Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference

A Framework to Manage Conditional Constraints and Qualitative Preferences

Eisa Alanazi [*] and **Malek Mouhoub**
Department of Computer Science
University of Regina
{alanazie,mouhoubm}@cs.uregina.ca

Abstract

A conditional CSP is a well known formalism for managing constraints in a dynamic environment where the possible changes are known a priori and can be enumerated beforehand. In this paper we argue that mixing preferences with Conditional CSPs brings more intuitive meaning to many real world applications. For example, in a product configuration problem the user preferences often co-exist with the configuration requirements in a dynamic environment. We propose here an extension of the Conditional CSP formalism in order to consider qualitative preferences. We use CP-nets as a representation for these preferences and define new concepts for preserving their semantics in dynamic settings.

Introduction

A Constraint Satisfaction Problem (CSP) (Dechter 2003) is a powerful formalism to represent constraints problems. A CSP instance consists of a set of variables along with their possible values and set of variables relations (constraints) stating which tuples are allowed. Given a CSP instance, different search techniques can be applied in order to find complete assignments satisfying all the constraints (Mackworth 1977; Dechter 2003). However, the CSP is limited when it comes to the dynamic nature of many tasks (Sabin and Freuder 1996; Mittal and Falkenhainer 1990). This is due to the fact that all CSP variables have to be part of any consistent solution and all the constraints of the initial problem have to be taken into account. This is however not the case in many real world applications (Mittal and Falkenhainer 1990).

There have been some work to extend the typical CSP formalism to naturally represent these dynamic constraint problems (Sabin and Freuder 1996; Mouhoub and Sukpan 2012; Mittal and Falkenhainer 1990; Mouhoub and Sukpan 2005; Mouhoub 2003). One extension of our interest is the Conditional Constraints Satisfaction Problem (CondCSP)(Mittal and Falkenhainer 1990) to represent synthesis tasks such as product configurations. A CondCSP represents the dynamic aspect of configuration through *activity constraints* which govern the activity of subset of variables to be part of the final solution via inclusion and exclusion constraints (Mittal and Falkenhainer 1990; Gelle and Sabin 2003).

Preferences, on the other hand, play a major role in many domains (Boutilier et al. 2004). In particular, in interactive product configuration problems where personalization is a key factor, users usually express their preferences and expect to have configurations satisfying their desires. One intuitive preference representation is the Conditional Preference networks (CP-nets) (Boutilier et al. 2004). A CP-net is a graphical model to represent qualitative preference statements including conditional preferences such as: "*I prefer A to B when X holds*". CP-nets have the same all-variables required limitation found in the standard CSPs. Each CP-net solution must include each variable in the network. In this work we identify sufficient conditions to reflect the dynamic aspect to the CP-nets and propose a framework to handle both qualitative preferences and conditional constraints simultaneously.

The paper is structured as follows. The next two sections provide the background information and related work respectively. Our framework is then presented in section four. In section five, we address the problem of finding optimal configurations. Finally, concluding remarks and future directions are presented in section six.

Background

Conditional Preference Networks

A Conditional Preferences network (CP-net) (Boutilier et al. 2004) is a graphical model to represent qualitative preference statements including conditional preferences such as: "*I prefer A to B when X holds*". A CP-net works by exploiting the notion of preferential independency based on the *ceteris paribus* (with all other things being without change) assumption. A CP-net can be represented by a directed graph where nodes represent features (or variables) along with their possible values (variables domains) and arcs represent preference independencies among features. Each variable X is associated with a ceteris paribus table (denoted as $CPT(X)$) expressing the order ranking over different values of X given the set of parents $Pa(X)$. An outcome for a CP-net is an assignment for each variable from its domain. Given a CP-net, the users usually have some queries about the set of preferences represented. One of the main queries

[*]The author is supported by the Saudi Arabian Cultural Bureau, Ottawa, Ontario Canada K1G 6C9

is to find the best outcome given the set of preferences. Another important query is to check whether one outcome is better than another. We say outcome o_i is better than outcome o_j if there is a sequence of worsening flips going from o_i to o_j (Boutilier et al. 2004). A Worsening flip is a change in the variable value to a less preferred value according to the variable's CPT.

