On the Integration of CP-nets in ASPRIN

Mario Alviano[1] , **Javier Romero**[2] and **Torsten Schaub**[2]
[1]University of Calabria, Italy
[2]University of Potsdam, Germany
alviano@mat.unical.it, {javier, torsten}@cs.uni-potsdam.de

Abstract

Conditional preference networks (CP-nets) express qualitative preferences over features of interest. A Boolean CP-net can express that a feature is preferable under some conditions, as long as all other features have the same value. This is often a convenient representation, but sometimes one would also like to express a preference for maximizing a set of features, or some other objective function on the features of interest. ASPRIN is a flexible framework for preferences in ASP, where one can mix heterogeneous preference relations, and this paper reports on the integration of Boolean CP-nets. In general, we extend ASPRIN with a preference program for CP-nets in order to compute most preferred answer sets via an iterative algorithm. For the specific case of acyclic CP-nets, we provide an approximation by partially ordered set preferences, which are in turn normalized by ASPRIN to take advantage of several highly optimized algorithms implemented by ASP solvers for computing optimal solutions. Finally, we take advantage of a linear-time computable function to address dominance testing for tree-shaped CP-nets.

1 Introduction

Answer set programming (ASP) is a language for knowledge representation and reasoning supporting several constructs to ease the specification of complex problems in terms of logic rules [van Harmelen *et al.*, 2008]. Each solution of a problem encoded in ASP maps to an answer set, that is, a set of propositional atoms satisfying all logic rules, and an additional stability condition empowering the language with nonmonotonic reasoning capabilities [Gelfond and Lifschitz, 1991].

Problems that find a natural representation in ASP are often characterized by several answer sets, which therefore represent the feasible solutions of the encoded problem. In some cases, feasible solutions can be compared according to some features of interest. As a use case, a car dealership may want to offer a car configurator to its customers. All valid car configurations constitute the feasible solutions, but not all of them are equally satisfactory for the customer using the car configurator. Hence, as an additional service, the car configurator may offer the possibility to specify preferences over some features. For example, a customer may want to specify that she prefers automatic transmission, convertible car, light paint for convertible cars and dark paint otherwise, and many other features. Such preferences are clearly *qualitative*, and naturally expressed by *ceteris paribus rules*, from the Latin *as long as everything else stays the same*. In the example, among two valid configurations differing only on the transmission, the automatic one is preferred; additionally, ceteris paribus rules may be conditional, so to express that the preference for the paint is subject to the selection of other features of interest, in this case whether the car is convertible or not.

ASP offers linguistic constructs for the specification of preference relations. Weak constraints [Buccafurri *et al.*, 2000; Simons *et al.*, 2002] are the most frequently used constructs for this purpose, and they encode objective functions to be minimized or maximized, that is, *quantitative* preference relations. While weak constraints are suitable for many applications, their use to express other common preference relations requires some additional effort. In particular, the core language of ASP misses constructs to easily express *conditional preference networks* (CP-nets) [Boutilier *et al.*, 2004a], that is, sets of ceteris paribus rules linked by conditional dependencies among the features of interest.

A more natural alternative is provided by ASPRIN [Brewka *et al.*, 2015a; Brewka *et al.*, 2015b], a framework handling user-defined preference relations encoded in ASP, and shipped with a rich library of common preference relations that can be used and combined by declarative statements. This paper reports on the integration of CP-nets in ASPRIN. Specifically, CP-nets are declared by *named preference statements* comprising *preference elements* that express conditional ceteris paribus rules; as usual in ASP systems, object variables can be used for a compact representation of a set of ground instances, and conditional ceteris paribus rules in ASPRIN are not an exception in this respect. CP-nets encoded by preference statements are mapped to ASP facts by the reification procedure implemented in ASPRIN. In ASPRIN, CP-nets can be combined with composite preference relations, such as *pareto* or *lexicographic* aggregation. The preference relations involved in a composition can be heterogeneous, which means that ASPRIN can naturally mix CP-nets with other qualitative and quantitative preference relations. Such

a mix is useful in several contexts [Domshlak *et al.*, 2006], as for example in the car configurator, where the customer is likely to prefer a less expensive configuration among those that are otherwise equally preferred. Further ties could be broken by minimizing the maximum shipping delays among all extras that are added to the configuration. A strength of ASPRIN is to provide a uniform syntax to express background knowledge bases, preference relations, and their aggregation.

The ASPRIN library is extended with a *preference program* for CP-nets, that is, an ASP program whose instantiation with respect to the reification of a CP-net and a pair of interpretations is consistent if and only if the pair of interpretations belongs to the preference relation associated with the CP-net (Section 3). The preference program is used by the iterative sat-unsat search algorithm implemented by ASPRIN for computing an optimal answer set, and for blocking all interpretations that are worse than the computed optimal answer sets.

ASPRIN provides also another solving technique for *acyclic* CP-nets, which is based on the notion of *approximation* [Alviano *et al.*, 2018]. Intuitively, a preference relation $\succeq$ approximates all preference relations being its subsets, and this property is sufficient to guarantee that all $\succeq$-optimal models are in turn optimal with respect to the approximated preference relations. For example, cardinal-minimality approximates subset-minimality. Approximation can be associated with *upstream expansion functions* for introducing auxiliary symbols, so that the mapping between different preference relations remains compact. In ASPRIN, CP-nets are approximated by preference relations induced by strict partial orders (*poset* [Rosa *et al.*, 2010]) over a set of auxiliary atoms. The mapping (given in Section 4) is intuitive, and links CP-nets to several preference relations in the ASPRIN library.

