The computational complexity of dominance and consistency in CP-nets

Judy Goldsmith
Dept. of Comp. Sci.
University of Kentucky
Lexington, KY
40506-0046, USA
goldsmit@cs.uky.edu

Jérôme Lang
IRIT
Université Paul Sabatier
31062 Toulouse Cedex
France
lang@irit.fr

Miroslaw Truszczyński
Dept. of Comp. Sci.
University of Kentucky
Lexington, KY
40506-0046, USA
mirek@cs.uky.edu

Nic Wilson
Cork Constraint
Computation Centre
University College Cork
Ireland
n.wilson@4c.ucc.ie

Abstract

We investigate the computational complexity of testing dominance and consistency in CP-nets. Up until now, the complexity of dominance has been determined only for restricted classes in which the dependency graph of the CP-net is acyclic. However, there are preferences of interest that define cyclic dependency graphs; these are modeled with general CP-nets. We show here that both dominance and consistency testing for general CP-nets are PSPACE-complete. The reductions used in the proofs are from STRIPS planning, and thus establish strong connections between both areas.

1 Introduction

The problems of eliciting, representing and reasoning with preferences over a multivariable (or, multiattribute) domain arise in many fields such as planning, design, and group decision making. An explicit representation of a preference ordering of elements (we refer to them as *outcomes*) of such multivariable domains is exponentially large in the number of attributes. Therefore, AI researchers have developed languages for representing preference orderings succinctly. The formalism of CP-nets [Boutilier *et al.*, 1999] is among the most popular ones. A CP-net provides a succinct representation of preference ordering on outcomes in terms of local preference statements of the form $p : x_i > x_j$, where x_i, x_j are values of a variable X and p is a logical condition. Informally, a preference statement $p : x_i > x_j$ means that given p, x_i is (strictly) preferred to x_j *ceteris paribus*, that is, *all other things being equal*. The meaning of a CP-net is given by a certain ordering relation (*dominance*) on the set of outcomes, derived from such reading of preference statements. If one outcome dominates another, we say that the dominant one is preferred.

Reasoning about the preference ordering (dominance relation) expressed by a CP-net is far from easy. The key problems include *dominance testing* and *consistency testing*. In the first problem, given a CP-net and two outcomes α and β, we want to decide whether β dominates α. The second problem asks whether there is a dominance cycle in the dominance ordering defined by an input CP-net, that is, whether there is an outcome that dominates (is preferred to) itself.

We study the computational complexity of these two problems. The results obtained so far concerned only restricted classes of CP-nets, all requiring that the graph of variable dependencies implied by preference statements in the CP-net be *acyclic*. Under certain assumptions, the dominance-testing problem is in NP and, under some additional assumptions, even in P [Domshlak and Brafman, 2002; Boutilier *et al.*, 2004a]. Its complexity in the general case has remained until now an open problem. We show that it is in fact PSPACE-complete, even for the propositional case, by exhibiting in Section 4 a PSPACE-hardness proof for dominance testing.

We then turn to consistency testing. While acyclic CP-nets are guaranteed to be consistent, this is not the case with general CP-nets (see [Domshlak and Brafman, 2002; Brafman and Dimopoulos, 2004] for detailed examples and discussions). In Section 5, we show that consistency testing is as hard as dominance testing.

To prove the hardness part of the results, we first establish the PSPACE-hardness of some problems related to propositional STRIPS planning. We then show that these problems can be reduced to CP-net dominance and consistency testing by exploiting connections between actions in STRIPS planning and preference statements in CP-nets.

We assume some familiarity with the complexity class PSPACE (we refer to [Papadimitriou, 1994] for details). In particular, we rely later on the equivalences NPSPACE = PSPACE = coPSPACE.

The complexity results in this paper address cyclic CP-nets. Most earlier work has concentrated on the acyclic model. However, we argue that acyclic CP-nets are not sufficiently expressive to capture human preferences on even some simple domains. Consider, for instance, a diner who has to choose either red or white wine, and either fish or meat. Given red wine, they prefer meat, and conversely, given meat they prefer red wine. On the other hand, given white wine, they prefer fish, and conversely, given fish they prefer white wine. This gives a consistent cyclic CP-net, and there is no acyclic CP-net giving rise to the same preferences on outcomes. So, such cyclicity of preference variables does not necessarily lead to a cyclic order on outcomes.

2 Generalized propositional CP-nets

Let $V = \{x_1, \dots, x_n\}$ be a finite set of *variables*. For each variable $x \in V$, we assume a finite *domain* D_x of *values*. An *outcome* is an n-tuple $(d_1, \ldots, d_n)$ of $D_{x_1} \times \cdots \times D_{x_n}$.

