Journal of Artificial Intelligence Research 25 (2006) 389–424 Submitted 09/05; published 03/06

On Graphical Modeling of Preference and Importance

Ronen I. Brafman BRAFMAN@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford CA 94305

Carmel Domshlak DCARMEL@IE.TECHNION.AC.IL
Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology
Haifa, Israel 32000

Solomon E. Shimony SHIMONY@CS.BGU.AC.IL
Department of Computer Science
Ben-Gurion University
Beer Sheva, Israel 84105

Abstract

In recent years, CP-nets have emerged as a useful tool for supporting preference elicitation, reasoning, and representation. CP-nets capture and support reasoning with qualitative conditional preference statements, statements that are relatively natural for users to express. In this paper, we extend the CP-nets formalism to handle another class of very natural qualitative statements one often uses in expressing preferences in daily life – statements of relative importance of attributes. The resulting formalism, TCP-nets, maintains the spirit of CP-nets, in that it remains focused on using only simple and natural preference statements, uses the *ceteris paribus* semantics, and utilizes a graphical representation of this information to reason about its consistency and to perform, possibly constrained, optimization using it. The extra expressiveness it provides allows us to better model tradeoffs users would like to make, more faithfully representing their preferences.

1. Introduction

The ability to make decisions and to assess potential courses of action is a corner-stone of numerous AI applications, including expert systems, autonomous agents, decision-support systems, recommender systems, configuration software, and constrained optimization applications. To make good decisions, we must be able to assess and compare different alternatives. Sometimes, this comparison is performed implicitly, as in many recommender systems (Burke, 2000; Resnick & Varian, 1997). But frequently, explicit information about the decision-maker's preferences is required.

In classical decision theory and decision analysis utility functions are used to represent the decision-maker's preferences. However, the process of obtaining the type of information required to generate a good utility function is involved, time-consuming and requires non-negligible effort on the part of the user (French, 1986). Sometimes such effort is necessary and possible, but in many applications the user cannot be engaged for a lengthy period of time and cannot be supported by a human decision analyst. For instance, this is the case in on-line product recommendation systems and other software decision-support applications.

When a utility function cannot be or need not be obtained, one should resort to other, more qualitative forms of preference representation. Ideally, this qualitative information should be easily obtainable from the user by non-intrusive means. That is, we should be able to extract such information from natural and relatively simple statements of preference provided by the user, and this elicitation process should be amenable to automation. In addition, automated reasoning about such qualitative preference information should be semantically effective and computationally efficient.

One framework for preference representation that addresses these concerns is that of *Conditional Preference Networks* (CP-nets) (Boutilier et al. 1999, 2004a). CP-nets is a graphical preference representation model grounded in the notion of conditional preferential independence. In preference elicitation with CP-nets, the decision maker (directly or indirectly) describes how her preference over the values of one variable depends on the value of other variables. For example, she may state that her preference for a dessert depends on the main-course as well as whether or not she had an alcoholic beverage. In turn, her preference for an alcoholic beverage may depend on the main course and the time of day. This information is described by a graphical structure in which the nodes represent variables of interest and edges capture direct preferential dependence relations between the variables. Each node is annotated with a *conditional preference table* (CPT) describing the user's preference over alternative values of this node given different values of the parent nodes. CP-nets capture a class of intuitive and useful natural language statements of the form "I prefer the value x_0 for variable X given that $Y = y_0$ and $Z = z_0$". Such statements do not require complex introspection nor a quantitative assessment.

From the practical perspective, there is another class of preference statements that is no less intuitive or important, yet is not captured by the CP-net model. These statements have the form: "It is more important to me that the value of X be better than that the value of Y be better." We call these *relative importance* statements. For instance, one might say "The length of the journey is more important to me than the choice of airline". A more refined notion of importance, though still intuitive and easy to communicate, is that of *conditional relative importance*: "The length of the journey is more important to me than the choice of airline if I need to give a talk the following day. Otherwise, the choice of airline is more important." The latter statement is of the form: "A better assignment for X is more important than a better assignment for Y given that $Z = z_0$." Notice that information about relative importance is different from information about preferential independence. For instance, in the example above, user's preference for an airline does not depend on the duration of the journey because, e.g., she compares airlines based only on their service, security levels, and the quality of their frequent flyer program. Informally, using statements of relative importance the user expresses her preference over *compromises* that may be required. Such information is very important in customized product configuration applications (Sabin & Weigel, 1998; Haag, 1998; Freuder & O'Sullivan, 2001), where production, supply, and other constraints are posed on the product space by the producer, and these constraints are typically even unknown to the customer. Indeed, in many applications, various resource (e.g., money, time, bandwidth) constraints exist, and the main computational task is that of finding a solution that is feasible and not preferentially dominated by any other solution.

In this paper we consider enhancing the expressive power of CP-nets by introducing information about importance relations, obtaining a preference-representation structure which we call TCP-nets (for *tradeoffs-enhanced* CP-nets). By capturing information about both conditional preferential independence and conditional relative importance, TCP-nets provide a richer framework for representing user preferences, allowing stronger conclusions to be drawn, yet remaining committed to the use of only very intuitive, qualitative information. At the same time, we show that the added relative importance information has significant impact on both the consistency of the specified relation, and the techniques used for reasoning about it. Focusing on these computational issues, we show that the graphical structure of the "mixed" set of preference statements captured in a TCP-net can often be exploited in order to achieve efficiency both in consistency testing and in preferential reasoning.

This paper is organized as follows: Section 2 describes the notions underlying TCP-nets: preference relations, preferential independence, and relative importance. In Section 3 we define TCP-nets, specify their semantics, and provide a number of examples. In Section 4 we characterize a class of conditionally acyclic TCP-nets whose consistency is guaranteed and then, in Section 5 we discuss the complexity of identifying members of this class. In Section 6 we present an algorithm for outcome optimization in conditionally acyclic TCP-nets, and discuss the related tasks of reasoning about preferences given a TCP-net. We conclude with a discussion of related and future work in Section 7.

2. Preference Orders, Independence, and Relative Importance

In this section we describe the semantic concepts underlying TCP-nets: preference orders, preferential independence, conditional preferential independence, as well as relative importance and conditional relative importance.

2.1 Preference and Independence

We model a *preference relation* as a strict partial order. Thus, we use the terms preference order and strict partial order interchangeably. A strict partial order is a binary relation over outcomes that is anti-reflexive, anti-symmetric and transitive. Given two outcomes o, o', we write $o \succ o'$ to denote that o is strictly preferred to o'.

The above choice implies that two outcomes cannot be equally preferred. This choice follows from the fact that the language of preferences we use in this paper does not allow statements of indifference (as opposed to incomparability), and thus there is no need for using weak orderings. Incorporating statements of indifference is pretty straightforward, as explained by Boutilier et al. (2004a), but introduces much overhead if we were to formally treat it throughout this paper.

The types of outcomes we are concerned with consist of possible assignments to some set of variables. More formally, we assume some given set $\mathbf{V} = \{X_1, \ldots, X_n\}$ of variables with corresponding domains $\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)$. The set of possible outcomes is then $\mathcal{D}(\mathbf{V}) = \mathcal{D}(X_1) \times \cdots \times \mathcal{D}(X_n)$, where we use $\mathcal{D}(\cdot)$ to denote the domain of a set of variables as well. For example, in the context of the problem of configuring a personal computer (PC), the variables may be *processor type, screen size, operating system* etc., where *screen size* has the domain {*17in, 19in, 21in*}, *operating system* has the domain {*LINUX, Windows98,* *WindowsXP*}, etc. Each complete assignment to the set of variables specifies an outcome – a particular PC configuration. Thus, a preference relation over these outcomes specifies a strict partial order over possible PC configurations.

The number of possible outcomes is exponential in n, while the set of possible orderings on them is more than doubly exponential in n. Therefore, explicit specification and representation of an ordering is not realistic, and thus we must describe it implicitly using a compact representation model. The notion of preferential independence plays a key role in such representations. Intuitively, $\mathbf{X} \subset \mathbf{V}$ is *preferentially independent* of $\mathbf{Y} = \mathbf{V} - \mathbf{X}$ if and only if for all assignments to $\mathbf{Y}$, our preference over $\mathbf{X}$ values is identical.

Definition 1 *Let* $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}(\mathbf{X})$ *for some* $\mathbf{X} \subseteq \mathbf{V}$*, and* $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{D}(\mathbf{Y})$*, where* $\mathbf{Y} = \mathbf{V} - \mathbf{X}$*. We say that* $\mathbf{X}$ *is* preferentially independent *of* $\mathbf{Y}$ *iff, for all* $\mathbf{x}_1$*,* $\mathbf{x}_2$*,* $\mathbf{y}_1$*,* $\mathbf{y}_2$ *we have that*

$$\mathbf{x}_1\mathbf{y}_1 \succ \mathbf{x}_2\mathbf{y}_1 \text{ iff } \mathbf{x}_1\mathbf{y}_2 \succ \mathbf{x}_2\mathbf{y}_2 \tag{1}$$

For example, in our PC configuration example, the user may assess *screen size* to be preferentially independent of *processor type* and *operating system*. This could be the case if the user always prefers a larger screen to a smaller screen, independent of the selection of processor and/or OS.

Preferential independence is a strong property, and is therefore not very common. A more refined notion is that of conditional preferential independence. Intuitively, $\mathbf{X}$ is *conditionally preferentially independent* of $\mathbf{Y}$ given $\mathbf{Z}$ if and only if for every fixed assignment to $\mathbf{Z}$, the ranking of $\mathbf{X}$ values is independent of the value of $\mathbf{Y}$.

Definition 2 *Let* $\mathbf{X}, \mathbf{Y}$ *and* $\mathbf{Z}$ *be a partition of* $\mathbf{V}$ *and let* $\mathbf{z} \in \mathcal{D}(\mathbf{Z})$*.* $\mathbf{X}$ *is* conditionally preferentially independent of $\mathbf{Y}$ given $\mathbf{z}$ *iff, for all* $\mathbf{x}_1$*,* $\mathbf{x}_2$*,* $\mathbf{y}_1$*,* $\mathbf{y}_2$ *we have that*

$$\mathbf{x}_1\mathbf{y}_1\mathbf{z} \succ \mathbf{x}_2\mathbf{y}_1\mathbf{z} \text{ iff } \mathbf{x}_1\mathbf{y}_2\mathbf{z} \succ \mathbf{x}_2\mathbf{y}_2\mathbf{z}, \tag{2}$$

and $\mathbf{X}$ *is* conditionally preferentially independent of $\mathbf{Y}$ given $\mathbf{Z}$ *iff* $\mathbf{X}$ *is conditionally preferentially independent of* $\mathbf{Y}$ *given every assignment* $\mathbf{z} \in \mathcal{D}(\mathbf{Z})$*.*

Returning to our PC example, the user may assess *operating system* to be independent of all other features given *processor type*. That is, he always prefers LINUX given an AMD processor and WindowsXP given an Intel processor (e.g., because he might believe that WindowsXP is optimized for the Intel processor, whereas LINUX is otherwise better). Note that the notions of preferential independence and conditional preferential independence are among a number of standard and well-known notions of independence in multi-attribute utility theory (Keeney & Raiffa, 1976).

2.2 Relative Importance

Although statements of preferential independence are natural and useful, the orderings obtained by relying on them alone are relatively weak. To understand this, consider two preferentially independent boolean attributes A and B with values a_1, a_2 and b_1, b_2, respectively. If A and B are preferentially independent, then we can specify a preference order over A values, say $a_1 \succ a_2$, independently of the value of B. Similarly, our preference over

B values, say $b_1 \succ b_2$, is independent of the value of A. From this we can deduce that a_1b_1 is the most preferred outcome and a_2b_2 is the least preferred outcome. However, we do not know the relative order of a_1b_2 and a_2b_1. This is typically the case when we consider independent variables: We can rank each one given a fixed value of the other, but often, we cannot compare outcomes in which both values are different. One type of information that can address some (though not necessarily all) such comparisons is information about relative importance. For instance, if we state that A is more important than B, it means that we prefer an improvement in A over an improvement in B. In that case, we know that $a_1b_2 \succ a_2b_1$, and can totally order the set of outcomes as $a_1b_1 \succ a_1b_2 \succ a_2b_1 \succ a_2b_2$.

One may ask why it is important for us to order a_1b_2 and a_2b_1 – after all, we know that a_1b_1 is the most preferred outcome. However, in many typical scenarios, we have auxiliary or user constraints that prevent us from providing the user with the most preferred (unconstrained) outcome. A simple and common example is that of budget constraints, other resource limitations, such are bandwidth and buffer size (as in the adaptive rich-media systems described by Brafman and Friedman (2005) are also common. In such cases, it is important to know which attributes the user cares about more strongly, and to try to maintain good values for these attributes, compromising on the others.

Returning to our PC configuration example, suppose that the attributes *operating system* and *processor type* are mutually preferentially independent. We might say that *processor type* is more important than *operating system*, e.g, because we believe that the effect of the processor's type on system performance is more significant than the effect of the operating system.

Definition 3 *Let a pair of variables X and Y be mutually preferentially independent given $\mathbf{W} = \mathbf{V} - \{X, Y\}$. We say that X is* more important than Y*, denoted by $X \rhd Y$, if for every assignment $\mathbf{w} \in \mathcal{D}(\mathbf{W})$ and for every $x_i, x_j \in \mathcal{D}(X)$, $y_a, y_b \in \mathcal{D}(Y)$, such that $x_i \succ x_j$ given $\mathbf{w}$, we have that:*

$$x_i y_a \mathbf{w} \succ x_j y_b \mathbf{w}. \tag{3}$$

Note that Eq. 3 holds even when $y_b \succ y_a$ given $\mathbf{w}$. For instance, when both X and Y are binary variables, and $x_1 \succ x_2$ and $y_1 \succ y_2$ hold given $\mathbf{w}$, then $X \rhd Y$ iff we have $x_1y_2\mathbf{w} \succ x_2y_1\mathbf{w}$ for all $\mathbf{w} \in \mathcal{D}(\mathbf{W})$. Notice that this is a strict notion of importance – any reduction in Y is preferred to any reduction in X. Clearly, this idea can be further refined by providing an actual ordering over elements of $\mathcal{D}(XY)$, and we discuss this extension in Section 3.4. In addition, one can consider relative importance assessments among more than two variables. However, we feel that the benefit of capturing such statements is small: We believe that statements of relative importance referring to more than two attributes are not very natural for users to articulate, and their inclusion would significantly reduce the computational advantages of graphical modeling. Therefore, in this work we focus only on relative importance statements referring to pairs of attributes.

Relative importance information is a natural enhancement of independence information. As such, relative importance retains a desirable property - it corresponds to statements that a naive user would find simple and clear to evaluate and articulate. Moreover, it can be generalized naturally to a notion of *conditional relative importance*. For instance, suppose that the relative importance of *processor type* and *operating system* depends on the primary usage of the PC. For example, when the PC is used primarily for graphical applications, then the choice of an operating system is more important than that of a processor because certain important software packages for graphic design are not available on LINUX. However, for other applications, the processor type is more important because applications for both Windows and LINUX exist. Thus, we say that X is more important than Y given $\mathbf{z}$ if we always prefer to reduce the value of Y rather than the value of X, whenever $\mathbf{z}$ holds.