This section concludes with a few considerations on the complexity of *dominance testing*, that is, checking that a pair of interpretations belongs to the preference relation associated with a CP-net. The problem is in general PSPACE-complete [Goldsmith *et al.*, 2008], but there are tractable cases. Among them, *tree-shaped* CP-nets, for which dominance testing is linear time computable [Bigot *et al.*, 2013]; accordingly, ASPRIN is equipped with a preference program specific to this class of CP-nets (Section 5). Concerning acyclic CP-nets, dominance testing is known to be NP-hard [Boutilier *et al.*, 2004a], while it is polynomial-time solvable for the approximation used by ASPRIN; practically, this fact implies that some optimal answer sets can be discarded at the beginning of the computation, and recovered after blocking some other optimal answer sets.

2 Background

Optimal models. A *knowledge base* Γ is a pair $(\mathcal{A}, \mathcal{M})$ such that $\mathcal{A}$ is a (finite) set of (propositional) atoms, and $\mathcal{M}$ is a set of subsets of $\mathcal{A}$. Set $\mathcal{A}$ is the *domain* of Γ, and each set in $\mathcal{M}$ is a *model* of Γ. A knowledge base is *consistent* if $\mathcal{M} \neq \emptyset$, and *inconsistent* otherwise. A *preference relation* $\succeq$ over a domain $\mathcal{A}$ is a partial order over $2^{\mathcal{A}}$, that is, $\succeq$ is a subset of $2^{\mathcal{A}} \times 2^{\mathcal{A}}$, and $\succeq$ is reflexive, antisymmetric and transitive. (Note that $\succeq$ also defines a preference relation over any superset of $\mathcal{A}$ in the obvious way, that is, $I \succeq J$ if and only if $I \cap \mathcal{A} \succeq J \cap \mathcal{A}$.) Let $\succ$ be the strict partial order obtained from $\succeq$, that is, $I \succ J$ if and only if $I \succeq J$ and $J \not\succeq I$. (Recall that a strict partial order is an irreflexive and transitive relation.) Let $\Gamma = (\mathcal{A}, \mathcal{M})$ be a knowledge base, and $\succeq$ be a preference relation over $\mathcal{A}$. $I \in \mathcal{M}$ is a $\succeq$-*optimal* model of Γ if there is no $J \in \mathcal{M}$ such that $J \succ I$.

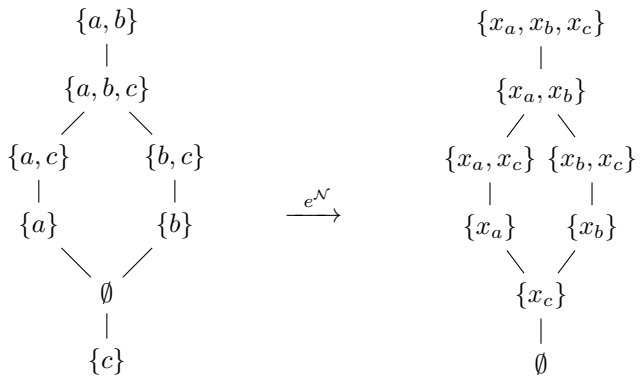

Figure 1: Hasse diagrams of preference relations from Examples 2 (left) and 1 (right). The two preference relations are linked by an approximation defined in Section 4.

Approximations. Let $\mathcal{A}, \mathcal{A}'$ be sets of atoms such that $\mathcal{A} \subseteq \mathcal{A}'$. An *expansion function* from $\mathcal{A}$ to $\mathcal{A}'$ is a function $e : 2^{\mathcal{A}} \to 2^{\mathcal{A}'}$ such that $e(I) \cap \mathcal{A} = I$, for all $I \subseteq \mathcal{A}$. Abusing notation, for a knowledge base $\Gamma = (\mathcal{A}, \mathcal{M})$, let $e(\Gamma)$ be the knowledge base $(\mathcal{A}', \{e(I) \mid I \in \mathcal{M}\})$. Let $\succeq$ be a preference relation for $\mathcal{A}$, and e be an expansion function from $\mathcal{A}$ to $\mathcal{A}'$. A preference relation $\succeq'$ is an approximation of $\succeq$ with respect to e, or e-*approximation*, if $I \succ J$ implies $e(I) \succ' e(J)$, for all $I, J \subseteq \mathcal{A}$. Approximations are closed under composition (Theorem 1 in [Alviano *et al.*, 2018]), and the optimal models they characterize are also optimal for the approximated preference relation (Theorem 2 in [Alviano *et al.*, 2018]). These two properties are stated next.

Proposition 1. *Let $e : 2^{\mathcal{A}} \to 2^{\mathcal{A}'}$, $e' : 2^{\mathcal{A}'} \to 2^{\mathcal{A}''}$ be expansion functions. Their composition $e' \circ e : 2^{\mathcal{A}} \to 2^{\mathcal{A}''}$ is an expansion function. Moreover, if $\succeq'$ is an e-approximation of $\succeq$, and $\succeq''$ is an e'-approximation of $\succeq'$, then $\succeq''$ is an $(e' \circ e)$-approximation of $\succeq$.*

Proposition 2. *Let $\Gamma = (\mathcal{A}, \mathcal{M})$ be a knowledge base, $\succeq$ be a preference relation over $\mathcal{A}$, $e : 2^{\mathcal{A}} \to 2^{\mathcal{A}'}$ be an expansion function, and $\succeq'$ be an e-approximation of $\succeq$. For any $I \subseteq \mathcal{A}$, if $e(I) \subseteq \mathcal{A}'$ is a $\succeq'$-optimal model of $e(\Gamma)$, then I is a $\succeq$-optimal model of Γ.*

Posets. Let $\succ$ be a strict partial order over a set $\mathcal{A}$ of atoms, that is, $\succ$ is a subset of $\mathcal{A} \times \mathcal{A}$, $\succ$ is irreflexive ($p \not\succ p$ for all $p \in \mathcal{A}$), and $\succ$ is transitive ($p_1 \succ p_2$ and $p_2 \succ p_3$ implies $p_1 \succ p_3$, for all $p_1, p_2, p_3 \in \mathcal{A}$). The preference relation $poset^{\succ}$ over $\mathcal{A}$ defined by $\succ$ is the reflexive (and transitive) closure of $\{(I, J) \mid I, J \subseteq \mathcal{A}, I \setminus J \neq \emptyset$, for all $p \in J \setminus I$ there is $p' \in I \setminus J$ such that $p' \succ p\}$.