In this paper, we focus on *propositional* variables: variables with *binary* domains. Let V be a finite set of propositional variables. For every $x \in V$, we set $D_x = \{x, \neg x\}$ (thus, we overload the notation and write x both for the variable and for one of its values). We refer to x and $\neg x$ as literals. Given a literal l we write $\neg l$ to denote the dual literal to l. The focus on binary variables makes the presentation clearer and has no impact on our complexity results.

A *conditional preference rule* (or, for short, a [preference] rule) over V is an expression $p : l > \neg l$, where l is a literal of some atom $x \in V$ and p is a propositional formula over V that does not involve variable x.

Definition 1 (Generalized CP-net) *A* generalized CP-net *C (for short, a* GCP-net*) over V is a set of conditional preference rules. For $x \in V$ we define $p_C^+(x)$ and $p_C^-(x)$, usually written just: $p^+(x)$ and $p^-(x)$, as follows: $p_C^+(x)$ is equal to the disjunction of all p such that there exists a rule $p : x > \neg x$ in C; $p_C^-(x)$ is the disjunction of all p such that there exists a rule $p : \neg x > x$ in C. We define the associated directed graph G_C (the* dependency graph*) over V to consist of all pairs (y, x) of variables such that y appears in either $p^+(x)$ or $p^-(x)$.*

In our complexity results we will also need the following representation of GCP-nets: a GCP-net C is said to be in *conjunctive form* if C only contains rules $p : l > \neg l$ such that p is a (possibly empty) conjunction of literals. In this case all formulas $p^-(x)$, $p^+(x)$ are in disjunctive normal form, that is, a disjunction of conjunctions of literals (including $\top$ – the empty conjunction of literals).

GCP-nets determine a transitive relation on outcomes, interpreted in terms of preference. A preference rule $p : l > \neg l$ represents the statement "given that p holds, l is preferred to $\neg l$ *ceteris paribus*". Its intended meaning is as follows. If outcome β satisfies p and l, then β is preferred to the outcome α which differs from β only in that it assigns $\neg l$ for variable x. In this situation we say that there is *an improving flip* from α to β sanctioned by the rule $p : l > \neg l$. If $\alpha_0, \ldots, \alpha_m$ is a sequence of outcomes with $m \geq 1$ and each next outcome in the sequence is obtained from the previous one by an improving flip, then we say that $\alpha_0, \ldots, \alpha_m$ is an *improving sequence* (from α_0 to α_m) for the GCP-net, and that α_m *dominates* α_0, written $\alpha_0 \prec \alpha_m$.

Finally, A GCP-net is *consistent* if there is no outcome α which is strictly preferred to itself, i.e., such that $\alpha \prec \alpha$.

The main objective of the paper is to establish the complexity of the following two problems concerning the notion of dominance associated with GCP-nets (sometimes under restrictions on the class of input GCP-nets).

Definition 2
GCP-DOMINANCE*: given a GCP-net C and two outcomes α and β, decide whether $\alpha \prec \beta$ in C.*
GCP-CONSISTENCY*: given a GCP-net C, decide whether C is consistent.*

There are two properties of GCP-nets that are essential in linking them to the original notion of CP-nets [Boutilier *et al.*, 1999; 2004a].

Definition 3
A GCP-net C over V is locally consistent *if for every $x \in V$, the formula $p_C^-(x) \wedge p_C^+(x)$ is unsatisfiable. It is* locally complete *if for every $x \in V$, the formula $p_C^-(x) \vee p_C^+(x)$ is a tautology.*

Definition 4 (Propositional CP-net) *A* CP-net *over the set V of (propositional) variables is a locally consistent and locally complete GCP-net over V.*

Problems CP-DOMINANCE and CP-CONSISTENCY are defined analogously to Definition 2.

This definition of a CP-net differs from the one given in [Boutilier *et al.*, 2004a], which uses explicit conditional preference tables. Our representation is often more compact, but it is easy to verify that it is equivalent in that it gives rise to the same definition of dominance.

When defining a decision problem, it is critical to specify the way to represent its input instances, as the representation may affect the complexity of the problem. Unless stated otherwise, we assume that GCP-nets (and so, also CP-nets) are represented as a set of preference rules, as described in Definition 1. Therefore, the size of a GCP-net is given by the total size of the formulas $p^-(x)$, $p^+(x)$, $x \in V$.

3 Propositional STRIPS planning

In this section we derive some technical results on propositional STRIPS planning which form the basis of our complexity results in Sections 4 and 5. We establish the complexity of plan existence problems for propositional STRIPS planning, under restrictions on input instances that make the problem useful in the studies of dominance and consistency in GCP-nets.

