TCP-nets - Introducing Variable Importance Tradeoffs into CP-nets

Ronen I. Brafman BRAFMAN@CS.BGU.AC.IL
Department of Computer Science
Ben-Gurion University
Beer Sheva, Israel 84105

Carmel Domshlak DCARMEL@CS.CORNELL.EDU
Department of Computer Science
Cornell University
Ithaca, NY 14853

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

Abstract

The ability to make decisions and to assess potential courses of action is a corner-stone of many AI applications. Typically, this ability requires explicit information about the decision-maker's preferences. In many applications, preference elicitation is a serious bottleneck. The user either does not have the time, the knowledge, or the expert support required to specify complex multi-attribute utility functions. In such cases, a method that is based on intuitive, yet expressive, preference statements is required. In this paper we suggest the use of TCP-nets, an enhancement of CP-nets, as a tool for representing, and reasoning about qualitative preference statements. We present and motivate this framework, define its semantics, and study various computational aspects of reasoning with TCP-nets. Finally, we show how to perform constrained optimization efficiently given a TCP-net.

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. 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 and time-consuming and requires non-negligible effort on the part of the user. 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 analysts. This is the case, e.g., 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 generate it from natural and relatively simple statements about preferences obtained from the user, and this elicitation process should be amenable to automation. In addition, automated reasoning with this representation should be feasible and efficient.

One framework for preference representation that addresses these concerns is that of *Conditional Preference Networks* (CP-nets) (Boutilier et al. (1997, 1999, 2002)). In CP-nets, the decision maker is asked to describe 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 value of the main-course as well as whether or not she had an alcoholic beverage. Her choice of an alcoholic beverage depends 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 the edges represent 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 preferential statements, not captured by the CP-net model, that is no less intuitive or important. These statements have the following form: "It is more important to me that the value of X be high than that the value of Y be high." 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 independence. For instance, in the example above, my preference for an airline does not depend on the duration of the journey because, e.g., I compare airlines based on their service, security levels, and the quality of their frequent flyer program. Using statements of relative importance, the user expresses her preference over compromises that may be required. Such information is very important in customised 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 completely unknown to the customer.

In this paper we suggest 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). TCP-nets capture both information about conditional independence and conditional relative importance. Thus, they provide a richer framework for representing user preferences, allowing stronger conclusions to be drawn. Yet, they remain committed to the use of intuitive, qualitative information as their source. The added relative importance information has significant impact on both the con-

sistency of the specified relation, and the techniques used for reasoning about it. We show that the structure of the "mixed" set of preferential statements captured in a TCP-net can be exploited in order to achive 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, provide a number of examples, discuss their semantics and various modeling issues. In Section 4 We characterize a class of TCP-nets whose consistency is guaranteed and discuss the complexity of identifying members of this class. In Section 5 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 future work in Section 6.

2. PREFERENCE ORDERS, INDEPENDENCE, AND RELATIVE IMPORTANCE

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

2.1 Preference and Independence

A *preference relation* is a total pre-order (a *ranking*) over a set of outcomes. Given two outcomes o, o', we write $o \succeq o'$ to denote that o is at least as preferred as o'. Likewise, $o \succ o'$ denotes that o is *strictly* preferred over o'. 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 $\mathcal{D}(X_1) \times \cdots \times \mathcal{D}(X_n)$. 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 assignment to the set of variables specifies an outcome – a particular PC configuration. Thus, a preference ordering over these outcomes specifies a ranking over possible PC configurations.

The number of possible outcomes is exponential in n, while the set of possible total orders on them is doubly exponential in n. Therefore, explicit specification and representation of a ranking is not realistic. We must implicitly describe this preference relation. Often, 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. More formally, let $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}(\mathbf{X})$ for some $\mathbf{X} \subseteq \mathbf{V}$ (where we use $\mathcal{D}(\cdot)$ to denote the domain of a set of variables as well), and let $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{D}(\mathbf{Y})$, 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

$$\mathbf{x}_1\mathbf{y}_1 \succeq \mathbf{x}_2\mathbf{y}_1 \text{ iff } \mathbf{x}_1\mathbf{y}_2 \succeq \mathbf{x}_2\mathbf{y}_2$$

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 $\mathbf{Y}$. Formally, 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

$$\mathbf{x}_1\mathbf{y}_1\mathbf{z} \succeq \mathbf{x}_2\mathbf{y}_1\mathbf{z} \text{ iff } \mathbf{x}_1\mathbf{y}_2\mathbf{z} \succeq \mathbf{x}_2\mathbf{y}_2\mathbf{z}$$

Finally, $\mathbf{X}$ 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 Windows98 given an Intel processor (e.g., because he might believe that Windows98 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 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 $a_1b_1 \succ a_1b_2 \succ a_2b_1 \succ a_2b_2$.

Returning to our PC configuration example, suppose that *operating system* and *processor type* are independent attributes. 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.

Formally, let a pair of variables X and Y be 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:

$$x_iy_a\mathbf{w} \succ x_jy_b\mathbf{w}.$$

(Note that the above preference 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 refined by providing an actual ordering over elements of $\mathcal{D}(XY)$. We have decided not to pursue this option farther because in our opinion it is less natural to specify. However, our results generalize to such specifications as well. In addition, one can consider relative importance assessments among more than two variables. However, we feel that such statements are somewhat artificial and less natural to articulate.

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.

Formally, let $X, Y, \mathbf{W}$ be as above, and let $\mathbf{Z} \subseteq \mathbf{W}$. We say that X is *more important* than Y given an assignment $\mathbf{z} \in \mathcal{D}(\mathbf{Z})$ (*ceteris paribus*) iff, for every assignment $\mathbf{w}$ on $\mathbf{W} = \mathbf{V} - (\{X, Y\} \cup \mathbf{Z})$ we have:

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

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 that 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., 1999) that encodes conditional relative importance statements, as well as the conditional preferences available in CP-nets. We use this graph-based representation for two reasons: First, it is an intuitive visual representation of preference independence and relative importance statements. Second, the structure of the graph has important consequences on issues such as consistency and complexity of reasoning. In particular, as we later show, when this structure is "acyclic" (for a suitable definition of this notion!), the preference statements contained in the graph are guaranteed to be consistent – that is, there is a total pre-order that satisfies all the preference statements.