Conditional Constraint Satisfaction Problems

A Conditional Constraint Satisfaction Problem (CondCSP) (Mittal and Falkenhainer 1990) is a formalism to handle constraint problems where a set of variables can be added during the search space. Unlike typical CSPs, CondCSPs have a set of optional variables that can participate in the final solution if they satisfy a particular type of constraints called activity constraints. Therefore, each variable in a CondCSP is associated with an activation status (i.e. active or not active). The set of activity constraints governs the participation of optional variables in the solutions.

More formally, a CondCSP is represented as a tuple $< V, V_I, D, C_C, C_A >$ where V is the set of all variables, V_I is the set of initial variables (i.e. variables that are initially active), D is the set of possible values for each variable (its domain) and C_C and C_A represent the set of compatibility and activity constraints respectively. A compatibility constraint is typically a hard constraint in CSPs (Gelle and Sabin 2003). Activity constraints have an inclusion or exclusion conditions in the form of $A \rightarrow B$ where A is a variable and B is an optional variable (belonging to $V - V_I$) to include or exclude and A can optionally have a value. An inclusion constraint $A = a \xrightarrow{incl} B$ means that B will be activated (i.e. included) iff a is assigned to A. Similarly, the exclusion constraint has the form $A = a \xrightarrow{excl} B$ where the value a is optional and B is excluded from the search space whenever the condition becomes true.

Related Work

Different methods have been proposed in the literature to solve CondCSPs (Mittal and Falkenhainer 1990; Gelle 2003; Gelle and Sabin 2003; Sabin, Freuder, and Wallace 2003). They differ by either solving CondCSPs directly or formulating the CondCSPs to standard CSPs. In (Mittal and Falkenhainer 1990) CondCSPs have been presented for the first time under the name of Dynamic CSPs. The authors based their discussion on the configuration problems and how it can be represented as CondCSPs. They proposed a backtracking algorithm for directly solving CondCSPs. They also discussed how to formulate the CondCSPs to standard CSPs by introducing special null values. (Gelle 2003) introduced an algorithm to formulate CondCSPs to standard CSPs for the problems containing discrete and continuous variables. The authors discussed how to prune the search space by adapting local consistency techniques. (Sabin, Freuder, and Wallace 2003) provides more general null-based formulation for CondCSPs problems. They also proposed new algorithms to maintain arc consistency and forward checking techniques in CondCSPs.

All the previous work except (Mouhoub and Sukpan 2008) and (Mouhoub and Sukpan 2012) have neglected the notion of preferences when looking for CondCSPs solutions. In (Mouhoub and Sukpan 2012) a Conditional and Composite CSP (CCCSP) framework has been extended to handle four types of quantitative preferences. Similarly, preferences in the context of temporal constraints have been proposed in (Mouhoub and Sukpan 2008). However, to our best knowledge, the co-existence of qualitative preferences and CondCSPs have not been discussed in the literature.

Extending CondCSPs with Preferences

Framework Definition

In this section we define our Preference Conditional Constraints Satisfaction Problem (PCCSP) model which extends the CondCSP with the CP-net as follows. A PCCSP is a tuple $< V, V_I, C_C, C_A, N >$ where:
- $V = \{v_1, ...v_n\}$ is the set of all variables, each defined on a domain D_{v_i} of discrete values.
- V_I is the set of initial variables. $V_I \subseteq V$
- C_C is the set of compatibility constraints.
- C_A is the set of activity constraints.
- N is the set of preference variables. $N \subseteq V$ and each $v_i \in N$ has a CPT(v_i).

It has been shown that the exclusion constraints can always be mapped to compatibility constraints (Gelle and Sabin 2003). Therefore, we consider only inclusion constraints in this work.