Definition 4 *Let X and Y be a pair of variables from $\mathbf{V}$, and let $\mathbf{Z} \subseteq \mathbf{W} = \mathbf{V} - \{X, Y\}$. We say that X is* more important than Y given $\mathbf{z} \in \mathcal{D}(\mathbf{Z})$ *iff, for every assignment $\mathbf{w}'$ on $\mathbf{W}' = \mathbf{V} - (\{X, Y\} \cup \mathbf{Z})$ we have:*

$$x_i y_a \mathbf{z}\mathbf{w}' \succ x_j y_b \mathbf{z}\mathbf{w}' \tag{4}$$

whenever $x_i \succ x_j$ given $\mathbf{z}\mathbf{w}'$. We denote this relation by $X \rhd_{\mathbf{z}} Y$. Finally, if for some $\mathbf{z} \in \mathcal{D}(\mathbf{Z})$ we have either $X \rhd_{\mathbf{z}} Y$, or $Y \rhd_{\mathbf{z}} X$, then we say that the relative importance of X and Y is conditioned on $\mathbf{Z}$, and write $\mathcal{RI}(X, Y|\mathbf{Z})$.

3. TCP-nets

The TCP-net (for *Tradeoff-enhanced CP-nets*) model is an extension of CP-nets (Boutilier et al., 2004a) that encodes conditional relative importance statements, as well as the conditional preference statements supported in CP-nets. The primary usage of the TCP-net graphical structure is in consistency analysis of the provided preference statements, and in classification of complexity and developing efficient algorithms for various reasoning tasks over these statements. In particular, as we later show, when this structure is "acyclic" (for a suitable definition of this notion!), the set of preference statements represented by the TCP-net is guaranteed to be consistent – that is, there is a strict total order over the outcomes that satisfies all the preference statements. In what follows we formally define the TCP-net model. As it subsumes the CP-net model, we will immediately define this more general model rather than proceed in stages.

3.1 TCP-net Definition

TCP-nets are annotated graphs with three types of edges. The nodes of a TCP-net correspond to the problem variables $\mathbf{V}$. The first type of (directed) edges comes from the original CP-nets model and captures direct preferential dependencies, that is, such an edge from X to Y implies that the user has different preferences over Y values given different values of X. The second (directed) edge type captures relative importance relations. The existence of such an edge from X to Y implies that X is more important than Y. The third (undirected) edge type captures conditional importance relations: Such an edge between nodes X and Y exists if there exists a non-empty variable subset $\mathbf{Z} \subseteq \mathbf{V} - \{X, Y\}$ for which $\mathcal{RI}(X, Y|\mathbf{Z})$ holds. Without loss of generality, in what follows, the set $\mathbf{Z}$ is assumed to be the minimal set of variables upon which the relative importance between X and Y depends.

As in CP-nets, each node X in a TCP-net is annotated with a *conditional preference table* (CPT). This table associates preferences over $\mathcal{D}(X)$ for every possible value assignment to the parents of X (denoted $Pa(X)$). In addition, in TCP-nets, each undirected edge is annotated with a *conditional importance table* (CIT). The CIT associated with such an edge

(X, Y) describes the relative importance of X and Y given the value of the corresponding importance-conditioning variables $\mathbf{Z}$.

Definition 5 *A TCP-net $\mathcal{N}$ is a tuple $\langle \mathbf{V}, \mathsf{cp}, \mathsf{i}, \mathsf{ci}, \mathsf{cpt}, \mathsf{cit} \rangle$ where:*

(1) $\mathbf{V}$ *is a set of nodes, corresponding to the problem variables $\{X_1, \dots, X_n\}$.*

(2) cp *is a set of directed* cp-arcs $\{\alpha_1, \dots, \alpha_k\}$ *(where* cp *stands for* conditional preference*). A* cp*-arc $\langle \overrightarrow{X_i, X_j} \rangle$ is in $\mathcal{N}$ iff the preferences over the values of X_j depend on the actual value of X_i. For each $X \in \mathbf{V}$, let $Pa(X) = \{X' | \langle \overrightarrow{X', X} \rangle \in \mathsf{cp}\}$.*

(3) i *is a set of directed* i-arcs $\{\beta_1, \dots, \beta_l\}$ *(where* i *stands for* importance*). An* i*-arc $(\overrightarrow{X_i, X_j})$ is in $\mathcal{N}$ iff $X_i \rhd X_j$.*

(4) ci *is a set of undirected* ci-arcs $\{\gamma_1, \dots, \gamma_m\}$ *(where* ci *stands for* conditional importance*). A* ci*-arc (X_i, X_j) is in $\mathcal{N}$ iff we have $\mathcal{RI}(X_i, X_j | \mathbf{Z})$ for some $\mathbf{Z} \subseteq \mathbf{V} - \{X_i, X_j\}$.*[1] *We call $\mathbf{Z}$ the **selector set** of (X_i, X_j) and denote it by $\mathcal{S}(X_i, X_j)$.*

(5) cpt *associates a CPT with every node $X \in \mathbf{V}$, where $CPT(X)$ is a mapping from $\mathcal{D}(Pa(X))$ (i.e., assignments to X's parent nodes) to strict partial orders over $\mathcal{D}(X)$.*

(6) cit *associates with every* ci*-arc $\gamma = (X_i, X_j)$ a (possibly partial) mapping $CIT(\gamma)$ from $\mathcal{D}(\mathcal{S}(X_i, X_j))$ to orders over the set $\{X_i, X_j\}$.*[2]

A TCP-net in which the sets i and ci (and therefore also cit) are empty, is also a CP-net. Thus, it is the elements i, ci, and cit that describe absolute and conditional importance of attributes provided by TCP-nets, beyond the conditional preference information captured by CP-nets.

3.2 TCP-net Semantics

The semantics of a TCP-net is defined in terms of the set of strict partial orders consistent with the set of constraints imposed by the preference and importance information captured by this TCP-net. The intuitive idea is rather straightforward: (1) A strict partial order $\succ$ satisfies the conditional preferences for variable X if any two complete assignments that differ only on the value of X are ordered by $\succ$ consistently with the ordering on X values in the CPT of X. Recall that this ordering can depend on the parent of X in the graph. (2) A strict partial order $\succ$ satisfies the assertion that X is more important than Y if given any two complete assignments that differ on the value of X and Y only, $\succ$ prefers that assignment which provides X with a better value. (3) A strict partial order $\succ$ satisfies the assertion that X is more important than Y given some assignment z to variable set Z if given any two complete assignments that differ on the value X and Y only, and in (both of) which Z is assigned z, $\succ$ prefers that assignment which provides X with a better value.

1. Observe that every i-arc $(\overrightarrow{X_i, X_j})$ can be seen as representing $\mathcal{RI}(X_i, X_j | \emptyset)$. However, a clear distinction between i-arcs and ci-arc simplifies specification of many forthcoming notions and claims (e.g., Lemma 3 in Section 4, as well as the related notion of root variables.)
2. That is, the relative importance relation between X_i and X_j may be specified only for certain values of the selector set.

This is defined more formally below. We use $\succ^X_{\mathbf{u}}$ to denote the preference relation over the values of X given an assignment $\mathbf{u}$ to $\mathbf{U} \supseteq Pa(X)$.

Definition 6 *Consider a TCP-net* $\mathcal{N} = \langle \mathbf{V}, \mathsf{cp}, \mathsf{i}, \mathsf{ci}, \mathsf{cpt}, \mathsf{cit} \rangle$.

1. *Let* $\mathbf{W} = \mathbf{V} - (\{X\} \cup Pa(X))$ *and let* $\mathbf{p} \in \mathcal{D}(Pa(X))$. *A preference (=strict partial) order* $\succ$ *over* $\mathcal{D}(\mathbf{V})$ *satisfies* $\succ^X_{\mathbf{p}}$ *iff* $x_i\mathbf{pw} \succ x_j\mathbf{pw}$, *for every* $\mathbf{w} \in \mathcal{D}(\mathbf{W})$, *whenever* $x_i \succ^X_{\mathbf{p}} x_j$ *holds.*

2. *A preference order* $\succ$ *over* $\mathcal{D}(\mathbf{V})$ *satisfies* $CPT(X) \in \mathsf{cpt}$ *iff it satisfies* $\succ^X_{\mathbf{p}}$ *for every assignment* $\mathbf{p}$ *of* $Pa(X)$.

3. *A preference order* $\succ$ *over* $\mathcal{D}(\mathbf{V})$ *satisfies* $X \rhd Y$ *iff for every* $\mathbf{w} \in \mathcal{D}(\mathbf{W})$ *such that* $\mathbf{W} = \mathbf{V} - \{X, Y\}$, $x_iy_a\mathbf{w} \succ x_jy_b\mathbf{w}$ *whenever* $x_i \succ^X_{\mathbf{w}} x_j$.

4. *A preference order* $\succ$ *over* $\mathcal{D}(\mathbf{V})$ *satisfies* $X \rhd_{\mathbf{z}} Y$ *iff for every* $\mathbf{w} \in \mathcal{D}(\mathbf{W})$ *such that* $\mathbf{W} = \mathbf{V} - (\{X, Y\} \cup \mathbf{Z})$, $x_iy_a\mathbf{zw} \succ x_jy_b\mathbf{zw}$ *whenever* $x_i \succ^X_{\mathbf{zw}} x_j$.

5. *A preference order* $\succ$ *over* $\mathcal{D}(\mathbf{V})$ *satisfies* $CIT(\gamma)$ *of the* ci*-arc* $\gamma = (X, Y) \in \mathsf{cit}$ *if it satisfies* $X \rhd_{\mathbf{z}} Y$ *whenever an entry in the table conditioned on* $\mathbf{z}$ *ranks* X *as more important.*

A preference order $\succ$ over $\mathcal{D}(\mathbf{V})$ satisfies a TCP-net $\mathcal{N} = \langle \mathbf{V}, \mathsf{cp}, \mathsf{i}, \mathsf{ci}, \mathsf{cpt}, \mathsf{cit} \rangle$ *iff:*

(1) for every $X \in \mathbf{V}$, $\succ$ *satisfies* $CPT(X)$,

(2) for every i*-arc* $\beta = (\overrightarrow{X_i, X_j}) \in \mathsf{i}$, $\succ$ *satisfies* $X \rhd Y$, *and*

(3) for every ci*-arc* $\gamma = (X_i, X_j) \in \mathsf{ci}$, $\succ$ *satisfies* $CIT(\gamma)$.

Definition 7 *A TCP-net is* satisfiable *iff there is some strict partial order* $\succ$ *over* $\mathcal{D}(\mathbf{V})$ *that satisfies it;* $o \succ o'$ *is* implied *by a TCP-net* $\mathcal{N}$ *iff it holds in all preference orders over* $\mathcal{D}(\mathbf{V})$ *that satisfy* $\mathcal{N}$.

Lemma 1 Preferential entailment with respect to a satisfiable TCP-net is transitive. That is, if $\mathcal{N} \models o \succ o'$ and $\mathcal{N} \models o' \succ o''$, then $\mathcal{N} \models o \succ o''$.

Proof: If $\mathcal{N} \models o \succ o'$ and $\mathcal{N} \models o' \succ o''$, then $o \succ o'$ and $o' \succ o''$ in all preference orders satisfying N. As each of these ordering is transitive, we must have $o \succ o''$ in all satisfying orderings. $\square$

Note that, strictly speaking, we should use the term "satisfiable" rather than "consistent" with respect to a set of preference statements, given that we provide a model theory, and not a proof theory. However, since the corresponding proof theory follows in a completely straightforward manner from our semantics combined with transitivity, this raises no problem.

3.3 TCP-net Examples

Having provided the formal specification of the TCP-nets model, let us now illustrate TCP-nets with a few examples. For simplicity of presentation, in the following examples all variables are binary, although the semantics of TCP-nets is given by Definitions 6 and 7 with respect to arbitrary finite domains.

Example 1 (Evening Dress) Figure 1(a) presents a CP-net that consists of three variables J, P, and S, standing for the jacket, pants, and shirt, respectively. I prefer black to white as a color for both the jacket and the pants, while my preference for the shirt color (red/white) is conditioned on the *color combination* of jacket and pants: If they are of the same color, a white shirt will make my dress too colorless, therefore, red shirt is preferable. Otherwise, if the jacket and the pants are of different colors, a red shirt will probably make my evening dress too flashy, therefore, a white shirt is preferable. The solid lines in Figure 1(c) show the preference relation induced directly by the information captured by this CP-net; The top and the bottom elements are the worst and the best outcomes, respectively, and the arrows are directed from less preferred to more preferred outcomes.

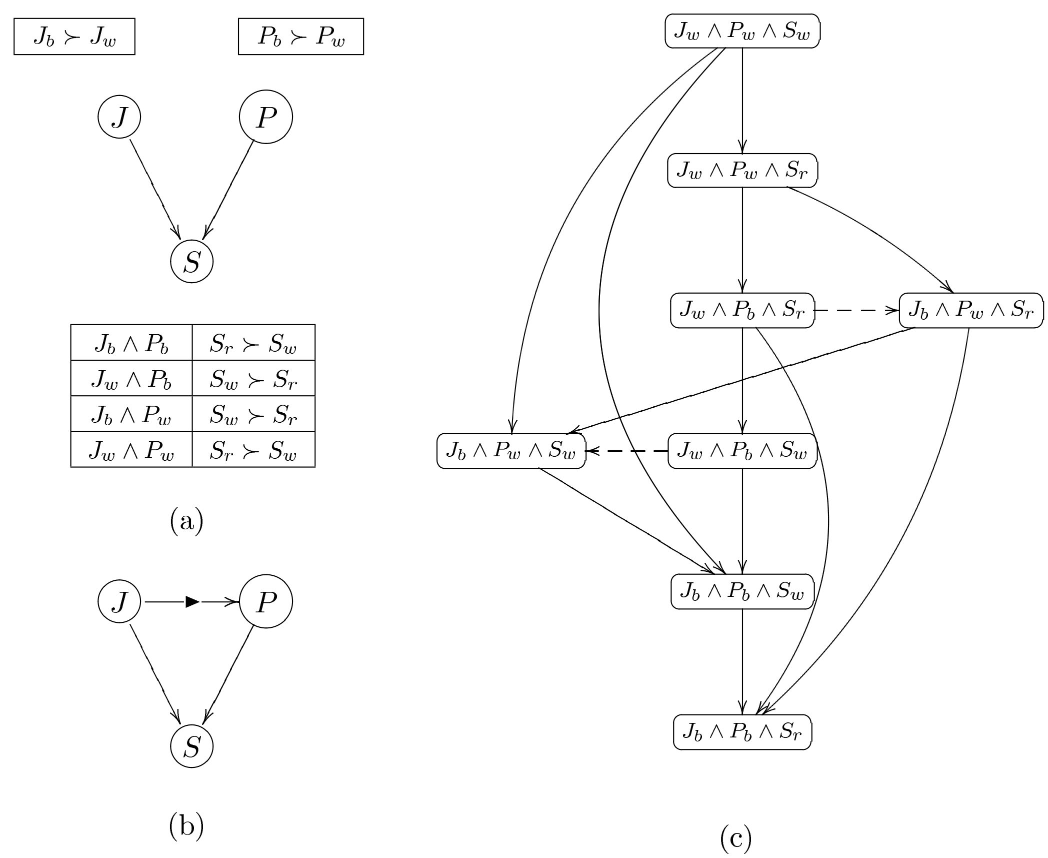


Figure 1: "Evening Dress" CP-net & TCP-net.

