is produced, or the incoherence of the propositional program is proved, i.e. no answer sets are admitted.

Example 2 (Propagation). Consider the propositional rule p_1 reported in Example 1 and assume that atoms `output(1,1,1)` and `output(1,2,1)` have been heuristically assigned to false. Then, the solver derives `output(1,3,1)` to true because it is the only way to satisfy the rule p_1. ◁

3.2 Existing Encoding

Instances of NSP^d and NSP^o are represented by means of ASP facts and constants. Specifically, the interval [1..365] of days is encoded by facts of the form `day(`d`)`, for all $d \in$ [1..365], and the number of days is fixed by the fact `days(365)`. The nurses are encoded by facts of the form `nurse(`n`)`, for all $n \in N$. Available shifts are encoded by facts of the form `shift(`id_x, x, h`)`, where $id_x \in$ [1..6] is a numerical identifier of the shift $x \in \{morning, afternoon, night, special\text{-}rest, rest, vacation\}$, and h is the number of working hours associated to the shift. Natural numbers x_{min}^{nurse}, x_{max}^{nurse} for $x \in \{morning, afternoon, night\}$ are represented by facts of the form `nurseLimits(`$id_x, x_{min}^{nurse}, x_{max}^{nurse}$`)`, where id_x is the identifier of the shift x.

```
      % Choose an assignment for each day and for each nurse.
r1 :  {assign(N,S,D) : shift(S,Name,H)} = 1 :- day(D), nurse(N).

      % Limits to nurses that must be present for each shift.
r2 :  :- day(D), #count{N : assign(N,S,D)} > Max, nurseLimits(S,Min,Max).
r3 :  :- day(D), #count{N : assign(N,S,D)} < Min, nurseLimits(S,Min,Max).

      % Each nurse works at least Min and at most Max hours per year.
r4 :  :- nurse(N), #sum{H,D : assign(N,S,D), shift(S,Na,H)} > Max, workLimits(Min,Max).
r5 :  :- nurse(N), #sum{H,D : assign(N,S,D), shift(S,Na,H)} < Min, workLimits(Min,Max).

      % Exactly 30 days of holidays. The ID 6 corresponds to the vacation.
r6 :  :- nurse(N), #count{D : assign(N,6,D)} != 30.

      % Each nurse cannot work twice in 24 hours (based on the order on the IDs).
r7 :  :- nurse(N), assign(N,T1,D), assign(N,T2,D+1), T2 < T1, T1 <= 3.

      % At least 2 rest days each 14 days. The ID 5 is associated to rest.
r8 :  :- nurse(N), day(D), days(DAYS), D <= DAYS-13,
            #count{D1 : assign(N,5,D1), D1 >= D, D1 <= D+13} < 2.

      % After two consecutive nights there is one rest day.
      % The ID 3 is associated to the shift night, while 4 is associated to special rest.
r9 :  :- not assign(N,4,D), assign(N,3,D-2), assign(N,3,D-1).
r10 : :- assign(N,4,D), not assign(N,3,D-2).
r11 : :- assign(N,4,D), not assign(N,3,D-1).

      % Balance requirements.
r12 : :- nurse(N), #count{D : assign(N,S,D)} > Max, dayLimits(S,T,Min,Max).
r13 : :- nurse(N), #count{D : assign(N,S,D)} < Min, dayLimits(S,T,Min,Max).

      % Added only in the optimization variant.
r14 : :~ nurse(N), X=#count{D : assign(N,S,D)}, dayLimits(S,T,Min,Max),
            X >= Min, X <= Max. [|X-T|@1,N]
```

Fig. 1. ASP encoding introduced in [21] for NSP^o (and for NSP^d if r_{14} is removed).