Let V be a finite set of variables. A *state* over V is a complete and consistent set of literals over V (which we often view as the conjunction of its members). A state is therefore equivalent to an *outcome*, defined in a CP-nets context.

Definition 5 (Propositional STRIPS planning) *A* propositional STRIPS instance *is a tuple $\langle V, \alpha_0, \gamma, ACT \rangle$, where*

1. *V is a finite set of propositional variables;*
2. *α_0 is a state (over V), called the* initial state*;*
3. *γ is a state called the* goal*;*
4. *ACT is a finite set of* actions*, where each action $a \in ACT$ is described by a consistent conjunction of literals $pre(a)$ (a* precondition*) and a consistent conjunction of literals $post(a)$ (a* postcondition, or effect*).*

An action a is executable *in a state α if $\alpha \models pre(a)$. The* effect *of a in state α is the state α' containing the same literals as α for all variables not mentioned in $post(a)$, and the literals of $post(a)$ otherwise. We assume that an action can be* applied *to any state, but that it has no effect if its preconditions do not hold (this assumption has no effect on complexity).*
The PROPOSITIONAL STRIPS PLAN EXISTENCE *problem, or* STRIPS PLAN *for short, is to decide whether for a given propositional STRIPS instance $\langle V, \alpha_0, \gamma, ACT \rangle$ there is a successful plan, that is, sequence of actions leading from the initial state α_0 to a state satisfying the goal γ.*
A plan is irreducible *if its every action changes the state.*

We assume, without loss of generality, that for any action a, no literal in $post(a)$ also appears in $pre(a)$; otherwise we can omit the literal from $post(a)$ without changing the effect of the action; (if $post(a)$ then becomes an empty conjunction, the action a can be omitted from ACT as it has no effect).

We have the following result, proved in [Bylander, 1994] :

Proposition 1 *([Bylander, 1994])*
STRIPS PLAN *is* PSPACE*-complete.*

Typically, propositional STRIPS instances do not require that goals be complete. We restrict consideration to complete goals. This restriction has no effect on the complexity: the plan existence problem remains PSPACE-complete under the goal-completeness restriction [Lang, 2004].

3.1 Acyclic STRIPS

Definition 6 (Acyclic sets of actions) *A set of actions ACT is* acyclic *if there is no state α such that $\langle V, \alpha, \alpha, ACT\rangle$ has a non-empty irreducible plan (informally, if there are no non-trivial directed cycles in the space of states induced by ACT).*

We will now consider the following two problems:

1. ACYCLIC STRIPS PLAN: given a propositional STRIPS instance $\langle V, \alpha_0, \gamma, ACT\rangle$ such that ACT is acyclic and $\alpha_0 \neq \gamma$, decide whether there is a plan for $\langle V, \alpha_0, \gamma, ACT\rangle$.
2. ACTION-SET ACYCLICITY: given a set ACT of actions, decide whether ACT is acyclic.

We will show that both problems are PSPACE-complete.

Proposition 2
ACYCLIC STRIPS PLAN *is* PSPACE*-complete.*

Proof: Membership in PSPACE is evident as the problem is a restriction of STRIPS PLAN. To prove PSPACE-hardness, we exhibit a polynomial-time reduction from STRIPS PLAN. Let *PE* $= \langle V, \alpha_0, \gamma, ACT\rangle$ be an instance of STRIPS PLAN. The idea behind the reduction is to introduce a *counter*, so that each time an action is executed, the counter is incremented. The counter may count up to 2^n, where $n = |V|$, making use of n additional variables. The counter is initialized to 0. Once it reaches $2^n - 1$ it can no longer be incremented and no action can be executed. Hence, the set of actions in the resulting instance of STRIPS PLAN is acyclic.

To describe the reduction, we write V as $\{x_1, \ldots, x_n\}$. We define *PE*$' = \langle V', \alpha_0', \gamma', ACT'\rangle$ as follows:

- $V' = \{x_1, \ldots, x_n, z_1, \ldots, z_n\}$, where z_i are new variables we will use to implement the counter;
- $\alpha_0' = \alpha_0 \wedge \neg z_1 \wedge \cdots \wedge \neg z_n$;
- $\gamma' = \gamma \wedge z_1 \wedge \cdots \wedge z_n$;
- for each action $a \in ACT$, we include in ACT' n actions a^i, $1 \leq i \leq n$, such that
  $pre(a^i) = pre(a) \wedge \neg z_i \wedge z_{i+1} \wedge \cdots \wedge z_n$
  $post(a^i) = post(a) \wedge z_i \wedge \neg z_{i+1} \wedge \cdots \wedge \neg z_n$
- Furthermore, we include in ACT' n actions b_i, $1 \leq i \leq n$, such that
  $pre(b_i) = \neg z_i \wedge z_{i+1} \wedge \cdots \wedge z_n$
  $post(b_i) = z_i \wedge \neg z_{i+1} \wedge \cdots \wedge \neg z_n$

