Extending Conditional Preference Networks to Handle Changes

Eisa Alanazi
Department of Computer Science
Umm Al-Qura University
Makkah, Saudi Arabia

***Abstract*—Conditional Preference Networks (CP-nets) are a compact and natural model to represent conditional qualitative preferences. In CP-nets, the set of variables is fixed in advance. That is, the set of alternatives available during the decision process are always the same no matter how long the process is. In many configuration and interactive problems, it is expected that some variables are subject to be included or excluded during the configuration process due to users showing interest or boredom on some aspects of the problem. Representing and reasoning with such changes is important to the success of the application and therefore, it is important to have a model capable of dynamically including or excluding variables. In this work, we introduce active CP-nets (aCP-nets) as an extension of CP-nets where variable participation is governed by a set of activation requirements. In particular, we introduce an activation status to the CP-net variables and analyze two possible semantics of aCP-nets along with their consistency requirements.**

***Keywords*—AI; changes; CP-nets; preferences; decision making; product configuration**

I. Introduction

Preferences, as a notion for desires and wishes, plays an important role in any decision making process [1]. Hence, representing and reasoning about preferences is an important task to develop successful applications. Classical decision theory approaches usually assume the existence of a utility function which maps an alternative u to a numerical value $\mu(u)$ that represent the desirability of the decision maker in having u among other alternatives. Another important type of preference representations is the qualitative models where preference statements are expressed in a comparative way i.e., u is more preferred than u' where u and u' are two distinct alternatives. The latter is known to be less demanding in terms of the cognitive effort required from the user [2]. One of the well-studied models for qualitative preferences is the of Conditional Preference Networks (CP-nets) [3]. A CP-net is a fixed model to represent and reason about qualitative conditional preferences. In CP-net, the preference of one attribute may depend on the value of other attributes. Roughly speaking, the CP-net is a directed graph where vertices represent attributes and the edges show the preferential dependencies. Given a set of attributes or variables V, one usually define a CP-net that is fixed and static through out all the process of decision making. In other words, the set of solutions available to the decision maker (DM) are the same through out the process.

Two important questions arise when dealing with CP-nets: i) What is the best alternative given the current preference information? and ii) Given two alternatives which one is better according to the underlying CP-net? The latter is also known as dominance testing, i.e., deciding which alternative dominates the other. Clearly, the best alternative is the one that is not dominated by any other alternative or solution. Solving or answering dominance questions require searching the space of solutions. Needless to say, for a fixed structure like CP-nets, the answer to the above questions is the same through out the decision process. However, one may expect the answer to differ from time to time due to some changes happening in the network.

Also, while having the same answer to both questions is acceptable on some static domains, it is not the case in interactive and configuration problems. In the latter, users are usually interested in different subsets of the variables satisfying certain requirements and hence, the answers need to take into account the changes. It is intuitive to assume the user interest in having one attribute to be part of the solution space is conditioned upon the existence of other attributes.

Consider, for example, a PC configuration website where customized PCs can be manufactured per customer's requirements and preferences. Assume the user is interested in the type of screen only if high performance graphic card was chosen as part of the customized configuration. In all other cases, she is not interested in knowing the screen type as all the market screens are indifferent to her. In this case, it is clear that there is no need to include the screen type preference for all configurations. But only to those where high performance graphic card is included. This is evident in situations where users build their own products or mass customization products. The users are interested in having a mechanism to include or exclude some variables based on some predefined criteria. This differs from situations where the system (or the user) has a vague idea on the preferences. The latter has been tackled by the literature in a probabilistic manner for uncertain preferences in CP-nets [4].

However, to the best of our knowledge, there exists no attempt to augment scenarios of the form *"If a variable X is included (resp. excluded) in the network then include (resp. exclude) variable Y from the search"* or *"If variable X has the value x then include (resp. exclude) variable Y"*. This has the potential to make the preference representation applicable to wide problems where the set of variables change during the decision making process. In this work, we propose an extension to CP-net called active CP-net (aCP-nets) to tackle variables' inclusion and exclusion in the problem. The main idea is to associate every variable with a status (active or inactive) where the status change is governed by a set of inclusion and exclusion constraints. And, at any given time of the decision making process, only those active variables are included in the search.

This paper is organized as follows: Background information is provided in the next section. Section 3 presents related attempts in the literature. In Section 4, we introduce aCP-nets and the participation constraints. Section 5 discusses two different possible semantics of aCP-nets and show how to solve dominance testing in each semantic. Lastly, conclusion and foreseeable work is discussed in Section 6.