Natural numbers $x^{day}, x^{day}_{min}, x^{day}_{max}$ for $x \in \{morning, afternoon, night\}$ are represented by `dayLimits`$(id_x, x^{day}, x^{day}_{min}, x^{day}_{max})$, while $work_{min}$, $work_{max}$ by `workLimits`$(work_{min}, work_{max})$. Hence, according to the specification given by the hospital, the following facts and constants are considered in our setting:

```
day(1..365).              days(365).               nurses(1..41).
shift(1,morning,7).       shift(2,afternoon,7).    shift(3,night,10).
shift(4,specialrest,0).   shift(5,rest,0).         shift(6,vacation,0).
nurseLimits(1,6,9).       nurseLimits(2,6,9).      nurseLimits(3,4,7).
dayLimits(1,78,74,82).    dayLimits(2,78,74,82).   dayLimits(3,60,58,61).
workLimits(1687,1692).
```

The computed schedule is encoded by atoms of the form `assign`(n, x, d), representing that nurse n is assigned shift x on day d, that is, $s(n, d) = x$. The same predicate `assign` is used to specify the partial schedule s' in input. x

The ASP encoding introduced in [21] is reported in Fig. 1. It implements the *Guess&Check* programming methodology: Choice rule r_1 is used to guess the schedule $s : N \times [1..365] \rightarrow [1..6]$ extending s' and assigning each day of each nurse to exactly one shift, and rules r_2–r_{13} are used to discard schedules not satisfying some of the desired requirements. Specifically, hospital requirements, for-

malized as property (3), are enforced by the integrity constraints r_2 and r_3, which filter out assignments exceeding the limits. Regarding nurse requirements, property (4) is enforced by integrity constraints r_4 and r_5, property (5) by integrity constraint r_6, property (6) by integrity constraint r_7, property (7) by integrity constraint r_8, and property (8) by integrity constraint r_9–r_{11}. Note that r_7 takes advantage of the numerical identifiers associated with shifts, and in particular by the fact that morning has ID 1, afternoon has ID 2, and night has ID 3. Concerning balance requirements, formalized as property (9), they are enforced by integrity constraints r_{12} and r_{14}. Rules r_1–r_{13} encode NSP^d, while for NSP^o we also need weak constraint r_{14}: It assigns a cost to each admissible schedule measured according to function (10). Optimum schedules are those minimizing such a cost.

3.3 Advanced Encoding

The aim of this section is to introduce a new encoding, shown on Fig. 2, which improves the encoding reported in the previous section. First of all, note that many constraints of the encoding in Fig. 1 only involve assignments to working shifts, that is, morning, afternoon and night. The *Guess* part of the encoding (i.e., rule r_1) can thus be replaced by two different choice rules, r'_{1a}, r'_{1b}, where r'_{1a} guesses among one of the working shifts or otherwise marks nurses as *not working*, and r'_{1b} eventually guesses among rest, special-rest and vacation for each nurse marked as not working. To achieve such a behavior, an additional *meta-shift* is added to the set of facts, namely `shift(7,notworking,0)`.

Table 1. Number of working hours assigned to nurse n, that is, $7 \cdot (M + A) + 10 \cdot N$, where $M = \mid \{d \in [1..365] : s(n, d) = morning\} \mid$, $A = \mid \{d \in [1..365] : s(n, d) = afternoon\} \mid$, and $N = \mid \{d \in [1..365] : s(n, d) = night\} \mid$. Admissible values, that is, those in the interval [1687..1692], are emphasized in bold.

N	$M + A$																
	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164
58	1616	1623	1630	1637	1644	1651	1658	1665	1672	1679	1686	1693	1700	1707	1714	1721	1728
59	1626	1633	1640	1647	1654	1661	1668	1675	1682	**1689**	1696	1703	1710	1717	1724	1731	1738
60	1636	1643	1650	1657	1664	1671	1678	1685	**1692**	1699	1706	1713	1720	1727	1734	1741	1748
61	1646	1653	1660	1667	1674	1681	**1688**	1695	1702	1709	1716	1723	1730	1737	1744	1751	1758

