On the Integration of CP-nets in ASPRIN

Mario Alviano\textsuperscript{1}, Javier Romero\textsuperscript{2} and Torsten Schaub\textsuperscript{2}

\textsuperscript{1}University of Calabria, Italy
\textsuperscript{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 reficiation 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 ⪰ approximates all preference relations being its subsets, and this property is sufficient to guarantee that all ⪰-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 (A, M) such that A is a (finite) set of (propositional) atoms, and M is a set of subsets of A. Set A is the domain of Γ, and each set in M is a model of Γ. A knowledge base is consistent if $M \neq \emptyset$, and inconsistent otherwise. A preference relation ⪰ over a domain A is a partial order over $2^A$, that is, ⪰ is a subset of $2^A \times 2^A$, and ⪰ is reflexive, antisymmetric and transitive. (Note that ⪰ also defines a preference relation over any superset of A in the obvious way, that is, $I \geq J$ if and only if $I \cap A \geq J \cap A$.) Let $\succ$ be the strict partial order obtained from ⪰; that is, $I \succ J$ if and only if $I \geq J$ and $J \not\geq I$. (Recall that a strict partial order is an irreflexive and transitive relation.) Let $\Gamma = (A, M)$ be a knowledge base, and ⪰ be a preference relation over A. $I \in \mathcal{M}$ is a $\preceq$-optimal model of $\Gamma$ if there is no $J \in \mathcal{M}$ such that $J \succ I$.

Approximations. Let $A, A'$ be sets of atoms such that $A \subseteq A'$. An expansion function from A to $A'$ is a function $e : 2^A \rightarrow 2^{A'}$ such that $e(I) \cap A = I$, for all $I \subseteq A$. Abusing notation, for a knowledge base $\Gamma = (A, M)$, let $e(\Gamma)$ be the knowledge base $\langle A', \{e(I) \mid I \in M\} \rangle$. Let $\preceq$ be a preference relation for A, and $e$ be an expansion function from A to $A'$. A preference relation $\preceq'$ is an approximation of $\preceq$ with respect to $e$, or e-approximation, if $I \succ J$ implies $e(I) \succ' e(J)$, for all $I, J \subseteq A$. 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 $e : 2^A \rightarrow 2^{A'}$, $e' : 2^A \rightarrow 2^{A''}$ be expansion functions. Their composition $e' \circ e : 2^A \rightarrow 2^{A''}$ 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 $\Gamma = (A, M)$ be a knowledge base, $\geq$ be a preference relation over A, e : $2^A \rightarrow 2^{A'}$ be an expansion function, and $\succeq'$ be an e-approximation of $\succeq$. For any $I \subseteq A$, if $e(I) \in A'$, $\succeq'$ is a $\succeq$-optimal model of $e(\Gamma)$, then I is a $\succeq$-optimal model of $\Gamma$.

Posets. Let $\succ$ be a strict partial order over a set $A$ of atoms, that is, $\succ$ is a subset of $A \times A$, $\succ$ is irreflexive ($p \not\succ p$ for all $p \in A$), and $\succ$ is transitive ($p_1 \succ p_2$ and $p_2 \succ p_3$ implies $p_1 \succ p_3$, for all $p_1, p_2, p_3 \in A$). The preference relation poset$^\succ$ over $A$ defined by $\succ$ is the reflexive (and transitive) closure of $\{(I, J) \mid I, J \subseteq A, I \cap J \neq \emptyset, \text{ for all } p \in J \setminus I \text{ there is } p' \in I \setminus J \text{ such that } p' \succ p\}$. [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 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.](image-url)
CP-nets. In order to uniform the notation, let us define a (Boolean) CP-net \( N \) over a domain \( A \) as a set comprising conditional ceteris paribus statements of the form \( p : (A_p, M_p) \) for each \( p \in A \), where \( A_p \subseteq A \setminus \{p\} \), and \((A_p, M_p)\) is a knowledge base. The dependency graph \( G_N \) of \( N \) has nodes \( A \), and arcs from all atoms in \( A_p \) to \( p \), for all \( p : (A_p, M_p) \) in \( N \). \( N \) is acyclic if \( G_N \) is acyclic. The preference relation \( cp^N \) defined by \( N \) is the reflexive, transitive closure of \[
\{(I \cup \{p\}, I \setminus \{p\}) \mid p : (A_p, M_p) \text{ belongs to } N, \ I \subseteq A, I \cap A_p \in M_p\} \cup \{(I \setminus \{p\}, I \cup \{p\}) \mid p : (A_p, M_p) \text{ belongs to } N, \ I \subseteq A, I \cap A_p \in M_p\}.
\]

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

Example 2. Let \( N \) be the following CP-net:

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

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

A CP-net \( N \) is tree-shaped if \( G_N \) is a forest (i.e., a set of trees), and each ceteris paribus statement \( p : (A_p, M_p) \) such that \( A_p \) is empty (i.e., \( p \) is a root) or \( M_p \) is a singleton (for otherwise, \( A_p \) can be reduced to the empty set). For a tree-shaped CP-net \( N \) over a domain \( A \), and two sets \( I, I' \subseteq A \), \( cp^N I \) can be checked in time \( O(|A|^2) \) 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 \( G \), let subtree(G, p) be the subtree of \( G \) whose root is \( p \). An atom \( p \in A \) such that (flip(p, I'), I') belongs to \( cp^N \) is said \( I' \)-improvable; \( p \) is \( G \)-I'-improvable if \( p \) is the only \( I' \)-improvable node in subtree(G, p). Hence, the algorithm removes leaves that match in \( I \) and \( I' \) (lines 3–4), and flips \( G \)-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).