Figure 1(b) depicts a TCP-net that extends this CP-net by adding an i-arc from J to P, i.e., having black jacket is (unconditionally) more important than having black pants. This induces additional relations among outcomes, captured by the dashed lines in Figure 1(c). ◇

The reader may rightfully ask whether the statement of importance in Example 1 is not redundant: According to my preference, it seems that I will always wear a black suit with a red shirt. However, while my preferences are clear, various constraints may make some outcomes, including the most preferred one, infeasible. For instance, I may not have a clean black jacket, in which case the most preferred feasible alternative is a white jacket, black pants, and a white shirt. Alternatively, suppose that the only clean clothes I have are velvet black jacket and white pants, and silk white jacket and black pants. My wife forbids me to mix velvet and silk, and so I will have to compromise, and to wear either the black (velvet) jacket with the white (velvet) pants, or the white (silk) jacket with the black (silk) pants. In this case, the fact that I prefer wearing the preferred jacket to wearing the preferred pants determines higher desirability for the velvet combination. Now, if my wife has to prepare my evening dress while I am late at work writing a paper, having this information will help her to choose among the *available* options an outfit that I would like most.

Indeed, as noted earlier, many applications involve limited resources, such as money, time, bandwidth, memory, etc. In many instances, the optimal assignment violates these resource constraints, and we must compromise and accept a less desirable, but feasible assignment. TCP-nets capture information that allows us make more informed compromises.

Example 2 (Flight to the USA) Figure 2(a) illustrates a more complicated CP-net, describing my preference over the flight options to a conference in the USA, from Israel. This network consists of five variables, standing for various parameters of the flight:

Day of the Flight The variable D distinguishes between flights leaving a day (D_{1d}) and two days (D_{2d}) before the conference, respectively. Since I am married, and I am really busy with my work, I prefer to leave on the day before the conference.

Airline The variable A represents the airline. I prefer to fly with British Airways (A_{ba}) than with KLM (A_{klm}).

Departure Time The variable T distinguishes between morning/noon (T_m) and evening/night (T_n) flights. Among flights leaving two days before the conference I prefer an evening/night flight, because it will allow me to work longer on the day of the flight. However, among flights leaving a day before the conference I prefer a morning/noon flight, because I would like to have a few hours before the conference opening in order to rest at the hotel.

Stop-over The variable S distinguishes between direct (S_{0s}) and indirect (S_{1s}) flights, respectively. On day flights I am awake most of the time and, being a smoker, prefer a stop-over in Europe (so I can have a smoking break). However, on night flights I sleep, leading to a preference for direct flights, since they are shorter.

Ticket Class The variable C stands for ticket class. On a night flight, I prefer to sit in economy class (C_e) (I don't care where I sleep, and these seats are significantly cheaper), while on a day flight I prefer to pay for a seat in business class (C_b) (Being awake, I can better appreciate the good seat, food, and wine).

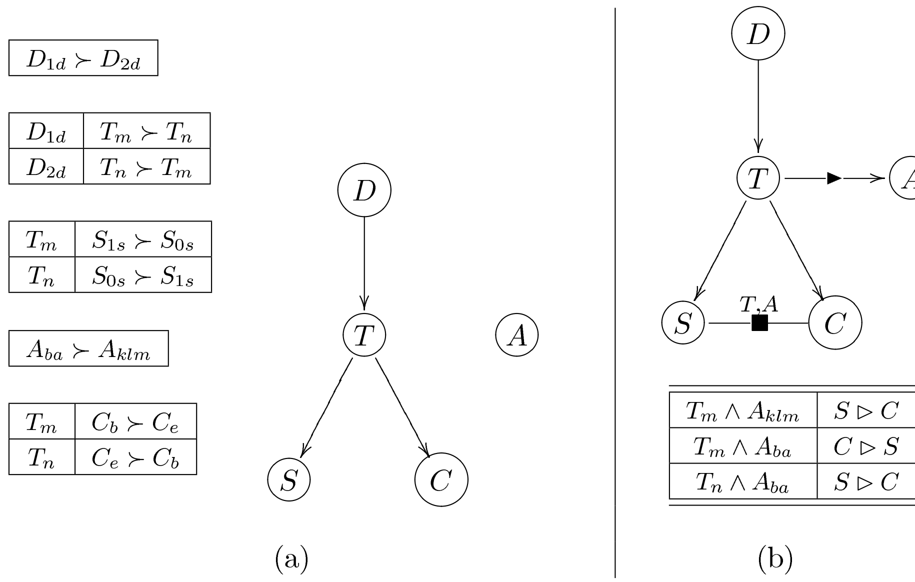


Figure 2: "Flight to the USA" CP-net & TCP-net from Example 2.

The CP-net in Figure 2(a) captures all these preference statements, and the underlying preferential dependencies, while Figure 2(b) presents a TCP-net that extends this CP-net to capture relative importance relations between some parameters of the flight. First, there is an i-arc from T to A, because getting more suitable flying time is more important to me than getting the preferred airline. Second, there is a ci-arc between S and C, where the relative importance of S and C depends on the values of T and A:[3]

1. On a KLM day flight, an intermediate stop in Amsterdam is more important to me than flying business class (I feel that KLM's business class does not have a good cost/performance ratio, while visiting a casino in Amsterdam's airport sounds to me like a good idea.)

2. For a British Airways night flight, the fact that the flight is direct is more important to me than getting a cheaper economy seat (I am ready to pay for business class, in order not to spend even one minute at Heathrow airport at night).

3. On a British Airways day flight, business class is more important to me than having a short intermediate break (it is hard to find a nice smoking area at Heathrow).

The CIT of this ci-arc is also shown in Figure 2(b). ◇

3.4 Relative Importance with Non-binary Variables

Having read so far, the reader may rightfully ask whether the notion of relative (conditional) importance *ceteris paribus*, as specified in Section 2.2 (Eq. 3 and 4), is not too strong when

3. For clarity, the ci-arc in Figure 2(b) is *schematically* labeled with its importance-conditioning variables T and A.

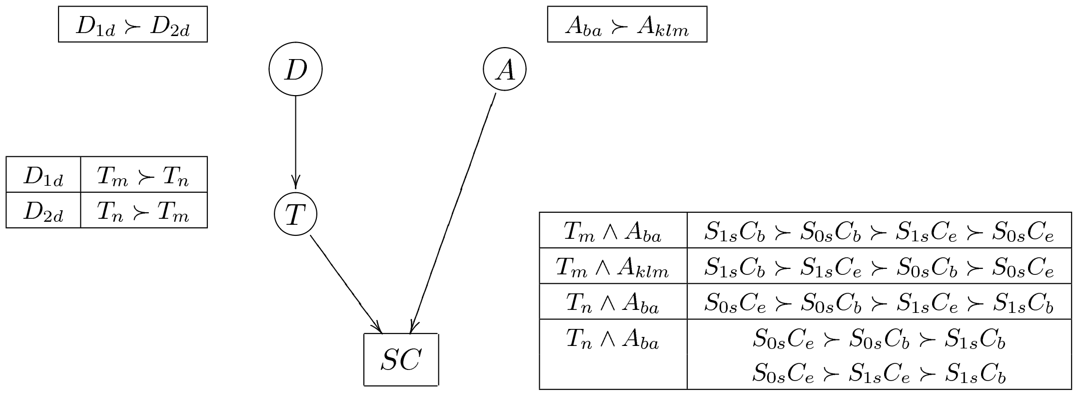
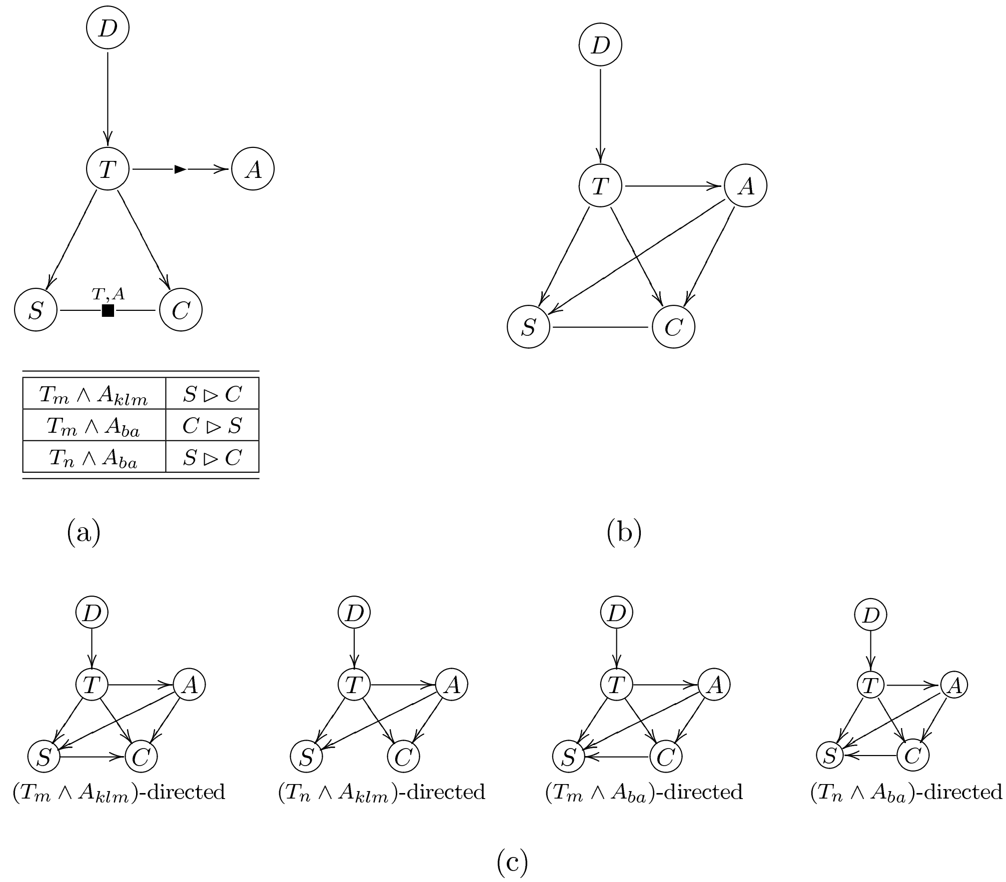


Figure 3: The network obtained by clustering variables S and C in Example 2.

the variables are not binary. For example, consider a more refined notion of departure time (variable T) in Example 2, and suppose there are more than two companies flying from Israel to the USA (variable A). In this case, one may prefer a better flight time, even if this requires a compromise in the airline, *as long as* this compromise is not too significant. For instance, to get a better flight time, one may be willing to compromise and accept any airline but only among those she ranks in the top i places in this context.

More generally, our notion of importance, as well as some more refined notions of it, are really means of specifying an ordering over assignments to variable pairs. In a sense, one could reduce TCP-nets into CP-nets by combining variables between which we have an importance relation. Thus, for instance, in the "Flight to the USA" example, we could combine the variables S and C (see Figure 3). The resulting variable, SC will have as its domain the Cartesian product of the domains of S and C. The preferences for the values of SC are now conditioned on T, the current parent of S and C, as well as on A, which belongs to the selector set of their CIT. In general, the selector set (and parents of) a pair of variables can be viewed as conditioning the preferences over the value combinations for this pair. Hence, such clustering can help us already in the case of binary variables as certain orderings over the assignments to two binary variables cannot be specified with a TCP-net. However, this is clearly more of an issue in the case of non-binary variables, where the number of combinations of pairs of values is much larger.

The bottom line is that more complex importance relations between pairs of variables can be captured. The main questions is how. The strict importance relation we use captures certain such relations in a very compact manner. As such, its specification (e.g., in terms of natural language statements) is very easy. This does not rule out the possibility of expressing more refined relations. Various linguistic constructs could be used to express such relations. However, technically, they can all be captured by clustering the relevant variables, and the resulting representation would be a TCP-net, or possibly simply a CP-net. Of course, it is quite possible that some relations have an alternative compact representation that could help make reasoning with them more efficient than simply collapsing them, and this can be a useful question for future research to examine.



$T_m \wedge A_{klm}$	$S \rhd C$
$T_m \wedge A_{ba}$	$C \rhd S$
$T_n \wedge A_{ba}$	$S \rhd C$

Figure 4: (a) "Flight to USA" TCP-net. (b) Its dependency graph. (c), Four **w**-directed graphs.

4. Conditionally Acyclic TCP-nets

Returning to the notion of TCP-net satisfiability, observe that Definition 7 provides no practical tools for verifying satisfiability of a given TCP-net. Tackling this issue, in this section we introduce a large class of TCP-nets whose members are guaranteed to be satisfiable. We refer to this class of TCP-nets as *conditionally acyclic*.

Let us begin with the notion of the dependency graph induced by a TCP-net.

Definition 8 *The* dependency graph $\mathcal{N}^\star$ *of TCP-net* $\mathcal{N}$ *contains all the nodes and edges of* $\mathcal{N}$*. Additionally, for every* ci*-arc* (X_i, X_j) *in* $\mathcal{N}$ *and every* $X_k \in \mathcal{S}(X_i, X_j)$, $\mathcal{N}^\star$ *contains a pair of directed edges* (X_k, X_i) *and* (X_k, X_j)*, if these edges are not already in* $\mathcal{N}$*.*

Figure 4(b) depicts the dependency graph of the TCP-net from the "Flight to USA" example, repeated for convenience in Figure 4(a). For the next definition, recall that the *selector set* of a ci-arc is the set of nodes whose value determines the "direction" of this arc. Recall also, that once we assign a value to the selector set, we are, in essence, orienting all the conditional importance edges. More generally, once all selector sets are assigned, we transform both $\mathcal{N}$ and $\mathcal{N}^\star$. This motivates the following definition.

Definition 9 *Let $\mathcal{S}(\mathcal{N})$ be the union of all selector sets of $\mathcal{N}$. Given an assignment $\mathbf{w}$ to all nodes in $\mathcal{S}(\mathcal{N})$, the* $\mathbf{w}$-directed graph *of $\mathcal{N}^\star$ consists of all the nodes and directed edges of $\mathcal{N}^\star$. In addition it has a directed edge from X_i to X_j if such an edge is not already in $\mathcal{N}^\star$, and (X_i, X_j) is a* ci*-arc of $\mathcal{N}$ and the CIT for (X_i, X_j) specifies that $X_i \rhd X_j$ given $\mathbf{w}$.*

Figure 4(c) presents all the four $\mathbf{w}$-directed graphs of the TCP-net from the "Flight to USA" example. Note that, for the KLM night flights, the relative importance of S and C is not specified, thus there is no edge between S and C in the $(T_n \wedge A_{klm})$-directed graph of $\mathcal{N}^\star$.

Using Definitions 8 and 9, we specify the class of conditionally acyclic TCP-nets, and show that it is satisfiable[4].

Definition 10 *A TCP-net $\mathcal{N}$ is* conditionally acyclic *if, for every assignment $\mathbf{w}$ to $\mathcal{S}(\mathcal{N})$, the induced $\mathbf{w}$-directed graphs of $\mathcal{N}^\star$ are acyclic.*

We now show that every conditionally acyclic TCP-net is satisfiable, and begin with providing two auxiliary lemmas.

Lemma 2 The property of conditional acyclicity of TCP-nets is hereditary. That is, given two TCP-nets $\mathcal{N} = \langle \mathbf{V}, \mathsf{cp}, \mathsf{i}, \mathsf{ci}, \mathsf{cpt}, \mathsf{cit} \rangle$ and $\mathcal{N}' = \langle \mathbf{V}', \mathsf{cp}', \mathsf{i}', \mathsf{ci}', \mathsf{cpt}', \mathsf{cit}' \rangle$, if

1. $\mathcal{N}$ is conditionally acyclic, and
2. $\mathbf{V}' \subseteq \mathbf{V}$, $\mathsf{cp}' \subseteq \mathsf{cp}$, $\mathsf{i}' \subseteq \mathsf{i}$, $\mathsf{ci}' \subseteq \mathsf{ci}$, $\mathsf{cpt}' \subseteq \mathsf{cpt}$, $\mathsf{cit}' \subseteq \mathsf{cit}$,

then $\mathcal{N}'$ is also conditionally acyclic.