A second improvement is obtained by combining the knowledge represented by Eqs. (4) and (9) with some observations on how rules r_4 and r_5 are evaluated. In fact, during the solving phase, rules obtained by instantiating r_4 and r_5 comprise aggregates with relatively large aggregation sets and few different weights. Specifically to our setting, where morning and afternoon shifts are fixed to 7 h, and night shifts to 10 h, each aggregation set contains 365 elements with weight 7, and 365 elements with weight 10. It turns out that several schedules result into exactly the same sum value. The question is now how many

```
      % Choose an assignment for each day and for each nurse.
r'1a: {assign(N,S,D):shift(S,Name,H), S != 4, S != 5, S != 6} = 1 :- day(D), nurse(N).
r'1b: {assign(N,S,D):shift(S,Name,H), S >= 4, S <= 6} = 1 :- day(D), nurse(N), assign(N,7,D).

      % Limits to nurses that must be present for each shift.
r'2 : :- day(D), #count{N : assign(N,S,D)} > Max, nurseLimits(S,Min,Max).
r'3 : :- day(D), #count{N : assign(N,S,D)} < Min, nurseLimits(S,Min,Max).

      % Nurses requirements.
r'4 : valid(Nu) :- nurse(Nu), admissible(N,M+A),
            countGE(1,Nu,M), not countGE(1,Nu,M+1),
            countGE(2,Nu,A), not countGE(2,Nu,A+1),
            countGE(3,Nu,N), not countGE(3,Nu,N+1).
r'5 : :- nurse(N), not valid(N).

      % Exactly 30 days of holidays. The id 6 corresponds to the vacation.
r'6 : :- nurse(N), #count{D : assign(N,6,D)} != 30.

      % Each nurse cannot work twice in 24 hours (based on the order on the IDs).
r'7 : :- nurse(N), assign(N,T1,D), assign(N,T2,D+1), T2 < T1, T1 <= 3.

      % At least 2 rest days each 14 days. The id 5 is associated to rest.
r'8 : :- nurse(N), day(D), days(DAYS), D <= DAYS-13,
            #count{D1 : assign(N,5,D1), D1 >= D, D1 <= D+13} < 2.

      % After two consecutive nights (ID 3) there is one rest day (ID 4).
r'9 : :- not assign(N,4,D), assign(N,3,D-2), assign(N,3,D-1).
r'10: :- assign(N,4,D), not assign(N,3,D-2).
r'11: :- assign(N,4,D), not assign(N,3,D-1).

      % A nurse should be assigned to the shift S at least Min and at most Max days.
r'12: :- dayLimits(S,T,Min,Max), nurse(N), not countGE(S,N,Min).
r'13: :- dayLimits(S,T,Min,Max), nurse(N), countGE(S,N,Max+1).

      % Added only in the optimization variant. The ID 7 corresponds to notworking.
r'14: :~ nurse(N), countGE(S,N,X), not countGE(S,N,X+1),
            dayLimits(S,T,Min,Max), S != 7, V = |X-T|. [V@1,N,S]

      % Admissible pairs of nights and morning+afternoon shifts, and limits for nonworking
    days.
r'15: admissible(N,M+A) :-
            dayLimits(1,T1,MinM,MaxM), dayLimits(2,T2,MinA,MaxA), dayLimits(3,T3,MinN,MaxN),
            M = MinM..MaxM, A = MinA..MaxA, N = MinN..MaxN,
            V = 7*(M+A) + 10*N, workLimits(MinW, MaxW), MinW <= V, V <= MaxW.

r'16: dayLimits(7,null,Min,Max) :- days(DAYS), Min = #min{DAYS-MA-N : admissible(N,MA)},
            Max = #max{DAYS-MA-N : admissible(N,MA)}.

      % countGE(S,N,V) is derived when a nurse N is assigned to the shift S at least V days.
r'17: countGE(S,N,V) :- nurse(N), dayLimits(S,T,Min,Max), V = Min..Max+1,
            #count{D : assign(N,S,D)} >= V.

      % The derivation of countGE(S,N,V) implies the derivation of countGE(S,N,V-1).
r'18: :- dayLimits(S,T,Min,Max), V > Min, countGE(S,N,V), not countGE(S,N,V-1).
```

Fig. 2. Advanced ASP encoding for NSP^o (and for NSP^d if r'_{14} is removed).