II. Conditional Preference Networks (CP-nets)

A CP-net [3] is a graphical model to represent qualitative preference statements including conditional preferences of the form "*I prefer x to x'* " or "*I prefer x to x' when y holds*". A CP-net works by exploiting the notion of preferential dependency based on the *ceteris paribus* (with all other things being without change) assumption. The CP-net is a set of ceteris paribus preference statements which assumed to be valid only when two alternatives differ in exactly one variable value. Graphically, a CP-net can be represented by a directed graph where vertices represent features (or variables) $V = \{V_1, V_2, \dots, V_n\}$ and arcs represent preference dependencies among features. Every variable $X \in V$ is associated with a set of possible values (its domain) D_X. An edge from X to Y means the preference of Y depends on the values of X. In such case we say X is a parent of Y and use Pa(Y) to denote the set of parents for Y. Every variable X is associated with a ceteris paribus table (denoted as $CPT(X)$) expressing the order ranking of different values of X given the values of the parents $Pa(X)$. An outcome for a CP-net is an assignment for each variable from its domain. A variable X is an ancestor of another variable Y if X resides in a path from any root node of the graph to Y and X is descendant of Y if Y is an ancestor of X.

Given a CP-net, the users usually have some queries about the preference statements in the network. One of the main queries is to find the best outcome given the set of preferences. We say an outcome o is better than another outcome o' if there exists a sequence of worsening flips going from o to o' [3]. A worsening flip is a change in the variable value to a less preferred value according to the variable's CPT. The relation between different outcomes for a CP-net can be captured through an induced graph. The graph is constructed as follows: Each node in the induced graph represents an outcome of the network. An edge going from o' to o exists if there is an improving flip according to the CPT of one of the variables in o' all else being equal.

Consider the simple CP-net and its induced graph shown in Fig. 1. The CP-net has three variables A, B and C where A and B are unconditionally prefer a and b to $\bar{a}$ and $\bar{b}$ respectively. However, the preference function for C depends on different values of A and B. For instance when $A = a$ and $B = \bar{b}$, the decision maker prefers $\bar{c}$ to c as value of the variable C. The induced graph represents all the information we need to answer different dominance relations between outcomes. An outcome o dominates another outcome o' if there is a path from o' to o in the induced graph otherwise they are incomparable (denoted as $o \bowtie o'$). An outcome is said to be optimal if there exists no other outcome that dominates it. It is known that for acyclic CP-nets there exists a single optimal outcome that dominates all other solution [3]. For example, the optimal outcome for CP-net in Fig. 1 is abc. Apparently, the size of the induced graph is exponential in the number of attributes. Due to the dependency nature of CP-nets, one needs to consult all the values of $Pa(X)$ before deciding which value is preferred for X. This is mainly because missing a value of any $Y \in Pa(X)$ may lead to inconsistent conclusions, i.e., x being preferred to x' and vice versa at the same time. This turns to be the main property we need to guarantee when including or excluding variables and dependencies.

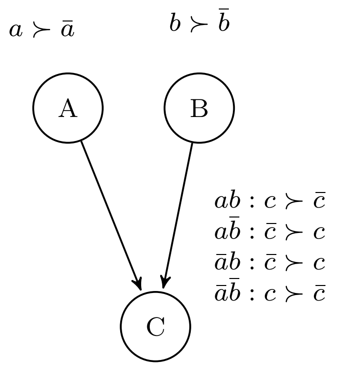
III. Related Work

Since its inception by Boutilier et al. [3], CP-nets have received a considerable attention from the artificial intelligence (AI) community. Many attempts have been proposed tackling different aspects of CP-nets including their semantics [5]–[8], learning [9], [10], and representation [11], [12]. In particular, several works have been made toward extending the semantics and the expressive power of the CP-nets. For instance, the work in [7] extended the CP-net to include preference languages beyond ceteris paribus and thus allow statements to differ in more than more attribute. Another extension of CP-net is the weighted CP-net [8] where the user is able to associate weights to variables. The work in [12] has introduced (conditional) importance over variables. Also, [13] extended the model to augment the notion of comfort when choosing one alternative over another. In [14], [15], the preference-based optimization problem was investigated where hard constraints are assumed to co-exist with a CP-net and the goal is to find a most preferred and feasible solution.

As for extending the semantics of CP-nets to dynamic situations, Bigot et al. [4], [9] studied the case where preferences are uncertain and a probability distribution is associated with a statement. The same problem has been also tackled by [16] where dependencies are associated with probability of existence and every variable X is associated with a distribution over the set of total orders for X's values. The work in [11] considered situations where a webpage content is governed by a CP-net in an adaptive way. Based on the user clicks, the most preferred content is rendered on the page.