Proof: The proof is straightforward from Definition 10 since removing nodes and/or edges from $\mathcal{N}$, as well as removing some preference and importance information from CPTs and CITs of $\mathcal{N}$, can only remove cycles from the $\mathbf{w}$-directed graphs of $\mathcal{N}^\star$. Hence, if $\mathcal{N}$ is conditionally acyclic, then so is any subnet of $\mathcal{N}$. $\Box$

Lemma 3 Every conditionally acyclic TCP-net $\mathcal{N} = \langle \mathbf{V}, \mathsf{cp}, \mathsf{i}, \mathsf{ci}, \mathsf{cpt}, \mathsf{cit} \rangle$ contains at least one variable $X \in \mathbf{V}$, such that, for each $Y \in \mathbf{V} \setminus \{X\}$, we have $\langle \overrightarrow{Y, X} \rangle \notin \mathsf{cp}$, $(\overrightarrow{Y, X}) \notin \mathsf{i}$, and $(X, Y) \notin \mathsf{ci}$.

Proof: To prove the existence of such a root variable $X \in \mathcal{N}$, consider the dependency graph $\mathcal{N}^\star$. Since $\mathcal{N}$ is conditionally acyclic, there has to be a node $X' \in \mathcal{N}^\star$ that has neither incoming directed nor undirected edges associated with it. The see the latter, observe that (i) every endpoint of an undirected edge in $\mathcal{N}^\star$ will also have an incoming directed edge, and

4. The authors would like to thank Nic Wilson for pointing out an error in the original definition of conditionally acyclic TCP-nets in (Brafman & Domshlak, 2002).

(ii) there has to be at least one node in $\mathcal{N}^\star$ with no incoming directed edges, or otherwise the conditional acyclicity of $\mathcal{N}$ will be trivially violated. However, such a node X' will also be a root node in $\mathcal{N}$ since the edge set of $\mathcal{N}^\star$ is a superset of that of $\mathcal{N}$. $\square$

Theorem 1 Every conditionally acyclic TCP-net is satisfiable.

Proof: We prove this constructively by building a satisfying preference ordering. In fact, our inductive hypothesis will be stronger: any conditionally acyclic TCP-net has a strict total order that satisfies it. The proof is by induction on the number of problem variables. The result trivially holds for one variable by definition of CPTs, since we can simply use any strict total order consistent with its CPT (and trivially satisfying Definition 6.)

Assume that the theorem holds for all conditionally acyclic TCP-nets with fewer than n variables. Let $\mathcal{N}$ be a TCP-net over n variables, and X be one of the root variables of $\mathcal{N}$. (The existence of such a root X is guaranteed by Lemma 3.) Let $\mathcal{D}(X) = \{x_1, \ldots, x_k\}$ be the domain of the chosen root variable X, and let $x_1 \prec \ldots \prec x_k$ be a total ordering of $\mathcal{D}(X)$ that is consistent with the (possibly partial) preferential ordering dictated by $CPT(X)$ in $\mathcal{N}$. For each x_i, $1 \le i \le k$, construct a TCP-net $\mathcal{N}_i$, with $n-1$ variables $\mathbf{V} - \{X\}$ by removing X from the original network, and:

1. For each variable Y, such that there is a cp-arc $\langle \overrightarrow{X, Y} \rangle \in \mathcal{N}$, revise the CPT of Y by restricting each row to $X = x_i$.

2. For each ci-arc $\gamma = (Y_1, Y_2)$, such that $X \in \mathcal{S}(\gamma)$, revise the CIT of γ by restricting each row to $X = x_i$. If, as a result of this restriction, all rows in the new CIT express the same relative importance between Y_1 and Y_2, replace γ in $\mathcal{N}_i$ by the corresponding i-arc, i.e., either $(\overrightarrow{Y_1, Y_2})$ or $(\overrightarrow{Y_2, Y_1})$. Alternatively, if the CIT of γ becomes empty, then γ is simply removed from $\mathcal{N}_i$.

3. Remove the variable X, together with all cp-arcs of the form $\langle \overrightarrow{X, Y} \rangle$, and all i-arcs of the form $(\overrightarrow{X, Y})$.

From Lemma 2 we have that conditional acyclicity of $\mathcal{N}$ implies conditional acyclicity of all the reduced TCP-nets $\mathcal{N}_i$. Therefore, by the inductive hypothesis we can construct a preference ordering $\succ_i$ for each of the reduced networks $\mathcal{N}_i$. Now we can construct the preferential ordering for the original network $\mathcal{N}$ as follows. Every outcome with $X = x_j$ is ranked as preferred to any outcome with $X = x_i$, for $1 \le i < j \le k$. All the outcomes with identical X value, x_i, are ranked according to the ordering $\succ_i$ associated with $\mathcal{N}_i$ (ignoring the value of X). Clearly, by construction, the ordering we defined is a strict total order: it was obtained by taking a set of strict total orders and ordering them, respectively. From Definition 6, it is easy to see that this strict total order satisfies $\mathcal{N}$. $\square$

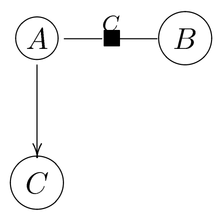
A close look at the proof of Theorem 1 reveals that the key property of conditionally acyclic TCP-nets is that they induce an "ordering" over the nodes of the network. This ordering is not fixed, but is context dependent. Different assignments to the variables in the prefix of this ordering will yield different suffixes. Put differently, the ordering depends on the values of the variables, and it captures the relative importance of each variable in each particular context. In particular, nodes that appear earlier in the ordering are more important in this particular context.

The above observation helps explain the rationale for our definition of the dependency graph (Definition 8). In some sense, this graph captures constraints on the ordering of variables. The TCP-net is conditionally acyclic if these constraints are satisfiable. We use this perspective to explain some choices made in the definition of the dependency graph which may seem arbitrary. First, consider the direction of (unconditional) importance edges from the more important to the less important variable. This simply goes in line with our desire to use a topological ordering in which the more important variables appear first. Second, consider the direction of CP-net edges from parent to children. It turns out that in CP-nets, there is an induced importance relationship between parents and children: parents are more important than their children (see (Boutilier et al., 2004a)). Thus, edges in the dependency graph must point from parent to child.

Finally, in order to make sense of this idea of context-dependent ordering, we must order the variables in the selector set of a ci-arc before the nodes connected by this arc. The motivation for this last choice may be a bit less clear. The following example shows the necessity of this (i.e., why Theorem 1 cannot be provided for a stronger notion of TCP-net acyclicity obtained by defining $\mathbf{w}$-directed graphs over $\mathcal{N}$ rather than over $\mathcal{N}^\star$).

c	$A \rhd B$
$\overline{c}$	$B \rhd A$

A	$a \succ \overline{a}$
B	$b \succ \overline{b}$
C	$a\ :\ \overline{c} \succ c$
	$\overline{a}\ :\ c \succ \overline{c}$



Consider a TCP-net as depicted above. This TCP-net $\mathcal{N}$ is defined over three boolean variables $\mathbf{V} = \{A, B, C\}$, and having $\mathsf{cp} = \{\langle\overrightarrow{A, C}\rangle\}$, $\mathsf{ci} = \{(A, B)\}$ with $\mathcal{S}(A, B) = \{C\}$, and $\mathsf{i} = \emptyset$. Clearly, the two possible $\mathbf{w}$-directed graphs of $\mathcal{N}$ (not of $\mathcal{N}^\star$) are acyclic. Now, suppose that there exists a strict partial order $\succ'$ over $\mathcal{D}(\mathbf{V})$ that satisfies $\mathcal{N}$. By Definition 6, we have

(1) $a\overline{b}\overline{c} \succ' a\overline{b}c$ (from $CPT(C)$),

(2) $\overline{a}b\overline{c} \succ' a\overline{b}\overline{c}$ (from $CIT((A, B))$ and $CPT(B)$),

(3) $\overline{a}bc \succ' \overline{a}b\overline{c}$ (from $CPT(C)$), and

(4) $a\overline{b}c \succ' \overline{a}bc$ (from $CIT((A, B))$ and $CPT(A)$).

However, this implies that $\succ'$ is not anti-symmetric, contradicting our assumption that $\succ'$ is a strict partial order.

5. Verifying Conditional Acyclicity

In contrast to standard acyclicity in directed graphs, the property of conditional acyclicity cannot be easily tested in general. Naive verification of the acyclicity of every $\mathbf{w}$-directed graph can require time exponential in the size of $\mathcal{S}(\mathcal{N})$. Here we study the complexity of verifying conditional acyclicity, discuss some hard and polynomial subclasses of this problem, and provide some sufficient and/or necessary conditions for conditional acyclicity that can be easily checked for certain subclasses of TCP-nets.

Let $\mathcal{N}$ be a TCP-net. If there are no cycles in the undirected graph underlying $\mathcal{N}^\star$ (i.e., the graph obtained from $\mathcal{N}^\star$ by making all directed edges into undirected edges), then clearly all $\mathbf{w}$-directed graphs of $\mathcal{N}^\star$ are acyclic, and this property of $\mathcal{N}^\star$ is simple to check. Alternatively, suppose that the underlying undirected graph of $\mathcal{N}^\star$ does contain cycles. If projection of each such cycle back to $\mathcal{N}^\star$ contains directed arcs oriented in different directions on the cycle (one "clockwise" and another "counter-clockwise"), then all $\mathbf{w}$-directed graphs of $\mathcal{N}^\star$ are still guaranteed to be acyclic. For instance, any subset (of size > 2) of the variables $\{T, A, S, C\}$ in our running example in Figure 4 forms a cycle in the undirected graph underlying $\mathcal{N}^\star$, yet each such cycle satisfies the aforementioned criterion. This sufficient condition for conditional acyclicity can also be checked in (low order) polynomial time.

The remaining cases are where the dependency graph $\mathcal{N}^\star$ contains what we define below as *semi-directed* cycles, and in the rest of this section we study these cases more closely.

Definition 11 *Let $\mathcal{A}$ be a mixed set of directed and undirected edges, and $\mathcal{A}^U$ be the undirected graph underlying $\mathcal{A}$ (that is, the graph obtained from $\mathcal{A}$ by dropping orientation of its directed edges.) We say that $\mathcal{A}$ is a* semi-directed cycle *if and only if*

(1) $\mathcal{A}^U$ forms a simple cycle (that is, $\mathcal{A}^U$ consists of a single connected component with all vertices having degree 2 w.r.t. $\mathcal{A}^U$).

(2) Not all of the edges in $\mathcal{A}$ are directed.

(3) All the directed edges of $\mathcal{A}$ point in the same direction along $\mathcal{A}^U$ (i.e., "clockwise" or "counter-clockwise").

Each assignment $\mathbf{w}$ to the selector sets of ci-arcs in a semi-directed cycle $\mathcal{A}$ of $\mathcal{N}^\star$ induces a direction for all these ci-arcs. We say that semi-directed cycle $\mathcal{A}$ is *conditionally acyclic* if under no such assignment $\mathbf{w}$ do we obtain a directed cycle from $\mathcal{A}$. Otherwise, $\mathcal{A}$ is called *conditionally directed*. Figure 5 illustrates a semi-directed cycle (based on the variables from our running example) with two possible configurations of its CITs that make this semi-directed cycle conditionally directed and conditionally acyclic, respectively.

Using these notions, Lemma 4 shows that testing conditional acyclicity for TCP-nets is naturally decomposable.

Lemma 4 A TCP-net $\mathcal{N}$ is conditionally acyclic if and only if every semi-directed cycle of $\mathcal{N}^\star$ is conditionally acyclic.

Proof: The proof is straightforward: If there is a variable assignment that makes one of the semi-directed cycles of $\mathcal{N}^\star$ conditionally directed, then no other cycle need be examined.

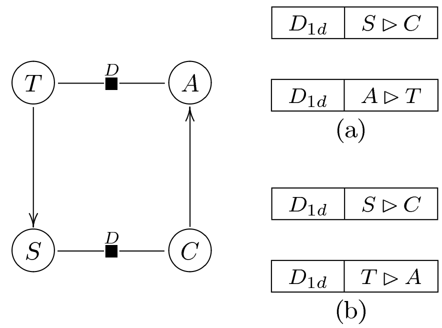


Figure 5: A semi-directed cycle: (a) conditionally directed, and (b) conditionally acyclic.

Conversely, consider one of the semi-directed cycles $\mathcal{A}$ of $\mathcal{N}^\star$. If no assignment to $\mathcal{S}(\mathcal{A})$ makes $\mathcal{A}$ conditionally directed, then additional assignments to variables in other selector sets do not change this property. $\square$

The decomposition presented by Lemma 4 allows us to prove our first complexity result for testing conditional acyclicity. Theorem 2 below shows that determining that a TCP-net is conditionally acyclic is coNP-hard.

Theorem 2 Given a binary-valued TCP-net $\mathcal{N}$, the decision problem: is there a conditionally directed cycle in $\mathcal{N}^\star$, is NP-complete, even if for every ci-arc $\gamma \in \mathcal{N}$ we have $|\mathcal{S}(\gamma)| = 1$.

Proof: The proof of hardness is by reduction from 3-SAT. Given a 3-CNF formula $\mathcal{F}$, construct the following TCP-net $\mathcal{N}$. For every variable X_i and every clause C_j in $\mathcal{F}$, construct a boolean variable X_i and variable C_j in $\mathcal{N}$, respectively (we retain the same names, for simplicity). In addition, for every clause C_j, construct three boolean variables $L_{j,k}$, $1 \leq k \leq 3$, corresponding to the literals appearing in C_j. Let n be the number of clauses in $\mathcal{F}$. The TCP-net $\mathcal{N}$ is somewhat degenerate, since it has no cp-arcs. However, it has an i-arc $(\overrightarrow{C_j, L_{j,k}})$ for each clause C_j and every literal $L_{j,k} \in C_j$. In addition, for every literal $L_{j,k} \in C_j$, there is a ci-arc $(L_{j,k}, C_{(j+1) \bmod n})$, whose selector variable is the variable X_i represented in $L_{j,k}$. The relative importance between $L_{j,k}$ and $C_{(j+1) \bmod n}$ on the selector X_i as follows: if $L_{j,k}$ is a positive literal, then variable $L_{j,k}$ is more important than $C_{(j+1) \bmod n}$ if X_i is true, and less important if X_i is false. For negative literals, the dependence on the selector variable is reversed. This completes the construction - clearly a polynomial-time operation. Figure 6 illustrates the subnet of $\mathcal{N}$ corresponding to a clause $C_j = (x_1 \vee x_2 \vee \overline{x_3})$, where $L_{j,1}, L_{j,2}, L_{j,3}$ correspond to $x_1, x_2, \overline{x_3}$, respectively.

We claim that $\mathcal{N}^\star$, the dependency graph for the network $\mathcal{N}$ we just constructed, has a conditionally directed cycle just when $\mathcal{F}$ is satisfiable[5]. It is easy to see that there is a path from C_j to $C_{(j+1) \bmod n}$ just when the values of the variables participating in C_j are such that C_j is satisfied. Thus, an assignment that creates a directed path from C_0 to C_0 is an

5. In this particular construction, the directed edges in $\mathcal{N}^\star$ outgoing from the selector variables X_i have no effect on the existence of conditionally directed cycles in $\mathcal{N}^\star$. Therefore, here we can simply consider the TCP-net $\mathcal{N}$ instead of its dependency graph $\mathcal{N}^\star$.