We will denote states over V' by pairs (α, k), where α is a state over V and k is an integer, $0 \leq k \leq 2^n - 1$. We view k as a compact representation of a state over variables $z_1, \ldots, z_n$: assuming that the binary representation of k is $d_1 \ldots d_n$ (with d_n being the least significant digit), k represents the state which contains z_i if $d_i = 1$ and $\neg z_i$, otherwise.

PE$'$ is acyclic, since executing any action in ACT' increments the counter k and no action can be executed once the counter has reached the value $2^n - 1$.

We claim that there is a plan for *PE* if and only if there is a plan for *PE*$'$. First, assume that there is a plan in *PE*. Let π be a shortest plan in *PE* and let m be its length. We have $m \leq 2^n - 1$, since no state along π repeats (otherwise, shorter plans than π for *PE* would exist). Let $\alpha_0, \alpha_1, \ldots, \alpha_m = \gamma$ be the sequence of states obtained by executing π. Let a be the action used in the transition from α_k to α_{k+1}. Since $k < 2^n - 1$, there is exactly one i, $1 \leq i \leq n$, such that the action a^i applies at the state (α, k) over V'. Replacing a with a^i in π yields a plan that when started at $(\alpha_0, 0)$ leads to $(\alpha_m, m) = (\gamma, m)$. Appending that plan with appropriate actions b_i to increment the counter to $2^n - 1$ yields a plan for *PE*$'$. Conversely, if τ is a plan for *PE*$'$, the plan obtained from τ by removing all actions of the form b_j and replacing each action a^i with a is a plan for *PE*.

Thus, the claim and the assertion follow. ■

Proposition 3
ACTION-SET ACYCLICITY *is* PSPACE*-complete.*

Proof: The argument for the membership in PSPACE is standard. To prove PSPACE-hardness, we proceed as follows. Let *PE* $= \langle V, \alpha_0, \gamma, ACT\rangle$ be a STRIPS instance such that ACT is acyclic and $\alpha_0 \neq \gamma$. Let a be a new action defined by $pre(a) = \gamma$ and $post(a) = \alpha_0$. It is easy to see that $ACT \cup \{a\}$ is not acyclic if and only if there exists a plan for *PE*. Thus, the PSPACE-hardness of the *complement* of the ACTION-SET ACYCLICITY problem follows from Proposition 2. Since PSPACE = coPSPACE, this suffices to prove the hardness part of the assertion. ■

3.2 Mapping STRIPS plans to single-effect STRIPS plans

Versions of the STRIPS PLAN and ACYCLIC STRIPS PLAN problems that are important for us allow only single-effect actions (actions with exactly one literal in their postconditions) in input propositional STRIPS instances. We refer to these restrictions as SE STRIPS PLAN and ACYCLIC SE STRIPS PLAN.

To prove PSPACE-hardness of both problems, we describe a mapping from STRIPS instances to single-effect STRIPS instances.

Consider an instance *PE* $= \langle V, \alpha_0, \gamma, ACT\rangle$ of the STRIPS PLAN problem (where ACT is not necessarily acyclic). For each action $a \in ACT$ we introduce a *new* variable x_a. We set $X = \bigwedge_{a \in ACT} \neg x_a$. That is, X is the conjunction of negative literals of all the additional variables. In addition, for each $a \in ACT$ we set $X_a = x_a \wedge \bigwedge_{b \in ACT - \{a\}} \neg x_b$. We now define an instance *PE*$' = \langle V', \alpha_0', \gamma', S(ACT)\rangle$ of the SE STRIPS PLAN problem as follows:

- Set of variables: $V' = V \cup \{x_a : a \in ACT\}$;
- Initial state: $\alpha_0' = \alpha_0 \wedge X$;
- Goal state: $\gamma' = \gamma \wedge X$;
- Set of actions: $S(ACT) = \{a^i : a \in ACT,\ i = 1, \ldots, 2|post(a)| + 1\}$.
  Let $a \in ACT$ and $post(a) = l_1 \wedge \cdots \wedge l_q$. For $i = 1, \ldots, q$, we define:
  $pre(a^i) = pre(a) \wedge X \wedge \neg l_i;\ post(a^i) = x_a;$
  $pre(a^{q+i}) = X_a;\ post(a^{q+i}) = l_i.$
  We also define:
  $pre(a^{2q+1}) = X_a \wedge l_1 \wedge \cdots \wedge l_q;\ post(a^{2q+1}) = \neg x_a.$