However, none of previous attempts has discussed the dynamic aspect of the CP-nets in handling changes that is deterministic and of incremental nature. In particular, the variables' inclusion and exclusion during the search. To this end, there are dynamic models to represent configuration problems similar to the problems tackled in this work. However, they target different knowledge information. One notable representation in this class is the conditional constraint satisfaction problem [17] where constraints and variables are included or excluded during the search. The conditional CSP formalism is limited to constraints and cannot directly applied to qualitative preferences as it is the case in this work.

A closely related area is the preference-based product configuration systems [18]–[22] where a configurator is responsible for customizing the product based on the user preferences. Such configurators allow for a greater flexibility in meeting the users needs and desires. However, we are not aware of


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

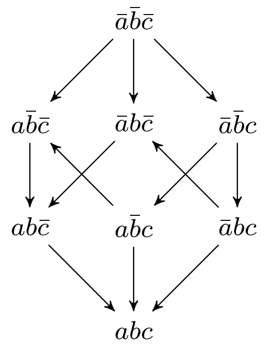


Fig. 1: An acyclic CP-net (left) and its induced preference graph (right).

any previous work targeting conditional qualitative preferences where attributes are included or excluded during the process of configuration.

IV. Active CP-nets (aCP-nets)

In this section we introduce aCP-net extension of CP-nets in the spirit of Conditional CSPs framework [17], [23]. In particular, we add participation conditions upon the CP-net variables that allow variables to be included or excluded during the search. However, unlike conditional CSPs, CP-nets have rich semantics that we need to take into account when reasoning about variables' participation. Specifically, the set of participant variables must be consistent with the preferential dependecies shown in the CP-net structure. Typical CP-net can be viewed as a pair $\langle V, \phi \rangle$ where V is the set of all possible variables in the domain and ϕ is the set of CPTs. Now we define our aCP-net framework.

Definition 1 (aCP-net)**.** *aCP-net is a tuple $\langle V, V_I, \hat{C}, \phi \rangle$ where V is the set of all variables s.t. each variable is associated with an activation status. V_I is the set of initial variables $V_I \subseteq V$ and $\hat{C}$ is the set of activity constraints and ϕ is the set of CPTs.*

Given a variable $X \in V$, we denote the status of X as STATUS(X) . Each variable can be in exactly one state either ACTIVE or INACTIVE at any given time. The set of participation requirements $\hat{C}$ describes different possible changes over the structure while V_I is the set of always active variables that cannot be removed from the domain. V_I can be viewed as the core variables that minimally describe any configuration and thus it is the default preference network for the system. When reasoning about aCP-net, only active variables are taken into account. Given aCP-net instance α, the two main operations over α are including and excluding variables. We describe variables inclusion in terms of Require Variable (RV) and Always Required Variable (ARV) conditions. Similarly, excluding variables from α is done through the conditions of Require Not (RN) and Always Require Not (ARN).

A. Participation Requirements

In this section, we discuss different possible participation conditions over the aCP-net structure. These conditions are Required Variable (RV) and Always Required Variable (ARV) to include variables and Require Not (RN) and Always Require Not (ARN) to exclude variables. Such participation requirements have been proven to be useful in many configuration problems and in particular in the framework of conditional CSP [17]. Generally speaking, all conditions have the following form:

```
condition::result
```

where `condition`$\subset V$ represents the condition variables and `result`$\in V - V_I$ represents a variable and $:: \in \{\xrightarrow{incl}, \xrightarrow{excl}\}$ represents the type of condition (exclusion or inclusion). Whenever `condition` becomes true, `result` is executed and a variable is either included or excluded from the problem domain.

Note that aCP-nets is a generalisation of CP-nets in the sense that the set of conditional dependencies can be preserved along with CPTs in aCP-nets under the special case where $V_I = V$. In such case, the aCP-net will have only one possible instance.

Example 1. *Consider CP-net in Fig. 2. It is easy to see that we can represent the original CP-net as aCP-net instance where $V = \{A, B, C, D, E, F\}$, $V_I = V$ and $\hat{C} = \emptyset$.*

1) Including Variables: One of the simplest yet effective participation requirements to include variables is the required variable (RV) condition. RV has the form $A_1 = a_1 \wedge A_2 = a_2 \wedge .... \wedge A_n = a_n \xrightarrow{incl} X$ where $A_i \in V$ and $X \in (V - V_I)$. This simply says X will be activated (i.e., included) into the problem if every $A_i = a_i$ becomes true.