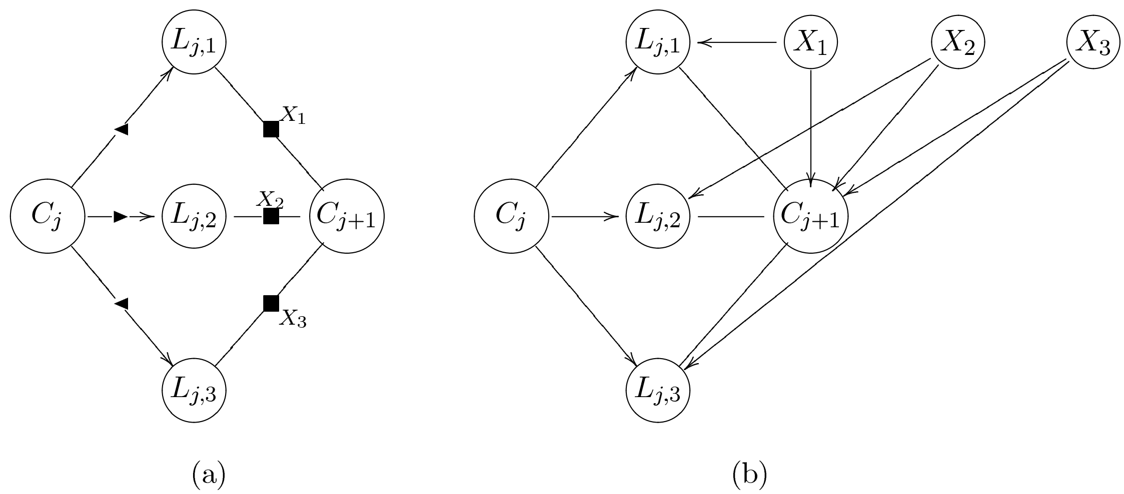


Figure 6: (a) TCP-net subnet for $C_j = (x_1 \vee x_2 \vee \overline{x_3})$, and (b) its dependency graph.

assignment that satisfies all clauses, and the problems are equivalent - hence our decision problem is NP-hard. Deciding existence of a conditional directed cycle is in NP: Indeed, verifying the existence of a semi-directed cycle $\mathcal{A}$ given an assignment to $\mathcal{S}(\mathcal{A})$ (the union of the selector sets of all ci-arcs in $\mathcal{A}$) can be done in polynomial time. Thus, the problem is NP-complete. $\square$

One reason for the complexity of the general problem, as emerges from the proof of Theorem 2, is the possibility that the number of semi-directed cycles in the TCP-net dependency graph is exponential in the size of the network. For example, the network in the reduction has 3^n semi-directed cycles, due to the three possible paths generated in each subnet as depicted in Figure 6(a). Thus, it is natural to consider networks for which the number of semi-directed cycles is not too large. In what follows, we call a TCP-net $\mathcal{N}$ *m-cycle bounded* if the number of different semi-directed cycles in its dependency graph $\mathcal{N}^\star$ is at most m.

From Lemma 4 it follows that, given an m-cycle bounded TCP-net $\mathcal{N}$, if m is polynomial in the size of $\mathcal{N}$, then we can reduce testing conditional acyclicity of $\mathcal{N}^\star$ into separate tests for conditional acyclicity of every semi-directed cycle $\mathcal{A}$ of $\mathcal{N}^\star$. A naive check for the conditional acyclicity of a semi-directed cycle $\mathcal{A}$ requires time exponential in the size of $\mathcal{S}(\mathcal{A})$ – where $\mathcal{S}(\mathcal{A})$ is the union of the selector sets of all ci-arcs in $\mathcal{A}$. Thus, if $\mathcal{S}(\mathcal{A})$ is small for each semi-directed cycle in $\mathcal{N}^\star$, then conditional acyclicity of $\mathcal{N}^\star$ can be checked quickly. In fact, often we can determine that a semi-directed cycle $\mathcal{A}$ is conditionally directed/acyclic even more efficiently than enumerating all possible assignments to $\mathcal{S}(\mathcal{A})$.

Lemma 5 Let $\mathcal{A}$ be a semi-directed cycle in $\mathcal{N}^\star$. If $\mathcal{A}$ is conditionally acyclic, then it contains a pair of ci-arcs γ_i, γ_j such that $\mathcal{S}(\gamma_i) \cap \mathcal{S}(\gamma_j) \neq \emptyset$.

In other words, if the selector sets of the ci-arcs in $\mathcal{A}$ are all pairwise disjoint, then $\mathcal{A}$ is conditionally directed. Thus, Lemma 5 provides a necessary condition for conditional acyclicity of $\mathcal{A}$ that can be checked in time polynomial in the number of variables.

Proof (Lemma 5) If all selector sets of the ci-arcs in $\mathcal{A}$ are pairwise disjoint, then trivially there exists an assignment to $\mathcal{S}(\mathcal{A})$ orienting all the ci-arcs of $\mathcal{A}$ in one direction. $\square$

Before developing sufficient conditions for conditional acyclicity, let us introduce some useful notation. First, given a ci-arc $\gamma = (X, Y)$, we say that an assignment $\mathbf{w}$ to a subset $\mathcal{S}'$ of $\mathcal{S}(\gamma)$ *orients* γ if all rows in $CIT(\gamma)$ consistent with $\mathbf{w}$ express the same relative importance between X and Y, if any. In other words, $\mathbf{w}$ orients γ if, given $\mathbf{w}$, the relative importance between X and Y is independent of $\mathcal{S}(\gamma) \setminus \mathcal{S}'$. Second, if a semi-directed cycle $\mathcal{A}$ contains some directed edges, we refer to their (by definition, unique) direction as the *direction of* $\mathcal{A}$.

Lemma 6 A semi-directed cycle $\mathcal{A}$ is conditionally acyclic if it contains a pair of ci-arcs γ_i, γ_j such that either:

(a) $\mathcal{A}$ contains directed edges, and for every assignment $\mathbf{w}$ to $\mathcal{S}(\gamma_i) \cap \mathcal{S}(\gamma_j)$, either γ_i or γ_j is oriented by $\mathbf{w}$ in the direction opposite to the direction of $\mathcal{A}$.

(b) All edges in $\mathcal{A}$ are undirected, and for every assignment $\mathbf{w}$ to $\mathcal{S}(\gamma_i) \cap \mathcal{S}(\gamma_j)$, γ_i and γ_j are oriented by $\mathbf{w}$ in opposite directions with respect to $\mathcal{A}$.

Proof: Follows immediately from the conditions in the lemma. $\square$

Lemma 6 provides a sufficient condition for conditional acyclicity of $\mathcal{A}$ that can be checked in time exponential in the maximal size of selector set intersection for a pair of ci-arcs in $\mathcal{A}$. Note that the size of the TCP-net is at least as large as the above exponential term, because the description of the CIT is exponential in the size of the corresponding selector set. Thus, checking this condition is only linear in the size of the network.

Definition 12 *Given a semi-directed cycle $\mathcal{A}$, let* $\mathsf{shared}(\mathcal{A})$ *denote the union of all pairwise intersections of the selector sets of the* ci*-arcs in $\mathcal{A}$:*

$$\mathsf{shared}(\mathcal{A}) = \bigcup_{\gamma_i, \gamma_j \in \mathcal{A}} \mathcal{S}(\gamma_i) \cap \mathcal{S}(\gamma_j)$$

Lemma 7

(a) If a semi-directed cycle $\mathcal{A}$ contains directed edges, then $\mathcal{A}$ is conditionally acyclic if and only if, for each assignment $\mathbf{u}$ on $\mathsf{shared}(\mathcal{A})$, there exists a ci-arc $\gamma_{\mathbf{u}} \in \mathcal{A}$ that is oriented by $\mathbf{u}$ in the direction opposite to the direction of $\mathcal{A}$.

(b) If a semi-directed cycle $\mathcal{A}$ contains only ci-arcs, then $\mathcal{A}$ is conditionally acyclic if and only if, for each assignment $\mathbf{u}$ on $\mathsf{shared}(\mathcal{A})$, there exist two ci-arcs $\gamma^1_{\mathbf{u}}, \gamma^2_{\mathbf{u}} \in \mathcal{A}$ that are oriented by $\mathbf{u}$ in opposite directions with respect to $\mathcal{A}$.

Proof: The sufficiency of the above condition is clear, since it subsumes the condition in Lemma 6. Thus, we are left with proving necessity. We start with the second case in which

$\mathcal{A}$ contains only ci-arcs. Assume to the contrary that $\mathcal{A}$ is conditionally acyclic, but there exists an assignment $\mathbf{u}$ on $\mathsf{shared}(\mathcal{A})$ such that no pair of ci-arcs in $\mathcal{A}$ are oriented by $\mathbf{u}$ in opposite directions with respect to $\mathcal{A}$.

For each ci-arc $\gamma \in \mathcal{A}$, let $\mathcal{S}^*(\gamma) = \mathcal{S}(\gamma) \setminus \mathsf{shared}(\mathcal{A})$. Consider the following disjoint partition $\mathcal{A} = \mathcal{A}^{\mathsf{i}}_{\mathbf{u}} \cup \mathcal{A}^{\mathsf{ci}}_{\mathbf{u}}$ induced by $\mathbf{u}$ on $\mathcal{A}$: For each ci-arc $\gamma \in \mathcal{A}$, if $\mathbf{u}$ orients γ, then we have $\gamma \in \mathcal{A}^{\mathsf{i}}_{\mathbf{u}}$. Otherwise, if the direction of γ is not independent of $\mathcal{S}^*(\gamma)$ given $\mathbf{u}$, we have $\gamma \in \mathcal{A}^{\mathsf{ci}}_{\mathbf{u}}$. We make two observations:

1. Our initial (contradicting) assumption implies that all the (now directed) edges in $\mathcal{A}^{\mathsf{i}}_{\mathbf{u}}$ agree on the direction with respect to $\mathcal{A}$.

2. If for some ci-arc $\gamma \in \mathcal{A}$ we have $\mathcal{S}^*(\gamma) = \emptyset$, then we have $\gamma \in \mathcal{A}^{\mathsf{i}}_{\mathbf{u}}$, since all the selectors of γ are instantiated by $\mathbf{u}$.

If we have $\mathcal{A}^{\mathsf{ci}}_{\mathbf{u}} = \emptyset$, then the first observation trivially contradicts our initial assumption that $\mathcal{A}$ is conditionally acyclic. Alternatively, if $\mathcal{A}^{\mathsf{ci}}_{\mathbf{u}} \neq \emptyset$, then, by definition of $\mathsf{shared}(\mathcal{A})$, we have that $\mathcal{S}^*(\gamma_i) \cap \mathcal{S}^*(\gamma_j) = \emptyset$ for each pair of ci-arcs $\gamma_i, \gamma_j \in \mathcal{A}^{\mathsf{ci}}_{\mathbf{u}}$. This means that we can assign each such (non-empty, by the second observation) $\mathcal{S}^*(\gamma_i)$ independently, and thus can extend $\mathbf{u}$ into an assignment on $\mathcal{S}(\mathcal{A})$ that will orient all the ci-arcs in $\mathcal{A}^{\mathsf{ci}}_{\mathbf{u}}$ either in the direction of $\mathcal{A}^{\mathsf{i}}_{\mathbf{u}}$ if $\mathcal{A}^{\mathsf{i}}_{\mathbf{u}} \neq \emptyset$, or in an arbitrary joint direction if $\mathcal{A}^{\mathsf{i}}_{\mathbf{u}} = \emptyset$. Again, this contradicts our assumption that $\mathcal{A}$ is conditionally acyclic. Hence, we have proved that our condition is necessary for the second case. The proof for the first case in which $\mathcal{A}$ contains some directed edges is similar. $\square$

In general, the size of $\mathsf{shared}(\mathcal{A})$ is $O(|\mathbf{V}|)$. Since we have to check the set of assignments over $\mathsf{shared}(\mathcal{A})$, this implies that the problem may be hard. Theorem 3 shows that this is indeed the case.

Theorem 3 Given a binary-valued, 1-cycle bounded TCP-net $\mathcal{N}$, the decision problem: is there a conditionally directed cycle in $\mathcal{N}^\star$, is NP-complete, even if for every ci-arc $\gamma \in \mathcal{N}$ we have $|\mathcal{S}(\gamma)| \leq 3$.

Proof: The proof of hardness is by reduction from 3-SAT. Given a 3-CNF formula $\mathcal{F}$, construct the following TCP-net $\mathcal{N}$. For every variable X_i and every clause C_j in $\mathcal{F}$, construct boolean variables X_i and C_j in $\mathcal{N}$, respectively. In addition, add a single dummy variable C, and an i-arc $(\overrightarrow{C, C_1})$. Let n be the number of clauses in $\mathcal{F}$. For $1 \leq j \leq n-1$, we have $n-1$ ci-arcs $E_j = (C_j, C_{j+1})$. In addition, we have ci-arc $E_n = (C_n, C)$. For all $1 \leq j \leq n$, the CIT for E_j is determined by clause C_j, as follows. The selector set for E_j is just the set of variables appearing in C_j, and the relative importance between the variables of E_j is determined as follows: C_j is less important than C_{j+1} just when the values of the variables in the selector set are such that C_j is false. (For $j = n$, read C instead of C_{j+1}).

The constructed TCP-net $\mathcal{N}$ is 1-cycle bounded, because there is only one semi-directed cycle in its dependency graph $\mathcal{N}^\star$, namely $C, C_1, \ldots, C_n, C$. We claim that this semi-directed cycle is conditionally directed just when $\mathcal{F}$ is satisfiable. It is easy to see that the directed path from C to C exists when all the ci-arcs are being directed from C_j to C_{j+1}, which occurs exactly when the variable assignment makes the clause C_j satisfiable. Hence, a directed cycle occurs in $\mathcal{N}$ exactly when the assignment makes all clauses satisfiable, making the two problems equivalent. Thus our decision problem is NP-hard. Finally, as deciding existence of a conditional directed cycle is in NP (see the proof of Theorem 3), the problem is NP-complete. □

Observe that the proof of Theorem 3 does not work when the size of all the selector sets is bounded by 2, because 2-SAT is in P. The immediate question is whether in this latter case the problem becomes tractable, and for binary-valued TCP-nets the answer is affirmative.

Theorem 4 Given a binary-valued, m-cycle bounded TCP-net $\mathcal{N}$, where m is polynomial in the size of $\mathcal{N}$ and, for every ci-arc $\gamma \in \mathcal{N}$ we have $|\mathcal{S}(\gamma)| \leq 2$, the decision problem: is there a conditional directed cycle in $\mathcal{N}^{\star}$, is in P.

Proof: The proof uses a reduction from conditional acyclicity testing to satisfiability. Let $\mathcal{A}$ be a semi-directed cycle with $|\mathcal{S}(\gamma)| \leq k$ for every ci-arc $\gamma \in \mathcal{A}$. We reduce the conditional acyclicity testing problem to an equivalent K-SAT problem instance. In particular, since 2-SAT is solvable in linear time (Even, Itai, & Shamir, 1976), together with Lemma 4 this proves the claim.