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) edge captures preferential

dependence, i.e., 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 some $\mathbf{Z}$ for which $\mathcal{RI}(X, Y, \mathbf{Z})$ holds.

As in CP-nets, each node X in a TCP-net is annotated with a *conditional preference table* (CPT). This table associates a 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 conditioning variables.

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

1. $\mathbf{V}$ is a set of nodes, corresponding to the problem variables $\{X_1, \ldots, X_n\}$.
2. cp is a set of directed cp-*arcs* $\{\alpha_1, \ldots, \alpha_k\}$ (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.
3. i is a set of directed i-*arcs* $\{\beta_1, \ldots, \beta_l\}$ (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* $\{\gamma_1, \ldots, \gamma_m\}$ (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]
5. cpt associates a CPT with every node $X \in \mathbf{V}$. A CPT is from $\mathcal{D}(Pa(X))$ (i.e., assignment's to X's parent nodes) to total pre-orders over $\mathcal{D}(X)$.
6. cit associates with every ci-arc (X_i, X_j) a subset $\mathbf{Z}$ of $\mathbf{V} - \{X_i, X_j\}$ and a mapping from a subset of $\mathcal{D}(\mathbf{Z})$ to total orders over the set $\{X_i, X_j\}$. We call $\mathbf{Z}$ the *selector set* of (X_i, X_j) and denote it by $\mathcal{S}(X_i, X_j)$.[2]

A CP-net (Boutilier et al., 1999) is simply a TCP-net in which the sets i and ci (and therefore cit) are empty.

3.2 TCP-net examples

This sub-section provides examples of TCP-net. For simplicity of presentation, in the following examples all variables are binary, although the semantics of TCP-net is defined 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 strictly prefer black to

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.
2. The set $\mathbf{Z}$ should be the minimal context upon which the relative importance between X_i and X_j depends.

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.

In turn, 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). $\diamond$

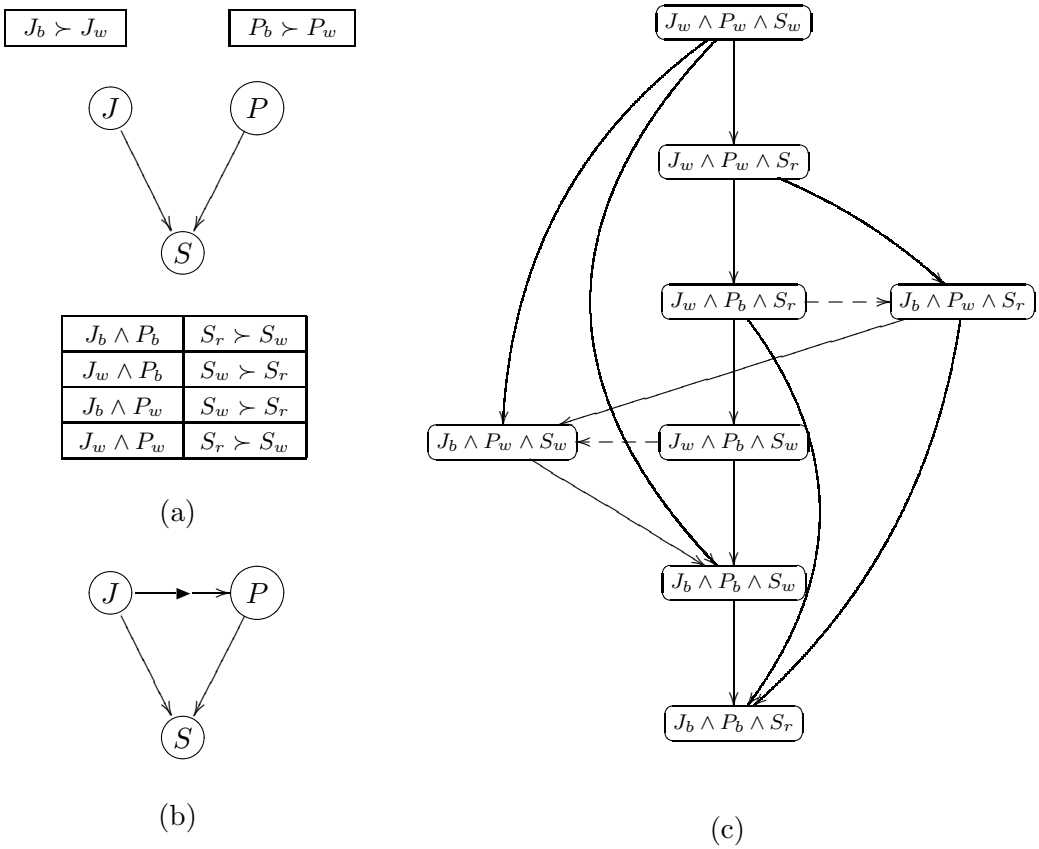

Figure 1: "Evening Dress" CP-net & TCP-net.

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 example, I may not have a clean black jacket, in which case the most prefered feasible alternative is a white jacket, black pants, and a white shirt. Or, 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 the most preferred, *available* evening dress for me.

As mentioned earlier, using statements of relative importance, the user expresses her preference over compromises that may be required. Such information is very important in customised 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 completely unknown to the customer. We return to the use of TCP-nets in this application area later in the paper.

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).

The CP-net in Figure 2(a) captures all these preferential 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, since 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:

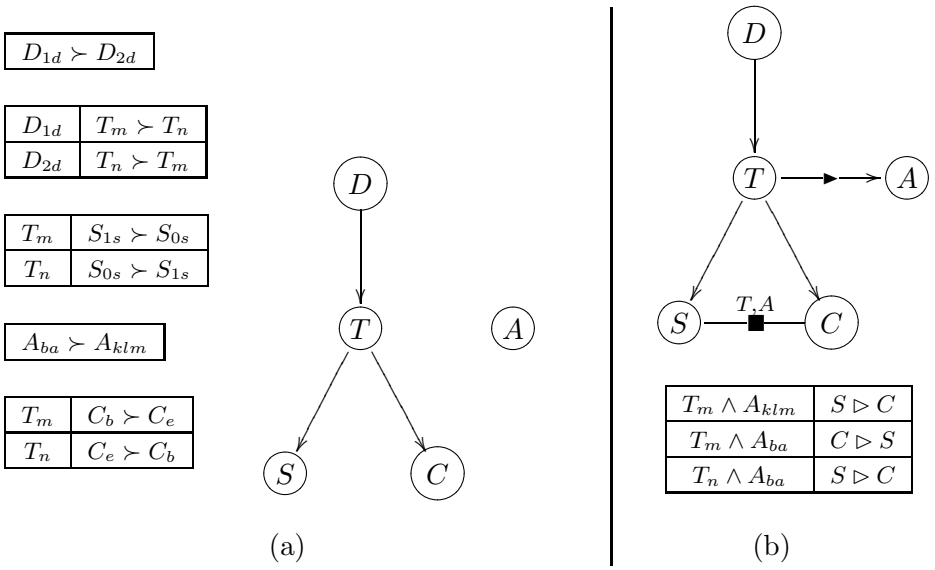

Figure 2: "Flight to the USA" CP-net & TCP-net from Example 2.