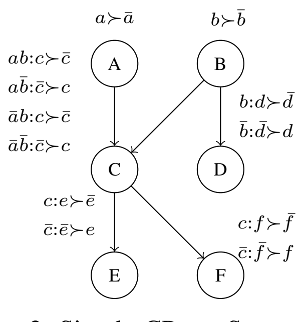

$a{\succ}\bar{a}$
$b{\succ}\bar{b}$
$ab{:}c{\succ}\bar{c}$
$a\bar{b}{:}\bar{c}{\succ}c$
$\bar{a}b{:}c{\succ}\bar{c}$
$\bar{a}\bar{b}{:}\bar{c}{\succ}c$
A
B
$b{:}d{\succ}\bar{d}$
$\bar{b}{:}\bar{d}{\succ}d$
C
D
$c{:}e{\succ}\bar{e}$
$\bar{c}{:}\bar{e}{\succ}e$
$c{:}f{\succ}\bar{f}$
$\bar{c}{:}\bar{f}{\succ}f$
E
F

Fig. 2: Simple CP-net Structure

Similarly, Always Required Variable (ARV) is used to include a variable X when subset of variables $A_1, ..., A_n \in V$ are active regardless of their values and has the form of $A_1 \wedge ... \wedge A_n \xrightarrow{incl} X$.

2) Excluding Variables: Intuitively, Required Not (RN) requirement asserts the exclusion of variable based on other variables values. RN has the form $A_1 = a_1 \wedge A_2 = a_2 \wedge ... \wedge A_n = a_n \xrightarrow{excl} X$ where $A_i \in V$ and $X \in (V - V_I)$. Similar to RN, ARN has the form of $A_1 \wedge ... \wedge A_n \xrightarrow{excl} X$.

V. Semantics of aCP-nets

So far we have described the conditions under which variables may be included or excluded from the network domain without relating them to the underlying aCP-net structure and semantics. Arbitrary changes might lead to violating the semantics of CP-nets. For instance, assume removing one of the parents of a variable X. How CPT(X) should be updated for such changes? or consider including a variable X where one of its parents is not active, how the aCP-net should behave in such circumstances? Therefore, in this work, we study different possible changes and define conservative and open rules for applying different changes into the aCP-net structure. The goal of conservative rules is to represent a valid CP-net (defined below) at anytime of the aCP-net process. On the other hand, the open semantics aim to represent most general case where the resulted instance of aCP-net are not necessarily a semantically correct CP-net.

A. Conservative Semantics

The core concept here is that the changes must result in a valid CP-net at any given time of the solution process.

Definition 2 (Valid CP-net)**.** *Given a set of variables R and their corresponding CPTs ψ, $\langle R, \psi \rangle$ represents a valid CP-net iff for any variable $X \in R$, $Pa(X)$ also exists in R.*

Example 2. *Consider $R = \{A, C, E, F\}$ and ψ is their CPTs for the CP-net in Fig. 2, here R does not represents valid CP-net since the variable $B \in Pa(C)$ is not in R.*

In the conservative semantics of aCP-nets the changes will always result in a valid CP-net. To reflect the conditional dependencies in the structure of CP-net, we assert the activation of a variable based on its parents activation. That is for any variable X with set of parents $Pa(X) \subset V$, for any parent variable $I \in Pa(X)$, either `STATUS(`I`)==ACTIVE` or there exists $c \in \hat{C}$ where I will be activated.

Definition 3 (Consistent Inclusion)**.** *An aCP-net has the consistent inclusion property if for any $c \in \hat{C}$ whenever a variable $X \in V$ to be included, $Pa(X)$ is also included in the domain.*

In the context of aCP-net, we need to be careful in excluding variables. The excluded variable X may be either a leaf node (thus there are no other variables depend on it) or not a leaf node in the aCP-net structure. In the first case, we can safely remove X since we are guaranteed that there will be no other variable S where $X \in Pa(S)$ that might be activated later. In the second case, we can use a procedure to look for whether any of X's descendants will be activated in $\hat{C}$.

Definition 4 (Consistent Exclusion)**.** *An aCP-net has the consistent exclusion property if for any $c \in \hat{C}$ whenever a variable $X \in V$ to be excluded, X has no descendants or for any variable $Y \in V$ of X's descendants, there is no $c \in \hat{C}$such that Y will be activated.*

Definition 5 (Consistent aCP-net)**.** *aCP-net is consistent if it satisfies the consistent inclusion and exclusion properties.*

The goal of conservative rules is to reflect precisely different valid CP-nets from the original CP-net $\langle V, \phi \rangle$ without violating its semantics and dependencies. This is formally proved by the following lemma:

Lemma 1. *If $\mathcal{A}$ is a consistent aCP-net then the set of variables available at any given time of the search represents a valid CP-net.*