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 \( A \) and answer sets \( M \) encodes the knowledge base \( (A, M) \), while preference relations are declared by

![Figure 2: Dependency graph from Example 2, and preference tables of domain atoms.](image)

Algorithm 1: TreeDT(N, I, I')

1. \( G := G_N \)
2. loop
3. while \( G \) has a leaf \( p \) s.t. \( \{p\} \cap I = \{p\} \cap I' \) 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. \( I' := \text{flip}(p, I') \) for some \( p \) being \( G \)-I'-improvable

states of the following form:

\[\#\text{ preference}(id, type)\{e_1; \ldots; e_n\}.\]

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 \( \ell \) 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 \( p : (A_p, M_p) \) with \( A_p \neq \emptyset \) is encoded by \( p \parallel L_1; \cdots; p \parallel L_m; \text{not} p \parallel L_{m+1}; \cdots; \text{not} p \parallel L_n \), where \( \bigvee_{i=1}^m L_i \) (and equivalently \( \bigwedge_{i=m+1}^n L_i \)) encodes \((A_p, M_p); p : (\emptyset, \{\emptyset\})\) is encoded by \( p \), and \( p : (\emptyset, 0) \) is encoded by not \( p \).

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

\[
\#\text{ preference}(cp\text{-from example}, cp) \{ \begin{array}{l}
a; b; c \parallel \{a; \text{not} b\}; c \parallel \{\text{not} a; b\};
\end{array}
\]

\[\text{not} c \parallel \{\text{not} a; \text{not} b\}; \text{not} c \parallel \{a; b\}.\]

(Preference element \( a \) can be also written \( a \) >> \text{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 \( N \) over a domain \( A \). ASPRIN reifies the \( \ell \)-th preference element \( \ell : \{\ell_1; \ldots; \ell_n\} (n \geq 0) \) as follows:

\[
\begin{align*}
\text{pref}(i, 1, \text{encode}(\ell_1)). \\
\text{pref}(i, 0, \text{encode}(\ell_j)). \\
\end{align*}
\]

where \( \text{encode} \) maps \( p \) to \( \text{atom}(p) \), and \( \text{not} p \) to \( \neg \text{atom}(p) \), for every atom \( p \). (Actually, ASPRIN uses predicate \( \text{pref} \) 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 \( \Pi_N \) be the reification of all preference elements in \( N \).

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

\[
\begin{align*}
\text{pref}(1, 1, \text{atom}(a)). \\
\text{pref}(2, 1, \text{atom}(b)). \\
\text{pref}(3, 1, \text{atom}(c)). \\
\end{align*}
\]

\[
\text{pref}(3, 0, \text{atom}(a)). \\
\text{pref}(3, 0, \text{not}\text{atom}(b)).
\]

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

\[ \text{eq} := \text{holds}(\text{atom}(A)) : \text{holds}'(\text{atom}(A)); \]
\[ \text{holds}'(\text{atom}(A)) : \text{holds}(\text{atom}(A)). \]

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

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

A sequence of preference elements so to reach a better interpretation encoded by predicate holds. A preference element \( \ell \mid L \) 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 \( \ell \) at time \( t \). The length of the plan is bounded by the numeric constant diameter, which generally is assigned the value \( 2^{|A|} \); smaller values can be used under some conditions (for example, \( |A|^2 \) is sufficient if \( G_N \) is a forest [Boutilier et al., 2004a]).

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

\[ \text{#const diameter = d.} \]
\[ :- \text{not better.} \]
\[ \text{holds( atom(p) ).} \quad \forall p \in I \]
\[ \text{holds'( atom(p) ).} \quad \forall p \in I' \]

The invariant is stated by the next claim.

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

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

Theorem 1 guarantees correctness of the iterative sat-unsat search algorithm implemented in ASPRIN for computing one \( cp^N \)-optimal answer set of a program \( \Pi \). 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 \( cp^N \)-optimal answer set \( I \), and any interpretation \( I' \) such that \( I \cap A, I' \cap A \in cp^N \). The process is completely automated by ASPRIN, and described in detail in the literature [Brewka et al., 2015a].

