Preference-based Constrained Optimization with CP-nets

Craig Boutilier
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3H8, Canada
cebly@cs.toronto.edu

Ronen I. Brafman
Department of Computer Science
Ben-Gurion University
Beer Sheva, Israel 84105 brafman@cs.bgu.ac.il

Carmel Domshlak
Department of Computer Science
Cornell University
Ithaca, NY 14853
dcarmel@cs.cornell.edu

Holger H. Hoos
Department of Computer Science
University of British Columbia
Vancouver, BC, V6T 1Z4, Canada
hoos@cs.ubc.ca

David Poole
Department of Computer Science
University of British Columbia
Vancouver, BC, V6T 1Z4, Canada
poole@cs.ubc.ca

December 18, 2003

Abstract

Many AI tasks, such as product configuration, decision support, and the construction of autonomous agents, involve a process of constrained optimization, that is, optimization of behavior or choices subject to given constraints. In this paper we present an approach for constrained optimization based on a set of hard constraints and a preference ordering represented using a CP-network—a graphical model for representing qualitative preference information. This approach offers both pragmatic and computational advantages. First, it provides a convenient and intuitive tool for specifying the problem, and in particular, the decision maker's preferences. Second, it admits an algorithm for finding the most preferred feasible (Pareto optimal) outcomes that has the following anytime property: the set of preferred feasible outcomes are enumerated without backtracking. In particular, the first feasible solution generated by this algorithm is Pareto optimal.

1 Introduction

The problem of finding a satisfying assignment of values to a set of variables given a collection of constraints is a well-studied problem in AI and other fields, and has a wide variety of practical applications. In some of these domains, a user may be satisfied with any solution satisfying the set of constraints. In others, however, the user has preferences over the set of solutions, and would like to obtain the *best* satisfying assignment—or at least a good assignment—with respect to these preferences. This latter problem, that of finding a solution to a constraint satisfaction problem that is optimal with respect to a user's preferences, is often referred to as *constrained optimization*.

How one approaches constrained optimization depends critically on the representation of preferences one adopts. If user preferences are captured using some quantitative objective function defined over problem variables, standard constraint-based optimization techniques can be used (e.g., standard linear- or integer-programming packages). Alternatively, depending on the nature of the constraints (e.g., their logical form), methods derived from constraint satisfaction algorithms can be applied. For example, some soft constraint representation (Bistarelli, Montanari, & Rossi, 1997; Bistarelli, Fargier, Montanari, Rossi, Schiex, & Verfaillie, 1999; Schiex, 1992; Fargier, Lang, & Schiex, 1993; Freuder & Wallace, 1992; Freuder, 1989) could be used to represent both feasibility constraints and preferences.

Unfortunately, in many situations, quantitative utility information capturing a user's preferences may be very difficult to obtain; decision analysts have recognized for decades that people have a hard time accurately assessing their preferences quantitatively (French, 1986). Fortunately, users are often much more comfortable assessing their preferences qualitatively, by *ordering* or *ranking* full outcomes, or instantiations of subsets of variables. Furthermore, unless uncertainty exists in the outcomes of decisions, quantitative utility information provides no benefit over simple ordering information.

In this paper, we consider the problem of constrained optimization with qualitative preference information. Specifically, we adopt *CP-networks* (Boutilier, Brafman, Hoos, & Poole, 1999; Boutilier, Brafman, Domshlak, Hoos, & Poole, 2003) to represent a user's qualitative preferences over variable assignments, and address the problem of optimization using this representation. Apart from providing a natural, concise, graphical representation of preferences, we show that CP-nets support effective techniques for constrained optimization. In particular, we derive a branch-and-bound algorithm for producing the set of Pareto optimal solutions satisfying a collection of constraints that exploits the CP-net structure for computational advantage. This algorithm also has an important anytime property: the collection of solutions it produces is guaranteed to contain only Pareto-optimal solutions at any point in time (i.e., no solutions will be retracted as new solutions are discovered).

1.1 Motivation for the Approach

We begin with some motivation for the general approach we adopt. We concentrate on constrained optimization problems, defined over a finite set of variables with finite domain size. Such problems are exemplified by configuration tasks, decision support problems and the construction of autonomous agents. As an example, consider the problem of adapting rich-media messages to specific user devices.[1] Rich-media messages contain multiple elements of distinct types, such as audio, video, text, image, and animation. Each element can be displayed in multiple ways. For example, a video segment can be displayed at different rates, using different color resolutions, and taking up different portions of the viewing area. Typically, the message author has a single most preferred manner for presenting her message. But this presentation may not be feasible on all user devices. For example, a PDA with a black-and-white screen cannot display color. Similarly, buffer size and screen size limit the types of elements that can be displayed simultaneously. To overcome this problem, the message transcoder needs to adapt the message presentation to the particular capabilities of the user device by finding a feasible presentation of the message. On the other hand, the

[1]This application is being developed by the STRIMM Consortium using the language and tools presented in this paper (Brafman & Friedman, 2003).

message author, who does not know at composition time on what devices her message will be displayed, needs to provide information that will help the transcoder select the best feasible presentation.

To help this process, we would like to provide the message author with a language for annotating her message. Using this language the author could rank different presentation options for various message elements. For example, she may specify that she prefers that a particular video segment be displayed in color rather than black and white. In addition, the author may prefer that color be displayed using a smaller image size (because of resolution problems) and black and white be displayed using a larger image size. When the message recipient requests the message, his device submits a description of device capabilities to the transcoder, which attempts to select among the feasible presentations that which is most preferred by the author. For example, given a user with a black-and-white PDA, the video is constrained to be black and white, and a large image size is selected because of the author's preferences.[2]

Problems of this type can be viewed abstractly as follows: we assume a finite set of variables, each with a finite domain (e.g., display color of a particular message component, display size, etc.). Various constraints are imposed on the value of these variables, such as the fact that video cannot be displayed in color on a black-and-white device, that the total display size of message components cannot exceed the size of the device display, and so on. Finally, some preference relation on the possible assignments is provided, as alluded to above. Our goal is to find assignments to the set of variables that satisfy the constraints and are optimal with respect to our preferences.

Although the specification of a constraint satisfaction problem (CSP) is quite standard, the specification of a preference relation on possible assignments leaves more room for choice. In particular, there are two aspects of the specification that we need to consider. First, we have to choose between quantitative and qualitative preference representations. Second, we have to choose between specifications that are *coupled*, where the same language is used for preferences and constraints, and *decoupled*, where preferences and constraints are treated as different sorts. We argue that for many applications, and in particular, customized product configuration problems, a qualitative, decoupled approach is the most appealing.

Let us consider the choice between *quantitative* and *qualitative* represen-

[2]This is just an illustrative example. In practice, there are more options, e.g., allowing for different color resolutions. In addition, the user can choose presentation templates that have built-in default preferences for different message aspects.

tations. In classical decision theory and decision analysis a *utility function* is used to represent the decision maker's preferences. Utility functions are a powerful form of knowledge representation. They provide a quantitative measure of the desirability of different outcomes, capture attitude toward risk, and support decision making under uncertainty. However, the process of obtaining the type of information needed to generate a good utility function is time consuming and requires considerable effort on the part of the user. In some applications, this effort may be worthwhile, for instance, when uncertainty plays a key role, and when the decision maker and a decision analyst are able and willing to engage in the required preference elicitation process. One would expect to see such effort invested when, say, important medical or business decisions are involved. However, there are many applications where: (a) uncertainty is not a crucial factor; (b) the user cannot be engaged in preference elicitation for a lengthy period of time (e.g., in an on-line product recommendation systems); or (c) the preference elicitation process cannot be supported by a human decision analyst and must be performed by a software system (e.g., due to replicability or mass marketing aims). In such cases, elicitation of a quantitative utility function of suitable accuracy and precision is not a realistic option.

When a utility function is unavailable or unnecessary, one can 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. For example, in an on-line product configuration or recommendation service, a recommendation should be generated quickly.[3]

Next, let us consider the question of whether the specification of constraints and preferences should be *coupled* or *decoupled*. Certainly, in many domains the specification of what people want is of a different sort than the specification of what is possible. There is no reason why these should be conflated, or specified using the same representation. One needs to seek

[3]An alternative that lies somewhere between quantitative and qualitative preference representations is the use of *imprecisely specified* utility functions. In such work, bounds (Wang & Boutilier, 2003; Boutilier, Patrascu, Poupart, & Schuurmans, 2003), constraints (White, III, Sage, & Dozono, 1984; Blythe, 2002), or distributions (Chajewska, Koller, & Parr, 2000; Boutilier, 2002) are placed on the (numerical) parameters of the underlying utility function. These models still often require users to assess quantitative information (such as tradeoff weights), though in a less precise fashion than full utility elicitation.

the best representations for preferences and the best representations for hard constraints; there is no a priori reason to assume that these should be similar, or to insist that they be the same. Furthermore, the decoupled perspective is often very natural: those most familiar with the constraints underlying a particular decision problem are different from the person most familiar with the preferences. This is especially true when one considers the use of systems that make or recommend decisions on behalf of multiple users—the constraints are known by (i.e., encoded within) the system, while the preferences vary from user to user. Consider, for example, customized *product configuration* (Haag, 1998; Sabin & Weigel, 1998), embodied, say, as a web site that assembles components for some type of system. In this class of problems, we typically have two parties involved, the manufacturer and the consumer. The manufacturer brings product expertise, and with it a set of hard constraints on possible system configurations. We expect that the manufacturer has knowledge of feasibility constraints regarding which parts fit together, and what functionality results from various combinations of parts. The builders of the web site, however, can't anticipate the specific preferences of individual users. The individual users, in contrast, do know their own preferences, but may be unaware of the hard constraints that define the set of feasible system configurations. In this situation there is a natural decoupling of the preferences and constraints. However, when choosing a system, both the constraints and the preferences need to be taken into account—by finding the most preferred, feasible product configuration—to provide the user with maximal satisfaction.