Proof: The proof is by contradiction. Assume $\mathcal{A}$ to be a consistent aCP-net but the set of variables available at time t form a CP-net that is not valid. By definition, this means there exists at least one variable X where $Y \in Pa(X)$ was not included at time t but X was included. First assume $X \in V_I$, this is impossible as V_I is available at any given time, and for any X, the set of parents must be part of the initial variables as well. Second, assume $X \notin V_I$, then there must exist at least one participation constraint $c \in \hat{C}$ that result in including X. Given that $Y \in Pa(X)$ was not included at time t, then the inclusion was not consistent and thus the aCP-net does not have the consistent inclusion property which contradicts with our assumption of $\mathcal{A}$ being a consistent aCP-net. ■

Algorithm 1: Consistency Test for aCP-nets

```
input : ⟨$V, V_I, \hat{C}, \phi$⟩: aCP-net Structure
output: True or False
1  foreach $c \in \hat{C}$ do
2      Let $X$ = result in $c$
3      if $c$ is inclusion condition then
4          foreach $Y \in Pa(X)$ do
5              if STATUS(Y)==INACTIVE then
6                  Return False
7              end
8          end
9      end
10     else
11         foreach $P \in$ Descendants($X$) do
12             if STATUS(P)==ACTIVE then
13                 Return False
14             end
15         end
16     end
17 end
18 Return True
```

Although this might seem too restrictive conditions, it may apply in different domains where the changes are known a priori and the dependencies between variables cannot be changed. We describe a procedure in Algorithm 1 to check whether a given aCP-net is consistent or not. The complexity of the algorithm is $O(n|\hat{C}|)$ where $n = |V|$ is the number of variables and $|\hat{C}|$ is the number of participation requirements assuming the parent size for any variable X is bounded by a constant.

1) Reasoning with aCP-net in Conservative Semantics: In the previous section, we have listed some conservative rules for aCP-nets to follow in order to be consistent with the information provided in V and ϕ. These consistency conditions are based on the valid CP-net definition and the set of activation requirements $\hat{C}$. It is clear that the order of conditions listed in $\hat{C}$ plays an important rule in verifying the consistency of a given aCP-net structure. Thus, we order $\hat{C}$ in the following way. First all inclusion conditions are sorted before exclusion ones. For the set of inclusion conditions, we order them from top to bottom according to V. For example for two conditions $c_1, c_2 \in C$ where c_1 and c_2 assert including A and B respectively such that $B \in Pa(A)$ then c_2 is ordered before c_1. The exclusion conditions are ordered in the opposite way from bottom to top.

Example 3. *Consider aCP-net structure θ= $\langle V, V_I, \hat{C}, \phi\rangle$ where V and ϕ represent the set of variables and their CPTs in Fig. 2, respectively. Let $V_I = \{A, B, D\}$ and $\hat{C}$ contains the following set of activity conditions: $a_1 : A = a \xrightarrow{incl} C$, $a_2 : B = \bar{b} \xrightarrow{incl} E$, $a_3 : C \xrightarrow{incl} F$. Fig. 3 shows the CP-net instances resulted from θ when executing a_1, a_2 and a_3 respectively. Observe that θ is a consistent aCP-net. Now, assume we add the activity constraint $a_4 : A = \bar{a} \xrightarrow{excl} C$, θ will not be consistent anymore as E is already activated. This will not be the case if for example there was an exclusion constraint for E.*

a) Answering aCP-net Queries:: Given the activation requirements, we are interested in answering different queries related to the underlying aCP-net structure. The key observation here is that any instance of aCP-net is a valid CP-net representation. In other words, it is a sub-network of the original CP-net. Therefore, for any aCP-net, the semantics of CP-net are directly inherited and utilized. For instance, finding the best outcome for a particular aCP-net instance is done through the sweep-forwarding procedure presented in [3]. Given two outcomes α and β, deciding which one is better can be classified into two cases: (i) α and β are derived from the same network. (ii) They are derived from different networks. In the former, it is clear that we can adopt the CP-net semantics to answer such query. In particular, assume α and β are derived from the network π , we can answer the dominance task by finding a sequence of flips from one outcome to another in π.

However, aCP-net structures usually contain outcomes with different domain spaces. Here, we need to have a mechanism under which we can conclude whether a given outcome is better than another. In other words, we need to find out a new *dominance* relation over aCP-net structure in case outcomes α and β were derived from different networks.