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 nignt 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). $\diamond$

3.3 TCP-net semantics

The semantics of a TCP-net is straightforward, and is defined in terms of the set of preference rankings consistent with the set of constraints imposed by its preference and importance information. We use $\succ_{\mathbf{u}}^{X}$ to denote the preference relation over the values of X given an assignment $\mathbf{u}$ to $\mathbf{U} \supseteq Pa(X)$.

Definition 1 Let $\mathcal{N}$ be a TCP-net over a set of variables $\mathbf{V}$.

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

2. A preference ranking $\succ$ satisfies the CPT of X iff it satisfies $\succ_{\mathbf{p}}^{X}$ for every assignment $\mathbf{p}$ of $Pa(X)$.

3. A preference ranking $\succ$ satisfies $X \rhd Y$ iff for every $\mathbf{w} \in \mathcal{D}(\mathbf{W})$ s.t. $\mathbf{W} = \mathbf{V} - \{X, Y\}$, $x_i y_a \mathbf{w} \succ x_j y_b \mathbf{w}$ whenever $x_i \succ^X_{\mathbf{w}} x_j$.

4. A preference ranking $\succ$ satisfies $X \rhd_{\mathbf{z}} Y$ iff for every $\mathbf{w} \in \mathcal{D}(\mathbf{W})$ s.t. $\mathbf{W} = \mathbf{V} - (\{X, Y\} \cup \mathbf{Z})$, $x_i y_a \mathbf{z}\mathbf{w} \succ x_j y_b \mathbf{z}\mathbf{w}$ whenever $x_i \succ^X_{\mathbf{z}\mathbf{w}} x_j$.

5. A preference ranking $\succ$ satisfies the CIT of the ci-arc (X, Y) 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 ranking $\succ$ satisfies a TCP-net $\mathcal{N}$ iff it satisfies:

1. every CPT in $\mathcal{N}$,

2. $X \rhd Y$ for every i-arc (X_i, X_j) in $\mathcal{N}$, and

3. every CIT in $\mathcal{N}$.

Definition 2 A TCP-net is *satisfiable* iff there is some ranking $\succ$ that satisfies it; $o \succ o'$ is *implied* by a TCP-net $\mathcal{N}$ iff it holds in all preference rankings 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 rankings satisfying N. As each of these rankings is transitive, we must have $o \succ o''$ in all satisfying rankings. ■

3.4 Discussion

Looking at Example 2, the reader may rightfully ask whether the notion of relative (conditional) importance *ceteris paribus*, as specified in Section 2.2, is not too strong when the variables are not binary. For example, consider a more refined notion of departure time (variable T), 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 great. For example, one may be willing to accept any airline among those she ranks in the top i places in this context. Such information can be maintained using a more refined CIT. In such a CIT, the relative importance of two variables X and Y will be conditioned on the actual values of X and Y, as well as on the values of the selector variables (as in ci-arcs), i.e., the preference ranking will be provided over elements of $\mathcal{D}(XY)$. As mentioned in Section 2.2, our results generalize to such specifications as well.

Depending on the application, a typical 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 web-page presentation (Domshlak, Brafman, & Shimony, 2001), the web-page designer chooses a set of content elements, which correspond to the set of variables. For an online shopper-assistent agent, the variables are likely to be fixed (e.g., if the agent is an online PC

customizer). 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.

Note that, in decision theory, the preference order reflects the preferences of a decision maker. The typical decision maker in preference-based product configuration is the consumer. However, in some domains, the product vendor may decide that customer preference elicitation is inappropriate, or simply impossible. In this case, the role of the decision maker may be relegated to a *product expert*, who is knowledgeable about appropriate component combinations for different customer classes. Following (Junker, 2001), consider the example of configuring a car. Suppose the expert indicates that:

(i) Older women typically prefer white cars to red cars, while the inverse is true for all other groups of customers, and

(ii) Young men typically value a fashionable look more than reliability, while the inverse is true for older men.

In this case, the product expert could specify a TCP-net as in Figure 3(a) over variables that stand for parameters both of the product and of the customer. The latter variables serve as *network parameters*, and are depicted by the shaded nodes in Figure 3(a). The network parameters are instantiated at the beginning of each session of personalized configuration once some personal information about the customer is obtained. Each such variable is a root variable in the TCP-net, as it does not participate in ci-arcs or in i-arcs, and it has only outgoing cp-arcs. However, it may serve as a selector variable in some ci-arcs of the network. No preference relation on its values is needed. After the network parameters are instantiated, the network is refined accordingly. For example, Figures 3(a), (b) and (c) depict the TCP-net from Figure 3(a), refined for young male, older male, and women customers, respectively.