A primary example of a coupled representation is the *soft constraints* formalism (see, e.g., Bistarelli *et al.* (1997, 1999)), developed to model constraint satisfaction problems that are either overconstrained, and thus "unsolvable" (Freuder & Wallace, 1992), or that suffer from imprecise knowledge about the actual constraints (Fargier & Lang, 1993). In the soft constraints formalism, the constrained optimization problem is presented in the form of a set of preference orderings over assignments to subsets of variables, together with some operator for combining these local preference relations to form a preference relation over the assignments to the complete set of variables. Each such subset of variables corresponds to an original constraint that now can be satisfied to a different extent by different variable assignments. There is much flexibility in how such local preferences are specified and 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 constraint satisfaction.

Some soft-constraint models have corresponding notions in standard decision-theoretic preference representations. For example, weighted CSPs—in which weight functions are associated with subsets of variables, and the aggregation is done via summation—correspond to the notion of additively decomposable utility functions (Keeney & Raiffa, 1976; Bertelè & Brioschi, 1972; Dechter, Dechter, & Pearl, 1988; Jensen, Jensen, & Dittmer, 1994), while fuzzy CSPs are grounded in the framework of possibilistic logic (Dubois & Prade, 1988). Unfortunately, while being very expressive, these quantitative models of soft constraints suffer from the elicitation disadvantages we noted above. Purely qualitative soft-constraint models that have been studied usually take the form of (unweighted) Max-CSP, that is, the CSP variant in which the goal is to satisfy as many constraints as possible. However, the expressivity of Max-CSP (and thus its expected practical usefulness) is very limited. Finally, we note that some recent work attempts to combine the capabilities of soft-constraints with the decoupled approach pursued in this paper (Domshlak, Rossi, Venable, & Walsh, 2003).

1.2 Overview

The main contribution of this paper is a qualitative, decoupled specification method for constrained optimization problems by means of an intuitive qualitative preference specification together with an efficient optimization algorithm. Our preference specification technique is based on *CP-networks* (Boutilier et al., 1999, 2003), a qualitative graphical formalism for specifying preference orders. CP-nets capture sets of *conditional ceteris paribus (CP)* preference statements, such as *"I prefer that a large display size be used if video clip 17 is played in black and white and no other images are displayed."* Under the *ceteris paribus* semantics ("all else being equal"), this statement asserts that given two message displays that differ *only* in the display size of clip 17, *and* both containing no other image and a black and white display of the clip, we prefer the message in which clip 17 is given more screen size. As argued by many philosophers (see Hansson (1996, 2001) for an overview) and AI researchers (Doyle, Shoham, & Wellman, 1991; Doyle & Wellman, 1994), *most of the preferences that we express or act upon seem to be of this type.*

In this paper we show that CP-nets are not only an intuitive tool for structuring one's preferences, but they also support an efficient optimization process. In fact, given preferences specified using this formalism, the problem of obtaining a single preferentially non-dominated satisfying assignment is, in some sense, no harder than the problem of solving the underlying CSP.

Moreover, our algorithm has a useful *anytime* property—the set of solutions it holds at any time during the search process contains only Pareto-optimal solutions. To the best of our knowledge, these properties are not common to other constrained optimization frameworks.

The paper is structured as follows. In Section 2 we describe the CP-network preference representation. In Section 3 we describe a branch-and-bound algorithm that, given a CSP and a CP-net, computes the set of Pareto-optimal solutions (i.e., those solutions of the CSP for whose no other solution is provably preferred). Section 4 concludes the paper with a discussion of future work.

2 Background

We begin with a discussion of the standard notions from decision theory that we require and of the CP-network formalism for preference representation.

2.1 Preferential Independence *Ceteris Paribus*

We assume a set of variables $\mathbf{V} = \{X_1, \ldots, X_n\}$ with finite domains $\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)$. The decision maker wants to specify a *preference ranking* over complete assignments on $\mathbf{V}$. Each such assignment can be seen as a possible outcome of the decision maker's action, where these could be some physical action or, as in configuration problems, simply a choice of a value for each variable. The set of all outcomes is denoted $\mathcal{O}$. A *preference ranking* is a total preorder $\succeq$ over the set of outcomes: $o_1 \succeq o_2$ means that outcome o_1 is equally as good as or preferred to o_2 by the decision maker. We use $o_1 \succ o_2$ to denote the fact that outcome o_1 is strictly preferred to o_2 (i.e., $o_1 \succeq o_2$ and $o_2 \not\succeq o_1$), while $o_1 \sim o_2$ denotes that the decision maker is indifferent between o_1 and o_2 (i.e., $o_1 \not\succeq o_2$ and $o_2 \not\succeq o_1$). We use the terms preference *ordering* and *relation* interchangeably with *ranking.*

Direct assessment of a preference relation is generally infeasible due to the exponential size of $\mathcal{O}$. Fortunately, a preference relation can be specified concisely, at least partially, if it exhibits sufficient structure—a property on which much recent work in the area of preference elicitation (and AI in general) focuses. Independence assumptions play a key role in such specifications.

Definition 1 A subset of variables $\mathbf{X}$ is *preferentially independent* of its complement $\mathbf{Y} = \mathbf{V} - \mathbf{X}$ iff, for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}(\mathbf{X})$, $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{D}(\mathbf{Y})$ (where we use $\mathcal{D}(\cdot)$ to denote the domain of a set of variables as well), we have:

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

If this relation holds, we say that $\mathbf{x}_1$ is preferred to $\mathbf{x}_2$ *ceteris paribus* (all else being equal). This implies that one's preferences for different values of $\mathbf{X}$ do not change as other attributes vary. The analogous notion of conditional preferential independence is defined as follows.

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

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

In other words, the preferential independence $\mathbf{X}$ of $\mathbf{Y}$ holds if $\mathbf{Z}$ is assigned to $\mathbf{z}$. If such a relation holds for all assignments on $\mathbf{Z}$ we say that $\mathbf{X}$ is conditionally preferentially independent of $\mathbf{Y}$ given $\mathbf{Z}$. Note that these notions are standard in multi-attribute utility theory (Keeney & Raiffa, 1976), and can be viewed as qualitative variants of (arguably more familiar) quantitative notions of utility independence.

2.2 CP-nets

CP-nets were introduced as a tool for compactly representing qualitative preference relations (Boutilier et al., 1999). This graphical model exploits conditional preferential independence in structuring a decision maker's preferences under a *ceteris paribus* assumption. Although reasoning about *ceteris paribus* statements has been explored within AI (Wellman & Doyle, 1991), CP-networks are the first graphical model based on the notions of purely qualitative preferential independence captured by the *ceteris paribus* assumption, and bear a superficial similarity to Bayesian networks (Pearl, 1988).[4] However, the nature of the relationship between nodes within a CP-net is generally quite weak compared with the probabilistic relations in Bayes nets.

During preference elicitation, for each variable X, the decision maker is asked to specify a set of *parent* variables $Pa(X)$ that can affect her preferences over the values of X. That is, given a particular value assignment

[4]Bacchus and Grove (1995) have developed undirected graphical preference and utility models that bear some relation to CP-networks.

to $Pa(X)$, the decision maker should be able to determine a preference ordering for the values of X, all other things being equal. Formally, X is conditionally preferentially independent of $\mathbf{V} - \{X, Pa(X)\}$ given $Pa(X)$. This information is used to create the graph of the CP-net in which each node X has $Pa(X)$ as its immediate predecessors. Given this structural information, the decision maker is asked to explicitly specify her preferences over the values of X for each assignment to $Pa(X)$. This conditional preference ranking over the values of X is captured by a *conditional preference table* (CPT) which annotates the node X in the CP-net. That is, for each assignment to $Pa(X)$, $CPT(X)$ specifies a total order over $\mathcal{D}(X)$, such that for any two values $x_i, x_j \in \mathcal{D}(X)$, either $x_i \succ x_j$ or $x_j \succ x_i$.