Example 4. *Consider the aCP-net structure θ= $\langle V, V_I, \hat{C}, \phi\rangle$ where V and ϕ represent the set of variables and their CPTs in Fig. 2, respectively. Let $V_I = \{A, B, D\}$ and $\hat{C}$ contains the following set of activity conditions: $a_1 : A = a \xrightarrow{incl} C$, $a_2 : B = \bar{b} \xrightarrow{incl} E$, $a_3 : C \xrightarrow{incl} F$ and $a_4 : D = \bar{d} \xrightarrow{excl} C$. Fig. 4 shows the complete search space for this example with classic baktracking search algorithm. Here, θ has different solutions with different domain spaces. For instance, solutions $ab\bar{d}$ and $abdcf$ are derived from different networks. The crossed out paths represent inconsistent aCP-net instances. For example, assignment $a\bar{b}\bar{d}$ is inconsistent since executing a_4 will lead to removing the variable C while E is included.*

2) Dominance Testing: In this section we provide a method to answer dominance queries when outcomes have different domain spaces. We do so by utilizing the original network structure Ω where $\Omega = \langle V, \phi\rangle$ for a given aCP-net structure θ=$\langle V, V_I, \hat{C}, \phi\rangle$. Let α be any outcome in θ. We know that α might be delivered from any valid CP-net π contained in Ω. This is due to the fact that consistent aCP-net always result in valid CP-nets. Thus, we have $\pi \subseteq \Omega$ for any outcome defined over network π in θ. Our method works as follow. Given two outcomes α and β over θ, if $D_\alpha \neq D_\beta$, we extent them to outcomes $\bar{\alpha}$ and $\bar{\beta}$ over Ω and then do dominance test over the new extended outcomes. Assume $\varphi = V_\Omega - V_\alpha$ to be the set of variables not listed in α. We go top to bottom assigning each variable $X \in \varphi$ to $X = x_i$ such that x_i is the least preferred value given $Pa(X)$.

Example 5. *Consider the aCP-net structure in Example 4 with the following two solutions $\alpha = ab\bar{d}$ and $\beta = abdcf$, it is clear that both Ω_α and Ω_β are subset of the original network Ω, thus we extend α and β to solutions over Ω and then do the dominance testing task. In particular, we have $\bar{\alpha} = ab\bar{d}\bar{c}fe$ and $\bar{\beta} = abdcf\bar{e}$ and $\bar{\beta} \succ_\Omega \bar{\alpha}$.*

B. Open Semantics

So far we have discussed a restrictive but useful semantics for changes over CP-net. The result of the conservative semantics is the guarantee of valid CP-nets during the decision process. This means that whenever we include a variable X, the set of parents are already included (i.e. active). Analogy, when removing a variable Y, we assert that Y has no other variables depend on it in the given aCP-net structure or its participation constraints. Particularly, we assumed a large and original CP-net exist in advance. From the imposed consistency conditions, we implicitly handled the inclusion and exclusion of variables.

However, it could be the case where the user is interested in changing the structure and semantics of the original network. In such cases, we need to distinguish between including/excluding variables through the participation requirements and adding/removing variables and dependencies to the domain. The latter will result in changing original network $\langle V, \phi\rangle$ of the aCP-net structure. In such cases we can relax the consistency conditions posed by the conservative semantics to a more flexible ones. In particular, we consider changes correspond to adding and removing variables and dependencies.

1) Adding Variables or Dependencies: Adding a variable X to the aCP-net structure can be done through two steps. First, we add X to V. Second, we pose an ARV condition in the form $\emptyset \xrightarrow{incl} X$. This will result in X included in the problem. This way, it is possible to add a completely new

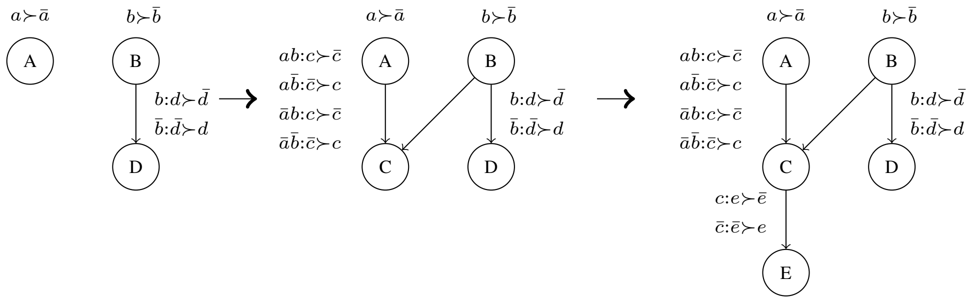

$a \succ \bar{a}$
$b \succ \bar{b}$
A
B
$b{:}d \succ \bar{d}$
$\bar{b}{:}\bar{d} \succ d$
D
$ab{:}c \succ \bar{c}$
$a\bar{b}{:}\bar{c} \succ c$
$\bar{a}b{:}c \succ \bar{c}$
$\bar{a}\bar{b}{:}\bar{c} \succ c$
C
$c{:}e \succ \bar{e}$
$\bar{c}{:}\bar{e} \succ e$
E

Fig. 3: Different CP-net instances resulted from the aCP-net in Example 3.

variable to the domain and then reason about its most preferred value.