Handling Preferences in a PCCSP

When considering the PCCSP, the following main questions arise. First, how to check the consistency of the preference statements in N? In other words, how to identify the set of feasible statements according to the problem constraints (C_C) in the framework? Second, how should N behave towards the dynamic aspect of the PCCSP?. It is known that the CP-net is a fixed preferences representation in a sense that it is not possible to add new variables during the searching process.

For the first issue, we can adopt local consistency techniques (Mackworth 1977) to prune some inconsistent statements as described in (Alanazi and Mouhoub 2012). In the latter, Arc Consistency is applied to prune some preference statements and produce what so called Arc Consistent CP-nets. The dynamic aspect of CondCSPs requires a closer attention in order to use CP-net as qualitative preference representation in our PCCSP framework. For instance, assume N has been consulted for determining the most preferred value to variable $v_i \in N$. It is possible that one of the v_i parents is not active (i.e. no instantiation). How the CP-net is supposed to behave in this case? For this issue we define a new concept called relevance for CP-nets. When solving the PCCSP only active variables are considered. This requires revising the CP-net N semantics to reflect the dynamic settings of the framework. Therefore, we adopt the relevant notion to the variables in N in order to tell whether the $CPT(X)$ should be consulted or not for a variable $X \in N$. In order to be successful in using the $CPT(X)$, we need to maintain the conditional independencies information. In other words, the set of parents for X ($Pa(X)$) must be active as well.

Definition 1 (Relevant Variable). *A variable $v_i \in N$ is relevant to the PCCSP iff v_i is active and for every $p_i \in Pa(v_i)$ the activity of p_i holds.*

In order to have complete specifications for different preference statements, we need to maintain not only the variables but also their dependencies (i.e. parents). Following the definition of CP-nets, this results in a valid CP-net instance. Therefore, we generalize the relevance notion and define it to the level of CP-nets.

Definition 2 (Relevant CP-net). *A CP-net CP is relevant to a given PCCSP iff each variable in CP is relevant.*

Given a CP-net N associated with a PCCSP P, we can classify N existence into two cases: (i) N exists only over initial variables (i.e. $N \subseteq V_I$).(ii) N exists, possibly partially, over non initial variables (i.e. $N \cap (V - V_I) \neq \emptyset$). The former case is a trivial one and can be solved by typical preference constrained optimization methods. Therefore, in the following we consider only the case where preference information exist over non initial variables.

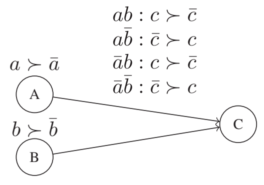
Existence over non-initial variables

In this case, a set of conditions must be defined in order to reason about the CP-net N correctly. For instance, assume $Y \subseteq N$ is a valid CP-net and has been consulted to determine the most preferred value for a variable $v_i \in Y$. It is possible that one of the v_i parents is not active (i.e. not instantiated). The definition of relevant CP-net guarantees that Y will be consulted only if all parents of v_i are active. Due to the dynamic behavior of the PCCSP, different solutions will have different arities. However, in CP-nets, the set of outcomes always have the same arity since, at any time, outcomes are defined over the same space. In our framework, preference outcomes are defined based on the relevant CP-net notion. Thus, we have different outcomes with different arities. In the following we define the specificity level (i.e. number of variables). Given two assignments $a_1 = (x_1, ..., x_n)$ and $a_2 = (y_1, ..., y_m)$ we say a_1 has specificity level n and a_2 has specificity level m where $m, n > 0$. Also, we say a_1 is more specific than a_2 iff $n > m$.