Example 1. Let $\succ$ be such that $x_a \succ x_c$ and $x_b \succ x_c$. Preference relation $poset^{\succ}$ is shown in Figure 1. ■

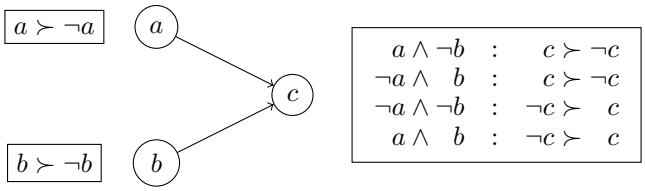

Figure 2: Dependency graph from Example 2, and preference tables of domain atoms.

CP-nets. In order to uniform the notation, let us define a (Boolean) CP-net $\mathcal{N}$ over a domain $\mathcal{A}$ as a set comprising conditional *ceteris paribus* statements of the form $p : (\mathcal{A}_p, \mathcal{M}_p)$ for each $p \in \mathcal{A}$, where $\mathcal{A}_p \subseteq \mathcal{A} \setminus \{p\}$, and $(\mathcal{A}_p, \mathcal{M}_p)$ is a knowledge base. The *dependency graph* $\mathcal{G}_\mathcal{N}$ of $\mathcal{N}$ has nodes $\mathcal{A}$, and arcs from all atoms in $\mathcal{A}_p$ to p, for all $p : (\mathcal{A}_p, \mathcal{M}_p)$ in $\mathcal{N}$. $\mathcal{N}$ is acyclic if $\mathcal{G}_\mathcal{N}$ is acyclic. The preference relation $cp^\mathcal{N}$ defined by $\mathcal{N}$ is the reflexive, transitive closure of

$$\{(I \cup \{p\}, I \setminus \{p\}) \mid p : (\mathcal{A}_p, \mathcal{M}_p) \text{ belongs to } \mathcal{N}, I \subseteq \mathcal{A}, I \cap \mathcal{A}_p \in \mathcal{M}_p\}$$
$$\cup \quad \{(I \setminus \{p\}, I \cup \{p\}) \mid p : (\mathcal{A}_p, \mathcal{M}_p) \text{ belongs to } \mathcal{N}, I \subseteq \mathcal{A}, I \cap \mathcal{A}_p \notin \mathcal{M}_p)\}.$$

(In this paper, $cp^\mathcal{N}$ is assumed to be antisymmetric, that is, $\mathcal{N}$ is consistent according to [Boutilier *et al.*, 2004a].)

Example 2. Let $\mathcal{N}$ be the following CP-net:

$$a : (\emptyset, \{\emptyset\}) \quad b : (\emptyset, \{\emptyset\}) \quad c : (\{a, b\}, \{\{a\}, \{b\}\})$$

Preference relation $cp^\mathcal{N}$ is shown in Figure 1. The dependency graph of $\mathcal{N}$ is shown in Figure 2 (which also reports the preference tables of domain atoms, a common way to express CP-nets). ■

A CP-net $\mathcal{N}$ is *tree-shaped* if $\mathcal{G}_\mathcal{N}$ is a forest (i.e., a set of trees), and each ceteris paribus statement $p : (\mathcal{A}_p, \mathcal{M}_p)$ is such that $\mathcal{A}_p$ is empty (i.e., p is a *root*) or $\mathcal{M}_p$ is a singleton (for otherwise, $\mathcal{A}_p$ can be reduced to the empty set). For a tree-shaped CP-net $\mathcal{N}$ over a domain $\mathcal{A}$, and two sets $I, I' \subseteq \mathcal{A}$, $I \, cp^\mathcal{N} \, I'$ can be checked in time $O(|\mathcal{A}|^2)$ by algorithm TREEDT [Boutilier *et al.*, 2004a], shown as Algorithm 1. Function *flip*(p, I') returns $I' \setminus \{p\}$ if $p \in I'$, and $I' \cup \{p\}$ otherwise. For a forest $\mathcal{G}$, let *subtree*$(\mathcal{G}, p)$ be the subtree of $\mathcal{G}$ whose root is p. An atom $p \in \mathcal{A}$ such that (*flip*$(p, I'), I')$ belongs to $cp^\mathcal{N}$ is said *I'-improvable*; p is *$\mathcal{G}$-I'-improvable* if p is the only I'-improvable node in *subtree*$(\mathcal{G}, p)$. Hence, the algorithm removes leaves that match in I and I' (lines 3–4), and flips $\mathcal{G}$-I'-improvable atoms (line 7), until either all nodes are eliminated (i.e., all nodes match; line 5) or there is no I'-improvable node (i.e., I' cannot be further improved).