An additional issue that, at least in some cases, can be addressed naturally using the TCP-net model, is *non-rigidness* of the variable set (Mittal & Falkenhainer, 1990; Sabin & Weigel, 1998; Soininen & Gelle, 1999; Veron & Aldanondo, 2000). As argued in (Sabin & Weigel, 1998), different components for the same functional role may need nonidentical sets of additional components, or functional roles. For example, if the a CD player is part of the car's configuration, one has the option of specifying preference for front/back speakers and an equilizer. However, if no audio component is included, the choice of speakers become irrelevant. In such cases, the set of variables in a solution changes dynamically on the basis of assignments of values to other variables – the related CSP formulation is known as a Dynamic CSP (Mittal & Falkenhainer, 1990). A DCSPs can be reduced to standard CSPs using a simple transformation, which can be applied in this case as well. We simply let one of the values in the domain of X stand for the *absence of* X. This way, the presence/absence of some variables may condition the presence of some other variables, their relative importance, and the preference on their values (and even their presence). Some of such conditionings may be entailed by the core configuration problem, and not by the user's preferences. In this case, they can be added into the TCP-net automatically, *after* the preference elicitation stage, by a straightforward extension of the specified TCP-net. This

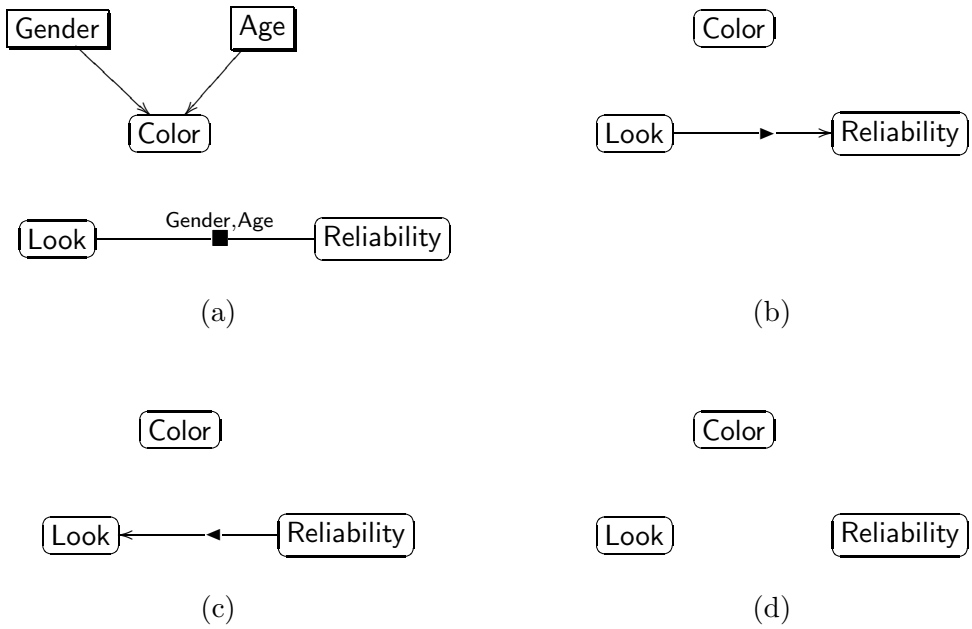

Figure 3: A parametrized TCP-net.

approach of automatically extending CP-nets was exploited in (Gudes, Domshlak, & Orlov, 2002) for preference-based presentation of tree-structured multimedia documents. In this domain, (i) the preferences on the value (= option of content appearance) of a document's component may be conditioned on the value of some other components, and (ii) whether it should be shown or hiden depends directly on which of its ancestors in the document hierarchical structure are shown.

To illustrate the above notion of activity values, consider the TCP-net graph in Figure 4(a), specifying preferences of a person looking for a sports utility car. This person prefers:

(i) convertibles (Convertible),

(ii) a sliding sun roof to a moon roof (Roof Window),

(iii) a multiple CD player to a single CD player (Audio System), and

(iv) a premium sound system to a standard one (Sound System).

Likewise, in a convertible, the customer values the quality of the sound system more than the robustness of the audio system.

A piece of information that is missing in the above description is that convertibles do not have sun or moon roofs. In general, the customer may be unaware of this fact. Therefore, once the preferences of the customer as above has been elicitated, the search/configuration engine of the dealership (possibly) adds a value `None` to the domain of the variable Roof Window, and sets this value to be the most preferable if Convertible = `Not` (see Figure 4(b)).

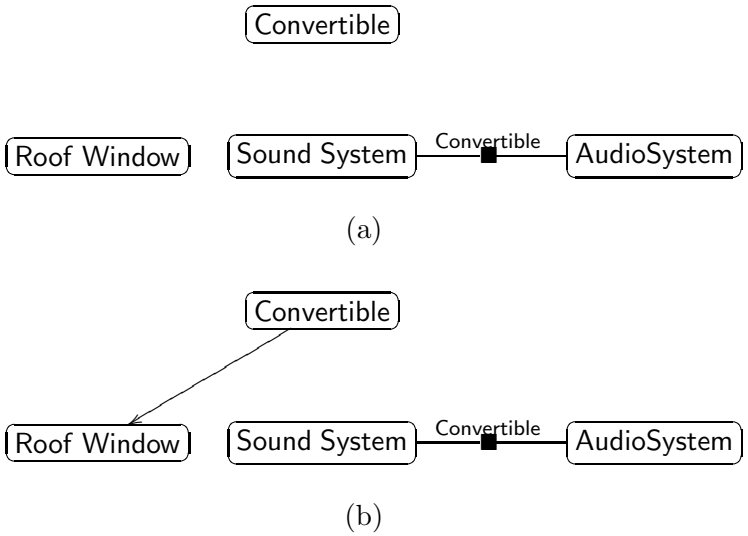

Figure 4: A TCP-net with activity values. (a) is the original TCP-net, and (b) is the modified one.

4. CONDITIONALLY ACYCLIC TCP-NETS

The definition of satisfiability in TCP-net (Definition 2) provides no practical tools for testing TCP-net satisfiability. A large class of TCP-nets, called *conditionally acyclic*, whose members are always satisfiable, is described in this section. Algorithms for testing TCP-nets for membership in this class, are developed.

First, let us define two types of directed graphs induced by all TCP-nets.

Definition 3 The *dependency graph* of TCP-net $\mathcal{N}$ contains all the nodes and directed edges of $\mathcal{N}$ (i.e., the cp-arcs and the i-arcs) as well as a pair of directed edges (X_k, X_i) and (X_k, X_j) for every ci-arc (X_i, X_j) in $\mathcal{N}$ and every $X_k \in \mathcal{S}(X_i, X_j)$.

Figure 5(b) depicts the dependency graph of the TCP-net from the "Flight to USA" example, repeated for convenience in Figure 5(a).

For the following definition, recall that the *selector set* of a ci-arc is the set of nodes whose value determines the "direction" of this arc.

Definition 4 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 $\mathcal{N}$ contains all nodes and directed edges of $\mathcal{N}$ and the edge from X_i to X_j if (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 5(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 $(n \wedge klm)$-directed graph of the TCP-net.