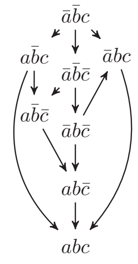
Definition 3. (Specificity level) *Given an assignment a_i for PCCSP, we say a_i is specific at level k (denoted as $sl(a_i) = k$) iff the relevant CP-net associated with a_i has k variables ($k = |N_i|$).*

Finding Optimal Outcomes in PCCSP

We adopt the longest chain of improving flips as a measurment for how close an outcome is to the optimal. The longest chain for an outcome o (denoted as $lc(o)$) to the optimal outcome opt is the maximum number of flips o requires to reach opt. For instance, the longest chain of flips for $d\bar{e}a$ to the optimal in Figure 1b is the chain $a\bar{b}c \to a\bar{b}\bar{c} \to ab\bar{c} \to abc$ with 3 moves. Generally, the longest chain shows us, in a pessimistic view, how close one outcome is to another. Beside closeness we are interested in answers (configurations) that reflect most of the preferences in N. One way


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

(a) The network



(b) The graph

Figure 1: Preference networks and its induced graph

to express this interest is through the specificity level. As a result, a new concept is defined for dominance testing which is a combination of both the specificity level and the longest chain concepts named *relative closeness (rc)*. The relative closeness for an outcome o with regard to another outcome d (denoted as $rc(o, d)$ or $rc(o)$ if d is obvious) is equal to the following: $rc(o, d) = lc(o) + (sl(d) - sl(o))$. As a result, we say o_i dominates o_j if the longest chain of o_i in addition to the difference in o_i specificity level is less than its analogue in o_j. That is $N \models o_i \succ_{rc} o_j$ iff $lc(o_i) + (sl(opt) - sl(o_i)) < lc(o_j) + (sl(opt) - sl(o_j))$.

Definition 4 (rc-Dominance). *An outcome o_i rc-dominates another outcome o_j according to N ($N \models o_i \succ_{rc} o_j$) iff $rc(o_i) < rc(o_j)$.*

The definition of rc-dominance poses no restrictions over the outcomes. In other words, it does not require the outcomes to be delivered from the same relevant CP-net nor having the same specificity levels. If two outcomes have the same relative closeness degree then they are incomparable according to the definition of rc-Dominance. The incomparability happens when neither the longest chain nor the specificity level favor one outcome over the other. Also, it is clear that the optimal outcome for N has relative closeness equal to zero. This is obvious since the longest chain for the optimal to itself equals zero and the optimal has specificity level of N ($sl(opt) = |N|$).

Example 1 Let us consider the PCCSP P such that P=($\{A, B, C, D, E\}$,$\{D, E\}$,$\{\}$,$\{a_1, a_2, a_3\}$,N) where $a_1 : \bar{d} \to B$, $a_2 : \bar{e} \to A$ and $a_3 : \bar{a}\bar{d} \to C$. Also, let $N = \{A, B, C\}$ as in Figure 1. Solving P requires finding a consistent assignment for $\{D, E\}$ with the best feasible preference values from N. Consider the topological order to be $\{D, E\}$. First, we are required to determine the value of D. When consulting N, we get the assignment de which forms the solution $s_1 = de$. Proceeding with the search we

```
input:
$P$ PCCSP $< V, V_I, C_C, C_A, N >$
output:
$S = \{\}$ set of Pareto optimals.
1. Run the AC algorithm to reduce the search space.
2. If the problem is inconsistent
     return $S$
3. Let $R \leftarrow V_I$
   Let $n \leftarrow \{\}$
   while $R \neq \emptyset$
     choose next variable $v \in R$
     Let $N_{n \cup v}$ be the preference network
     Let $v_i$ be the preferred value given $Pa(v)$
     $v = v_i$
     if any $a \in C_A$ becomes relevant
       $R \leftarrow R \cup result(a)$
     if $v_i$ is inconsistent
       backtrack
     $n \leftarrow n \cup (v, v_i)$
4. if $n$ dominates any $s \in S$
     remove $s$ from $S$
5. if $n$ is dominated by any $s \in S$
     backtrack
   else
     $S \leftarrow n$
   return $S$
```

Figure 2: Procedure to Find Pareto Optimals

will have $s_2 = d\bar{e}a$ with the relevant network N_A and we will obtain $s_2 \succ_{rc} s_1$.