Algorithm 1: TREEDT($\mathcal{N}$, I, I')

```
1 G := G_N
2 loop
3     while G has a leaf p s.t. {p} ∩ I = {p} ∩ I' do
4         remove p from G
5     if G is empty then return YES
6     if no node in G is I'-improvable then return NO
7     I' := flip(p, I') for some p being G-I'-improvable
```

3 CP-nets in ASPRIN

In ASPRIN, knowledge bases and preference relations are encoded by ASP programs (refer to [Gebser *et al.*, 2015] for syntax and semantics of ASP, and [Brewka *et al.*, 2015b] for details on ASPRIN). Specifically, a (ground) ASP program with atoms $\mathcal{A}$ and answer sets $\mathcal{M}$ encodes the knowledge base $(\mathcal{A}, \mathcal{M})$, while preference relations are declared by statements of the following form:

$$\#\texttt{preference}(id, type)\{e_1; \cdots ; e_n\}.$$

where $n \geq 1$ and each e_i ($i \in [1..n]$) is a *preference element*. For CP-nets, *type* is `cp`, and preference elements have the form $\ell \parallel L$, where ℓ is a literal and L is a (semicolon-separated) set of literals (a literal is an atom possibly preceded by `not`). Intuitively, a *ceteris paribus* statement $p : (\mathcal{A}_p, \mathcal{M}_p)$ with $\mathcal{A}_p \neq \emptyset$ is encoded by $p \parallel L_1; \cdots ; p \parallel L_m;$ `not` $p \parallel L_{m+1}; \cdots ;$ `not` $p \parallel L_n$, where $\bigvee_{i=1}^{m} \bigwedge L_i$ (and equivalently $\neg \bigvee_{i=m+1}^{n} \bigwedge L_i$) encodes $(\mathcal{A}_p, \mathcal{M}_p)$; $p : (\emptyset, \{\emptyset\})$ is encoded by p, and $p : (\emptyset, \emptyset)$ is encoded by `not` p.

Example 3. In ASPRIN, the CP-net from Example 2 can be declared as follows:

```
#preference(cp_from_example, cp) {
  a; b; c || {a; not b}; c || {not a; b};
  not c || {not a; not b}; not c || {a; b}
}.
```

(Preference element `a` can be also written `a >> not a`, and similar for the other preference elements, a syntax closer to preference tables in the literature.) ■

To simplify the presentation, the input is assumed to include exactly one preference statement of type `cp`, encoding a CP-net $\mathcal{N}$ over a domain $\mathcal{A}$. ASPRIN reifies the i-th preference element $\ell : \{\ell_1; \cdots ; \ell_n\}$ ($n \geq 0$) as follows:

```
pref(i,1,encode(ℓ)).
pref(i,0,encode(ℓ_j)).        ∀j ∈ [1..n]
```

where *encode* maps p to `atom(p)`, and `not` p to `neg(atom(p))`, for every atom p. (Actually, ASPRIN uses predicate `pref` and arguments for the name and the type of the preference relation, as well as other extra arguments which are not given here for simplicity.) Let $\Pi_\mathcal{N}$ be the reification of all preference elements in $\mathcal{N}$.

Example 4. The reification of the first three preference elements in Example 3 is the following:

```
pref(1,1,atom(a)). pref(2,1,atom(b)).
pref(3,1,atom(c)). pref(3,0,atom(a)).
pref(3,0,neg(atom(b))).
```
■

Figure 3 shows the *preference program* Π_{CP} associated with CP-nets (note that the conditional literal `holds(atom(A)): holds'(atom(A))` is expanded into a conjunction and is essentially interpreted as `#count{A : not holds(atom(A)), holds'(atom(A))} = 0`). The program encodes the following planing problem: Start from an interpretation encoded by predicate `holds'`, and choose

```
better :- not eq, not fail.
eq :- holds(atom(A)) : holds'(atom(A));
   holds'(atom(A)) : holds(atom(A)).
% guess plan
{do(I,T) : pref(I,1,_)} = 1 :- T=1..diameter.
% initial situation
true(A,0) :- holds'(atom(A)).
% positive effects and inertia
true(A,T) :- do(I,T), pref(I,1,atom(A)).
true(A,T) :- true(A,T-1), T <= diameter,
   not do(I,T) : pref(I,1,neg(atom(A))).
% preconditions
fail :- do(I,T), pref(I,0,atom(A)),
   not true(A,T-1).
fail :- do(I,T), pref(I,0,neg(atom(A))),
   true(A,T-1).
% goal
fail :- true(A,diameter), not holds(atom(A)).
fail :- not true(A,diameter), holds(atom(A)).
```

Figure 3: Preference program Π_{CP} for (possibly cyclic) CP-nets

a sequence of preference elements so to reach a better interpretation encoded by predicate `holds`. A preference element $\ell \parallel L$ chosen at time t must be applicable, that is, each literal in L must be true at time $t-1$, and causes the truth of ℓ at time t. The length of the plan is bounded by the numeric constant `diameter`, which generally is assigned the value $2^{|\mathcal{A}|}$; smaller values can be used under some conditions (for example, $|\mathcal{A}|^2$ is sufficient if $\mathcal{G}_\mathcal{N}$ is a forest [Boutilier *et al.*, 2004a]).

The preference program satisfies the following invariant: if $I, I' \subseteq \mathcal{A}$ are encoded by predicates `holds` and `holds'`, respectively, then an answer set exists if and only if I is better than I' in $cp^\mathcal{N}$. More formally, let $\Pi_{\mathcal{N},I,I',d}$ be the program $\Pi_{CP} \cup \Pi_\mathcal{N}$ extended with the following rules:

```
#const diameter = d.
:- not better.
holds (atom(p)).          ∀p ∈ I
holds'(atom(p)).          ∀p ∈ I'
```

The invariant is stated by the next claim.