To add dependency between X and Y where $X, Y \in V$. The only condition here is that the new dependency will not lead to cycles in the aCP-net structure. We refer to $I(X, Y)$ as a dependency of Y on the values of X (i.e., for different $x \in X$ we have an order over D_Y). Adding dependency $I(X, Y)$ will result in updating the CPT(Y)$\in \phi$ in a way that $Pa(Y) = Pa(Y) \cup X$ and for each unique assignments of the parents, we have an order over D_Y.

2) Removing Dependencies and Variables: Before removing a dependency or variable, we first introduce the process of marginalization of a variable $X \in Pa(Y)$ in CPT(Y).

Definition 6 (Marginalization). *Given a CPT ℓ for a variable Y , marginalising $X \in Pa(Y)$ over ℓ (denoted as $\ell^{\downarrow X}$) is a new CPT λ where D_λ=$D_{\ell - X}$ and for any value $x \in D_X$, x has been removed from λ.*

After marginalising X over CPT(Y) ($CPT(Y)^{\downarrow X}$) it might be the case where we have the same assignment of the parents with different orders over D_Y. Thus, we next provide a definition for valid CPTs in the aCP-net structure.

Definition 7 (Valid CPT). *CPT(X) is a valid CPT iff for each assignment γ of the parents, we have the same ordering over D_X.*

For instance, consider the CP-net in Fig. 1, assume we are interested in removing the dependency between B and C ($I(B, C)$). First we marginalise B over $CPT(C)$. The resulted CPT is not valid since for $A = a$ we have two different orders.

Lastly, in order to remove a variable X from the domain, we first need to remove the set of dependencies hold between X and its immediate descendants (i.e., children) and then we can safely remove X.

3) Posing Queries: Consider removing $I(B, C)$ from the CP-net structure in Fig. 1, CPT(C) will have the following statements: $a : c \succ \bar{c}$, $a : \bar{c} \succ c$, $\bar{a} : c \succ \bar{c}$ and $\bar{a} : \bar{c} \succ c$. Obviously, these statements contradict with each other and breaks the intuitive meaning of CP-net of having exactly one order for the same assignment of the parents. How the CPT(C) should be updated in such cases? We can overcome such contradictions by revising the order of the variable. One way to do so is to engage the user by asking different questions. In this particular example, we can ask the user whether she prefers abc to $ab\bar{c}$ in order to know the order when $A = a$. In particular, if $abc \succ ab\bar{c}$ then the CPT(C) is updated to $c \succ \bar{c}$ for $A = a$. The same goes for $A = \bar{a}$ with the query whether $\bar{a}bc$ is preferred to $\bar{a}b\bar{c}$. Such queries hold the promise of revising invalid CPTs and make them valid during the process of decision making.

VI. Conclusion and Future Work

This paper presented *aCP-net*, an extension for CP-nets to include and exclude variables during the search. We listed some consistency conditions under which the resulted changes always form valid CP-nets and, thus will preserve the semantics of CP-nets. We have also analyzed the situation of changes leading to inconsistencies in the preference information and suggested possible techniques to overcome the inconsistency and answer the dominance testing.

Foreseeable work include defining relaxed conditions to allow arbitrary changes over variables and dependencies. Another important future work is to learn the participation requirements from historical interactions with the system. This holds the promise of lowering the burden of specifying participating requirements by the end users.

References

[1] T. Walsh, "Representing and reasoning with preferences," *AI Magazine*, vol. 28, no. 4, pp. 59–70, 2007.

[2] C. Domshlak, R. I. Brafman, and S. E. Shimony, "Preference-based configuration of web page content," in *IJCAI*, 2001, pp. 1451–1456.

[3] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements," *J. Artif. Intell. Res. (JAIR)*, vol. 21, pp. 135–191, 2004.

[4] D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini, "Probabilistic conditional preference networks," in *Proc. 29th Conference on Uncertainty in Artificial Intelligence (UAI 2013)*, 2013.

[5] R. I. Brafman and Y. Dimopoulos, "A new look at the semantics and optimization methods of cp-networks," in *IJCAI*, 2003, pp. 1033–1038.

[6] C. Boutilier, F. Bacchus, and R. I. Brafman, "Ucp-networks: A directed graphical representation of conditional utilities," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 56–64.