Definition 3 A *CP-net* over variables $\mathbf{V} = \{X_1, \ldots, X_n\}$ is a directed graph G over $X_1, \ldots, X_n$ whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in \mathbf{V}$. Each conditional preference table $CPT(X_i)$ associates a total order $\succ^i_{\mathbf{u}}$ with each instantiation $\mathbf{u}$ of X_i's parents $Pa(X_i) = \mathbf{U}$.

For simplicity of presentation, we insist that the ordering $\succ^i_{\mathbf{u}}$ expressed in the CPTs of the network be total. As such, conditional expressions of indifference are not allowed. Nothing critical hinges on this, but algorithms for processing CP-nets with statements of indifference are slightly more involved. We refer to (Boutilier et al., 2003) for a discussion of this point.

Generally, nothing in the semantics forces CP-nets to be acyclic. However, acyclicity of a CP-net ensures that it specifies a consistent preference order (Boutilier et al., 1999). We note in passing that cyclic CP-nets are of potential interest in some domains, but their applicability is limited by the fact that asymmetry of the induced preference relation is not always guaranteed in the presence of cycles. A general effective algorithm for verifying the consistency of cyclic CP-nets does not yet exist, and we believe that the problem is NP-hard. The analysis of preference orderings induced by cyclic CP-nets and the development of effective algorithms for testing their consistency is an area of ongoing research, and some initial results on these issues currently exist (Domshlak & Brafman, 2002; Domshlak, 2002). In this paper we restrict the discussion to acyclic CP-nets. Likewise, for a formal specification of the semantics of CP-networks and other properties of this representation, we refer the reader to (Boutilier et al., 2003).

2.3 The Induced Preference Graph

The preference information captured by an acyclic CP-net $\mathcal{N}$ can be viewed as a set of logical assertions about a user's preference ordering over complete assignments to variables in the network. These statements are generally not complete, that is, they do not determine a unique preference ordering. Those orderings consistent with $\mathcal{N}$ can be viewed as possible models of the user's preferences, and any preference assertion that holds in all such models can be viewed as a consequence of the CP-net. More precisely, we say that the preference $o \succ o'$ is a *consequence* of $\mathcal{N}$, written $\mathcal{N} \models o \succ o'$, iff $o \succ o'$ holds in all preference orderings consistent with the *ceteris paribus* preference statements encoded by the CPTs of $\mathcal{N}$ (Boutilier et al., 2003).

The set of consequences $o \succ o'$ of an acyclic CP-net constitutes a partial order over outcomes: o is preferred to o' in this ordering iff $\mathcal{N} \models o \succ o'$. This partial order can be represented by an acyclic directed graph, referred to as the *induced preference graph*:

(i) The nodes of the induced preference graph correspond to the complete assignments to the variables of the network, and

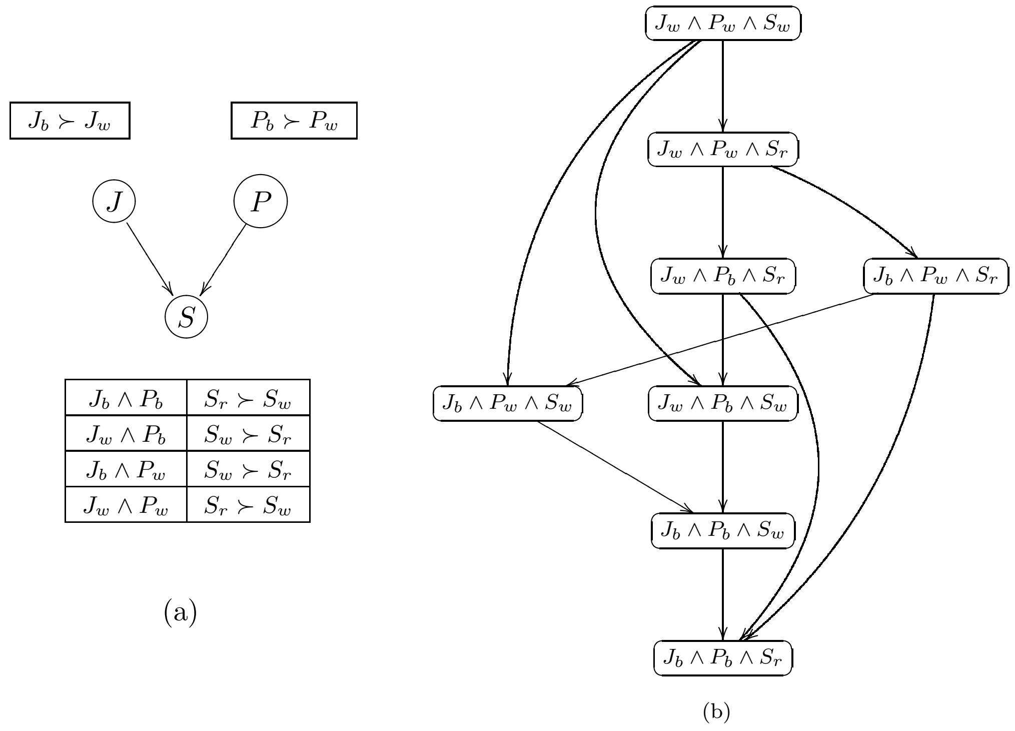
(ii) there is an edge from node o' to node o if and only if the assignments at o' and o differ only in the value of a single variable X, and given the values assigned by o' and o to $Pa(X)$, the value assigned by o to X is preferred to the value assigned by o' to X.

In turn, the transitive closure of this graph specifies the (asymmetric) partial order over outcomes induced by the CP-net. More precisely, we have that $\mathcal{N} \models o \succ o'$ iff there is a directed path from o' to o in the induced preference graph for $\mathcal{N}$.

We illustrate the CP-net model and the induced preference graph using the following simple example. For simplicity of presentation, all variables in this example are Boolean. However, the semantics of CP-nets is defined for variables with arbitrary finite domains.

Figure 1(a) illustrates a CP-net that expresses my preferences for evening dress. This network consists of three variables J, P, and S, standing for the color of my jacket, pants, and shirt, respectively. I unconditionally prefer black to white as a color for both the jacket and the shirt, while my preference between red and white shirts is conditioned on the *combination* of jacket and pants: if they are the same color, then a white shirt will make my outfit too colorless, thus I prefer a red shirt. Otherwise, if the jacket and pants are different colors, then a red shirt will probably appear too flashy, thus I prefer a white shirt.

Figure 1(b) shows the graph corresponding to the partial order over outcomes that this CP-net induces. The top element ($J_w \wedge P_w \wedge S_w$) is the worst outcome while the bottom element ($J_b \wedge P_b \wedge S_r$) is the best. Arrows are directed from less preferred to more preferred outcomes according to the immediate preference relations specified by $CPT(P)$, $CPT(P)$ and $CPT(S)$. Finally, the transitive closure of this graph precisely captures everything that can be proven about my preference relation for alternative evening dress given the information present in the CP-net.



$J_b \succ J_w$
$P_b \succ P_w$
J
P
S
$J_b \wedge P_b$ | $S_r \succ S_w$
$J_w \wedge P_b$ | $S_w \succ S_r$
$J_b \wedge P_w$ | $S_w \succ S_r$
$J_w \wedge P_w$ | $S_r \succ S_w$
(a)
$J_w \wedge P_w \wedge S_w$
$J_w \wedge P_w \wedge S_r$
$J_w \wedge P_b \wedge S_r$
$J_b \wedge P_w \wedge S_r$
$J_b \wedge P_w \wedge S_w$
$J_w \wedge P_b \wedge S_w$
$J_b \wedge P_b \wedge S_w$
$J_b \wedge P_b \wedge S_r$
(b)

Figure 1: "Evening Dress" CP-net.

2.4 Reasoning with CP-nets

The main purpose of any preference representation is to support answering various queries about the preferences of the decision maker. Two fundamental types of queries are:

1. *Outcome optimization*—determining the best outcome consistent with certain evidence (or one of the nondominated outcomes, if the best outcome is not unique).

2. *Outcome comparison*—preferential comparison between a pair of outcomes.

General procedures for both of these tasks with respect to a CP-net have been introduced (Boutilier et al., 1999, 2003). However, while outcome optimization with respect to a CP-net has been shown to be computationally easy, there are both polynomial and NP-hard cases of outcome comparison. In this section we briefly overview relevant results on reasoning about preferences modeled by a CP-net (Boutilier et al., 2003).

2.4.1 Outcome optimization queries

Given an acyclic CP-net $\mathcal{N}$, we can easily determine the best outcome among those preference rankings that satisfy $\mathcal{N}$. Such a query is called *outcome optimization.* Intuitively, we simply need to sweep through the network from top to bottom (i.e., from ancestors to descendants) setting each variable to its most preferred value given the instantiation of its parents. Indeed, while the network does not usually determine a unique ranking, it does determine a unique best outcome.[5] More generally, suppose that we are given evidence constraining possible outcomes in the form of an instantiation $\mathbf{z}$ of some subset $\mathbf{Z} \subseteq \mathbf{V}$ of the network variables. Determining the best completion of $\mathbf{z}$ (that is, the best outcome consistent with $\mathbf{z}$) can be achieved in a similar fashion.