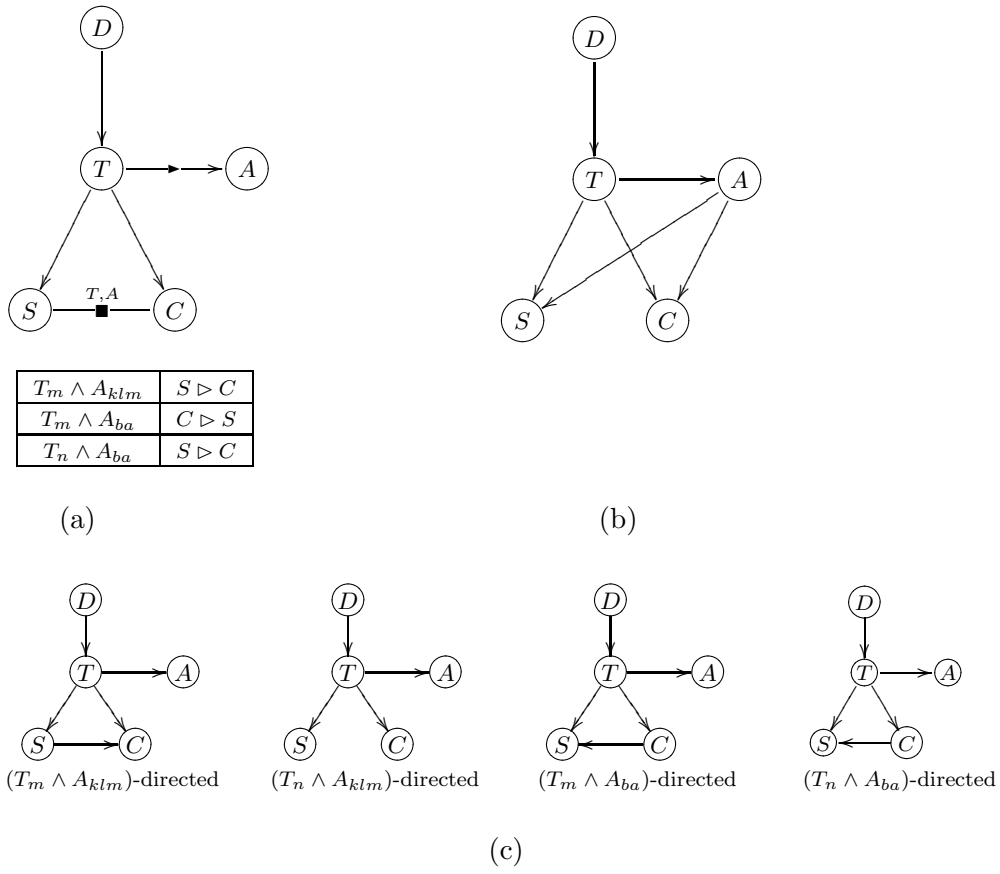

Figure 5: "Flight to USA" TCP-net (a), its dependency graph (b), and four **w**-directed graphs (c).

Now, using Definitions 3 and 4, we specify a relatively wide subclass of TCP-nets, and show that it is satisfiable.

Definition 5 A TCP-net $\mathcal{N}$ is *conditionally acyclic* if its induced dependency graph is acyclic and for every assignment **w** to $\mathcal{S}(\mathcal{N})$, the induced **w**-directed graphs are acyclic.

Theorem 1 Every conditionally acyclic TCP-net is satisfiable.

The constructive proof of Theorem 1 is given in Appendix A, p. 26. Note that, unlike acyclicity in directed graphs, the property of conditional acyclicity cannot be easily tested, in the general case. Despite that, we will argue that in practical cases, the problem is not hard.

4.1 Verifying conditional acyclicity

Verifying conditional acyclicity requires testing for two properties: Verifying the acyclicity of the dependency graph is simple, and can be done in time linear in the number of edges

in $\mathcal{N}$. However, naive verification of the acyclicity of every **w**-directed graph can require time exponential in the combined size of the selector sets. Here we study the complexity of verifying conditional acyclicity, discuss some hard and polynomial subclasses of this problem, and provide some sufficient and/or neccessary conditions for conditional acyclicity that can be easily checked for some subclasses of TCP-nets.

Let $\mathcal{N}$ be a TCP-net. If $\mathcal{N}$ contains directed cycles, then surely both the induced dependency graph and every **w**-directed graph is cyclic. Since such directed cycles are simple to detect, let us assume that they do not arise in $\mathcal{N}$. Next, note that if there are no cycles in the undirected graph induced by $\mathcal{N}$ (i.e., the graph obtained from $\mathcal{N}$ by removing the direction of its directed edges) then clearly all **w**-directed graphs are acyclic. This case is also quite simple to check. Finally, if there are cycles in the induced undirected graph, but each such cycle, when projected back to $\mathcal{N}$, contains directed arcs in different directions, then all **w**-directed graphs are still acyclic. This latter sufficient condition can be checked in (low order) polynomial time.

We are left with the situation where $\mathcal{N}$ contains sets $\mathcal{A}$ of edges that form a cycle in the induced undirected graph, not all of these edges are directed, yet all the directed edges point in the same direction (i.e., clockwise or counter-clockwise). We call these *semi-directed* cycles, and focus on their investigation in the rest of this section.

Each assignment **w** to the selector sets of ci-arcs in a semi-directed cycle $\mathcal{A}$ induces a direction for all these arcs. We say that cycle $\mathcal{A}$ is *conditionally acyclic* if under no such assignment **w** do we obtain a directed cycle from $\mathcal{A}$. Otherwise, $\mathcal{A}$ is called *conditionally directed*.

Showing that a TCP-net is conditionaly acyclic, i.e., that it does not contain any directed cycles, is hard in the general case.

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

For the proof of Theorem 2, see Appendix A, p. 27.

Thus, determining that a TCP-net is conditionally acyclic is Co-NP hard. In practice, this may not be too bad, as we do not expect to see users specify preference relations over more than a few dozen variables. However, we are interested in characterizing classes of conditionally acyclic networks that are easy to identify. First, we note that testing conditional acyclicity for TCP-nets is naturally decomposable.

Lemma 2 A TCP-net $\mathcal{N}$ is conditionally acyclic if and only if each one of its semi-directed cycles is conditionally acyclic.