Theorem 1. *Let $\mathcal{N}$ be a CP-net over a domain $\mathcal{A}$. For all $I, I' \subseteq \mathcal{A}$, program $\Pi_{\mathcal{N},I,I',2^{|\mathcal{A}|}}$ is consistent if and only if $(I, I') \in cp^\mathcal{N}$ and $I \neq I'$.*

Example 5. Continuing with Example 4, for $I = \{a, b\}$ and $I' = \{a, c\}$, an answer set for $\Pi_{\mathcal{N},I,I',2^3}$ is obtained by choosing `do(2,1)` and `do(6,t)` for all $t \in [2..8]$ (as well as all other implied atoms). Indeed, `do(2,1)` transforms $\{a, c\}$ into $\{a, b, c\}$, and `do(6,2)` transforms $\{a, b, c\}$ into $\{a, b\}$. All other `do(6,t)` atoms just maintain $\{a, b\}$. ■

Theorem 1 guarantees correctness of the iterative sat-unsat search algorithm implemented in ASPRIN for computing one $cp^\mathcal{N}$-optimal answer set of a program Π. Specifically, better and better answer sets are searched until the processed program has no answer sets, witnessing the optimality of the latest computed answer set. Moreover, when computing many optimal answer sets, ASPRIN uses the preference program to block any computed $cp^\mathcal{N}$-optimal answer set I, and any interpretation I' such that $(I \cap \mathcal{A}, I' \cap \mathcal{A}) \in cp^\mathcal{N}$. The process is completely automated by ASPRIN, and described in detail in the literature [Brewka *et al.*, 2015a].

Algorithm 2: TREEDT-restated($\mathcal{N}$, I, I')

```
// Lines 1–6 from TREEDT
```
7 $I' := \mathit{flip}(\mathit{subtree}(\mathcal{G}_\mathcal{N}, p), I')$ for all $\mathcal{G}$-I'-improvable p

4 From Acyclic CP-nets to Posets

Let $\mathcal{N}$ be a CP-net over $\mathcal{A}$. Let $e^\mathcal{N}$ be the following expansion function from $\mathcal{A}$ to $\mathcal{A} \cup \{x_p \mid p \in \mathcal{A}\}$:

$$\begin{aligned} e^\mathcal{N}(I) \;:=\; & I \cup \{x_p \mid p : (\mathcal{A}_p, \mathcal{M}_p) \text{ belongs to } \mathcal{N}, \\ & \qquad p \in I, I \cap \mathcal{A}_p \in \mathcal{M}_p\} \\ \cup\; & \{x_p \mid p : (\mathcal{A}_p, \mathcal{M}_p) \text{ belongs to } \mathcal{N}, \\ & \qquad p \in \mathcal{A} \setminus I, I \cap \mathcal{A}_p \notin \mathcal{M}_p\}. \end{aligned}$$

Let $\succ^\mathcal{N}$ be the poset over $\{x_p \mid p \in \mathcal{A}\}$ such that $x_p \succ x_q$ if $\mathcal{G}_\mathcal{N}$ has a path from p to q (that is, if p is an ancestor of q).

Example 6. For the CP-net $\mathcal{N}$ from Example 2, the expansion function $e^\mathcal{N}$ is schematically shown in Figure 1: $\{a, b\} \mapsto \{a, b, x_a, x_b, x_c\}$, $\{a, b, c\} \mapsto \{a, b, c, x_a, x_b\}$, $\{a, c\} \mapsto \{a, c, x_a, x_c\}$, and so on. Moreover, $\succ^\mathcal{N}$ is the strict partial order from Example 1. ■

The next theorem states that acyclic CP-nets are approximated by poset preference relations.

Theorem 2. *Let $\mathcal{N}$ be a (Boolean) acyclic CP-net over $\mathcal{A}$. Relation poset$^{\succ^\mathcal{N}}$ is an $e^\mathcal{N}$-approximation of $cp^\mathcal{N}$.*

The ASPRIN library is extended with an ASP encoding to map acyclic CP-nets to poset, and in turn to either weak constraints or lexicographic composition of subset preference relations. The underlying ASP solver is then asked to compute some optimal models, which are in turn optimal models according to the CP-net.

5 Tree-shaped CP-nets

For tree-shaped CP-nets, TREEDT iteratively removes matching leaves, and then flips some $\mathcal{G}$-I'-improvable node p. After flipping p, all children of p (if any) become $\mathcal{G}$-I'-improvable, and then they are flipped in subsequent steps of the algorithm. It turns out that all the nodes in $\mathit{subtree}(p)$ are flipped in this way, so that the leaves of $\mathit{subtree}(p)$ will be removed. Moreover, all $\mathcal{G}$-I'-improvable nodes can be processed in a single step of the algorithm. Hence, TREEDT is restated as shown in Algorithm 2.

An interesting observation is that, after each step of the restated algorithm, nodes in $\mathit{subtree}(p)$ are not I'-improvable (because they have been just flipped). Hence, in a run of Algorithm 2, each node is I'-improvable at most once, and anyhow before being flipped. Therefore, I'-improvable nodes can be determined before starting the loop of the algorithm, on the I' in input.