First, assume that $\mathcal{A}$ has at least one directed edge (either i-arc or cp-arc). By definition of semi-directed cycles, all directed edges of $\mathcal{A}$ point in the same direction, specifying the only possible cyclic orientation ω of $\mathcal{A}$. For each ci-arc $\gamma_i \in \mathcal{A}$, let the selector set be $\mathcal{S}(\gamma_i) = \{X_{i,1}, ..., X_{i,k}\}$.[6] Clearly, $\mathcal{A}$ is conditionally directed if and only if all the ci-arcs of $\mathcal{A}$ can be directed consistently with ω.

Given such a semi-directed cycle $\mathcal{A}$, we create a corresponding K-CNF formula $\mathcal{F}$, such that $\mathcal{F}$ is satisfiable just when $\mathcal{A}$ is conditionally directed. Let us call all $CIT(\gamma_i)$ entries that are consistent with ω by the term *ω-entries*. Since $\mathcal{S}(\gamma) = \{X_{i,1}, ..., X_{i,k}\}$ and $\mathcal{N}$ is binary valued, we can represent the non-ω entries in $CIT(\gamma_i)$ as a conjunction of disjunctions, i.e., in CNF form. The number of disjunctions is equal to the number of non-ω entries in $CIT(\gamma_i)$, and each disjunction is comprised of k literals. Thus, the representation of $CIT(\gamma_i)$ is a k-CNF formula, of size linear in the size of $CIT(\gamma_i)$. (In fact, the size of the resulting formula can sometimes be significantly smaller, as one can frequently simplify the component CNF fragments, but this property is not needed here.)

Finally, compose all the CNF representations of the $CIT(\gamma_i)$, for every $\gamma_i \in \mathcal{A}$, resulting in a k-CNF formula of size linear in the combined number of table entries. The construction of $\mathcal{F}$ is clearly a linear-time operation. Likewise, it is easy to see that $\mathcal{F}$ is satisfiable just when there is an assignment to $\mathcal{S}(\mathcal{A})$ converting $\mathcal{A}$ into a directed cycle.

The minor unresolved issue is with semi-directed cycles consisting of ci-arcs only. Given such a semi-directed cycle $\mathcal{A}$, we reduce the problem into two sub-problems with a directed arc. Let $\mathcal{A}'$ and $\mathcal{A}''$ be cycles created from $\mathcal{A}$ by inserting one dummy variable and one i-arc into $\mathcal{A}$ – clockwise for $\mathcal{A}'$, counter-clockwise for $\mathcal{A}''$. Now, $\mathcal{A}$ is conditionally directed if and only if either $\mathcal{A}'$ or $\mathcal{A}''$ (or both) are conditionally directed. □

6. If $|\mathcal{S}(\gamma_i)| < k$, the only impact will be a more compact reduction below.

To summarize our analysis of verifying conditional acyclicity, one must first identify the semi-directed cycles in the dependency graph of the TCP-net. Next, one must show that given each assignment $\mathbf{w}$ to the importance-conditioning variables of each semi-directed cycle, the $\mathbf{w}$-directed graph is acyclic. This problem is coNP-hard in general networks[7], but there are interesting classes of networks in which it is tractable. This is the case when the number of semi-directed cycles is not too large and either the size of $\mathsf{shared}(\mathcal{A})$ for each such cycle or the size of each selector set is not too large. Note that in practice, one would expect to have small selector sets – statements such as "X is more important than Y when $A = a$ and $B = b$ and $\dots$ and $Z = z$" appear to be more complex than what one would expect to hear. Thus, Lemma 6, Lemma 7 (for semi-directed cycles with small $\mathsf{shared}(\mathcal{A})$), and Theorem 4 are of more than just theoretical interest. Naturally, extending the toolbox of TCP-net subclasses that can be efficiently tested for consistency is clearly of both theoretical and practical interest.

6. Reasoning about Conditionally Acyclic TCP-nets

While automated consistency verification is the core part of the preference elicitation stage, efficiency of reasoning about user preferences is one of the main desiderata of any model for preference representation. Of particular importance is the task of preference-based optimization and constrained optimization, which we discuss in the first part of this section. Another important task, which provides an important component in the algorithm for constrained optimization we present, is outcome comparison – discussed in the second part of this section.

6.1 Generating Optimal Assignments

Following the notation of Boutilier et al. (2004a), if $\mathbf{x}$ and $\mathbf{y}$ are assignments to disjoint subsets $\mathbf{X}$ and $\mathbf{Y}$ of the variable set $\mathbf{V}$, respectively, we denote the combination of $\mathbf{x}$ and $\mathbf{y}$ by $\mathbf{xy}$. If $\mathbf{X} \cap \mathbf{Y} = \emptyset$ and $\mathbf{X} \cup \mathbf{Y} = \mathbf{V}$, we call $\mathbf{xy}$ a *completion* of assignment $\mathbf{x}$, and denote by $Comp(\mathbf{x})$ the set of all completions of $\mathbf{x}$.

One of the central properties of the original CP-net model (Boutilier et al., 2004a) is that, given an acyclic CP-net $\mathcal{N}$ and a (possibly empty) partial assignment $\mathbf{x}$ on its variables, it is simple to determine an outcome consistent with $\mathbf{x}$ (a completion of $\mathbf{x}$) that is *preferentially optimal* with respect to $\mathcal{N}$. The corresponding linear time *forward sweep* procedure is as follows: Traverse the variables in some topological order induced by $\mathcal{N}$, and set each unassigned variable to its most preferred value given its parents' values.

Our immediate observation is that this procedure works *as is* also for conditionally acyclic TCP-nets: The relative importance relations do not play a role in this case, and the network is traversed according to a topological order induced by the CP-net part of the given TCP-net. In fact, Corollary 1 holds for any TCP-net that has no directed cycles consisting only of cp-arcs.

Corollary 1 Given a conditionally acyclic TCP-net and a (possibly empty) partial assignment $\mathbf{x}$ on its variables, the forward sweep procedure constructs the most preferred outcome in $Comp(\mathbf{x})$.

7. This actually means that when the network is not too large, we can probably solve this in a reasonable amount of time.

This strong computational property of outcome optimization with respect to acyclic CP-nets (and conditionally acyclic TCP-nets) does not hold if some of the TCP-net variables are constrained by a set of hard constraints, $\mathcal{C}$. In this case, determining the set of *preferentially non-dominated*[8] *feasible* outcomes is not trivial. For acyclic CP-nets, a branch and bound algorithm for determining the optimal feasible outcomes was introduced by Boutilier, Brafman, Domshlak, Hoos, and Poole (2004b). This algorithm has the important *anytime* property – once an outcome is added to the current set of non-dominated outcomes, it is never removed. An important implication of this property is that the *first* generated assignment that satisfies the set of hard constraints is also preferentially non-dominated. In other words, finding just one non-dominated solution in this algorithm boils down to solving the underlying CSP under certain variable and value ordering strategies.

Here we develop an extension/modification of the algorithm of Boutilier et al. (2004b) to conditionally acyclic TCP-nets. The extended algorithm Search-TCP retains the anytime property and is shown in Figure 7. The key difference between processing an acyclic CP-net and a conditionally acyclic TCP-net is that the semantics of the former implicitly induces a single partial order of importance over the variables (where each node precedes its descendants) (Boutilier et al., 2004a), while the semantics of the latter induces a hierarchically-structured *set* of such partial orders: Each such partial order corresponds to a single assignment to the set of selector variables of the network, or, more specifically, to a certain $\mathbf{w}$-directed graph.

Formally, the problem is defined by a conditionally acyclic TCP-net $\mathcal{N}_{orig}$, and a set of hard constraints $\mathcal{C}_{orig}$, posed on the variables of $\mathcal{N}_{orig}$. The Search-TCP algorithm (depicted in Figure 7) is recursive, and each recursive call receives three parameters:

1. A TCP-net $\mathcal{N}$, which is a subnet of the original conditionally acyclic TCP-net $\mathcal{N}_{orig}$,
2. A set $\mathcal{C}$ of hard constraints among the variables of $\mathcal{N}$, which is a subset of the original set of constraints $\mathcal{C}_{orig}$ obtained by restricting $\mathcal{C}_{orig}$ to the variables of $\mathcal{N}$, and
3. An assignment $\mathcal{K}$ to all the variables in $\mathcal{N}_{orig} - \mathcal{N}$. In what follows, we refer to this assignment $\mathcal{K}$ as a *context*.

The initial call to Search-TCP is done with $\mathcal{N}_{orig}$, $\mathcal{C}_{orig}$, and $\{\}$, respectively.

Basically, the Search-TCP algorithm starts with an empty set of solutions, and gradually extends it by adding new non-dominated solutions to $\mathcal{C}_{orig}$. At each stage of the algorithm, the current set of solutions serves as a "lower bound" for future candidates; A new candidate at any point is compared to all solutions generated up to that point. If the candidate is dominated by no member of the current solution set, then it is added into this set.

The Search-TCP algorithm is guided by the graphical structure of $\mathcal{N}_{orig}$. It proceeds by assigning values to the variables in a top-down manner, assuring that outcomes are generated in an order that satisfies (i.e., consistent with) $\mathcal{N}$. On a recursive call to the Search-TCP procedure with a TCP-net $\mathcal{N}$, the eliminated variable X is one of the root variables of $\mathcal{N}$ (line 1). Recall that, by Lemma 3, conditional acyclicity of $\mathcal{N}$ guarantees the existence of such a root variable X. The values of X are considered according to

8. An outcome o is said to be non-dominated with respect to some preference order $\succ$ and a set of outcomes S if there is no other $o' \in S$ such that $o' \succ o$.

<u>Search-TCP</u> ($\mathcal{N}$, $\mathcal{C}$, $\mathcal{K}$)
Input: Conditionally acyclic TCP-net $\mathcal{N}$,
Hard constraints $\mathcal{C}$ on the variables of $\mathcal{N}$,
Assignment $\mathcal{K}$ to the variables of $\mathcal{N}_{orig} \setminus \mathcal{N}$.
Output: Set of all, non-dominated w.r.t. $\mathcal{N}$, solutions for $\mathcal{C}$.

1. Choose any variable X s.t. there is no cp-arc $\langle\overrightarrow{Y,X}\rangle$,
   no i-arc $(\overrightarrow{Y,X})$, and no (X,Y) in $\mathcal{N}$.
2. Let $x_1 \succ \ldots \succ x_k$ be a total order on $\mathcal{D}(X)$ consistent with the preference ordering of $\mathcal{D}(X)$ by the assignment on $Pa(X)$ in $\mathcal{K}$.
3. Initialize the set of local results by $\mathcal{R} = \emptyset$
4. **for** $(i = 1;\ i \leq k;\ i++)$ **do**
5. $\quad X = x_i$
6. $\quad$ Strengthen the constraints $\mathcal{C}$ by $X = x_i$ to obtain $\mathcal{C}_i$
7. $\quad$ **if** $\mathcal{C}_j \subseteq \mathcal{C}_i$ for some $j < i$ **or** $\mathcal{C}_i$ is inconsistent **then**
8. $\quad\quad$ **continue** with the next iteration
   $\quad$ **else**
9. $\quad\quad$ Let $\mathcal{K}'$ be the partial assignment induced by $X = x_i$ and $\mathcal{C}_i$
10. $\quad\quad$ $\mathcal{N}_i$ = Reduce ($\mathcal{N}$,$\mathcal{K}'$)
11. $\quad\quad$ Let $\mathcal{N}_i^1, \ldots, \mathcal{N}_i^m$ be the components of $\mathcal{N}_i$, connected
    $\quad\quad\quad$ either by the edges of $\mathcal{N}_i$ or by the constraints $\mathcal{C}_i$.
12. $\quad\quad$ **for** $(j = 1;\ j \leq m;\ j++)$ **do**
13. $\quad\quad\quad$ $\mathcal{R}_i^j$ = Search-TCP($\mathcal{N}_i^j$,$\mathcal{C}_i$,$\mathcal{K} \cup \mathcal{K}'$)
14. $\quad\quad$ **if** $\mathcal{R}_i^j \neq \emptyset$ for all $j \leq m$ **then**
15. $\quad\quad\quad$ **foreach** $o \in \mathcal{K}' \times \mathcal{R}_i^1 \times \cdots \times \mathcal{R}_i^m$ **do**
16 $\quad\quad\quad\quad$ **if** $\mathcal{K} \cup o' \not\succ \mathcal{K} \cup o$ holds <u>for each</u> $o' \in \mathcal{R}$ **then** add o to $\mathcal{R}$
17. **return** $\mathcal{R}$

Figure 7: The Search-TCP algorithm for conditionally acyclic TCP-net based constrained optimization.

the preference ordering induced on $\mathcal{D}(X)$ by the assignment provided by the context $\mathcal{K}$ to $Pa(X)$ (where $Pa(X)$ is defined with respect to $\mathcal{N}_{orig}$). Note that $\mathcal{K}$ necessarily contains some assignment to $Pa(X)$ since X is a root variable of the currently considered subnet $\mathcal{N}$ of $\mathcal{N}_{orig}$. Any additional variable assignment $X = x_i$ converts the current set of constraints $\mathcal{C}$ into a strictly non-weaker constraint set $\mathcal{C}_i$. As a result of this propagation of $X = x_i$, values for some variables (at least, the value of X) are fixed automatically, and this partial assignment $\mathcal{K}'$ extends the current context $\mathcal{K}$ in recursive processing of the next variable. The Reduce procedure, presented in Figure 8, refines the TCP-net $\mathcal{N}$ with respect to $\mathcal{K}'$: For each variable assigned by $\mathcal{K}'$, we reduce both the CPTs and the CITs involving this variable, and remove this variable from the network. This reduction of the CITs may remove conditioning of relative importance between some variables, and thus convert some ci-arcs into i-arcs, and/or remove some ci-arcs completely. The main point is that, in contrast to CP-nets, for a pair of X values x_i, x_j, the variable elimination orderings for processing the networks $\mathcal{N}_i$ and $\mathcal{N}_j$, resulting from propagating $\mathcal{C}_i$ and $\mathcal{C}_j$, respectively, may *disagree* on the ordering of some variables.

```
Reduce (𝒩, 𝒦′)
foreach {X = x_i} ∈ 𝒦′ do
  foreach cp-arc ⟨X→Y⟩ ∈ 𝒩 do
    Restrict the CPT of Y to the rows dictated by X = x_i.
  foreach ci-arc γ = (Y1, Y2) ∈ 𝒩 s.t. X ∈ 𝒮(γ) do
    Restrict the CIT of γ to the rows dictated by X = x_i.
    if, given the restricted CIT of γ, relative importance
      between Y1 and Y2 is independent of 𝒮(γ), then
        if CIT of γ is not empty then
          Replace γ by the corresponding i-arc.
        else Remove γ.
  Remove from 𝒩 all the edges involving X.
return 𝒩.
```

Figure 8: The Reduce procedure.

If the partial assignment $\mathcal{K}'$ causes the current CP-net to become disconnected with respect to both the edges of the network and the inter-variable hard constraints, then each connected component invokes an independent search (lines 11-16). This is because optimization of the variables within such a component is independent of the variables outside that component. In addition, after strengthening the set of constraints $\mathcal{C}$ by $X = x_i$ to $\mathcal{C}_i$ (line 6), some pruning takes place in the search tree (lines 7-8): If the set of constraints $\mathcal{C}_i$ is strictly more restrictive than some other set of constraints $\mathcal{C}_j = \mathcal{C} \cup \{X = x_j\}$ where $j < i$, then the search under $X = x_i$ is not continued. The reason for this pruning is that it can be shown that any feasible outcome a involving $X = x_i$ is dominated by (i.e., less preferable than) some feasible outcome b involving $X = x_j$ and thus a cannot be in the set of non-dominated solutions for the original set of constraints[9]. Therefore, the search is depth-first branch-and-bound, where the set of non-dominated solutions generated so far is a proper subset of the required set of all the non-dominated solutions for the problem, and thus it corresponds to the current lower bound.