Proof: The proof is straightforward: If there is a variable assignment that makes one of the semi-directed cycles of N conditionally directed, then no other cycle need be examined. Conversely, consider one of the semi-directed cycles $\mathcal{A}$ of $\mathcal{N}$. If no assignment to $\mathcal{S}(\mathcal{A})$ makes $\mathcal{A}$ conditionally directed, then additional assigments to variables in other selector sets do not change this property. ■

Given this property, it is natural to consider networks in which the number of cycles is not too large. Indeed, one reason for the complexity of the general problem, as emerges from

the proof of Theorem 2, is the possibility that the number of simple cycles is exponential in the size of the TCP-net. A TCP-net $\mathcal{N}$ where the number of different simple (possibly mixed) cycles is at most m is called *m-cycle bounded*.

From Lemma 2 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 $\mathcal{N}$ to testing conditional acyclicity of every semi-directed cycle $\mathcal{A}$ of $\mathcal{N}$. 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 $\mathcal{A}$ in the network, then conditional acyclicity can be checked for quickly. In fact, often we can determine that a semi-directed cycle is conditionally directed/acyclic even more efficiently.

Lemma 3 Let $\mathcal{A}$ be a semi-directed cycle in $\mathcal{N}$. 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 3 provides a necessary condition for conditional acyclicity of $\mathcal{A}$ that can be checked in time polynomial in the number of variables.

Proof (Lemma 3) If all selector sets, $\mathcal{S}(\gamma_i)$, are pairwise disjoint, then we can find an assignment on $\mathcal{S}(\mathcal{A})$ that will orient all ci-arcs in one direction.
■

Before we develope 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 $CIT(\gamma)$ 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 this (obviously unique) direction as the *direction of* $\mathcal{A}$.

Lemma 4 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 $\mathcal{S}(\gamma_i) \cap \mathcal{S}(\gamma_j)$, 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. ■

Lemma 4 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 this large 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 6 Given a semi-directed cycle $\mathcal{A}$, we let $shared(\mathcal{A})$ denote the union of all pairwise intersections of the selector sets of the ci-arcs in $\mathcal{A}$:

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

Lemma 5 (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 $shared(\mathcal{A})$, 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 $shared(\mathcal{A})$, 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 4. Thus, we are left with proving necessity. We start with the second case in which $\mathcal{A}$ contains only ci-arcs, assuming to the contrary that $\mathcal{A}$ is conditionally acyclic, but there exist an assignment $\mathbf{u}$ on $shared(\mathcal{A})$ 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 $\mathcal{S}^*(\gamma) = \mathcal{S}(\gamma) \setminus shared(\mathcal{A})$. 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 induces 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 $shared(\mathcal{A})$, 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. ■

In general, the size of $shared(\mathcal{A})$ is $O(|\mathbf{V}|)$. Since we have to check the set of assignments over $shared(\mathcal{A})$, 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 $\mathcal{N}$, is NP-complete, even if for every ci-arc $\gamma \in \mathcal{N}$ we have $|\mathcal{S}(\gamma)| \leq 3$.

The proof of Theorem 3 by reduction from 3-SAT appears in Appendix A, p. 28. This proof does not work when the size of all the selector sets is bounded by 2, since 2-SAT is in P. The question is whether in this latter case the problem becomes tractable, and 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 $\mathcal{N}$, is in P.

The proof of Theorem 4 by reduction to 2-SAT appears in Appendix A, p. A. To summarize the above analysis, one must first identify the semi-directed cycles in the network – a network without semi-directed cycles is consistent. Next, one must show that given each assignment $\mathbf{w}$ to the context variables of each semi-directed cycle, the $\mathbf{w}$-directed graph is acyclic. This problem is Co-NP hard in general networks,[3] 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 $shared(\mathcal{A})$ 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 $C = c$" appear to be more complex than what one would expect to hear. Thus, Lemma 4, Lemma 5 (for semi-directed cycles with small $shared(\mathcal{A})$), 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.

5. REASONING ABOUT CONDITIONALLY ACYCLIC TCP-NETS

While automated consistency verification is the core part of the preference elicitation stage, efficient reasoning over 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.

5.1 Generating Optimal Assignments

One of the central properties of the original CP-net model (Boutilier et al., 1999) is that, given an acyclic CP-net $\mathcal{N}$ and a (possibly empty) partial assignment π on its variables, it is simple to determine an outcome consistent with $\mathbf{z}$ (in $Comp(\mathbf{z})$) 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

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

Search-TCP ($\mathcal{N}$, $\mathcal{C}$, $\mathcal{K}$)
Input: Conditionally acyclic TCP-net $\mathcal{N}$, Constraints $\mathcal{C}$,
Partial assignment $\mathcal{K}$ on the variables of the original TCP-net.
Output: Set of all, Pareto-optimal w.r.t. $\mathcal{N}$, solutions for $\mathcal{C}$.

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 $\mathcal{N}$.
Let $x_1 \succ \ldots \succ x_k$ be the preference ordering of $\mathcal{D}(X)$
given the assignment on $Pa(X)$ in $\mathcal{K}$.
Initialize the set of local results by $\mathcal{R} = \emptyset$
for $(i = 1;\ i \leq k;\ i++)$ **do**
$X = x_i$
Strengthen the constraints $\mathcal{C}$ by $X = x_i$ to obtain $\mathcal{C}_i$
if $\mathcal{C}_j \subseteq \mathcal{C}_i$ for some $j < i$ **or** $\mathcal{C}_i$ is inconsistent **then**
continue with the next iteration
else
Let $\mathcal{K}'$ be the partial assignment induced by $X = x_i$ and $\mathcal{C}_i$
$\mathcal{N}_i$ = Reduce ($\mathcal{N}$,$\mathcal{K}'$)
Let $\mathcal{N}_i^1, \ldots, \mathcal{N}_i^m$ be the components of $\mathcal{N}_i$, connected
either by the edges of $\mathcal{N}_i$ or by the constraints $\mathcal{C}_i$.
for $(j = 1;\ j \leq m;\ j++)$ **do**
$\mathcal{R}_i^j$ = Search-TCP($\mathcal{N}_i^j, \mathcal{C}_i, \mathcal{K} \cup \mathcal{K}'$)
if $\mathcal{R}_i^j \neq \emptyset$ for all $j \leq m$ **then**
foreach $o \in \mathcal{K}' \times \mathcal{R}_i^1 \times \cdots \times \mathcal{R}_i^m$ **do**
if $\mathcal{K} \cup o' \not\succ \mathcal{K} \cup o$ holds for each $o' \in \mathcal{R}$ **then** add o to $\mathcal{R}$
return $\mathcal{R}$

Figure 6: The Search-TCP algorithm for conditionally acyclic TCP-net based constrained optimization.

the network is traversed according to a topological order induced by the CP-net part of the given TCP-net.