of these schedules actually satisfy both (4) and (9). Restricting to the specification given by the hospital, that is, $morning^{day}_{min} = afternoon^{day}_{min} = 74$, $morning^{day}_{max} = afternoon^{day}_{max} = 82$, $night^{day}_{min} = 58$, and $night^{day}_{max} = 61$, the possible sum values are those reported in Table 1, where we also highlight admissible values in the interval $[work_{min}..work_{max}] = [1687..1692]$. The new encoding therefore determines the admissible pairs of the form $(N, M + A)$, where M, A, N are the number of morning, afternoon and nights assigned to a given nurse, by means of rule r'_{15}. These pairs are then used to check whether the assignment of working shifts is valid for each nurse by means of rules r'_4 and r'_5.

Actually, rules r'_4 and r'_5 take advantage from a third improvement of the advanced encoding. The number of morning, afternoon and night shifts that can be assigned to a nurse must adhere to Eq. (9), and are therefore limited to a few different values. The possible values of these aggregations are therefore encoded by means of atoms of the form $\texttt{countGE}(x, n, v)$, being true whenever $|\ \{d \in [1..365] : s(n, d) = x\}\ | \geq v$. It turns out that any answer set satisfies the following property: for each shift x and for each nurse n, there is exactly one value v such that $\texttt{countGE}(x, n, v), \texttt{not countGE}(x, n, v + 1)$ is true. In the advanced encoding, predicate `countGE` is defined by rule r'_{17}. Moreover, rule r'_{18} is used to enforce truth of $\texttt{countGE}(x, n, v - 1)$ whenever $\texttt{countGE}(x, n, v)$ is true; it is not required for correctness, but convenient to prune the search space in case $\texttt{countGE}(x, n, v)$ is assigned to true during the computation even if $|\ \{d \in [1..365] : s(n, d) = x\}\ | \geq v$ does not yet hold (for example, in case $\texttt{countGE}(x, n, v)$ is selected as a branching literal).

The fourth improvement is obtained by noting that rules r_{12}–r_{14} aggregate on sets $\{d \in [1..365] : s(n, d) = x\}$ for $x \in \{morning, afternoon, night\}$. It is therefore convenient to rewrite these rules in terms of predicate `countGE`, hence obtaining rules r'_{12}–r'_{14}. Finally, a further improvement is obtained by checking the number of nonworking days assigned to each nurse. For the specification given by the hospital it must range between 149 and 150, and in general the admitted range can be determined by rule r'_{16}. The check itself is then performed by rules r'_{12} and r'_{13} (for `S` being 7). Note that also this last check is not required to guarantee correctness of the encoding.

4 Empirical Evaluation

In this section the results of the empirical evaluation conducted on the same setting of [21] is reported. The experiments consider real data provided by the Italian hospital unit, which comprises a set of 41 nurses and holidays selected using the preferences of nurses of the year 2015. Moreover, the scalability of the approach has been evaluated by considering different number of nurses. In particular, an additional experiment was run by considering 10, 20, 41, 82 and 164 nurses without fixed holidays. We consider both the decisional (NSP^d) and the optimization (NSP^o) variants of NSP. Concerning the decisional variant, we compared our new ASP-based approach with the previous ASP encoding, with a solution based on SAT and one based on ILP.

The ASP encodings have been tested using the system CLINGO (v. 5.1.0) [24] and the solver WASP (v. 996bfb3) [5] combined with the grounder GRINGO [25], both configured with the core-based algorithms [4] for NSP^o. Solvers LINGELING (v. bbc-9230380-160707) [12], GLUCOSE (v. 4.1) [8] and CLASP (v. 3.2.2) have been executed on the SAT encoding, while the commercial tool GUROBI (v. 7.0.2) [1] on the ILP encoding. Concerning the optimization variant, the same tools for ASP and ILP have been used, whereas LINGELING and GLUCOSE have been replaced by the MaxSAT tools MSCG [32] and MAXINO [6], both binaries taken from MaxSAT Competition 2016.