We can also determine the number of times a node p *has* to flip back to its original value in I' before being removed

(the algorithm returns NO if such a number is unreachable). This number is referred to as the *loops* of a node, and defined inductively as follows (for a fixed CP-net $\mathcal{N}$ and sets I, I'):

$$lps(p) := \begin{cases} 0 & \text{if } p \text{ is a leaf} \\ \max_{pq \in \mathcal{G}_\mathcal{N}} lps(p,q) & \text{otherwise} \end{cases}$$

$$lps(p,q) := \begin{cases} lps(q)+1 & \text{if } q \text{ is not } I'\text{-improv.,} \\ & \quad \{p\} \cap I = \{p\} \cap I', \\ & \quad \{q\} \cap I \neq \{q\} \cap I' \\ \max(0, lps(q)-1) & \text{if } q \text{ is } I'\text{-improvable,} \\ & \quad \{p\} \cap I \neq \{p\} \cap I', \\ & \quad \{q\} \cap I = \{q\} \cap I' \\ lps(q) & \text{otherwise} \end{cases}$$

Intuitively, a leaf does not loop because it must be removed in the first two iterations of the restated algorithm. Internal nodes, instead, may loop in order to let their descendants be removed. Specifically, if a node q is not I'-improvable and $\{q\} \cap I \neq \{q\} \cap I'$, its flipping requires a flip of its parent p (so that q becomes improvable); if $\{p\} \cap I = \{p\} \cap I'$ holds, then p has to be flipped back to its value, i.e., p requires an additional loop. On the other hand, if q is I'-improvable, $\{q\} \cap I = \{q\} \cap I'$, and $\{p\} \cap I \neq \{p\} \cap I'$, then q can do one loop (if any) without imposing a loop of p as follows: flip q, then flip p and make q improvable again, so that q can be flipped back to its original value. In all other cases, each loop of q implies a loop of its parent p.

Theorem 3. *Let $\mathcal{N}$ be a tree-shaped CP-net over a domain $\mathcal{A}$. For all distinct $I, I' \subseteq \mathcal{A}$, $(I, I') \in cp^\mathcal{N}$ if and only if (i) for each $p : (\emptyset, \{\emptyset\})$ in $\mathcal{N}$, $lps(p) = 0$ and either $\{p\} \cap I = \{p\} \cap I'$ or $p \in I$, and (ii) for each $p : (\emptyset, \emptyset)$ in $\mathcal{N}$, $lps(p) = 0$ and either $\{p\} \cap I = \{p\} \cap I'$ or $p \notin I$. The time complexity to check (i) and (ii) is $O(|\mathcal{A}|)$.*

Figure 4 reports an ASP encoding of the conditions used in Theorem 3 to check that an interpretation is preferred to another according to a tree-shaped CP-net. The encoding in the ASPRIN library is more involved (because it must serve all use cases of the framework), and does not use the `#max` aggregate (because it is involved in a recursive definition).

```
better :- not eq, not fail.
eq :- holds(atom(A)) : holds'(atom(A));
   holds'(atom(A)) : holds(atom(A)).

p(I,N,A,pos) :- pref(I,N,atom(A)).
p(I,N,A,neg) :- pref(I,N,neg(atom(A))).

edge(P,Q) :- p(I,0,P,_), p(I,1,Q,_).
leaf(P) :- p(_,_,P,_), #count{Q:edge(P,Q)}=0.
root(Q) :- p(_,_,Q,_), #count{P:edge(P,Q)}=0.

match(A) :- atom(A), #count{
  1:holds(atom(A)); 2:holds'(atom(A))} != 1.

improvable(Q) :- p(I,0,P,SP), p(I,1,Q,SQ),
  #count{1: SP=pos; 2: holds'(atom(P))} != 1,
  #count{1: SQ=pos; 2: holds'(atom(Q))} = 1.

inc(P,Q) :- edge(P,Q), not improvable(Q),
  match(P), not match(Q).
dec(P,Q) :- edge(P,Q), improvable(Q),
  not match(P), match(Q).

lps(P,0) :- leaf(P).
lps(P,M) :- edge(P,_), M=#max{L:lps(P,Q,L)}.

lps(P,Q,L-1) :- dec(P,Q), lps(Q,L), L > 0.
lps(P,Q,L) :- dec(P,Q), lps(Q,L), L = 0.
lps(P,Q,L) :- edge(P,Q), not inc(Q),
  not dec(Q), lps(Q,L).
lps(P,Q,L+1) :- inc(P,Q), lps(Q,L).

fail :- root(A), not match(A), #count{
  1 : pref(I,1,atom(A)); 2 : holds'(A)} != 1.
fail :- root(A), lps(A,L), L > 0.
```

Figure 4: Preference program for tree-shaped CP-nets

6 Experiments

The different solving techniques for CP-nets available in ASPRIN are evaluated empirically on testcases generated from the 193 instances of [Brewka *et al.*, 2015b]. We obtained three testcases for each instance, generating a CP-net with a given structure (list-shaped, tree-shaped or acyclic) involving all atoms in the optimization statement of the original instance. We measured the time to compute one optimal answer set with up to five techniques: LOOPS, the algorithm in Section 5; USC, the approximation (Section 4) provided by weak constraints using unsatisfiable core analysis (`--opt-strat=usc` option); HEUR, the approximation provided by lexicographic compositions of subset preferences using domain heuristics (`#heuristic` directives); PL-SQ and PL-LIN, the algorithm in Section 3 respectively with quadratic and linear diameter. Experiments ran on an Intel Xeon 2.20GHz processor under Linux, and resources were limited to 3600 seconds of runtime and 8 GB of memory. Testcases and details can be downloaded at https://potassco.org/asprin/.