Corollary 1 The forward sweep procedure constructs the most preferred outcome in $Comp(\mathbf{z})$ with respect to a given conditionally acyclic TCP-net.

This strong 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 mutually constrained by a set of hard constraints, $\mathcal{C}$. In this case, determining the set of Pareto-optimal[4] feasible outcomes is not trivial. For the acyclic CP-nets, a branch and bound

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

algorithm for determining the optimal feasible outcomes was introduced in (Boutilier et al., 1997). 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* assignment generated that satisfies the set of constraints is also Pareto-optimal. Thus, in some sense, the optimization problem is not harder than the underlying CSP!

Here we develop an extension/modification of the algorithm of (Boutilier et al., 1997) to conditionally acyclic TCP-nets. The extended algorithm Search-TCP retains the anytime property and it is shown in Figure 6. 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 preceeds its descendents) (Boutilier et al., 2002), 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 and is consistent with the dependency graph of $\mathcal{N}$.

Reduce ($\mathcal{N}$, $\mathcal{K}'$)
foreach $\{X = x_i\} \in \mathcal{K}'$ **do**
  foreach cp-arc $\langle \overrightarrow{X, Y} \rangle \in \mathcal{N}$ **do**
    Restrict the CPT of Y to the rows dictated by $X = x_i$.
  foreach ci-arc $\gamma = (Y_1, Y_2) \in \mathcal{N}$ s.t. $X \in \mathcal{S}(\gamma)$ **do**
    Restrict the CIT of γ to the rows dictated by $X = x_i$.
    if, given the restricted CIT of γ, relative importance
      between Y_1 and Y_2 is independent of $\mathcal{S}(\gamma)$, **then**
        if CIT of γ is not empty **then**
          Replace γ by the corresponding i-arc.
        else Remove γ.
  Remove from $\mathcal{N}$ all the edges involving X.
return $\mathcal{N}$.

Figure 7: The Reduce procedure.

The Search-TCP algorithm is guided by the underlying TCP-net $\mathcal{N}$. It proceeds by assigning values to the variables in a top-down manner, assuring that outcomes are generated according to the preferential ordering induced by $\mathcal{N}$ – on a call to the Search-TCP procedure with a TCP-net $\mathcal{N}$, the eliminated variable X is one of the root variables of $\mathcal{N}$. The values of X are considered according to the preferential ordering induced by the assignment on $Pa(X)$. Note that X is observed in some context $\mathcal{K}$ which necessarily contains some assignment on $Pa(X)$. Any additional variables assignment $X = x_i$ converts the current set of constraints $\mathcal{C}$ into a stronger constraint set $\mathcal{C}_i$. As a result of this propagation of $X = x_i$, values for some variables (at least for the variable X) are fixed automatically, and this partial assignment $\mathcal{K}'$ extends the current context $\mathcal{K}$ in processing of the next variable. The Reduce procedure, presented in Figure 7, 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 dependency graphs of the networks $\mathcal{N}_i$ and $\mathcal{N}_j$, resulted by propagating $\mathcal{C}_i$ and $\mathcal{C}_j$, respectively, may *disagree* on the ordering of some variables.

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 constraints, then each connected component invokes an independent search. 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 $\mathcal{C}_i$, some pruning takes place in the search tree (see the **continue** instruction[5] in the algorithm): 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 Pareto-optimal solutions for the original set of constraints. Therefore, the search is depth-first branch-and-bound, where the set of nondominated solutions generated so far is a proper subset of the required set of the Pareto-optimal solutions for the problem, and thus it corresponds to the current lower bound.

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

Note that, if we are interested in getting *one* Pareto-optimal solution for the given set of constraints (which is usually the case), we can output the *first* feasible outcome generated by Search-TCP. No dominance queries between pairs of outcomes are required because there is nothing to compare with the first generated solution. If we are interested in getting *all*, or even *a few* Pareto-optimal solutions, then the efficiency of dominance queries becomes an important factor in the entire complexity of the Search-TCP algorithm.

5.2 Dominance testing for TCP-nets

A fundamental query we may pose to any preference-representation formalism is whether some outcome o dominates (i.e., is strictly preferred to) some other outcome o'. As discussed above, dominance queries are required whenever we wish to generate more than one Pareto-optimal outcome. 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 sanctioned by

5. This pruning was introduced in (Boutilier et al., 1997) for acyclic CP-nets, and it remains valid for conditionally acyclic TCP-nets.

the given TCP-net. Formally, an improving flipping sequence in the context of TCP-nets can be defined as follows:

Definition 7 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}.

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., 1999).

Theorem 5 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.

The proof of Theorem 5 is given in Appendix A, p. 29. 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 in (Domshlak & Brafman, 2002; Domshlak, 2002; Boutilier et al., 2002) can be applied to the TCP-net model – an issue left for future research.

6. DISCUSSION

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 a TCP-net. 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 is motivated by the need for preference elicitation, representation, and reasoning techniques in diverse areas of AI. One of the main application areas we have in mind is product configuration. Keeping this application area in mind, we now review some related approaches to preference-based optimiziation 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 overconstrained (and thus unsolvable in the standard meaning of the 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 an original constraint that now 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, H.Fargier, Montanari, Rossi, Schiex, & Verfaillie, 1999), fuzzy (Schiex, 1992), probabilistic (Fargier & Lang, 1993), and lexicographic (Fargier, Lang, & Schiex, 1993) CSPs, are discussed in the literature on constraint satisfaction.

The conceptual difference between our approach and the soft-constraints formalisms 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. In customized 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 the most preferred, feasible product configuration. This naturally leads to a decoupled approach.

Freuder and O'Sullivan (Freuder & O'Sullivan, 2001) proposed a framework of interactive sessions for overconstrained 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 & 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.