In order to test SAT and ILP solutions, we created a pseudo-Boolean formula based on the ideas of the advanced ASP encoding. The pseudo-Boolean formula was represented using the OPB format, which is parsed by the tool GUROBI. Concerning the SAT-based solutions, we use the tool PBLIB [33] to convert the pseudoBoolean formula into a CNF. The running time of PBLIB has not been included in the analysis.

Time and memory were limited to 1 h and 8 GB, respectively. All the material can be found at http://www.star.dist.unige.it/~marco/Data/material.zip.

Results. The results of the run on the instance provided by the Italian hospital are reported in Table 2. The best result overall is obtained by CLINGO executed on the advanced encoding for both NSP^d and NSP^o, which is able to find a schedule in 42 and 70 s, respectively. This is a clear improvement with respect to the original encoding. Indeed, CLINGO executed on the original encoding was able to find a schedule in 22 and 7 min for the decisional and optimization variant, respectively. However, the advanced encoding does not help the other ASP solver WASP: its bad performance seems related to the branching heuristic, which is not effective on this particular domain. SAT-based (and MaxSAT-based) approaches are also not able to find a schedule within the allotted time and memory. In this case their performance can be explained by looking at the large size of the formula to evaluate (approximately 65 millions of clauses), which makes the solvers exceed the allotted memory. The tool GUROBI obtained good performance on both NSP^d and NSP^d instances. In particular for NSP^d GUROBI is faster than CLINGO executed on the original encoding. On the contrary, GUROBI is slower than CLINGO on NSP^o instances.

Scalability. We also performed an analysis about the scalability of the encoding, considering different numbers of nurses. In particular, for both NSP^d and NSP^o we considered five instances containing 10, 20, 41, 82 and 164 nurses, respectively. For each instance, we proportionally scaled the number of working nurses during each shift and holidays are randomly generated, whereas other requirements are not modified. Results are reported in Table 3.

The best results overall is obtained again by CLINGO executed on the advanced encoding, which outperforms all other tested approaches. Concerning ASP-based approaches, their performance is much better when they are executed on the advanced encoding. Indeed, the running time of CLINGO decreases considerably for all tested instances in both NSP^d and NSP^o. Moreover, Fig. 3

Table 2. Results of the experiment with 41 nurses and fixed holidays.

NSP^d		NSP^o	
Solver	Solving time (s)	Solver	Solving time (s)
CLINGO (ORIG ENC)	1352	CLINGO (ORIG ENC)	431
CLINGO (ADV ENC)	43	CLINGO (ADV ENC)	70
WASP (ORIG ENC)	-	WASP (ORIG ENC)	-
WASP (ADV ENC)	-	WASP (ADV ENC)	-
GLUCOSE (SAT ENC)	-	MSCG (MAXSAT ENC)	-
LINGELING (SAT ENC)	-	MAXINO (MAXSAT ENC)	-
CLASP (SAT ENC)	-	CLASP (MAXSAT ENC)	-
GUROBI (ILP ENC)	1018	GUROBI (ILP ENC)	1073

Table 3. Scalability of the approach. Solving time (s) for each solver.

Solver		Nurses				
		10	20	41	82	164
NSP^d	CLINGO (ORIG ENC)	155	117	738	1486	2987
	CLINGO (ADV ENC)	4	9	70	351	1291
	WASP (ORIG ENC)	-	-	-	-	-
	WASP (ADV ENC)	5	20	-	-	-
	GLUCOSE (SAT ENC)	-	-	-	-	-
	LINGELING (SAT ENC)	-	-	-	-	-
	CLASP (SAT ENC)	-	-	-	-	-
	GUROBI (ILP ENC)	62	172	1018	-	-
NSP^o	CLINGO (ORIG ENC)	37	94	339	798	1689
	CLINGO (ADV ENC)	4	13	72	482	1590
	WASP (ORIG ENC)	-	-	-	-	-
	WASP (ADV ENC)	4	-	-	-	-
	MSCG (MAXSAT ENC)	-	-	-	-	-
	MAXINO (MAXSAT ENC)	-	-	-	-	-
	CLASP (MAXSAT ENC)	-	-	-	-	-
	GUROBI (ILP ENC)	113	411	2004	-	-