Solving Method

The procedure to solve the PCCSP is shown in Figure 2. The solving method visits the nodes in a depth first manner. The algorithm takes a PCCSP instance as an input and returns the set of pareto optimals corresponding to the instance. An outcome is said to be Pareto optimal if it is not dominated by any other outcome. A solution to the PCCSP is an assignment for the set of active variables (R). Every time we encounter a solution n, we check whether it is dominated by any other solution in the set of optimals S. If n is dominated, the algorithm backtracks and looks for another solution. At any point of the execution time, N_n refers to the relevant CP-net associated with the (possibly partial) assignment n. The function $result(a)$ is the result of executing the activity constraint a according to the PCCSP P.

Example 2 Recall the PCCSP in Example 1 in addition to a constraint $C_{CB} = \{(c, \bar{b}), (\bar{c}, b), (\bar{c}, \bar{b})\}$. Applying the procedure in Figure 2 will result in (a, b) and $(\bar{a}, b, \bar{c})$ as the set of optimal outcomes (S) with relative closeness equal two for each. As a direct enhancement for the brute force approach, we can bound the longest chain of a partial assignment x in a relevant CP-net N_x and prune unpromising branches where their longest chain are greater than S. For instance, if w is a partial assignment with $lc(w) > rc(S)$ then we are sure that w will not lead to an optimal outcome and thus we safely exclude it from the searching process.

Conclusion

In this paper, a framework is defined to handle activity constraints and qualitative preferences together. The paper motivation is based on the observation that many interactive configuration problems have a room for user desires and interests. We defined sufficient conditions under which the consistency of preference statements are preserved and exploited during the searching process for optimal outcomes. Finally, we proposed an algorithm to find the set of pareto optimals given the set of preference statements represented as a CP-net. Our ultimate goal is to have a framework to represent different configuration tasks and able to effectively handle preference information over different components. Future work includes experimenting the framework with different real life applications, investigating other methods for assessing outcomes and handling composite variables in addition to conditional ones.

References

Alanazi, E., and Mouhoub, M. 2012. Arc consistency for cp-nets under constraints. In *Proceedings of the FLAIRS Conference, Marco Island, Florida. May 23-25, 2012*. AAAI Press.

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

Dechter, R. 2003. *Constraint processing*. Elsevier Morgan Kaufmann.

Gelle, E., and Sabin, M. 2003. Solving methods for conditional constraint satisfaction. In *In IJCAI-2003*, 7–12.

Gelle, E. 2003. Solving mixed and conditional constraint satisfaction problems. *Constraints* 8:2003.

Mackworth, A. K. 1977. Consistency in networks of relations. *Artificial Intelligence* 8(1):99 – 118.

Mittal, S., and Falkenhainer, B. 1990. Dynamic constraint satisfaction problems. In *AAAI*, 25–32.

Mouhoub, M., and Sukpan, A. 2005. A new temporal csp framework handling composite variables and activity constraints. In *ICTAI*, 143–149.

Mouhoub, M., and Sukpan, A. 2008. Managing temporal constraints with preferences. *Spatial Cognition & Computation* 8(1-2):131–149.

Mouhoub, M., and Sukpan, A. 2012. Managing dynamic csps with preferences. *Appl. Intell.* 37(3):446–462.

Mouhoub, M. 2003. Arc consistency for dynamic csps. In *Proceedings of Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems(KES'2003), Oxford, Lectures Notes in Computer Science(LNCS) 2773*, 393–400.

Sabin, D., and Freuder, E. C. 1996. Configuration as composite constraint satisfaction. In *Proc. Artificial Intelligence and Manufacturing. Research Planning Workshop*, 153–161. AAAI Press.

Sabin, M.; Freuder, E. C.; and Wallace, R. J. 2003. Greater efficiency for conditional constraint satisfaction. In *CP*, 649–663.