Experimental results are shown in Figure 5. For list-shaped and tree-shaped CP-nets we registered timeouts for almost all CP-nets with thousands of atoms, while for smaller CP-nets a good performance is achieved using the encoding from Section 5, or the approximation from Section 4 combined with unsatisfiable core analysis [Andres *et al.*, 2012; Alviano and Dodaro, 2016; Alviano and Dodaro, 2017] or domain heuristics [Gebser *et al.*, 2013]. The encoding from Section 3 is less competitive, even limiting the diameter to be quadratic or linear with respect to the domain of the processed CP-net. As expected, using a linear diameter results into better performance, but then the optimality of the solutions is not guaranteed, and indeed 3 of the 62 computed solutions were non-optimal. It is interesting to observe that tree-shaped instances are simpler than list-shaped instances, which is due to the fact that list-shaped instances have larger diameters.

Concerning acyclic CP-nets, LOOPS is not applicable, and the encoding from Section 3 using linear or quadratic diameters may compute non-optimal solutions (larger diameters led to much worse performance). Graphs were generated starting from a list comprising node 1 and 2, and then setting the parents of each $n \geq 3$ to be $n/2$ and some other randomly chosen node. Also for these testcases the encoding from Section 3 is

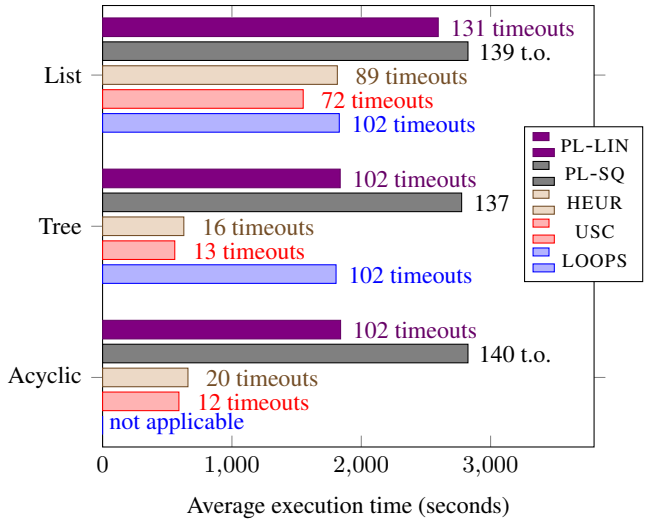

Figure 5: Experimental results aggregated per structure of generated CP-net. Each timeout contributes 3600 seconds to the average execution time.

not competitive with the approximation from Section 4.

To sum up, these preliminary experimental results well motivate the several encodings added to the ASPRIN library in order to integrate CP-nets. The linear encoding from Section 5 and the approximation from Section 4 are necessary in practice with CP-nets involving hundreds of atoms.

7 Related Work

Several works in constraint programming are related to this paper. CP-nets were used to express a preference relation over the models of knowledge bases encoded by hard constraints [Boutilier *et al.*, 2004b], and acyclic CP-nets were approximated by weighted soft constraints (essentially weak constraints) and used in combination with hard and soft constraints [Domshlak *et al.*, 2006]. In comparison, an advantage of ASPRIN is that CP-nets can be combined with many other preference relations, even those defined by the user. Moreover, the notion of approximation used in this paper is more general, as it is parametrized by an upstream expansion function for introducing auxiliary symbols [Alviano *et al.*, 2018]. Since approximations are closed under composition, the one in Section 4 is sufficient to map CP-nets to weak constraints or lexicographic compositions of subset preferences.

To the best of our knowledge, no implementation related to [Boutilier *et al.*, 2004b; Domshlak *et al.*, 2006] is available. Moreover, systems for reasoning with CP-nets alone cannot express some quantitative preferences [Domshlak *et al.*, 2006]. For example, minimizing (2 if `a,not b`) + (1 if `not a,b`) corresponds to the order $\{a,b\} \succ \{b\} \succ \{a\}$, $\emptyset \succ \{b\}$, and $\emptyset \succ \{a\}$, which cannot be represented by any CP net: for any CP net such that $\{b\} \succ \{a\}$, given that sets $\{b\}$ and $\{a\}$ differ in two atoms (a and b), there must be some set X such that $\{b\} \succ X$ and $X \succ \{a\}$; neither $X = \{a,b\}$ nor $X = \emptyset$ have this property. The same example applies to *more-or-less* CP nets [Yaman and desJardins, 2007].

Acyclic CP-nets were also approximated in the language of *answer set optimization* (*aso*) [Brewka *et al.*, 2003]. Actually, the ASPRIN library provides approximations for *aso* preference relations [Alviano *et al.*, 2018], and therefore one could encode acyclic CP-nets in *aso* and use ASPRIN for reasoning on them, as an alternative to the approximation presented in Section 4. However, the approximation provided by poset preserves more structure of the approximated CP-net. For example, consider a CP-net comprising $a : (\emptyset, \{\emptyset\})$, $b : (\emptyset, \{\emptyset\})$, and $c : (\{a\}, \{\{a\}\})$. Atoms b and c are unrelated in the approximation provided by poset, as well as in the CP-net. On the other hand, b is preferred to c in the approximation provided by aso (whose ranking gives priority to a and b over c).

The preference program in Section 3 is inspired by an analogous SAT encoding for dominance testing [Allen *et al.*, 2017]. In ASPRIN, the preference program finds several other applications. In fact, it is not only used by the sat-unsat search algorithm, but also processed by a meta-programming encoding in order to block all computed optimal answer sets and undesired interpretations [Gebser *et al.*, 2008]. Moreover, the preference program in the ASPRIN library is parametrized by the name of the processed CP-net, so to enable combinations with other preference relations. As for tree-shaped CP-nets, function *lps* is inspired by algorithm TREEDT, and provides an alternative proof for the nontrivial linear time complexity of the problem [Bigot *et al.*, 2013]; in fact, one can build a tree-shaped CP-net $\mathcal{N}$ over a domain $\mathcal{A}$, and interpretations I, I' such that the smallest sequence $I_0 = I, \ldots, I_n = I'$ with $(I_i, I_{i+1}) \in cp^{\mathcal{N}}$ (for all $i \in [1..n-1]$) contains $\Theta(|\mathcal{A}|^2)$ interpretations [Boutilier *et al.*, 2004a].