shows a comparison among the number of conflicts found by CLINGO executed on the original and on the advanced encodings. The new encoding takes advantage of the better propagations, thus it is able to find a solution with a smaller number of conflicts. Concerning NSP^o, it can also be observed that the performance of the two versions of CLINGO are comparable on the instance with 164 nurses, even if the number of conflicts are much lower when CLINGO is executed on the advanced encoding. To explain this discrepancy we analyzed the number

of branching choices performed by CLINGO, which are around 128 millions for the original encoding, and around 300 millions for the advanced encoding. Thus, for this specific instance, the branching heuristic of CLINGO seems to be more effective when the original encoding is considered. Moreover, WASP executed on the advanced encoding is able to find a schedule for NSP^d when 10 and 20 nurses are considered whereas WASP executed on the original one does not terminate the computation in 1 h. The performance of SAT (and MaxSAT) solvers are also in this case not satisfactory since they cannot solve any of the tested instances. GUROBI can solve instances up to 41 nurses, whereas it is not able to find a schedule when 82 and 164 nurses are considered.

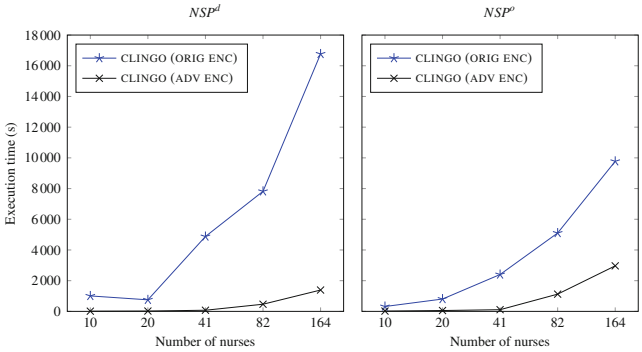

Fig. 3. Comparison of the number of conflicts (in thousands) of CLINGO executed on the original and on the advanced encodings for both NSP^d and NSP^o with different number of nurses.

5 Related Work

In recent years, several approaches to solve NSP have been proposed. The main differences concern (i) the planning periods; (ii) the different type of shifts; (iii) the requirements related to the coverage of shifts, i.e. the number of personnel needed for every shift; and (iv) other restrictions on the rules of nurses (see [14] for more detailed information). In this paper a one-year window of time has been considered as in [17], where however the same requirements on nurses and hospitals were not reported. Concerning the shifts, we considered three different shifts (morning, afternoon and night) with no overlapping among shifts, whereas in the literature other approaches were based on one single shift only (see e.g. [31]). Other requirements depend on the different policies of the considered hospitals. Thus, this makes the different strategies not directly comparable with each other.

Concerning other solving technologies reported in the literature, they range from mathematical to meta-heuristics approaches, including solutions based on integer programming [9,11], genetic algorithms [3], fuzzy approaches [35], and

ant colony optimization algorithms [28], to mention a few. Detailed and comprehensive surveys on NSP can be found in [14,18].

The approach described in this paper represents an enhancement of the one proposed in [21]. The two encodings mainly differ with respect to how the constraints related to hospital and balance requirements are modeled. Indeed, the new encoding takes into account only combinations of parameters values that can lead to a valid schedule.

Finally, we report that ASP has been already successfully used for solving hard combinatorial and application problems in several research areas, including Artificial Intelligence [10,20], Bioinformatics [22,29], Hydroinformatics [23], Databases [30] and also in industrial applications [2,19]. ASP encodings were proposed for scheduling problems other than NSP: *Incremental Scheduling Problem* [16], where the goal is to assign jobs to devices such that their executions do not overlap one another; and *Team Building Problem* [34], where the goal is to allocate the available personnel of a seaport for serving the incoming ships. However, to the best of our knowledge, the only ASP encodings for NSP are those shown in Sect. 3.