More precisely, outcome optimization queries can be answered using the following *forward sweep* procedure, taking time linear in the size of the network. Given an acyclic CP-net $\mathcal{N}$ over a set of variables $\mathbf{V}$, and an assignment $\mathbf{z}$ to some $\mathbf{Z} \subseteq \mathbf{V}$, denote by $\mathit{Comp}(\mathbf{z})$ the set of all assignments over $\mathbf{V}$ in which $\mathbf{Z}$ takes the value $\mathbf{z}$. The goal is to determine the (unique) $o \in \mathit{Comp}(\mathbf{z})$ such that $\mathcal{N} \models o \succ o'$ for all $o' \in \mathit{Comp}(\mathbf{z}) - \{o\}$. This can be effected by a straightforward sweep through the network. Let $X_1, \ldots, X_n$ be any topological ordering of the variables in $\mathbf{V}$ with respect to the graph of $\mathcal{N}$. We set $\mathbf{Z} = \mathbf{z}$, and iterate from X_1 to X_n, assigning each $X_i \notin \mathbf{Z}$ in turn to its most preferred value given the instantiation of its parents. This procedure exploits the considerable power of the *ceteris paribus* semantics to easily find an optimal outcome given certain observed evidence.

[5] We refer the reader to (Boutilier et al., 2003) for a discussion on *partial CP-nets* that specify more than one best outcome. Determining the set of all best outcomes with respect to such acyclic CP-nets is linear in the size of the output (i.e., the set of best outcomes).

2.4.2 Outcome Comparison Queries

Outcome optimization is not the only task that should be supported by a preference representation. Another basic query with respect to such a model is *preferential comparison* between outcomes. Two outcomes o and o' can stand in one of three possible relations with respect to a CP-net $\mathcal{N}$: either $\mathcal{N} \models o \succ o'$; or $\mathcal{N} \models o' \succ o$; or $\mathcal{N} \not\models o \succ o'$ and $\mathcal{N} \not\models o' \succ o$. The third case, specifically, means that the network $\mathcal{N}$ does not contain enough information to prove that either outcome is preferred to the other (i.e., there exist preference orderings satisfying $\mathcal{N}$ in which $o \succ o'$ and in which $o' \succ o$). There are two distinct ways in which we can compare two outcomes using a CP-net:

1. *Dominance queries* – Given a CP-net $\mathcal{N}$ and a pair of outcomes o and o', ask whether $\mathcal{N} \models o \succ o'$. If this relation holds, o is preferred to o', and we say that o *dominates* o' with respect to $\mathcal{N}$.

2. *Ordering queries* – Given a CP-net $\mathcal{N}$ and a pair of outcomes o and o', ask if $\mathcal{N} \not\models o' \succ o$. If this relation holds, there exists a preference ordering consistent with $\mathcal{N}$ in which $o \succ o'$. In other words, it is consistent with the knowledge expressed by $\mathcal{N}$ to order o "over" o' (i.e., to assert that o is preferred to o'). In such a case we say o is *consistently orderable over* o' with respect to $\mathcal{N}$.

Ordering queries are a weak version of dominance queries, but they are sufficient in many applications where, given a pair of outcomes o and o', one can be satisfied by knowing only that $o' \not\succ o$. For example, consider a set of products that a human or automated seller would like to present to a customer in decreasing order of customer preference. In this case, there seems to be no difference between sorting the products according to a strong relation $\succ$, or according to its weaker counterpart $\not\prec$. Ordering queries with respect to acyclic CP-nets can be answered in time *linear* in the number of variables (Boutilier et al., 2003). Informally, ordering a pair of outcomes o and o' is based on a procedure that, given a query $\mathcal{N} \models o \not\prec o'$, returns in linear time either "yes" or "don't know". However, if $o \neq o'$, then this procedure returns "yes" for at least one of the two queries $\mathcal{N} \models o' \not\prec o$ and $\mathcal{N} \models o \not\prec o'$. In what follows, we refer to this procedure as the *ordering procedure*.

In some applications ordering queries are not sufficient. Indeed, the constrained optimization algorithm presented later requires the ability to answer dominance queries. An understanding of the computational properties of dominance queries is an important part of the research on CP-nets (Domshlak & Brafman, 2002; Domshlak, 2002). Comparisons between outcomes that differ in the value of a single variable X only are easy: we simply check the CPT of X and determine which outcome sets it to a more preferred value. We can view the improved outcome as a product of a single improving "flip" in the value of X. The only current technique for solving more general dominance queries is based on the idea of a *flipping sequence* (Boutilier et al., 1999). An *improving* flipping sequence is a sequence of progressively better outcomes, each obtained from its predecessor via a single value flip.

Definition 4 Let $\mathcal{N}$ be a CP-net over variables $\mathbf{V}$, with $X \in \mathbf{V}$, $\mathbf{U}$ the parents of X, and $\mathbf{Y} = \mathbf{V} - (\mathbf{U} \cup \{X\})$. Let $\mathbf{u}x'\mathbf{y} \in \mathcal{D}(\mathbf{V})$ be any outcome. An *improving flip* of outcome $\mathbf{u}x'\mathbf{y}$ with respect to variable X is any outcome $\mathbf{u}x\mathbf{y}$ such that $x \succ x'$ given $\mathbf{u}$ in network $\mathcal{N}$. Note that an improving flip with respect to X may not exist if x' is the most preferred value of X given $\mathbf{u}$.

An *improving flipping sequence* with respect to network $\mathcal{N}$ is any sequence of outcomes $o_1, \ldots, o_k$ such that o_{i+1} is an improving flip of o_i with respect to some variable (for each $i < k$). An improving flipping sequence for a pair of outcomes o' and o is any improving sequence $o_1, \ldots, o_k$ with $o' = o_1$ and $o = o_k$.[6]

Informally, each improving flipping sequence from an outcome o' to an o corresponds to a directed path from the node o' to the node o in the preference graph induced by $\mathcal{N}$. It is intuitively clear that the existence of an improving flipping sequence from o' to o implies that all the total orderings of the outcomes that are consistent with $\mathcal{N}$ must prefer o' to o. It is also not too difficult to show that the lack of a flipping sequence from o' to o implies the existence of such a total ordering in which o' is preferred to o.

Indeed, given a CP-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 (Boutilier et al., 1999). It turns out that answering dominance queries even with respect to CP-nets with only Boolean variables is hard in general. In particular, we have that (Boutilier et al., 2003):

- When the binary CP-net forms a directed tree, the complexity of dominance testing is quadratic in the number of variables.

[6] *Worsening flips* and *worsening flipping sequences* are defined analogously. Obviously, any worsening flipping sequence is the inverse of an improving flipping sequence, and vice versa.

- When the binary CP-net forms a polytree (i.e., the induced undirected graph is acyclic), dominance testing is polynomial in the size of the CP-net description.

- When the binary CP-net is directed-path singly connected (i.e., there is at most one directed path between any pair of nodes), dominance testing is NP-complete. The problem remains hard even if the node in-degree in the network is bounded by a small constant.

- Dominance testing remains in NP-complete if the number of alternative paths between any pair of nodes in the CP-net is polynomially bounded.

3 Constrained Optimization over CP-nets

In Section 2.4.1 it was shown that, given an acyclic CP-net and a (possibly empty) partial assignment to its variables, determining the optimal outcome is straightforward and can be done in time linear in the number of variables. However, if some of the variables are mutually constrained by a set of hard constraints, then the optimal outcome with respect to a CP-net may not be feasible. In this case, our goal should be to determine either one, some, or all outcomes that are (i) feasible (i.e., consistent with the hard constraints), and (ii) not dominated by any other feasible outcome. Given a set of feasible outcomes S, an outcome $o \in S$ is said to be *Pareto optimal* with respect to preference order $\succ$ if and only if there is no $o' \in S$ such that $o' \succ o$. Determining the set of Pareto-optimal feasible outcomes is not trivial, since for general sets of constraints without preferences the problem is NP-hard. In this section we introduce a branch-and-bound algorithm for determining such a set of outcomes, given an acyclic CP-net and a set of hard constraints on its variables, and analyze various computational properties of this algorithm.