Let π be a sequence of actions in ACT. We define $S(\pi)$ to be the sequence of actions in $S(ACT)$ obtained by replacing each action a in π by $a^1, \ldots, a^{2q+1}$ (where $q = |post(a)|$). Now consider a sequence τ of actions from $S(ACT)$. Remove from τ any action a^i such that $i \neq 2|post(a)| + 1$, and replace actions of the form $a^{2|post(a)|+1}$ by a. We denote the resulting sequence of actions from ACT by $S'(\tau)$. The following properties are easy to check (details are omitted):

Lemma 1 *With the above definitions,*

(i) if π is a plan for PE then $S(\pi)$ is a plan for PE$'$;

(ii) if τ is an irreducible plan for PE$'$ then $S'(\tau)$ is a plan for PE;

(iii) ACT is acyclic if and only if $S(ACT)$ is acyclic.

Proposition 4
SE STRIPS PLAN *and* ACYCLIC SE STRIPS PLAN *are* PSPACE*-complete.*

Proof: SE STRIPS PLAN and ACYCLIC SE STRIPS PLAN problems are restrictions of STRIPS PLAN, from which membership in PSPACE follows. PSPACE-hardness of ACYCLIC SE STRIPS PLAN (and so, also of the other problem) is shown by reduction from ACYCLIC STRIPS PLAN. Consider an instance $PE = \langle V, \alpha_0, \gamma, ACT\rangle$ of ACYCLIC STRIPS PLAN. Define $PE' = \langle V', \alpha_0', \gamma', S(ACT)\rangle$, which by Lemma 1(iii) is an instance of the ACYCLIC SE STRIPS PLAN problem. By Lemma 1(i) and (ii) there exists a plan for *PE* if and only if there exists a plan for *PE$'$*. ■

4 Dominance

The goal of this section is to prove that the GCP-DOMINANCE problem is PSPACE-complete, and that the complexity does not go down even when we restrict the class of inputs to CP-nets. We use the results on propositional STRIPS planning from Section 3 to prove that the general GCP-DOMINANCE problem is PSPACE-complete. We then show that the complexity does not change if we impose the requirements of local consistency and local completeness on input GCP-nets.

The similarities between dominance testing in CP-nets and propositional STRIPS planning were first noted in [Boutilier *et al.*, 1999], where a reduction (presented in more detail in [Boutilier *et al.*, 2004a]) was given from the dominance problem to the plan existence problem for a class of propositional STRIPS planning specifications consisting of *unary* actions (actions with single effects). We prove our results for the GCP-DOMINANCE and GCP-CONSISTENCY problems by constructing a reduction in the other direction.

This reduction is much more complex than the one used in [Boutilier *et al.*, 1999], due to the fact that CP-nets impose more restrictions than STRIPS planning. Firstly, STRIPS planning allows multiple effects, but GCP-nets only allow flips $x > \neg x$ or $\neg x > x$ that change the value of one variable; this is why we constructed the reduction from STRIPS planning to single-effect STRIPS planning in the last section. Secondly, CP-nets impose two more restrictions, local consistency and local completeness, which do not have natural counterparts in the context of STRIPS planning.

For all dominance and consistency problems considered in the paper, the membership in PSPACE can be demonstrated by considering nondeterministic algorithms consisting of repeatedly guessing appropriate improving flips. Such algorithms work in polynomial space and show the membership of problems they solve in NPSPACE and consequently in PSPACE, since NPSPACE = PSPACE. Therefore, due to space restrictions, from now on we only provide arguments for the PSPACE-hardness of problems we consider.

4.1 Dominance for generalized CP-nets

We will prove that the GCP-DOMINANCE problem is PSPACE-complete by a reduction from the problem SE STRIPS PLAN, which we now know to be PSPACE-complete.

Mapping single-effect STRIPS problems to GCP-nets dominance problems

Let $\langle V, \alpha_0, \gamma, ACT\rangle$ be an instance of the SE STRIPS PLAN problem. For every action $a \in ACT$ we denote by l_a the unique literal in the postcondition of a, that is, $post(a) = l_a$. We denote by $pre'(a)$ the conjunction of all literals in $pre(a)$ different from $\neg l_a$ (we recall that by a convention we adopted earlier, $pre'(a)$ does not contain l_a either). We then define c_a to be the conditional preference rule $pre'(a) : l_a > \neg l_a$ and define $M(ACT)$ to be the GCP-net $C = \{c_a : a \in ACT\}$.