The tradeoffs exploited in (Freuder & 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 in (Freuder & O'Sullivan, 2001), in the TCP-net based constrained optimization presented in Section 5, we exploit this information to guide the constraint solver to the preferable feasible solutions. On the other hand, the motivation and the ideas behind the work of Freuder and O'Sullivan as well as these in some other works on interactive search (see work, e.g., on interactive goal programming (Dyer, 1972)) 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.

Our work opens several directions for future theoretical and applied 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 was analyzed in (Domshlak, 2002; Boutilier

et al., 2002). 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 appears in (Domshlak & Brafman, 2002).

An interesting practical question related to elicitation of qualitative preferences is model acquisition from speech/text (Asher & Morreau, 1995; James, 1999). 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 preferential statements seems to form a linguistic domain that is apriori 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 (Boutilier et al., 1997).

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.

Bistarelli, S., H.Fargier, 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.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2002). CP-nets: A tool for representing and reasoning about conditional *ceteris paribus* preference statements. *submitted for publication.*

Boutilier, C., Brafman, R., Geib, C., & Poole, D. (1997). A constraint-based approach to preference elicitation and decision making. In *AAAI Spring Symposium on Qualitative Decision Theory*, Stanford.

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.

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

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.

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.

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.

Freuder, E., & O'Sullivan, B. (2001). Modeling and generating tradeoffs for constraint-based configuration. In *Proceedings of 4th Workshop on Configuration (IJCAI-01)*, pp. 38–44, Seattle, US.

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

Gudes, E., Domshlak, C., & Orlov, N. (2002). Remote conferencing with multimedia objects. In *Proceedings of the Second International Workshop on Multimedia Data Document Engineering (MDDE'02)*.

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

James, G. (1999). Challenges for spoken dialogue systems. In *Proceedings of the IEEE ASRU Workshop*.

Junker, U. (2001). Preference programming for configuration. In *Proceedings of 4th Workshop on Configuration (IJCAI-01)*, pp. 50–56, Seattle, US.

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

Mittal, S., & Falkenhainer, B. (1990). Dynamic constraint satisfaction problem. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 25–32. AAAI Press.

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.

Soininen, T., & Gelle, E. M. (1999). Dynamic constraint satisfaction in conguration. In *Proceedings of AAAI Workshop on Conguration*.

Veron, M., & Aldanondo, M. (2000). Yet another approach to CCSP for configuration problem. In *Proceedings of 3rd Workshop on Configuration (ECAI-00)*, Berlin, Germany.

Appendix A. Proofs

Theorem 1 Every conditionally acyclic TCP-net is satisfiable.

Proof: We prove this constructively by building a satisfying preference ordering. The proof is by induction on the number of problem variables, and the result trivially holds for one variable, as the total ordering is specified by the TCP-net.

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. Since $\mathcal{N}$ is conditionally acyclic, there is at least one "root" variable X, such that there is:

1. no cp-arc $\langle \overrightarrow{Y, X} \rangle$, and
2. no i-arc $(\overrightarrow{Y, X})$, and
3. no ci-arc (X, Y).

Note that every root node in the dependency graph has this property.

Let $\mathcal{D}(X) = \{x_1, \ldots, x_k\}$ be the domain of the variable X, and let $x_1 \prec \ldots \prec x_k$ be the preferential ordering over $\mathcal{D}(X)$ dictated by $\mathcal{N}$. For each x_i, $1 \leq i \leq k$, 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 $(\overrightarrow{X, Y})$.

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 $1 \leq i < j \leq k$. 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). It is easy to see that this preference ordering satisfies $\mathcal{N}$. ■

Theorem 2 Given a binary-valued, TCP-net $\mathcal{N}$, the decision problem: is there a conditionally directed cycle in $\mathcal{N}$, is NP-hard, 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 $(\overrightarrow{L_{j,k}, C_{(j+1) \bmod n}})$, 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 (i.e., X_i is not a negated 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 8 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.

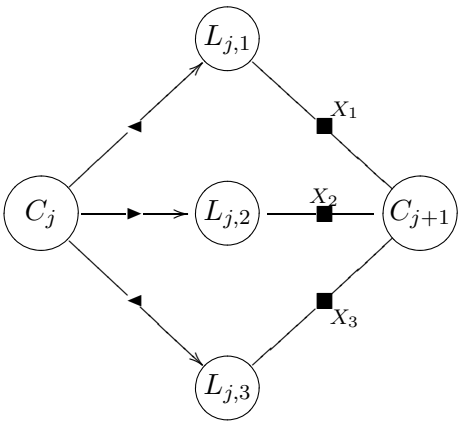

Figure 8: TCP-net subnet for $C_j = (x_1 \vee x_2 \vee \overline{x_3})$.

We claim that $\mathcal{N}$ has a conditionally directed cycle just when $\mathcal{F}$ is satisfiable. 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 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: we need a semi-directed cycle $\mathcal{A}$ and an assigment on $\mathcal{S}(\mathcal{A})$ (the union of the selector sets of all ci-arcs in $\mathcal{A}$) which can be checked in polynomial time. Thus, the problem is NP-complete. ■

Theorem 3 Given a binary-valued, 1-cycle bounded TCP-net $\mathcal{N}$, the decision problem: is there a conditionally directed cycle in N, 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 (we retain the same names, for simplicity). In addition, add a single dummy variable C, and an i-arc $(\overrightarrow{C, C_1})$. Let n be the number of clauses in $\mathcal{F}$. Then, there are also $n-1$ ci-arcs $E_j = (C_j, C_{j+1})$, one for each clause, except the last, for which we have ci-arc (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}).

We claim that a conditionally directed cycle in $\mathcal{N}$ exists 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. ■

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 $\mathcal{N}$, is in P.

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

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

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 generate two different $\mathcal{A}'$ and $\mathcal{A}''$, by inserting one dummy variable and one i-arc to $\mathcal{A}$ – clockwise in $\mathcal{A}'$, counter-clockwise in $\mathcal{A}''$. Now, $\mathcal{A}$ is conditionally directed if and only if either $\mathcal{A}'$ or $\mathcal{A}''$ (or both) are conditionally directed. ■

Theorem 5 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 $\mathcal{F}$ from o' to o with respect to $\mathcal{N}$, by Definition 7, $\mathcal{N} \models o_2 \succ o_1$ for any flip from o_1 to o_2 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 2), 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 $\mathcal{N} \not\models o \succ o'$. ■