3.1 An Algorithm for Constrained Optimization

Formally, the problem is defined by an acyclic CP-net $\mathcal{N}_{orig}$, and a set of hard constraints $\mathcal{C}_{orig}$, posed on the variables of $\mathcal{N}_{orig}$. The Search-CP algorithm (depicted in Figure 2) is recursive, and each recursive call accepts three arguments:

1. A CP-net $\mathcal{N}$, which is a subnet of the original acyclic CP-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 $\mathcal{C}_{orig}$ restricted to the variables of $\mathcal{N}$, and

3. An assignment $\mathcal{K}$ to all variables in $\mathcal{N}_{orig} - \mathcal{N}$.

The initial call to Search-CP is done with $\mathcal{N}_{orig}$, $\mathcal{C}_{orig}$, and $\{\}$, respectively. The Search-CP algorithm works as follows:

1. Search-CP starts with an empty set of solutions, and continuously extends it by adding new non-dominated solutions.

2. At each stage of the algorithm, the current set of solutions serves as a lower bound for future candidates.

3. 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 most important property of this algorithm is that the set of iteratively generated solutions never shrinks. Indeed, as we will prove later on: once a particular solution is generated, no solution dominating it can follow. This means that the Search-CP algorithm has a clear *anytime* behavior (Dean & Boddy, 1988; Horvitz & Seiver, 1997; Zilberstein, 1993)—at any moment we can stop the execution of the algorithm, and the solution set generated to that point will be a subset of the set of all problem solutions.[7] This property of the algorithm is very important, since the entire set of non-dominated feasible solutions can be exponential in the number of variables. Without this property, we would have to maintain a potentially unmanageable set of candidate solutions. As far as we know, this property of the CP-net model together with the Search-CP algorithm is unique in the area of preference-based constrained optimization.

The algorithm proceeds by assigning values to the variables in a top-down manner according to a topological ordering of the CP-net; this ordering is determined in the recursive steps of the program. In each call, the values for a topmost variable X in the current subnetwork $\mathcal{N}$ (i.e., X has no parents in $\mathcal{N}$) are considered according to the preferential ordering induced by the assignment to $Pa(X)$ (where $Pa(X)$ is defined in $\mathcal{N}_{orig}$). Whenever a

[7]Using Zilberstein's (1993) terminology, this algorithm is anytime with respect to the *specificity* quality measure: the intermediate result is always correct, but the level of detail (e.g., completeness) is increased over time.

variable X is assigned a value x_i, the current set of constraints $\mathcal{C}$ is strengthened into $\mathcal{C}_i$ (line 5). As a result of this propagation of $X = x_i$, values for some variables (at least for X) will be fixed automatically. The resulting partial assignment $\mathcal{K}'$ will extend the current context $\mathcal{K}$ and this extended context $\mathcal{K} \cup \mathcal{K}'$ will be used in subsequent recursive calls of the algorithm. The CP-net induced by this partial assignment $\mathcal{K}'$ to the variables of $\mathcal{N}$ is denoted by $\mathcal{N}_i$. Note that our algorithm is independent of the constraint propagation technique being used to determine $\mathcal{K}'$.

After strengthening the set of constraints $\mathcal{C}$ by $X = x_i$ to $\mathcal{C}_i$, some pruning can take place in the search tree. Suppose that the set of constraints $\mathcal{C}_i$ is at least as restrictive as some other set of constraints $\mathcal{C}_j = \mathcal{C} \cup \{X = x_j\}$ where $j < i$. We claim that there is no point pursuing the branch $X = x_i$, as any feasible assignment with $X = x_i$ will be dominated by some previously achieved assignment in which $X = x_j$. While the formal claim is provided by Lemma 2 in Section 3.3, informally, this is not difficult to see: *given* $\mathcal{K}$, for any feasible assignment o_i consistent with x_i, the assignment o_j, which is identical to o_i except that x_i is replaced by x_j, is also feasible. This is because $\mathcal{C}_i$ is at least as restrictive as $\mathcal{C}_j$. However, x_j is preferred to x_i given $\mathcal{K}$, so o_j is preferred to o_i. This justifies the second condition of line 6.

The generated partial assignment $\mathcal{K}'$ may cause some of the constraints and some of the edges of the CP-net to become redundant. Intuitively, $\mathcal{K}'$ makes a constraint C redundant if, for each variable X involved in C, either we have $\mathcal{K}'$ instantiating X, or $\mathcal{K}'$ refines the domain of X to contain only values permitted by C. In turn, edges in the CP-net become redundant because one's conditional preferences may not depend on some of the conditioning variables in certain contexts. As a simple example, consider a variable $X \in \mathcal{N}$ with two parents Y and Z. It may be the case that if Y is assigned the value y_1, one prefers x_1 to x_2 regardless of the value of Z, while if Y is assigned y_2, one prefers $X = x_1$ iff $Z = z_1$. Thus, although X depends on both Y and Z, if Y is assigned y_1, the edge from Z to X is redundant.[8] Note that the extent of such problem refinement (with respect to both constraints and preferential dependencies) depends on the depth of constraint propagation performed in step 5.

If $\mathcal{N}_i$, the network obtained from projecting $\mathcal{K}'$ on $\mathcal{N}$, is disconnected with respect to both the edges of the network and the constraints, then each connected component invokes an *independent* search in the extended

[8]This reflects a form of *context-specific independence* often used in the representation of conditional distributions in Bayesian networks (Boutilier, Friedman, Goldszmidt, & Koller, 1996).

context $\mathcal{K} \cup \mathcal{K}'$. This is because optimization of the variables within such a component is independent of the variables outside that component. The m separate cases of lines 9–14 reflect these independent components. This potential for substantial computational savings is why we care about identifying redundant edges and constraints.

Note that when the nondominated solutions for a particular subnet generated after the assignment $X = x_i$ are returned, each such solution is compared to *all* nondominated solutions involving assignments that are more preferred in the current context $\mathcal{K}$, that is, those corresponding to the assignments $X = x_j$, for all $j < i$ (line 14). The key point is that a solution involving $X = x_i$ cannot dominate any solution involving $X = x_j$, $j < i$; this follows from the *ceteris paribus* semantics. However, such a pair of solutions can be incomparable. This explains the need for dominance testing and the dependence of the the Search-CP algorithm on the complexity of dominance testing. It also justifies the claim that no solution has to be retracted. The corresponding formal claim is provided by Lemma 3 in Section 3.3.

The complexity results for the CP-net based preferential comparison between the outcomes (presented in Section 2.4.2) induce the following, somewhat surprising, computational property of constrained optimization over the preference relation described by a CP-net.

1. If we are interested in getting *any* single non-dominated solution for the given set of constraints (which is often the case), then we can output the *first* feasible outcome generated by the Search-CP algorithm. Note that no dominance queries will be required in this case since there is nothing to compare with the first generated solution. Thus, although it is hard to compare the complexity of a decision problem (CSP) with an optimization problem, in some sense we can say that this constrained optimization problem is no harder than the underlying constraint satisfaction task. Of course, we are constrained to assign values to variables in a topological order, and this order may not be optimal for solving the underlying CSP quickly.

2. If we are interested in getting *all*, or even *some* non-dominated solutions for the given constrained optimization problem then, generally, we need to compare different solutions. The overhead of the required dominance queries makes our problem (at least in a practical sense) much harder than the corresponding CSP, except for cases in which dominance testing in the given CP-net is polynomial (see Section 2.4.2).

However, it is worth noting that in many cases, the dominance query required in step 14 of Search-CP can often be solved quickly. This follows from the fact that when comparing $\mathcal{K} \cup o'$ and $\mathcal{K} \cup o$, we actually want to determine whether $\mathcal{K} \cup o' \not\succ \mathcal{K} \cup o$, and not whether $\mathcal{K} \cup o' \succ \mathcal{K} \cup o$. This can sometimes be resolved by a simple, linear-time, ordering query (Boutilier et al., 2003), as discussed in Section 2.4.2.

3.2 Example Application of Search-CP

We now illustrate the execution of the Search-CP algorithm on an example. Consider the CP-net $\mathcal{N}_{orig}$ over six Boolean variables $\{A, B, C, D, E, F\}$, represented by the nodes and the (solid) directed arcs in Figure 3(a), while the CPTs of the variables appear in Figure 3(b). Suppose that the set of hard constraints $\mathcal{C}_{orig}$ consists of three binary constraints, represented by the dashed, undirected arcs in Figure 3(a), as follows:

$$\begin{aligned} c(A,E) &: \{A=a\} \leftrightarrow \{E=e\} \\ c(C,E) &: \{C=c\} \leftrightarrow \{E=\bar{e}\} \\ c(D,F) &: \{D=d\} \rightarrow \{F=\bar{f}\} \end{aligned}$$

For instance, the constraint between the variables A and E asserts that if A takes the value a, then E has to take the value e, and vice versa.

The initial call to Search-CP (numbered as call 0) has arguments: $\mathcal{N} = \mathcal{N}_{orig}$, $\mathcal{C} = \mathcal{C}_{orig}$, and the empty partial assignment $\mathcal{K}_0 = \{\}$. The candidates for variable selection in step 1 are A, C and E. Let us suppose that A has been chosen. The variable A is preferentially independent of all others, and $a \succ \bar{a}$ is the preference ordering of its values. Therefore, first we perform steps 5–14 with the value a, then with the value $\bar{a}$.