A sequence of states in a plan corresponds to an improving sequence from α_0 to γ, which leads to the following result.

Lemma 2 *With the above notation,*

(i) there is a non-empty irreducible plan for $\langle V, \alpha_0, \gamma, ACT\rangle$ if and only if γ dominates α_0 in C;

(ii) ACT is acyclic if and only if $M(ACT)$ is consistent.

Theorem 1 *The* GCP-DOMINANCE *problem is* PSPACE*-complete. Moreover, this remains so under the restrictions that the GCP-net is consistent and is in conjunctive form.*

Proof: PSPACE-hardness is shown by reduction from ACYCLIC SE STRIPS PLAN (Proposition 4). Let $\langle V, \alpha_0, \gamma, ACT\rangle$ be an instance of the ACYCLIC SE STRIPS PLAN problem. By Lemma 2(ii), $M(ACT)$ is a consistent GCP-net in conjunctive form. Since $\alpha_0 \neq \gamma$, there is a plan for $\langle V, \alpha_0, \gamma, ACT\rangle$ if and only if there is a non-empty irreducible plan for $\langle V, \alpha_0, \gamma, ACT\rangle$, which, by Lemma 2(i), is if and only if γ dominates α_0 in C. ■

4.2 Dominance in CP-nets

In this section we show that GCP-DOMINANCE remains PSPACE-complete under the restriction to locally-consistent and locally-complete GCP-nets, i.e., CP-nets. We refer to this restriction of GCP-DOMINANCE as CP-DOMINANCE.

We will show PSPACE-hardness for CP-DOMINANCE by a reduction from GCP-DOMINANCE for consistent GCP-nets.

Mapping locally-consistent GCP-nets to CP-nets

Let C be a locally-consistent GCP-net. Let $V = \{x_1, \dots, x_n\}$ be the set of variables of C. We define $V' = V \cup \{y_1, \dots, y_n\}$, where $\{y_1, \dots, y_n\} \cap V = \emptyset$. We define a GCP-net C' over V', which we will show is a CP-net. To this end, for every $z \in V'$ we will define conditional preference rules $q^+(z) : z > \neg z$ and $q^-(z) : \neg z > z$ to be included in C' by specifying formulas $q^+(z)$ and $q^-(z)$.

First, for each variable $x_i \in V$, we set

$$q^+(x_i) = y_i \quad \text{and} \quad q^-(x_i) = \neg y_i.$$

Thus, x_i depends only on y_i. We also note that the formulas $q^+(x_i)$ and $q^-(x_i)$ satisfy local-consistency and local-completeness requirements.

Next, for each variable y_i, $1 \leq i \leq n$, we define

$$\begin{aligned} e_i &= (x_1 \leftrightarrow y_1) \wedge \cdots \wedge (x_{i-1} \leftrightarrow y_{i-1}) \\ &\quad \wedge (x_{i+1} \leftrightarrow y_{i+1}) \wedge \cdots \wedge (x_n \leftrightarrow y_n), \end{aligned}$$

$$f_i^+ = e_i \wedge p^+(x_i) \quad \text{and} \quad f_i^- = e_i \wedge p^-(x_i).$$

Finally, we define

$$q^+(y_i) = f_i^+ \vee (\neg f_i^- \wedge x_i)$$

and

$$q^-(y_i) = f_i^- \vee (\neg f_i^+ \wedge \neg x_i)$$

Thus, y_i depends on every variable in V' but itself.

We note that by the local consistency of C, formulas $f_i^+ \wedge f_i^-$, $1 \leq i \leq n$, are unsatisfiable. Consequently, formulas $q^+(y_i) \wedge q^-(y_i)$, $1 \leq i \leq n$, are unsatisfiable. Thus, C' is locally consistent. Finally, $q^+(y_i) \vee q^-(y_i)$ is equivalent to $f_i^+ \vee \neg x_i \vee f_i^- \vee x_i$, so is a tautology. Thus, C' is locally complete and hence a CP-net over V'.

Let α and β be outcomes over $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_n\}$, respectively. By $\alpha\beta$ we denote the outcome over V' obtained by concatenating n-tuples α and β. Conversely, every outcome for C' can be written in this way.

Let α be an outcome over V. We define $\bar{\alpha}$ to be the outcome over $\{y_1, \dots, y_n\}$ obtained by replacing in α every component of the form x_i with y_i and every component $\neg x_i$ with $\neg y_i$. Then for every i, $1 \leq i \leq n$, $\alpha\bar{\alpha} \models e_i$.