At a semantic level, in ASPRIN CP-nets are associated with a domain comprising all atoms occurring in their preference statements, and ceteris paribus rules are applied only to these atoms. A *don't care* atom `a` can be introduced with the help of a fresh atom `na` by adding the preference elements `a >> not a` and `na >> not na`, as well as the rule `na :- not a` to the knowledge base.

8 Conclusion

The integration of CP-nets in ASPRIN involved the definition of a preference program for the general case to enable the iterative sat-unsat search algorithm of the system. Moreover, driven by the liner time complexity of dominance checking for tree-shaped CP-nets, the ASPRIN library was further extended with a preference program tailored for tree-shaped CP-nets. Finally, the approximation given in terms of *poset* preference relations enables several other algorithms implemented by ASP solvers for computing optimal solutions.

References

[Allen *et al.*, 2017] Thomas E. Allen, Judy Goldsmith, Hayden Elizabeth Justice, Nicholas Mattei, and Kayla Raines. Uniform random generation and dominance testing for cp-nets. *J. Artif. Intell. Res.*, 59:771–813, 2017.

[Alviano and Dodaro, 2016] Mario Alviano and Carmine Dodaro. Anytime answer set optimization via unsatisfiable core shrinking. *TPLP*, 16(5-6):533–551, 2016.

[Alviano and Dodaro, 2017] Mario Alviano and Carmine Dodaro. Unsatisfiable core shrinking for anytime answer set optimization. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4781–4785. ijcai.org, 2017.

[Alviano *et al.*, 2018] Mario Alviano, Javier Romero, and Torsten Schaub. Preference relations by approximation. In Michael Thielscher, Francesca Toni, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018.*, pages 2–11. AAAI Press, 2018.

[Andres *et al.*, 2012] Benjamin Andres, Benjamin Kaufmann, Oliver Matheis, and Torsten Schaub. Unsatisfiability-based optimization in clasp. In Agostino Dovier and Vítor Santos Costa, editors, *Technical Communications of ICLP 2012, September 4-8, 2012, Budapest, Hungary*, volume 17 of *LIPIcs*, pages 211–221. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[Bigot *et al.*, 2013] Damien Bigot, Bruno Zanuttini, Hélène Fargier, and Jérôme Mengin. Probabilistic conditional preference networks. In Ann Nicholson and Padhraic Smyth, editors, *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*, pages 72–81. AUAI Press, 2013.

[Boutilier *et al.*, 2004a] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res.*, 21:135–191, 2004.

[Boutilier *et al.*, 2004b] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Preference-based constrained optimization with cp-nets. *Computational Intelligence*, 20(2):137–157, 2004.

[Brewka *et al.*, 2003] Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski. Answer set optimization. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 867–872. Morgan Kaufmann, 2003.

[Brewka *et al.*, 2015a] Gerhard Brewka, James P. Delgrande, Javier Romero, and Torsten Schaub. Implementing preferences with ASPRIN. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, volume 9345 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 2015.

[Brewka *et al.*, 2015b] Gerhard Brewka, James P. Delgrande, Javier Romero, and Torsten Schaub. ASPRIN: Customizing answer set preferences without a headache. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1467–1474. AAAI Press, 2015.

[Buccafurri *et al.*, 2000] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing Disjunctive Datalog by Constraints. *IEEE Trans. Knowl. Data Eng.*, 12(5):845–860, 2000.

[Domshlak *et al.*, 2006] Carmel Domshlak, Steven David Prestwich, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Hard and soft constraints for reasoning about qualitative conditional preferences. *J. Heuristics*, 12(4-5):263–285, 2006.

[Gebser *et al.*, 2008] Martin Gebser, Jörg Pührer, Torsten Schaub, and Hans Tompits. A meta-programming technique for debugging answer-set programs. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 448–453. AAAI Press, 2008.

[Gebser *et al.*, 2013] Martin Gebser, Benjamin Kaufmann, Javier Romero, Ramón Otero, Torsten Schaub, and Philipp Wanko. Domain-specific heuristics in answer set programming. In Marie desJardins and Michael L. Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.*, pages 350–356. AAAI Press, 2013.

[Gebser *et al.*, 2015] Martin Gebser, Amelia Harrison, Roland Kaminski, Vladimir Lifschitz, and Torsten Schaub. Abstract gringo. *TPLP*, 15(4-5):449–463, 2015.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Comput.*, 9(3/4):365–386, 1991.

[Goldsmith *et al.*, 2008] Judy Goldsmith, Jérôme Lang, Miroslaw Truszczynski, and Nic Wilson. The computational complexity of dominance and consistency in cp-nets. *J. Artif. Intell. Res.*, 33:403–432, 2008.

[Rosa *et al.*, 2010] Emanuele Di Rosa, Enrico Giunchiglia, and Marco Maratea. Solving satisfiability problems with preferences. *Constraints*, 15(4):485–515, 2010.

[Simons *et al.*, 2002] Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artif. Intell.*, 138(1-2):181–234, 2002.

[van Harmelen *et al.*, 2008] Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors. *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier, 2008.

[Yaman and desJardins, 2007] Fusun Yaman and Marie desJardins. More-or-less cp-networks. In Ronald Parr and Linda C. van der Gaag, editors, *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pages 434–441. AUAI Press, 2007.