Let us consider the iteration with $A = a$. The branch of the search tree corresponding to the assignment $A = a$ is depicted in Figure 4. Since $A = a$ participates in constraint $c(A,E)$, in step 5 the constraint set $\mathcal{C}$ is strengthened to $\mathcal{C}_a$, implying $E = e$. The pruning step 6 is not relevant in this iteration since a is the first (i.e., best) value of A, and $\mathcal{C}_a$ is consistent. We continue with steps 7–8, resulting in $\mathcal{K}'_0 = \{a, e\}$ and $\mathcal{N}_a$ as in Figure 3(c-d). Since $\mathcal{N}_a$ consists of only one connected component, in steps 10–11 we perform only one call (call 1) to Search-CP with $\mathcal{N} = \mathcal{N}_a$, $\mathcal{C} = \mathcal{C}_a$, and $\mathcal{K} = \{a, e\}$.

The candidates for variable selection step 1 in call 1 are B, C and F. Let us suppose that B has been chosen. Given the value of A in $\mathcal{K}_1 = \{a, e\}$, $b \succ \bar{b}$ is the preference ordering of the values of B. Therefore, we start with the iteration corresponding to $B = b$. Again, we skip the pruning part of step 6, and since B does not participate in any of the hard constraints, we have $\mathcal{C}_b = \mathcal{C}$ and $\mathcal{K}'_1 = \{b\}$. The network $\mathcal{N}_b$ achieved after step 8 is presented in Figure 3(e-f). Observe that $\mathcal{N}_b$ consists of two connected components, since D is preferentially independent of C given $B = b$. Thus, in steps 10–11 we perform two (unordered) calls to Search-CP: call 2 with $\mathcal{N}_b^1$ (consisting of C), and call 3 with $\mathcal{N}_b^2$ consisting of D and F. Both these calls are performed with the (unchanged from call 1) set of constraints $\mathcal{C} = \mathcal{C}_b$ and with the context $\mathcal{K} = \{a, e, b\}$.

In call 2 the network consists of a single, preferentially independent node C, and the preferential ordering of the C values is $c \succ \bar{c}$. Here, the first iteration with c fails at step 6 since $\mathcal{C}_c$ implies $E = \bar{e}$, which contradicts $\mathcal{K}_2 = \{a, e, b\}$. (The corresponding (leftmost) branch of the search tree appears as $\bigotimes$.) The second iteration with $\bar{c}$ is successful, and, since $\mathcal{N}_{\bar{c}}$ is empty, call 2 terminates by returning the local result set $\mathcal{R}_2 = \{\bar{c}\}$.[9]

In call 3 the network consists of two, preferentially independent nodes D and F. Suppose that D has been chosen in step 1, and consider the first iteration with $D = d$. The strengthened constraint set $\mathcal{C}_d$ consistently implies $F = \bar{f}$, resulting in $\mathcal{K}'_3 = \{d, \bar{f}\}$, and the empty network $\mathcal{N}_d$. Therefore, this iteration extends the empty set of local results $\mathcal{R}_3$ to $\mathcal{R}_3 = \{d\bar{f}\}$. The next iteration is with $D = \bar{d}$; we have $\mathcal{C}_{\bar{d}} = \emptyset$ and $\mathcal{K}'_3 = \{d\}$. The reduced network $\mathcal{N}_{\bar{d}}$ consists of a single node F, thus the only call (call 4) to Search-CP is with $\mathcal{N} = \mathcal{N}_{\bar{d}}$, $\mathcal{C} = \mathcal{C}_{\bar{d}}$, and $\mathcal{K} = \{a, e, b, \bar{d}\}$.

In call 4 the preferential ordering of the values of F, given the value of its parent E in $\mathcal{K}_4 = \{a, e, b, \bar{d}\}$, is $f \succ \bar{f}$. The first iteration with $F = f$ extends the (empty so far) local set of results to $\mathcal{R}_4 = \{f\}$. Now consider the next iteration with $F = \bar{f}$. This iteration terminates at step 6 due to the pruning technique, since $\mathcal{C}_{\bar{f}} = \mathcal{C}_f (= \mathcal{C} = \emptyset)$ and $\bar{f} \prec f$. (The corresponding branch of the tree in Figure 4 appears as $\top$). Note that this is the first time that the pruning condition was effective, but later we show that, at least in this particular example, this condition prunes most of the search tree.

The returned set of results $\{f\}$ from call 4 is received as (the only) $\mathcal{S}_{\bar{d}}^1$ in steps 10–11 of the iteration with $\bar{d}$ in call 3. Recall that, at this stage of call 3, we have $\mathcal{R}_3 = \{d\bar{f}\}$, $\mathcal{K}_3 = \{a, e, b\}$, and $\mathcal{K}'_3 = \{\bar{d}\}$. The cross product $\mathcal{K}'_3 \times \mathcal{S}_{\bar{d}}^1$ provides us with $\{\bar{d}f\}$, and in step 14 we test the assignments $abde\bar{f}$ and $ab\bar{d}ef$ for dominance with respect to the original

[9]Note that a simple forward checking procedure would have immediately restricted the domain of C to $\{\bar{c}\}$ when variable E was set to e. This would have eliminated C as a variable upon which to branch. We note that this and other domain pruning methods can easily be incorporated into our algorithm without difficulty.

CP-net $\mathcal{N}_{orig}$ minus the variables that are not involved in the query. These two outcomes are incomparable in $\mathcal{N}_{orig}$, thus we extend the set of local results to $\mathcal{R}_3 = \{d\bar{f}, \bar{d}f\}$, and return it back to call 1.

Now, after we finished with call 2 and call 3, we are back in call 1, moving to the steps 11–14 with $\mathcal{S}_b^{(1)} = \mathcal{R}_2 = \{\bar{c}\}$ and $\mathcal{S}_b^{(2)} = \mathcal{R}_3 = \{d\bar{f}, \bar{d}f\}$. Recall that at this stage we have $\mathcal{R}_1 = \{\}$, $\mathcal{K}_1 = \{a, e\}$, and $\mathcal{K}'_1 = \{b\}$. The cross product $\mathcal{K}'_1 \times \mathcal{S}_b^{(1)} \times \mathcal{S}_b^{(2)}$ provides us with $\{b\bar{c}d\bar{f}, b\bar{c}\bar{d}f\}$. Now, since $\mathcal{R}_1 = \{\}$, i.e., there is nothing to compare with, we extend the set of local results to $\mathcal{R}_1 = \{b\bar{c}d\bar{f}, b\bar{c}\bar{d}f\}$, and move to the next iteration. However, the next iteration terminates at step 6 due to the pruning technique, since $\mathcal{C}_{\bar{b}} = \mathcal{C}_b$ and $\bar{b} \prec b$. Returning back to the initial call 0, we extend the (empty so far) local set of results $\mathcal{R}_0$ to contain the result of call 2 extended by $\mathcal{K}'_0 = \{a, e\}$; that is, $\mathcal{R}_0 = \{ab\bar{c}de\bar{f}, ab\bar{c}\bar{d}ef\}$, and continue with the next iteration corresponding to $A = \bar{a}$.

The branch of the search tree corresponding to the assignment $A = \bar{a}$ is depicted in Figure 5. We omit a detailed, step-by-step discussion of this part of the search, but we would like to emphasize several points with respect to it. First, it is easy to see from the illustration that the order in which the values of the variable B are processed in the subtree $A = \bar{a}$ is different from the search branch corresponding to $A = a$. In general, this order can be different for all the variables of a CP-net. Second, during the processing of this branch we test three candidate solutions. However, only one of these candidates, $\bar{a}\bar{b}cd\bar{e}\bar{f}$, is successfully added to the set of solutions, while the other two candidates are found to be dominated and thus pruned: $\bar{a}\bar{b}c\bar{d}\bar{e}\bar{f}$ is pruned since it is dominated by $\bar{a}\bar{b}cd\bar{e}\bar{f}$, and $\bar{a}\bar{b}\bar{c}\bar{d}ef$ is dominated by $ab\bar{c}\bar{d}ef$. Note that this example emphasizes the need to compare a new candidate to *all* solutions generated so far, since $\bar{a}\bar{b}cd\bar{e}\bar{f}$ is the only solution that dominates $\bar{a}\bar{b}c\bar{d}\bar{e}\bar{f}$, and $ab\bar{c}\bar{d}ef$ is the only solution that dominates $\bar{a}\bar{b}\bar{c}\bar{d}ef$.

3.3 Formal Properties of Search-CP

Our main claim is the correctness of the algorithm:

Theorem 1 *Given a CP-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-CP *if and only if o is consistent with $\mathcal{C}$, and there is no other outcome o' consistent with $\mathcal{C}$ such that $\mathcal{N} \models o' \succ o$.*

To prove this theorem, we prove two main lemmas. First, we show that the pruning technique is sound—specifically, that no undominated solution is pruned. This means that we generate a superset of all undominated solutions. To show that our solution set is precisely the set of undominated solutions it is enough to demonstrate that if o were added to the generated set of solutions after o' then it is not the case that $o \succ o'$. This implies that a newly generated solution cannot dominate an existing solution.