When the potentially non-dominated solutions for a particular subgraph are returned with some assignment $X = x_i$, each such solution is compared to all non-dominated solutions involving more preferred (in the current context $\mathcal{K}$) assignments $X = x_j$, $j < i$ (line 16). A solution with $X = x_i$ is added to the set of the non-dominated solutions for the current subgraph and context if and only if it passes this non-domination test. From the semantics

9. This pruning was introduced by Boutilier et al. (2004b) for acyclic CP-nets, and it remains valid the same way for conditionally acyclic TCP-nets. For the proof of soundness of this pruning technique we refer the reader to Lemma 2 in (Boutilier et al., 2004b).

of the TCP-nets, given the same context $\mathcal{K}$, a solution involving $X = x_i$ can not be preferred to a solution involving $X = x_j$, $j < i$. Thus, the generated global set $\mathcal{R}$ never shrinks.

Theorem 5 Given a conditionally acyclic TCP-net $\mathcal{N}$ and a set of hard constraints $\mathcal{C}$ over the variables of $\mathcal{N}$, an outcome o belongs to the set $\mathcal{R}$ generated by the algorithm Search-TCP if and only if o is consistent with $\mathcal{C}$, and there is no other outcome o' consistent with $\mathcal{C}$ such that $\mathcal{N} \models o' \succ o$.

Proof: Let $\mathcal{R}_\mathcal{C}$ be the desired set of all the preferentially non-dominated solution to $\mathcal{C}$. To prove this theorem, we should show that:

1. *Completeness:* No preferentially non-dominated solution to $\mathcal{C}$ is pruned out, that is, we have $\mathcal{R} \supseteq \mathcal{R}_\mathcal{C}$, and
2. *Soundness:* The resulting set $\mathcal{R}$ contains no preferentially dominated solution to $\mathcal{C}$, that is, $\mathcal{R} \subseteq \mathcal{R}_\mathcal{C}$.

(1) The solutions to $\mathcal{C}$ are pruned by Search-TCP only in two places, namely at the search space pruning in lines 7-8, and at the non-dominance test step in line 16. For the first case, the correctness of the pruning technique used in lines 7-8 is given by Lemma 2 in (Boutilier et al., 2004b), and thus this pruning does not violate completeness of Search-TCP. For the second case, if an explicitly generated solution o is rejected due to the failure of its non-dominance test, then $o \notin \mathcal{R}_\mathcal{C}$ is apparent since the rejection of o here is based on presenting a concrete solution o' such that $\mathcal{N} \models o' \succ o$. Hence, we have $\mathcal{R} \supseteq \mathcal{R}_\mathcal{C}$.

(2) To show $\mathcal{R} \subseteq \mathcal{R}_\mathcal{C}$ it is enough to prove that a newly generated solution cannot dominate an existing solution, that is, if o was added to the generated set of solutions after o' then it is not the case that $\mathcal{N} \models o \succ o'$. The proof is by induction on the number of problem variables. First, the claim trivially holds for any one-variable TCP-net, as the order in which the solutions are examined in line 16 coincides with the total order selected for the single variable of the network in line 2. Now, assume that the claim holds for all conditionally acyclic TCP-nets with fewer than n variables. Let $\mathcal{N}$ be a TCP-net over n variables, $\mathcal{C}$ be a set of hard constraints on these variables, and X be the root variable of $\mathcal{N}$ selected in line 1. Let $\mathcal{R} = \{o_1, \ldots, o_r\}$ be the output of Search-TCP for these $\mathcal{N}$ and $\mathcal{C}$, where the elements of $\mathcal{R}$ are numbered according to the order of their non-dominance examination in line 16. Now, assume that there exists a pair of assignments $o_i, o_j \in \mathcal{R}$, such that $i < j$, yet $\mathcal{N} \models o_j \succ o_i$.

First, suppose that o_i and o_j provide the same value to X, that is $o_i = x_l o'_i$ and $o_j = x_l o'_j$, for some $x_l \in \mathcal{D}(X)$. In this case, however, o'_i and o'_j belong to the output of the same recursive call to Search-TCP with $\mathcal{N}_l$ and $\mathcal{C}_l$, and thus, by our inductive hypothesis, o'_i and o'_j are preferentially incomparable. Likewise, $\mathcal{N}_l$ is obtained in line 10 by reducing $\mathcal{N}$ with respect to x_l, and thus the variables of $\mathcal{N}_l$ are preferentially independent of X. Hence, preferential incomparability of o'_i and o'_j implies preferential incomparability of o_i and o_j, and thus $\mathcal{N} \models o_j \succ o_i$ cannot be the case.

Alternatively, suppose that o_i and o_j provide two different values to X, that is $o_i = x_l o'_i$ and $o_j = x_m o'_j$, $x_l, x_m \in \mathcal{D}(X)$, where $\mathcal{D}(X)$ is numbered according to the total ordering of its values selected in line 2. Observe that, by the construction of Search-TCP, $i < j$ trivially implies $l < m$. However, using the arguments identical to these in the constructive proof of Theorem 6, there exists at least one preference order $\succ$ of the complete assignments to the variables of $\mathcal{N}$ in which we have $o_i \succ o_j$. Hence, it cannot be the case that $\mathcal{N} \models o_j \succ o_i$, and thus contradiction of our assumption that $\mathcal{N} \models o_j \succ o_i$ is now complete. $\square$

Note that, if we are interested in getting *one* non-dominated solution for the given set of hard constraints (which is often the case), we can output the *first* feasible outcome generated by Search-TCP. No comparisons between pairs of outcomes are required because there is nothing to compare with the first generated solution. However, if we are interested in getting *all*, or even *a few* non-dominated solutions, then the efficiency of preferential comparison between pairs of outcomes becomes an important factor in the entire complexity of the Search-TCP algorithm. Hence, in the next section we consider such preferential comparisons more closely.

6.2 Dominance Testing for TCP-nets

One of the most fundamental queries in any preference-representation formalism is whether some outcome o dominates (i.e., is strictly preferred to) some other outcome o'. As discussed above, such *dominance queries* are required whenever we wish to generate more than one non-dominated solution to a set of hard constrains. Much like in CP-nets, a dominance query $\langle \mathcal{N}, o, o' \rangle$ with respect to a TCP-net can be treated as a search for an improving flipping sequence from the (purported) less preferred outcome o' to the (purported) more preferred outcome o through a sequence of successively more preferred outcomes, such that each flip in this sequence is directly sanctioned by the given TCP-net. Formally, an improving flipping sequence in the context of TCP-nets can be defined as follows:

Definition 13 *A sequence of outcomes*

$$o' = o_0 \prec o_1 \prec \cdots \prec o_{m-1} \prec o_m = o$$

is an improving flipping sequence with respect to a TCP-net $\mathcal{N}$ *if and only if, for* $0 \leq i < m$, *either*

1. *(CP-flips) outcome o_i is different from the outcome o_{i+1} in the value of exactly one variable X_j, and $o_i[j] \prec o_{i+1}[j]$ given the (identical) values of $Pa(X_j)$ in o_i and o_{i+1}, or*

2. *(I-flips) outcome o_i is different from the outcome o_{i+1} in the value of exactly* two *variables X_j and X_k, $o_i[j] \prec o_{i+1}[j]$ and $o_i[k] \succ o_{i+1}[k]$ given the (identical) values of $Pa(X_j)$ and $Pa(X_k)$ in o_i and o_{i+1}, and $X_j \rhd X_k$ given $\mathcal{RI}(X_j, X_k|\mathbf{Z})$ and the (identical) values of $\mathbf{Z}$ in o_i and o_{i+1}.*[10]

Clearly, each value flip in such a flipping sequence is sanctioned by the TCP-net $\mathcal{N}$, and the CP-flips are exactly the flips allowed in CP-nets (Boutilier et al., 2004a).

10. We implicitly assumed that neither node is the parent of the other. An implicit consequence of the standard semantics of conditional preferences is a node is more important than its children. Thus, there is no need to specify this explicitly.

Theorem 6 Given a TCP-net $\mathcal{N}$ and a pair of outcomes o and o', we have that $\mathcal{N} \models o \succ o'$ if and only if there is an improving flipping sequence with respect to $\mathcal{N}$ from o' to o.

Proof:

$\Longleftarrow$ Given an improving flipping sequence $\mathcal{F}$:

$$o' = o_0 \prec o_1 \prec \cdots \prec o_{m-1} \prec o_m = o$$

from o' to o with respect to $\mathcal{N}$, by Definition 13, we have $\mathcal{N} \models o_i \succ o_{i+1}$ for any improving flip from $\mathcal{F}$. The proposition follows from the transitivity of preferential entailment with respect to TCP-nets (Lemma 1).

$\Longrightarrow$ Let $\mathcal{G}$ be the graph of preferential ordering induced by $\mathcal{N}$, i.e., nodes of $\mathcal{G}$ stand for all outcomes, and there is a directed edge from o_1 to o_2 if and only if there is an improving CP-flip or I-flip of o_1 to o_2, sanctioned by $\mathcal{N}$. Clearly, directed paths in $\mathcal{G}$ are equivalent to improving flipping sequences with respect to $\mathcal{N}$.

First, we show that any preference ordering $\succ$ that respects the paths in $\mathcal{G}$ (that is, if there is a path from o_1 to o_2 in $\mathcal{G}$, then we have $o_2 \succ o_1$) satisfies $\mathcal{N}$. Assume to the contrary that $\succ^*$ respects the paths in $\mathcal{G}$, and does not satisfy $\mathcal{N}$. Then, by the definition of satisfiability (Definition 7), there must exist either:

1. Some variable X, assignment $\mathbf{p} \in \mathcal{D}(Pa(X))$, values $x, x' \in \mathcal{D}(X)$, and assignment $\mathbf{w}$ to the remaining variables $\mathbf{W} = \mathbf{V} - (X \cup Pa(X))$, such that $\mathbf{p}x\mathbf{w} \succ^* \mathbf{p}x'\mathbf{w}$, but $CPT(X)$ dictates that $x' \succ x$ given $\mathbf{p}$, or

2. Some importance arc ξ between a pair of variables X and Y, assignment $\mathbf{z} \in \mathcal{D}(\mathcal{S}(\xi))$ (if ξ is an i-arc, then $\mathcal{S}(\xi) = \emptyset$), values $x, x' \in \mathcal{D}(X), y, y' \in \mathcal{D}(Y)$, and assignment $\mathbf{w}$ to the remaining variables $\mathbf{W} = \mathbf{V} - (\{X, Y\} \cup \mathcal{S}(\xi))$, such that $\mathbf{p}xy\mathbf{w} \succ^* \mathbf{p}x'y'\mathbf{w}$, but (i) the $CPT(X)$ dictates that $x' \succ x$, and (ii) the (possibly empty) CIT of ξ dictates that $X \rhd Y$ given $\mathbf{z}$.

However, in the first case, if $\mathcal{N}$ specifies $x' \succ x$ given $\mathbf{p}$, there is a CP-flip from $\mathbf{p}x'\mathbf{w}$ to $\mathbf{p}x\mathbf{w}$, contradicting the fact that $\succ^*$ extends $\mathcal{G}$. Similarly, in the second case, if $\mathcal{N}$ specify $x' \succ x$ given $\mathbf{w}$, and $X \rhd Y$ given z, then there is an I-flip from $\mathbf{p}x'y'\mathbf{w}$ to $\mathbf{p}xy\mathbf{w}$, contradicting the fact that $\succ^*$ extends $\mathcal{G}$.

Now, by the construction of $\mathcal{G}$, if there is no improving flipping sequence from o' to o, then there is no directed path in $\mathcal{G}$ from o' to o. Therefore, there exist a preference ordering $\succ^*$ respecting the paths in $\mathcal{G}$ in which $o' \succ^* o$. However, based on the above observation on preference orderings respecting the paths in $\mathcal{G}$, $\succ^*$ also satisfies $\mathcal{N}$, which implies $\mathcal{N} \not\models o \succ o'$. $\Box$

Various methods can be used to search for a flipping sequence. In particular, we believe that at least some of the techniques, developed for this task with respect to CP-nets by Domshlak and Brafman (2002), Domshlak (2002), and Boutilier et al. (2004a) can be applied to the TCP-net model – an issue left for future research. However, in general, dominance testing with respect to CP-nets (and thus TCP-nets) is known to be

NP-hard (Boutilier et al., 2004a), thus in practice one may possibly consider performing approximate constrained optimization, using the Search-TCP algorithm with a dominance testing based on one of the tractable refinements of TCP-nets such as those discussed by Brafman, Domshlak, and Kogan (2004a).

7. Discussion

CP-nets (Boutilier et al., 1999, 2004a) is a relatively new graphical model for representation and reasoning about preferences. Its development, however, already stimulated research in several directions (e.g., see (Brafman & Chernyavsky, 2005; Brafman & Dimopoulos, 2004; Brewka, 2002; Boutilier et al., 2001; Domshlak et al., 2003; Rossi et al., 2004; Lang, 2002; Wilson, 2004b, 2004a)). In this paper we introduced the qualitative notions of absolute and conditional relative importance between pairs of variables and extended the CP-net model to capture the corresponding preference statements. The extended model is called TCP-nets. We identified a wide class of TCP-nets that are satisfiable, notably the class of conditionally acyclic TCP-nets, and analyzed complexity and algorithms for testing membership in this class of networks. We also studied reasoning about TCP-nets, focusing on outcome optimization in conditionally acyclic TCP-nets with and without hard constraints.

Our work opens several directions for future research. First, an important open theoretical question is the precise complexity of dominance testing in TCP-nets. In the context of CP-nets this problem has been studied by Domshlak (2002), Boutilier et al. (2004a), Goldsmith et al. (2005). Another question is the consistency of TCP-nets that are not conditionally acyclic. A preliminary study of this issue in context of cyclic CP-nets has been done by Domshlak and Brafman (2002) and Goldsmith et al. (2005).

The growing research on preference modeling is motivated by the need for preference elicitation, representation, and reasoning techniques in diverse areas of AI and user-centric information systems. In particular, one of the main application areas we have in mind is this of automatic personalized product configuration (Sabin & Weigel, 1998). Thus, in the remaining part of this section, we first consider the process of preference elicitation with TCP-nets, listing a few practical challenges that should be addressed to make this process appealing to users en-masse. Then, we relate our work to some other approaches to preference-based optimization.

7.1 Preference Elicitation with TCP-nets (and Other Logical Models of Preference)

The process of preference elicitation is known to be complex as into account should be taken not only the formal model of the user's preferences but also numerous important factors of human-computer interaction (e.g., see (Faltings, Pu, Torrens, & Viappiani, 2004; Pu & Faltings, 2004)). In this paper we focus on a formalism for structuring and analyzing the user's preferences, although for some (probably offline) applications, this formalism could actually be used to drive the input process, much like a Bayes network can be used to help experts express their beliefs.

Depending on the application, a *schematic* process of constructing a TCP-net would commence by asking the decision maker to identify the variables of interest, or by presenting them to the user, if they are fixed. For example, in the application of CP-net to adaptive document presentation (Domshlak, Brafman, & Shimony, 2001; Brafman, Domshlak, & Shimony, 2004b), the content provider chooses a set of content elements, which correspond to the set of variables. For an online shopper-assistant agent, the variables are likely to be fixed (e.g., if the agent is an online PC customizer) (Brafman et al., 2004a). Next, the user is asked to consider for each variable, the value of which other variables influence her preferences over the values of this variable. At this point cp-arcs and CPTs are introduced. Next, the user is asked to consider relative importance relations, and the i and ci-arcs are added. For each ci-arc, the corresponding CIT is filled.