6 Conclusion

In this paper an advanced ASP encoding for addressing a variant of NSP has been proposed. The new encoding overcomes the limitations of the one proposed in [21] by taking into account intrinsic properties of NSP and internal details of ASP solvers. The resulting approach has been compared with the previous one and with other declarative approaches on real setting provided by an Italian hospital. Results clearly show that CLINGO executed on the new encoding outperforms all alternatives, being able to solve all instances within 30 min, even with more than 100 nurses.

Acknowledgments. We would like to thank Nextage srl for providing support for this work. Mario Alviano has been partially supported by the Italian Ministry for Economic Development (MISE) under project "PIUCultura – Paradigmi Innovativi per l'Utilizzo della Cultura" (no. F/020016/01-02/X27), and under project "Smarter Solutions in the Big Data World (S2BDW)" (no. F/050389/01-03/X32) funded within the call "HORIZON2020" PON I&C 2014-2020, and by Gruppo Nazionale per il Calcolo Scientifico (GNCS-INdAM).

References

1. The website of Gurobi. http://www.gurobi.com
2. Abseher, M., Gebser, M., Musliu, N., Schaub, T., Woltran, S.: Shift design with answer set programming. Fundam. Inform. **147**(1), 1–25 (2016). https://doi.org/10.3233/FI-2016-1396
3. Aickelin, U., Dowsland, K.A.: An indirect genetic algorithm for a nurse-scheduling problem. Comput. OR **31**(5), 761–778 (2004). https://doi.org/10.1016/S0305-0548(03)00034-0

4. Alviano, M., Dodaro, C.: Anytime answer set optimization via unsatisfiable core shrinking. TPLP **16**(5–6), 533–551 (2016). https://doi.org/10.1017/S147106841600020X
5. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: Calimeri, F., Ianni, G., Truszczynski, M. (eds.) LPNMR 2015. LNCS (LNAI), vol. 9345, pp. 40–54. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23264-5_5
6. Alviano, M., Dodaro, C., Ricca, F.: A MaxSAT algorithm using cardinality constraints of bounded size. In: IJCAI 2015, pp. 2677–2683. AAAI Press (2015)
7. Alviano, M., Faber, W.: The complexity boundary of answer set programming with generalized atoms under the FLP semantics. In: Cabalar, P., Son, T.C. (eds.) LPNMR 2013. LNCS (LNAI), vol. 8148, pp. 67–72. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40564-8_7
8. Audemard, G., Simon, L.: Extreme cases in SAT problems. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 87–103. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_7
9. Azaiez, M.N., Sharif, S.S.A.: A 0–1 goal programming model for nurse scheduling. Comput. OR **32**, 491–507 (2005). https://doi.org/10.1016/S0305-0548(03)00249-1
10. Balduccini, M., Gelfond, M., Watson, R., Nogueira, M.: The USA-advisor: a case study in answer set planning. In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR 2001. LNCS (LNAI), vol. 2173, pp. 439–442. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45402-0_39
11. Bard, J.F., Purnomo, H.W.: Preference scheduling for nurses using column generation. Eur. J. Oper. Res. **164**(2), 510–534 (2005). https://doi.org/10.1016/j.ejor.2003.06.046
12. Biere, A., Fröhlich, A.: Evaluating CDCL variable scoring schemes. In: Heule, M., Weaver, S. (eds.) SAT 2015. LNCS, vol. 9340, pp. 405–422. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24318-4_29
13. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011). https://doi.org/10.1145/2043174.2043195
14. Burke, E.K., Causmaecker, P.D., Berghe, G.V., Landeghem, H.V.: The state of the art of nurse rostering. J. Sched. **7**(6), 441–499 (2004). https://doi.org/10.1023/B:JOSH.0000046076.75950.0b
15. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T.: ASP-Core-2 Input Language Format (2013). https://www.mat.unical.it/aspcomp.2013/files/ASP-CORE-2.01c.pdf
16. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the fifth answer set programming competition. Artif. Intell. **231**, 151–181 (2016). https://doi.org/10.1016/j.artint.2015.09.008
17. Chan, P., Weil, G.: Cyclical staff scheduling using constraint logic programming. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 159–175. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44629-X_10
18. Cheang, B., Li, H., Lim, A., Rodrigues, B.: Nurse rostering problems - a bibliographic survey. Eur. J. Oper. Res. **151**(3), 447–460 (2003). https://doi.org/10.1016/S0377-2217(03)00021-3
19. Dodaro, C., Gasteiger, P., Leone, N., Musitsch, B., Ricca, F., Schekotihin, K.: Combining answer set programming and domain heuristics for solving hard industrial problems (application paper). TPLP **16**(5–6), 653–669 (2016). https://doi.org/10.1017/S1471068416000284
20. Dodaro, C., Leone, N., Nardi, B., Ricca, F.: Allotment problem in travel industry: a solution based on ASP. In: Cate, B., Mileo, A. (eds.) RR 2015. LNCS, vol. 9209, pp. 77–92. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22002-4_7

21. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: Balduccini, M., Janhunen, T. (eds.) LPNMR 2017. LNCS (LNAI), vol. 10377, pp. 301–307. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61660-5_27
22. Erdem, E., Öztok, U.: Generating explanations for biomedical queries. TPLP **15**(1), 35–78 (2015). https://doi.org/10.1017/S1471068413000598
23. Gavanelli, M., Nonato, M., Peano, A.: An ASP approach for the valves positioning optimization in a water distribution system. J. Log. Comput. **25**(6), 1351–1369 (2015). https://doi.org/10.1093/logcom/ext065
24. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with Clingo 5. In: ICLP TCs. OASICS, vol. 52, pp. 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016). https://doi.org/10.4230/OASIcs.ICLP.2016.2
25. Gebser, M., Kaminski, R., König, A., Schaub, T.: Advances in *gringo* series 3. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS (LNAI), vol. 6645, pp. 345–351. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20895-9_39
26. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.T.: Potassco: the Potsdam answer set solving collection. AI Commun. **24**(2), 107–124 (2011). https://doi.org/10.3233/AIC-2011-0491
27. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: from theory to practice. Artif. Intell. **187**, 52–89 (2012). https://doi.org/10.1016/j.artint.2012.04.001
28. Gutjahr, W.J., Rauner, M.S.: An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria. Comput. OR **34**(3), 642–666 (2007). https://doi.org/10.1016/j.cor.2005.03.018
29. Koponen, L., Oikarinen, E., Janhunen, T., Säilä, L.: Optimizing phylogenetic supertrees using answer set programming. TPLP **15**(4–5), 604–619 (2015). https://doi.org/10.1017/S1471068415000265
30. Marileo, M.C., Bertossi, L.E.: The consistency extractor system: answer set programs for consistent query answering in databases. Data Knowl. Eng. **69**(6), 545–572 (2010). https://doi.org/10.1016/j.datak.2010.01.005
31. Miller, H.E., Pierskalla, W.P., Rath, G.J.: Nurse scheduling using mathematical programming. Oper. Res. **24**(5), 857–870 (1976). https://doi.org/10.1287/opre.24.5.857
32. Morgado, A., Dodaro, C., Marques-Silva, J.: Core-guided MaxSAT with soft cardinality constraints. In: O'Sullivan, B. (ed.) CP 2014. LNCS, vol. 8656, pp. 564–573. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10428-7_41
33. Philipp, T., Steinke, P.: PBLib – a library for encoding pseudo-boolean constraints into CNF. In: Heule, M., Weaver, S. (eds.) SAT 2015. LNCS, vol. 9340, pp. 9–16. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24318-4_2
34. Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N.: Team-building with answer set programming in the Gioia-Tauro seaport. TPLP **12**(3), 361–381 (2012). https://doi.org/10.1017/S147106841100007X
35. Topaloglu, S., Selim, H.: Nurse scheduling using fuzzy modeling approach. Fuzzy Sets Syst. **161**(11), 1543–1563 (2010). https://doi.org/10.1016/j.fss.2009.10.003