Lemma 2 *The pruning technique of* Search-CP *(step 6) is sound.*

Proof: Let X be the current instantiated variable and let $\mathcal{K}$ be the current context. We know that all of X's parents in the CP-net are in $\mathcal{K}$. Suppose that C_i, the set of constraints obtained after assigning x_i to X, is equal to or contains C_j, the set of constraints obtained *earlier* when the branch $X = x_j$ was processed. The algorithm specification ensures that x_j is a more preferred value for X than x_i in context $\mathcal{K}$. Because $C_i \supseteq C_j$, any assignment to the remaining variables that satisfies C_i satisfies C_j. Consider such a satisfying assignment o_i. From the above, we know that o_j which is identical to o_i except for the value of X is also a satisfying assignment. By the CP-semantics, o_j is preferred to o_i. Hence, all solutions obtained in the pruned branch are dominated. ▮

Lemma 3 Search-CP *generates solutions in a non-increasing order.*

Proof: Let o and o' be two solutions such that o was generated before o'. Let X denote the earliest variable (in terms of instantiation order) on which these outcomes differ. Since o was generated before o', it must assign X a more preferred value (say x) than does o' (say x') given the (identical) values of X's parents. We now show by induction on X's position in the instantiation order that there cannot exist an improving flipping sequence from o to o'; hence, $\mathcal{N} \not\models o' \succ o$.

Suppose that X is the first variable in the instantiation ordering of Search-CP; this requires that X be a root node in $\mathcal{N}$. Since X is a root node, there is no way to flip X from x to x' while improving an outcome. Hence there can be no improving flipping sequence from o to o'.

Suppose that the inductive hypothesis holds for any two outcomes whose first difference (with respect to instantiation order) occurs in some variable at any position $p \leq k-1$ in the instantiation order. Let X occur at position k. If an improving sequence exists from o to o', at some point X must be flipped from x to x'. However, since o and o' agree on the values of the parents of X, and since x is preferred to x' given this parent instantiation, this cannot occur in the sequence until some parent of X is flipped. This implies that some ancestor of X must be flipped along any improving path from o to o', hence that some variable earlier in the instantiation order than X must be flipped earlier in the path than X. Let o'' be any outcome along the improving sequence from o to o' in which some variable earlier in the instantiation order than X has a value distinct from that assigned by o'. By the inductive hypothesis, we cannot have an improving sequence from o'' to o'; hence no improving sequence from o to o' exists. ▮

In particular, Lemma 3 implies the anytime property.

Lemma 4 (anytime property) *At each point during* Search-CP *the current set of solutions is contained in the set of all solutions.*

4 Summary and Future Work

In this paper we introduced a qualitative, decoupled framework for preference-based constrained optimization problems, adopting CP-nets as the underlying preference representation. The core of this framework is an efficient optimization algorithm which exploits the structure of user preferences and behaves in an anytime fashion—at any point we can stop the algorithm's execution, and the current solution set will contain only Pareto-optimal solutions.

Our work leaves open a number of important issues for future research. Of particular importance to real-world product configuration systems is the development of an *interactive* preference-based constrained optimization process, some aspects of which we discuss below.

One assumption made in our framework is that user preferences (in form of a CP-net) are specified before optimization is undertaken. Indeed, we saw that preference information can drastically prune the search space. On the other hand, by pre-computing the set of feasible outcomes, we can reduce the degree of intrusion upon the user required for preference assessment. In particular, in many circumstances, we may not need to rank all values of a variable X. For instance, we could ask only for a "top" portion of the ranking associated with a certain assignment to $Pa(X)$, assuming (at this stage) that any non-dominated feasible outcome will only involve these top values. For similar reasons, we may not want to ask the user a priori to assess the conditional rankings associated with each assignment to $Pa(X)$.

If we start with some unranked assignments, the search algorithm may find itself at some point lacking the information it needs to proceed. At this point, an appropriate query should be posed to the user. In addition, other queries could be posed. For instance, one could ask queries that are expected to offer the greatest potential for pruning based on the information about problem constrainedness obtained by the search algorithm so far.

Evidently, there is some tension between the two desiderata of minimizing user effort and reducing computation time, which points to a clear need for interactive search algorithms in which user preference in a particular region of the search space can be obtained through user queries in an attempt to minimize computational effort. This observation is not novel (see work, e.g., on interactive goal programming (Dyer, 1972)). There are clear tradeoffs involved between the number and complexity of the user queries required to prune part of the search space and the amount of search space expected to be pruned. Further study of these issues is clearly of great practical importance.

Other practical issues include the detailed study of the computational properties of our search algorithm. Refinements of the algorithm should also be explored with an eye toward improved computational performance. Possible refinements include adapting techniques for constraint satisfaction to our problem. For example, specific variable and value ordering techniques that depend on the underlying constraint set could be used in addition to our current technique (which orders variables and values in a way that reflects the preference structure). In addition, more sophisticated constraint propagation methods could easily be incorporated into our algorithm.

Acknowledgments: Thanks to Chris Geib for his contributions to earlier, related models of CP-nets, and to the anonymous referees for their useful comments. Boutilier, Hoos, and Poole were supported by the Natural Sciences and Engineering Research Council, and the Institute for Robotics and Intelligent Systems. Brafman gratefully acknowledges the support of the Paul Ivanier Center for Robotics Research and Production Management.

References

Bacchus, F., & Grove, A. (1995). Graphical models for preference and utility. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 3–10, Montreal, Canada.

Bertelè, U., & Brioschi, F. (1972). *Nonserial dynamic programming*, Vol. 91 of *Mathematics in Science and Engineering*. Academic Press.

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 Eighteenth National Conference on Artificial Intelligence*, pp. 526–532, Edmonton, Canada.

Boutilier, C. (2002). A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp. 239–246, Edmonton, Canada.

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., & Poole, D. (2003). CP-Nets: A tool for representing and reasoning with conditional *Ceteris Paribus* preference statements. *Journal of Artificial Intelligence Research (JAIR)*. to appear.

Boutilier, C., Brafman, R. I., Hoos, H. H., & Poole, D. (1999). Reasoning with conditional *Ceteris Paribus* preference statements. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 71–80, Stockholm, Sweden.

Boutilier, C., Friedman, N., Goldszmidt, M., & Koller, D. (1996). Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 115–123, Portland (OR), USA.

Boutilier, C., Patrascu, R., Poupart, P., & Schuurmans, D. (2003). Constraint-based optimization with the minimax decision criterion. In *Ninth International Conference on Principles and Practice of Constraint Programming*, pp. 168–182, Kinsale, Ireland.

Brafman, R. I., & Friedman, D. (2003). Synchronized rich-media adaptation. Tech. rep. 03-19, Dept. of CS, Ben-Gurion University, Israel.

Chajewska, U., Koller, D., & Parr, R. (2000). Making rational decision using adaptive utility elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pp. 363–369, Austin (TX), USA.

Dean, T., & Boddy, M. (1988). An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI–88)*, pp. 49–54, Minneapolis (MN), USA.

Dechter, R., Dechter, A., & Pearl, J. (1988). Optimization in constraint networks. In Oliver, R. M., & Smith, J. Q. (Eds.), *Influence Diagrams, Belief Nets and Decision Analysis*, pp. 411–425. John Wiley & Sons.

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., Rossi, F., Venable, C., & 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.

Doyle, J., Shoham, Y., & Wellman, M. (1991). A logic of relative desire (preliminary report). In *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems (ISMIS–91)*, Lecture Notes in Computer Science, pp. 16–31. Springer-Verlag.

Doyle, J., & Wellman, M. (1994). Representing preferences as ceteris paribus comparatives. In *Proceedings of the AAAI Spring Symposium on Decision-Theoretic Planning*, pp. 69–75, Stanford (CA), USA.

Dubois, D., & Prade, H. (1988). *Possibility Theory.* Plenum Press, New York (NY), USA.

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

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. Springer-Verlag.

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, Aachen, Germany.

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

Freuder, E. C. (1989). Partial constraint satisfaction. In Sridharan, N. S. (Ed.), *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 278–283, San Francisco (CA), USA. Morgan Kaufmann Publishers.

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

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

Hansson, S. O. (1996). What is ceteris paribus preference. *Journal of Philosophical Logic*, *25*(3), 307–332.

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 Academic Publishers.

Horvitz, E., & Seiver, A. (1997). Time-critical action: Representations and application. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pp. 250–257, Providence (RI), USA.

Jensen, F., Jensen, F. V., & Dittmer, S. (1994). From influence diagrams to junction trees. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 367–373, Seattle (WA), USA.

Keeney, R. L., & Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo (CA), USA.