Let s be a sequence $\alpha_0, \dots, \alpha_m$ of outcomes over V. Define $L(s)$ to be the sequence of V'-outcomes: $\alpha_0\overline{\alpha_0}, \alpha_0\overline{\alpha_1}, \alpha_1\overline{\alpha_1}, \alpha_1\overline{\alpha_2}, \dots, \alpha_m\overline{\alpha_m}$. Further, let t be a sequence $\epsilon_0, \epsilon_1, \dots, \epsilon_m$ of V'-outcomes with $\epsilon_0 = \alpha\bar{\alpha}$ and $\epsilon_m = \beta\bar{\beta}$. Define $L'(t)$ to be the sequence obtained from t by projecting each element in t to V and iteratively removing elements in the sequence which are the same as their predecessor (until any two consecutive outcomes are different).

Lemma 3 *With the above definitions,*

(i) if s is an improving sequence for C from α to β then $L(s)$ is an improving sequence for C' from $\alpha\bar{\alpha}$ to $\beta\bar{\beta}$;

(ii) if t is an improving sequence from $\alpha\bar{\alpha}$ to $\beta\bar{\beta}$ then $L'(t)$ is an improving sequence from α to β;

(iii) C is consistent if and only if C' is consistent

Sketch of proof: Let $e = \bigwedge_{i=1}^n (x_i \leftrightarrow y_i)$. The definitions have been arranged to ensure the following for CP-net C' :

(a) Suppose e holds in an outcome, so the outcome can be written as $\alpha\bar{\alpha}$ for some α; then no improving flip changes any variable x_i; furthermore, there is an improving flip changing variable y_i if and only if there is an improving flip for the GCP-net C from outcome α changing variable x_i. After applying this flip changing variable y_i, there is exactly one improving flip possible, changing x_i, after which e holds again (this follows using (b), as y_i cannot be immediately flipped back again, because C' is locally consistent).

(b) If e does not hold in an outcome then the only improving flips possible change the value of some variable (x_i or y_i) to make $x_i \leftrightarrow y_i$ hold for some i.

(a) implies (i) and (ii). Also, (i) implies half of (iii), that if C is inconsistent then C' is inconsistent. Conversely, suppose that C' is inconsistent, so there exists an improving sequence t for C' from some outcome to itself. By (b), any improving flip applied to an outcome in which e does not hold increases (by one) the number of i such that $x_i \leftrightarrow y_i$ holds. This implies that e must hold in some outcome in t, because t is cyclic. Write this outcome as $\alpha\bar{\alpha}$. We can cyclically permute t to form an improving sequence from $\alpha\bar{\alpha}$ to itself. Part (ii) then implies that there exists an improving flipping sequence for C from α to itself, showing that C is inconsistent. ■

Theorem 2 CP-DOMINANCE *is* PSPACE*-complete. This holds even if we restrict the CP-nets to being consistent.*

Proof: We use a reduction from PSPACE-hardness of the GCP-DOMINANCE problem when the GCP-nets are restricted to being consistent (Theorem 1). Let C be a consistent (and hence locally consistent) GCP-net over V, and let α and β be outcomes over V. Consider the CP-net C' over variables V' constructed above. Lemma 3(i) and (ii) imply that β dominates α in C if and only if $\beta\bar{\beta}$ dominates $\alpha\bar{\alpha}$ in C'. Moreover, C' is consistent by Lemma 3(iii). Thus, the hardness part of the assertion follows. ■

5 Consistency of GCP-nets

In this section we show that the GCP-CONSISTENCY problem is PSPACE-complete, using results from Sections 3 and 4.

Theorem 3

GCP-CONSISTENCY *is* PSPACE*-complete. This holds even under the restriction to GCP-nets in conjunctive form.*

Proof: PSPACE-hardness is shown by reduction from ACTION-SET ACYCLICITY. We apply function S from Section 3.2 followed by M from Section 4.1. This maps instances of ACTION-SET ACYCLICITY to instances of GCP-CONSISTENCY in conjunctive form. By Lemma 1(iii) and Lemma 2 (ii), an instance of ACTION-SET ACYCLICITY is acyclic if and only if the corresponding instance of GCP-CONSISTENCY is consistent, proving the result. ■

We now show that consistency testing remains PSPACE-complete for CP-nets.