Clearly, one may prefer to keep the preference elicitation process more user-driven, allowing the user simply provide us with a set of preference statements. But if such a set of statements fits the language expressible by the TCP-nets model, then the specific TCP-net underlying these statements can be constructed from a simple analysis of referents and conditionals of these statements. Such TCP-net extraction from the statements will be simpler if these statements will be provided in this or another formal language, or obtained via some carefully designed, structured user interface. However, for the user it is obviously more natural to provide these statements in natural language. Hence, an interesting practical question related to elicitation of qualitative preferences is model acquisition from speech and/or text (Asher & Morreau, 1995; Glass, 1999; Bethard, Yu, Thornton, Hatzivassiloglou, & Jurafsky, 2004). Observe that the intuitiveness of the qualitative preferential statements is closely related to the fact that they have a straightforward representation in natural language of everyday life. In addition, collections of typical preferential statements seem to form a linguistic domain that is a priori constrained in a very special manner. This may allow us to develop specialized techniques and tools for understanding the corresponding language. Both offline and online language understanding should be considered, since a user can either describe her preferences offline, as a self-contained text, or can be asked online, as a part of interactive process of (possibly mixed) preference elicitation and preference-based constrained optimization.

Yet another possible approach for eliciting TCP-nets, as well as some alternative logical models of preferences, would be to allow the user expressing pair-wise comparisons between completely specified choices, and then construct a TCP-net consistent with this input. In the scope of quantitative models for preference representation, such an example-based model generation has been adopted in numerous user-centric optimization systems (e.g., see (Linden, Hanks, & Lesh, 1997; Blythe, 2002).) However, devising such a framework for learning qualitative models of preference seems to be somewhat more challenging. In theory, nothing prevents us from adopting example-based generation of TCP-nets since the latter can be seen as just a compact representation of a preference relation over a space of choices. The question, however, is whether a reasonably small set of pair-wise choice comparisons can provide us with a sufficient basis for learning not just a TCP-net consistent with these "training examples", but a compact TCP-net that will generalize in a justifiable manner beyond the provided examples. To the best of our knowledge, so far this question has been studied for no logical preference-representation models, and hence it clearly poses a challenging venue for future research.[11]

11. Note that, if we are only interested in compact modeling of pair-wise comparisons between the choices, then numerous techniques from the area of machine learning can be found useful. For instance, one can learn a decision tree classifying *ordered pairs* of choices as "preferred" (first choice to the second choice)

7.2 Related Work

As we show in Section 6, extending CP-nets to TCP-nets is appealing mainly in the scope of decision scenarios where the space of all syntactically possible choices is (either explicitly or implicitly) constrained by some hard constraints. We now review some related approaches to preference-based optimization that appeared in the literature.

A primary example of preference-based optimization is the soft-constraints formalism (e.g., see Bistarelli et al. (1997)), developed to model constraint satisfaction problems that are either over-constrained (and thus unsolvable according to the standard meaning of satisfaction) (Freuder & Wallace, 1992), or suffer from imprecise knowledge about the actual constraints (Fargier & Lang, 1993). In this formalism, the constrained optimization problem is represented as a set of preference orders over assignments to subsets of variables, together with some operator for combining the preference relations over these subsets of variables to a preference relation over the assignments to the whole set of variables. Each such subset of variables corresponds to a soft constraint that can be satisfied to different extent by different variable assignments. There is much flexibility in how such "local" preference orders are specified, and how they are combined. Various soft-constraints models, such as weighted (Bistarelli et al., 1999), fuzzy (Schiex, 1992), probabilistic (Fargier & Lang, 1993), and lexicographic (Fargier et al., 1993) CSPs, are discussed in the literature on soft constraint satisfaction.

The conceptual difference between our approach and the soft-constraints formalism is that the latter is based on tightly coupled representation of preferences and constraints, while our representation of these two paradigms is completely decoupled. Informally, soft constraints machinery has been developed for optimization *of* partial constraint satisfaction, while we are dealing with optimization *in the face* of constraint satisfaction. For instance, in personalized product configuration, there are two parties involved typically: the manufacturer and the consumer. The manufacturer brings forth its product expertise, and with it a set of hard constraints on possible system configurations and the operating environment. The user expresses her preferences over properties of the final product. Typically numerous configurations satisfy the production constraints, and the manufacturer strives to provide the user with maximal satisfaction by finding one of the most preferred, feasible product configuration. This naturally leads to a decoupled approach.

Freuder and O'Sullivan (2001) proposed a framework of interactive sessions for over-constrained problems. During such a session, if the constraint solver discovers that no solution for the current set of constraints is available, the user is asked to consider "tradeoffs". For example, following Freuder and O'Sullivan (2001), suppose that the set of user requirements for a photo-camera configuration tool is that the weight of the camera should be less that 10 ounces and the zoom lens should be at least 10X. If no camera meets these requirements, the user may specify tradeoffs such as "I will increase my weight limit to 14 ounces, if I can have a zoom lens of at least 10X" (possibly using some suggestions automatically generated by the tool). In turn, these tradeoffs are used for refining the current set of requirements, and the system goes into a new constraint satisfaction process.

or "not preferred". However, such a classification does not guarantee in general that the resulting binary relation over the space of choices will be anti-symmetric under the assumption of preference transitivity (a joint property that considered to be extremely natural in the literature on preference structures (Hansson, 2001).)

The tradeoffs exploited by Freuder and O'Sullivan (2001) correspond to the information captured in TCP-nets by the i-arcs. However, instead of treating this information as an incremental "compromising" update to the set of hard constraints as done by Freuder and O'Sullivan (2001), in the TCP-net based constrained optimization presented in Section 6, we exploit this information to guide the constraint solver to the preferable feasible solutions. On the other hand, the motivation and ideas behind the work of Freuder and O'Sullivan as well as these in some other works on interactive search (see works, e.g., on interactive goal programming (Dyer, 1972), and interactive optimization based on example critique (Pu & Faltings, 2004)) open a venue for future research on interactive preference-based constrained optimization with TCP-nets, where elicitation of the user preferences is to be interleaved with the search for feasible solution.

The notion of lexicographic orders/preferences (Fishburn, 1974; Schiex et al., 1995; Freuder et al., 2003) is closely related to our notion of importance. The idea of a lexicographic ordering is often used in qualitative approaches for multi-criteria decision making. Basically, it implies that if one item does better than another on the most important (lexicographically earlier) criteria on which they differ, it is considered better overall, regardless on how poorly it may do on all other criteria. Thus, if we have four criteria (or attributes) A, B, C, D, thus ordered, and o does well on A but miserably on B, C and D, whereas o' is slightly worse on A but much better on all other criteria, o is still deemed better. In terms of our notion of variable importance, a lexicographic ordering of attributes denotes a special form of relative importance of an attribute versus a set of attributes. Thus, in the example above, A is more important than B, C and D combined; B is more important than C and D combined, and C is more important than D. We note that Wilson (2004b) provides a nice language that can capture such statements and more. Wilson allows statements of the form "$A = a$ is preferred to $A = a$ all-else-being-equal, except for B and C." That is, given two outcomes that differ on A, B and C only, the one that assigns a to A is preferred to the one that assigns a', regardless of the value of B and C in these outcomes. Hence, this richer language can in particular capture lexicographic preferences.

While we believe a lexicographic ordering over attributes is typically too strong, we think the flexibility provided by Wilson's language could be quite useful. However, once one starts analyzing relationships between sets of attributes, the utility of graphical models and their analysis power becomes questionable. Indeed, we are not aware of a graphical analysis of Wilson's approach, except for the special case covered by TCP-nets. Moreover, our intuition is that relative importance of sets will be a notion that users are much less comfortable specifying in many applications. However, this hypothesis requires empirical verification, as well as a more general study of the exact expressive power of TCP-nets, i.e., characterizing partial orders that are expressible using this language. We believe that this is an important avenue for future research.

Acknowledgments

Ronen Brafman and Solomon Shimony were partly supported by the Paul Ivanier Center for Robotics and Production Management. Ronen Brafman was partly supported by NSF grants SES-0527650 and IIS-0534662. Ronen Brafman's permanent address is: Department of Computer Science, Ben Gurion University, Israel.

References

Asher, N., & Morreau, M. (1995). What some generic sentences mean. In Carlson, G., & Pelletier, F. J. (Eds.), *The Generic Book*, pp. 300–338. Chicago University Press.

Bethard, S., Yu, H., Thornton, A., Hatzivassiloglou, V., & Jurafsky, D. (2004). Automatic extraction of opinion propositions and their holders. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T., & Verfaillie, G. (1999). Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, *4*(3), 275–316.

Bistarelli, S., Montanari, U., & Rossi, F. (1997). Semiring-based constraint solving and optimization. *Journal of the ACM*, *44*(2), 201–236.

Blythe, J. (2002). Visual exploration and incremental utility elicitation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 526–532.

Boutilier, C., Bacchus, F., & Brafman, R. I. (2001). UCP-networks: A directed graphical representation of conditional utilities. In *Proceedings of Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 56–64.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2004a). CP-nets: A tool for representing and reasoning about conditional *ceteris paribus* preference statements. *Journal of Artificial Intelligence Research (JAIR)*, *21*, 135–191.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2004b). Preference-based constrained optimization with CP-nets. *Computational Intelligence (Special Issue on Preferences in AI and CP)*, *20*(2), 137–157.

Boutilier, C., Brafman, R., Hoos, H., & Poole, D. (1999). Reasoning with conditional *ceteris paribus* preference statements. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 71–80. Morgan Kaufmann Publishers.

Brafman, R., & Chernyavsky, Y. (2005). Planning with goal preferences and constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pp. 182–191, Monterey, CA.

Brafman, R., & Domshlak, C. (2002). Introducing variable importance tradeoffs into CP-nets. In *Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 69–76, Edmonton, Canada.

Brafman, R., Domshlak, C., & Kogan, T. (2004a). Compact value-function representations for qualitative preferences. In *Proceedings of the Twentieth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 51–58, Banff, Canada.

Brafman, R., Domshlak, C., & Shimony, S. E. (2004b). Qualitative decision making in adaptive presentation of structured information. *ACM Transactions on Information Systems*, *22*(4), 503–539.

Brafman, R. I., & Dimopoulos, Y. (2004). Extended semantics and optimization algorithms for cp-networks. *Computational Intelligence (Special Issue on Preferences in AI and CP)*, *20*(2), 218–245.

Brafman, R. I., & Friedman, D. (2005). Adaptive rich media presentations via preference-based constrained optimization. In *Proceedings of the IJCAI-05 Workshop on Advances in Preference Handling*, pp. 19–24, Edinburgh, Scotland.

Brewka, G. (2002). Logic programming with ordered disjunction. In *Proceedings of Eighteenth National Conference on Artificial Intelligence*, pp. 100–105, Edmonton, Canada. AAAI Press.

Burke, R. (2000). Knowledge-based recommender systems. In Kent, A. (Ed.), *Encyclopedia of Library and Information Systems*, Vol. 69, pp. 180–200. Marcel Dekker, New York.

Domshlak, C. (2002). *Modeling and Reasoning about Preferences with CP-nets*. Ph.D. thesis, Ben-Gurion University, Israel.

Domshlak, C., & Brafman, R. (2002). CP-nets - reasoning and consistency testing. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 121–132, Toulouse, France.

Domshlak, C., Brafman, R., & Shimony, S. E. (2001). Preference-based configuration of web page content. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pp. 1451–1456, Seattle.

Domshlak, C., Rossi, F., Venable, K. B., & Walsh, T. (2003). Reasoning about soft constraints and conditional preferences: Complexity results and approximation techniques. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 215–220, Acapulco, Mexico.

Dyer, J. S. (1972). Interactive goal programming. *Management Science*, *19*, 62–70.

Even, S., Itai, A., & Shamir, A. (1976). On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, *5*, 691–703.

Faltings, B., Pu, P., Torrens, M., & Viappiani, P. (2004). Designing example-critiquing interaction. In *Proceedings of the International Conference on Intelligent User Interfaces*, pp. 22–29, Funchal, Madeira, Portugal.

Fargier, H., & Lang, J. (1993). Uncertainty in constraint satisfaction problems: A probabilistic approach. In *Proceedings of the European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty*, Vol. 747 of *LNCS*, pp. 97–104.

Fargier, H., Lang, J., & Schiex, T. (1993). Selecting preferred solutions in fuzzy constraint satisfaction problems. In *Proceedings of the First European Congress on Fuzzy and Intelligent Technologies*, pp. 1128–1134.

Fishburn, P. (1974). Lexicographic orders, utilities, and decision rules: A survey. *Management Science*, *20*(11), 1442–1471.

French, S. (1986). *Decision Theory*. Halsted Press, New York.

Freuder, E., & O'Sullivan, B. (2001). Generating tradeoffs for interactive constraint-based configuration. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pp. 590–594, Paphos, Cyprus.

Freuder, E. C., & Wallace, R. J. (1992). Partial constraint satisfaction. *Artificial Intelligence*, *58*, 21–70.

Freuder, E. C., Wallace, R. J., & Heffernan, R. (2003). Ordinal constraint satisfaction. In *Proceedings of the Fifth International Workshop on Soft Constraints*.

Glass, J. (1999). Challenges for spoken dialogue systems. In *Proceedings of the IEEE ASRU Workshop*, Keystone, CO.

Goldsmith, J., Lang, J., Truszczynski, M., & Wilson, N. (2005). The computational complexity of dominance and consistency in CP-nets. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 144–149, Edinburgh, Scotland.

Haag, A. (1998). Sales configuration in business processes. *IEEE Intelligent Systems and their Applications*, *13*(4), 78–85.

Hansson, S. O. (2001). Preference logic. In Gabbay, D. M., & Guenthner, F. (Eds.), *Handbook of Philosophical Logic* (2 edition)., Vol. 4, pp. 319–394. Kluwer.

Keeney, R. L., & Raiffa, H. (1976). *Decision with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley.

Lang, J. (2002). From preference representation to combinatorial vote. In *Proceedings of the Eight International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 277–288.

Linden, G., Hanks, S., & Lesh, N. (1997). Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of the Sixth International Conference on User Modeling*, pp. 67–78.

Pu, P., & Faltings, B. (2004). Decision tradeoff using example critiquing and constraint programming. *Constraints: An International Journal*, *9*(4), 289–310.

Resnick, P., & Varian, H. R. (Eds.). (1997). *Special Issue on Recommender Systems*, Vol. 40 of *Communications of the ACM*.

Rossi, F., Venable, K. B., & Walsh, T. (2004). mCP nets: Representing and reasoning with preferences of multiple agents. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 729–734, San Jose, CL.

Sabin, D., & Weigel, R. (1998). Product conguration frameworks - A survey. *IEEE Intelligent Systems and their Applications*, *13*(4), 42–49.

Schiex, T. (1992). Possibilistic cosntraint satisfaction, or "How to handle soft constraints". In *Proceedings of Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 269–275.

Schiex, T., Fargier, H., & Verfaillie, G. (1995). Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 631–637.

Wilson, N. (2004a). Consistency and constrained optimisation for conditional preferences. In *Proceedings of the Sixteenth European Conference on Artificial Intelligence*, pp. 888–894, Valencia.

Wilson, N. (2004b). Extending CP-nets with stronger conditional preference statements. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 735–741, San Jose, CL.