[7] N. Wilson, "Extending cp-nets with stronger conditional preference statements," in *AAAI*, vol. 4, 2004, pp. 735–741.

[8] H. Wang, J. Zhang, W. Sun, H. Song, G. Guo, and X. Zhou, "Wcp-nets: a weighted extension to cp-nets for web service selection," in *International Conference on Service-Oriented Computing*. Springer, 2012, pp. 298–312.

[9] D. Bigot, J. Mengin, and B. Zanuttini, "Learning probabilistic cp-nets from observations of optimal items." in *STAIRS*, 2014, pp. 81–90.

[10] F. Koriche and B. Zanuttini, "Learning conditional preference networks," *Artificial Intelligence*, vol. 174, no. 11, pp. 685–703, 2010.

[11] R. I. Brafman, "Adaptive rich media presentations via preference-based constrained optimization," in *Proceedings of the IJCAI-05 Workshop on Advances in Preference Handling*, 2005.

[12] R. I. Brafman, C. Domshlak, and S. E. Shimony, "On graphical modeling of preference and importance," *J. Artif. Intell. Res. (JAIR)*, vol. 25, pp. 389–424, 2006.

[13] S. Ahmed and M. Mouhoub, "Extending conditional preference network with user's genuine decisions," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2018, pp. 4216–4223.

[14] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "Preference-based constrained optimization with cp-nets," *Computational Intelligence*, vol. 20, no. 2, pp. 137–157, 2004.

[15] E. Alanazi and M. Mouhoub, "Variable ordering and constraint propagation for constrained cp-nets," *Applied Intelligence*, vol. 44, no. 2, pp. 437–448, 2016.

[16] C. Cornelio, J. Goldsmith, N. Mattei, F. Rossi, and K. B. Venable, "Updates and uncertainty in cp-nets," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2013, pp. 301–312.

[17] S. Mittal and B. Falkenhainer, "Dynamic constraint satisfaction problems," in *AAAI*, 1990, pp. 25–32.

[18] Y. Wang, D. Y. Mo, and M. M. Tseng, "Relative preference-based product configurator design," *Procedia CIRP*, vol. 83, pp. 575–578, 2019.

[19] H. L. Jakubovski Filho, T. N. Ferreira, and S. R. Vergilio, "Preference based multi-objective algorithms applied to the variability testing of software product lines," *Journal of Systems and Software*, vol. 151, pp. 194–209, 2019.

[20] P. Zheng, Z. Sang, R. Y. Zhong, Y. Liu, C. Liu, K. Mubarok, S. Yu, X. Xu *et al.*, "Smart manufacturing systems for industry 4.0: Conceptual framework, scenarios, and future perspectives," *Frontiers of Mechanical Engineering*, vol. 13, no. 2, pp. 137–150, 2018.

[21] S. Shafiee, A. Felfernig, L. Hvam, P. Piroozfar, and C. Forza, "Cost benefit analysis in product configuration systems," in *Configuration Workshop 2018 (ConfWS 2018)*. CEUR-WS, 2018.

[22] E. Gençay, P. Schüller, and E. Erdem, "Applications of non-monotonic reasoning to automotive product configuration using answer set programming," *Journal of Intelligent Manufacturing*, vol. 30, no. 3, pp. 1407–1422, 2019.

[23] M. Sabin, E. C. Freuder, and R. J. Wallace, "Greater efficiency for conditional constraint satisfaction," in *CP*, 2003, pp. 649–663.

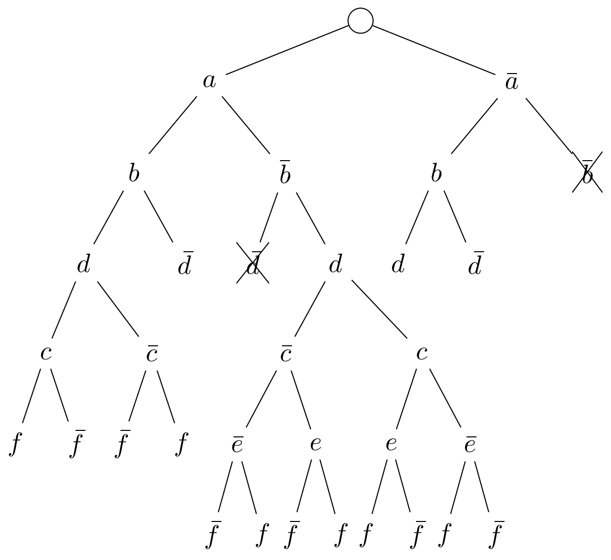


Fig. 4: The complete search space for Example 4