Theorem 4 CP-CONSISTENCY *is* PSPACE*-complete.*

Proof: Let C be a GCP-net in conjunctive form. We define a CP-net C' as follows. If C is locally inconsistent (the property can be decided in polynomial time), we set C' to be any fixed (independent of C) inconsistent but locally consistent CP-net. (Such CP-nets exist.) Otherwise, C is locally-consistent and for C' we take the CP-net we constructed in Section 4.2. The mapping from locally consistent GCP-nets to CP-nets, described in Section 4.2, preserves consistency (Lemma 3 (iii)). Since local inconsistency implies inconsistency, we have that the GCP-net C is consistent if and only if the CP-net C' is consistent. Thus, PSPACE-hardness of the CP-CONSISTENCY problem follows from Theorem 3. ■

Notice the huge complexity gap with the problem of deciding whether there exists a nondominated outcome, which is "only" NP-complete [Domshlak *et al.*, 2003].

6 Concluding remarks

We have shown that dominance and consistency testing in CP-nets is PSPACE-complete. The repeated use of reductions from planning problems confirms the importance of the structural similarity between STRIPS planning and reasoning with CP-nets. This suggests that the well-developed field of planning algorithms for STRIPS representations, especially for unary operators [Brafman and Domshlak, 2003], could be useful for implementing algorithms for dominance and consistency in CP-nets.

Our theorems extend to CP-nets with non-binary domains, and to extensions and variations of CP-nets, such as TCP-nets [Brafman and Domshlak, 2002] that allows for explicit priority of some variables over others, and the more general preference language described in [Wilson, 2004b; 2004a].

The complexity result for dominance is also relevant for constrained optimisation, where a complete algorithm [Boutilier *et al.*, 2004b] involves many dominance checks. Our results reinforce the need for work on finding special classes of problems where dominance and consistency can be tested efficiently [Domshlak and Brafman, 2002; Boutilier *et al.*, 2004a], and for incomplete methods for checking consistency and constrained optimisation [Wilson, 2004a].

Several open problems remain. We do not know whether dominance and consistency testing remain PSPACE-complete when the number of parents in the dependency graph is bounded by a constant. We also do not know whether these two problems remain PSPACE-complete for CP-nets in conjunctive form (the reduction used to prove Theorems 2 and 4 yields CP-nets that are not in conjunctive form). Finally, we do not know the complexity of deciding whether the preference relation induced by a CP-net is complete.

Acknowledgements

We are grateful for a reviewer's simple example of a cyclic CP-net. This material is supported in part by the NSF under Grants No. IIS-0325063 and IIS-0097278, by Science Foundation Ireland under Grant No. 00/PI.1/C075, and by the CNRS program *RTP 11*: "raisonner et décider".

References

[Boutilier *et al.*, 1999] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of UAI99*, pages 71–80, 1999.

[Boutilier *et al.*, 2004a] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus statements. *JAIR*, 21:135–191, 2004.

[Boutilier *et al.*, 2004b] C. Boutilier, R. I. Brafman, C. Domshlak, H. Hoos, and D. Poole. Preference-based constrained optimization with CP-nets. *Computational Intelligence*, 20(2):137–157, 2004.

[Brafman and Dimopoulos, 2004] R. Brafman and Y. Dimopoulos. Extended semantics and optimization algorithms for CP-networks. *Computational Intelligence*, 20(2):218–245, 2004.

[Brafman and Domshlak, 2002] R. Brafman and C. Domshlak. Introducing variable importance trade-offs into CP-nets. In *Proceedings of UAI-02*, pages 69–76, 2002.

[Brafman and Domshlak, 2003] R. Brafman and C. Domshlak. Structure and complexity of planning with unary operators. *JAIR*, 18:315–439, 2003.

[Bylander, 1994] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence Journal*, 69(1-2):165–204, 1994.

[Domshlak and Brafman, 2002] C. Domshlak and R. I. Brafman. CP-nets—reasoning and consistency testing. In *Proceedings of KR2002*, pages 121–132, 2002.

[Domshlak *et al.*, 2003] C. Domshlak, F. Rossi, K. B. Venable, and T. Walsh. Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques. In *Proceedings of IJCAI-03*, pages 215–220, 2003.

[Lang, 2004] J. Lang. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence*, 42(1):37–71, 2004.

[Papadimitriou, 1994] Ch. Papadimitriou. *Computational complexity*. Addison–Wesley, 1994.

[Wilson, 2004a] N. Wilson. Consistency and constrained optimisation for conditional preferences. In *Proceedings of ECAI-04*, pages 888–892, 2004.

[Wilson, 2004b] N. Wilson. Extending CP-nets with stronger conditional preference statements. In *Proceedings of AAAI-04*, pages 735–741, 2004.