**Algorithm 2:** TREEDET-restated(\( N, I, I' \))

\[ // \text{ Lines 1-6 from TREEDET} \]
\[ 7 \text{ I':= flip(subtree}(G_N, p), I') \text{ for all } G-\text{I'}-\text{improvable } p \]

4 From Acyclic CP-nets to Posets

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

\[ e^N(I) := I \cup \{x_p \mid \{A_p, M_p\} \text{ belongs to } N, \]
\[ p \in I, I \cap A_p \in M_p\} \]
\[ \cup \{x_p \mid \{A_p, M_p\} \text{ belongs to } N, \]
\[ p \in A \setminus I, I \cap A_p \notin M_p\}. \]

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

**Example 6.** For the CP-net \( N \) from Example 2, the expansion function \( e^N \) is schematically shown in Figure 1:

\[ \{a, b\} \rightarrow \{a, b, x_a, x_b, x_c\}, \{a, b, c\} \rightarrow \{a, b, c, x_a, x_b\}, \]
\[ \{a, c\} \rightarrow \{a, c, x_a, x_c\}, \]

and so on. Moreover, \( \succ^N \) 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 \( N \) be a (Boolean) acyclic CP-net over \( A \). Relation \( poset^{\succ^N} \) is an \( e^N \)-approximation of \( cp^N \).

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

An interesting observation is that, after each step of the restated algorithm, nodes in \( subtree(p) \) 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 \( N \) and sets \( I, I' \)):

\[
\text{lps}(p) := \begin{cases} 
0 & \text{if } p \text{ is a leaf} \\
\max_{q \in G} \text{lps}(p, q) & \text{otherwise}
\end{cases}
\]

\[
\text{lps}(q, p) := \begin{cases} 
\text{lps}(q) + 1 & \text{if } q \text{ is not } I' \text{-improvable,} \\
\max(0, \text{lps}(q) - 1) & \text{if } q \text{ is } I' \text{-improvable,} \\
\text{lps}(q) & \text{otherwise}
\end{cases}
\]

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

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
to check all use cases of the framework), and does not use the \#max
aggregate (because it is involved in a recursive definition).

## 6 Experiments

The different solving techniques for CP-nets available in ASPRIN are evaluated empirically on testcases generated from
the 193 instances of I\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 bet-
ter 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

\[
\begin{align*}
\text{edge}(P,Q) & : p(I,0,P,\_), \ p(I,1,Q,\_). \\
\text{leaf}(P) & : p(\_,\_,P,\_), \ #\text{count}(Q:\text{edge}(P,Q)) = 0. \\
\text{root}(Q) & : p(\_,\_,\_,\_), \ #\text{count}(P:\text{edge}(P,Q)) = 0. \\
\text{match}(A) & : \text{atom}(A), \ #\text{count}(1 : \text{holds}(\text{atom}(A)); 2 : \text{holds}'(\text{atom}(A))) = 1. \\
\text{improvable}(Q) & : p(I,0,P,SP), \ p(I,1,Q,SO), \ #\text{count}(1 : SP=\text{pos}; 2 : \text{holds}'(\text{atom}(P))) = 1, \ #\text{count}(1 : SQ=\text{pos}; 2 : \text{holds}'(\text{atom}(Q))) = 1. \\
\text{inc}(P,Q) & : \text{edge}(P,Q), \ \text{not} \ \text{improvable}(Q), \ \text{match}(P), \ \text{not} \ \text{match}(Q). \\
\text{dec}(P,Q) & : \text{edge}(P,Q), \ \text{improvable}(Q), \ \text{not} \ \text{match}(P), \ \text{match}(Q). \\
\text{lps}(P,0) & : \text{leaf}(P). \\
\text{lps}(P,M) & : \text{edge}(P,\_), \ M = \max(L: \text{lps}(P,Q,L)). \\
\text{lps}(P,Q,L) & : \text{dec}(P,Q), \ \text{lps}(Q,L), \ L > 0. \\
\text{lps}(P,Q,L) & : \text{dec}(P,Q), \ \text{lps}(Q,L), \ L = 0. \\
\text{lps}(P,Q,L) & : \text{edge}(P,Q), \ \text{not} \ \text{inc}(Q), \ \text{not} \ \text{dec}(Q), \ \text{lps}(Q,L). \\
\text{lps}(P,Q,L+1) & : \text{inc}(P,Q), \ \text{lps}(Q,L). \\
\end{align*}
\]
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 \( \{a, b\} \succ \{b\} \succ \{a\} \succ \emptyset \), 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 \( a : (\emptyset, \{0\}) \), \( b : (\emptyset, \{0\}) \), 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 \( \text{lps} \) is inspired by algorithm TREEEDT, 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 \( N \) over a domain \( A \), and interpretations \( I, I' \) such that the smallest sequence \( I_0 = I, \ldots, I_n = I' \) with \( (I_i, I_{i+1}) \in \text{cp}^N \) (for all \( i \in [1..n-1] \)) contains \( \Theta(|A|^2) \) 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 \( \text{don’t care} \) atom \( a \) can be introduced with the help of a fresh atom \( \text{na} \) by adding the preference elements \( a \succ \text{not } a \) and \( \text{na} \succ \text{not } \text{na} \), as well as the rule \( \text{na} \succ \text{not } a \) 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 linear 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 \( \text{poset} \) preference relations enables several other algorithms implemented by ASP solvers for computing optimal solutions.

References