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

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

Wang, T., & Boutilier, C. (2003). Incremental utility elicitation with the minimax regret decision criterion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 309–316, Acapulco, Mexico.

Wellman, M., & Doyle, J. (1991). Preferential semantics for goals. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pp. 698–703, Anaheim (CA), USA.

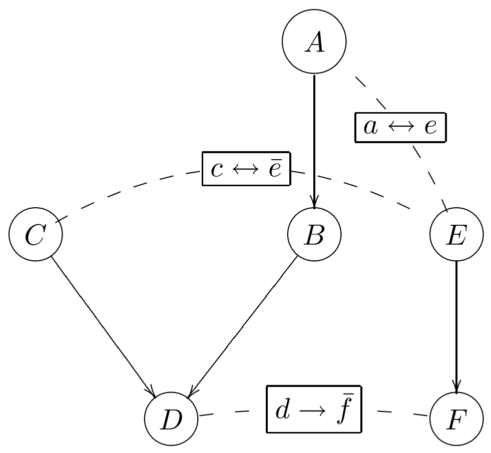
White, III, C. C., Sage, A. P., & Dozono, S. (1984). A model of multiattribute decisionmaking and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man and Cybernetics*, *14*(2), 223–229.

Zilberstein, S. (1993). *Operational Rationality through Compilation of Anytime Algorithms*. Ph.D. thesis, University of California at Berkeley (CA), USA.

```
Search-CP
Input: Acyclic CP-net $\mathcal{N}$, constraints $\mathcal{C}$,
          assignment $\mathcal{K}$ to the variables of $\mathcal{N}_{orig} - \mathcal{N}$.
Output: Set of all solutions for $\mathcal{C}$ that are Pareto-optimal w.r.t. $\mathcal{N}$.

1.   choose any variable $X$ with no parents in $\mathcal{N}$
2.   let $x_1 \succ \ldots \succ x_k$ be the preference ordering of $\mathcal{D}(X)$
        given the assignment to $Pa(X)$ in $\mathcal{K}$
3.   $\mathcal{R} := \emptyset$ (i.e., initialize the set of local results)
4.   for $i := 1$ to k do
5.      strengthen the constraints $\mathcal{C}$ by $X = x_i$ to obtain $\mathcal{C}_i$
6.      if $\mathcal{C}_i$ is consistent and $\mathcal{C}_j \nsubseteq \mathcal{C}_i$ for all $j < i$ then
7.         let $\mathcal{K}'$ be the partial assignment induced by $\mathcal{C}_i$
8.         reduce $\mathcal{N}$ to $\mathcal{N}_i$ by removing the variables assigned by $\mathcal{K}'$
9.         let $\mathcal{N}_i^{(1)}, \ldots, \mathcal{N}_i^{(m)}$ be the components of $\mathcal{N}_i$ that are connected either
              by the edges of $\mathcal{N}_i$ or by the constraints $\mathcal{C}_i$
10.        for each $j \in \{1, \ldots, m\}$ do
11.           $\mathcal{S}_i^{(j)} =$ Search-CP$(\mathcal{N}_i^{(j)}, \mathcal{C}_i, \mathcal{K} \cup \mathcal{K}')$
12.        if $\mathcal{S}_i^{(j)} \neq \emptyset$ for all $j \in \{1, \ldots, m\}$ then
13.           for each $o \in \mathcal{K}' \times \mathcal{S}_i^{(1)} \times \cdots \times \mathcal{S}_i^{(m)}$ do
14.              if $\mathcal{K} \cup o' \not\succ \mathcal{K} \cup o$ holds for each $o' \in \mathcal{R}$ then add $o$ to $\mathcal{R}$
        end if
15.     $i := i + 1$
     end for
16.  return $\mathcal{R}$
```

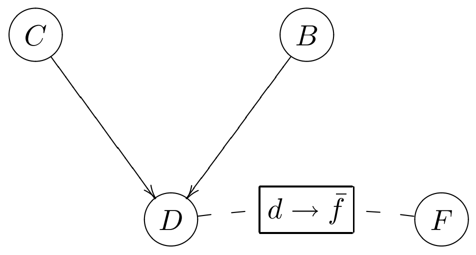
Figure 2: The Search-CP algorithm for acyclic CP-net based constrained optimization.



(a)

Variable	CPT
A	$a \succ \bar{a}$
B	$a : b \succ \bar{b}$ $\bar{a} : \bar{b} \succ b$
C	$c \succ \bar{c}$
D	$b : d > \bar{d}$ $\bar{b} \wedge c : d \succ \bar{d}$ $\bar{b} \wedge \bar{c} : \bar{d} \succ d$
E	$e \succ \bar{e}$
F	$e : f \succ \bar{f}$ $\bar{e} : \bar{f} \succ f$

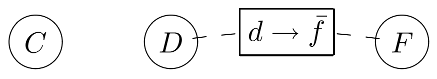
(b)



(c)

Variable	CPT
B	$b \succ \bar{b}$
C	$c \succ \bar{c}$
D	$b : d > \bar{d}$ $\bar{b} \wedge c : d \succ \bar{d}$ $\bar{b} \wedge \bar{c} : \bar{d} \succ d$
F	$f \succ \bar{f}$

(d)



(e)

Variable	CPT
C	$c \succ \bar{c}$
D	$d > \bar{d}$
F	$f \succ \bar{f}$

(f)

Figure 3: CP-net and the hard constraints for different steps in the recursion of the algorithm.

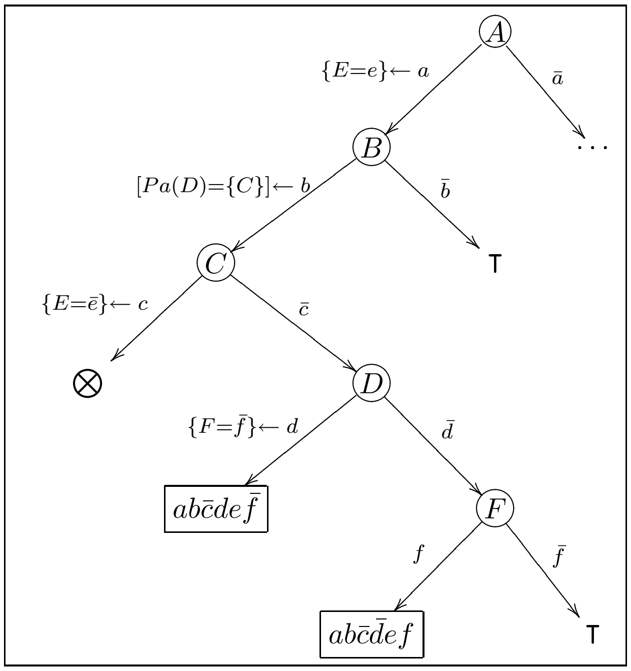


Figure 4: Search subtree corresponding to $A = a$.

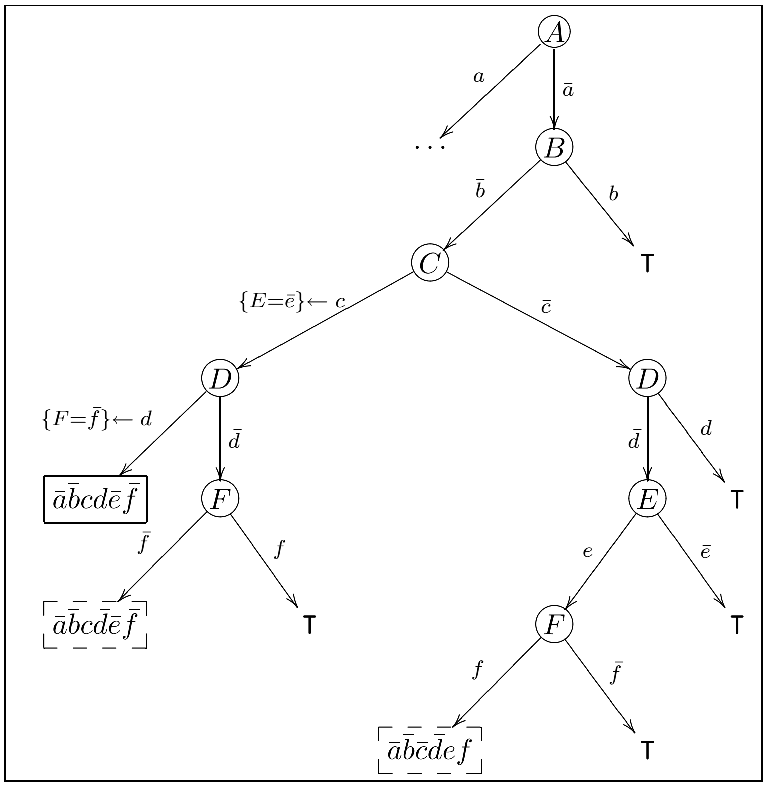


Figure 5: Search subtree corresponding to $A = \bar{a}$.