Constrained TCP-nets

Malek Mouhoub and Samira Sadaoui and Shu Zhang
University of Regina, Canada

{sadaouis,mouhoubm}@uregina.ca, jessie630527@gmail.com

Abstract

A Conditional Preference Network (CP-net) is a widely used graphical model for expressing qualitative and conditional preferences over attributes values. CP-nets have been recently extended to Tradeoffs-enhanced Conditional Preference Networks (TCP-nets) in order to capture the relative importance among attributes. In this paper, we extend the TCP-net to hard constraints and call the new proposed model Constrained TCP-net (CTCP-net). More precisely, the CTCP-net has the ability to represent and manage a given application under constraints as well as qualitative and conditional preferences over the attributes and their values. Solving the CTCP-net consists of finding the set of Pareto optimal solutions satisfying all the constraints and maximizing all the preferences. This task is addressed in this paper using a variant of the backtrack search algorithm enhanced with constraint propagation and variable ordering heuristics.

Keywords. Constraint Satisfaction, Qualitative Preferences, CP-nets, TCP-nets.

1 Introduction

Preference elicitation, representation and reasoning plays an important role in many real life applications, such as collaborative filtering, product configuration, automated decision making systems and recommender systems. In most cases, helping users to make a decision efficiently and correctly, based on a set of preferences, is important as discussed by researchers in the field.

Some of the past research works have focused on the quantitative representation of preferences through a utility function based on the well-known Multi-Attribute Utility Theory (MAUT) [Keeney and Raiffa, 1993; Sadaoui and Shil., 2014] or the C-Semiring based CSP (SCSP) and the Valued CSP (VCSP) based on a totally ordered commutative monoid [Bistarelli et al., 1999]. It is however more natural to describe preferences in a qualitative way. In this regard, many logical and graphical compact preference representation languages and formalisms have been proposed [Baier and McIlraith, 2009; Kaci, 2011]. In particular, the Conditional Preference Networks (CP-Net) [Boutilier et al., 2004] is an intuitive qualitative graphical model for representing qualitative preference information to reflect the conditional preference dependency under ceteris paribus (all else being equal) interpretation. The Tradeoffs-enhanced Conditional Preference Network (TCP-net) [Brafman et al., 2006] is introduced by extending CP-nets, allowing users to describe their relative importance on variables, thus improving the limitations of CP-Nets. In this paper, we extend the TCP-net with constraints, producing a more expressive model, called Constrained TCP-net (CTCP-net), to address problems under both constraints and qualitative preferences. These latter can be conditional and are defined on both attributes and their values. Given a problem represented as a CTCP-net, one important task is to look for a set of Pareto optimal outcomes satisfying all the constraints and optimizing all the qualitative preferences. This is a hard to solve problem that we tackle using a variant of backtrack search improved with constraint propagation techniques [Dechter, 2003]. These latter techniques will act as a filtering process that will be applied before and during the backtrack search to enhance the efficiency of this latter by reducing the size of the search space. In order to assess the time performance of our proposed solving algorithm, we have conducted several experiments on problem instances taken from Kijiji.ca. The results are very promising and demonstrate the efficiency of our method thanks to the constraint propagation techniques.

Note that CP-nets have been extended in the past in order to consider hard constraints. In [Boutilier et al., 2001], the CP-net has been augmented to a new model, called constrained CP-net, in order to include hard constraints. A generalized version of this solving method has been proposed in [Boerkoel Jr et al., 2010], where a hybrid parameterized approach (“alternative algorithms”) allows the solving of the constrained CP-net with the flexibility of trading solution quality for computational time. Recently, the constrained CP-net has been solved in [Alanazi and Mouhoub, 2016] using a powerful backtrack search algorithm including constraint propagation and variable ordering heuristics. Another approach for solving constrained CP-nets consists of converting the CP-net into a set of hard constraints that are then added to the initial set of (hard and soft) constraints of the problem to solve. The solutions to the newly obtained constrained prob-

1http://www.kijiji.ca/b-cars-vehicles/regina-area/c2711700194
lem are the optimal solutions of the initial constrained CP-net [Prestwich et al., 2005]. This approach is often called a coupled approach as opposed to the above method [Boutilier et al., 2001] and its generalization [Boerkoel Jr et al., 2010], referred to as decoupled approaches, since in these two approaches constraints and preferences are modeled separately. In [Domshlak et al., 2009], a new framework based on CP-nets and soft constraints was proposed in order to manage both hard and soft constraints as well as conditional preferences. Here, the CP-net representing conditional preferences is first approximated via soft constraints into an SCSP which will then be completed with other hard and soft constraints. Note that, since this is an approximation method, the Pareto optimal solutions returned are not guaranteed to be optimal. Comparing to the above contributions, our proposed work considers relative importance between variables and deals with conditional constraints rather than general hard constraints.

The rest of the paper is organized as follows. Section 2 reviews the related research work on constraints and preferences representation and reasoning. Basic concepts, CSPs, CP-Nets and TCP-nets are discussed in details in this section. In Section 3, our proposed CTCP-net model is defined together with its solving techniques. Section 4 is dedicated to the experiments we conducted to assess the time performance of our solving method. Finally, section 5 concludes this research work and lists some possible future works.

2 Background

2.1 CSPs
A Constraint Network (CN) includes a finite set of variables with finite domains, and a finite set of constraints restricting the possible combinations of variable values [Dechter, 2003]. Given a CN, a Constraint Satisfaction Problem (CSP) consists of finding a set of assigned values to variables that satisfy all the constraints. A CSP is known to be an NP-hard problem in general\(^2\), and is solved with a backtrack search algorithm of exponential time cost. In order to reduce this cost in practice, constraint propagation techniques have been proposed [Dechter, 2003; Haralick and Elliott, 1980; Mackworth and Freuder, 1985]. The idea here is to reduce the size of the search space before and during the backtrack search. In the past four decades the CSP framework, with its solving techniques, has demonstrated its ability to efficiently model and solve a large size real-life applications, such as scheduling and planning problems, configuration, bioinformatics, vehicle routing and scene analysis [Meseguer et al., 2006].

2.2 CP-Nets and TCP-nets
A Conditional Preference Network (CP-Net) [Boutilier et al., 2004] is a graphical model for representing and reasoning on conditional ceteris paribus preferences in a compact, intuitive and structural manner. This model allows users to express their preferences in a qualitative way, which is more natural and comfortable for users compared to quantitative descriptions. Non-conditional preferential independence is used to represent the fact that customers’ preference relation over values of a given feature is the same regardless of the values given to other features [Keeney and Raiffa, 1993]. This can be formalized as shown in Definition 1. Here, \(X\) and \(Y\) are two given variables, each defined over a discrete and a finite domain (denoted respectively as \(D(X)\) and \(D(Y)\)) of values (denoted respectively as \(x_i\) and \(y_j\)).

**Definition 1** [Brafman et al., 2006] Let \(x_1, x_2 \in D(X)\) for some \(X \subseteq V\) and \(y_1, y_2 \in D(Y)\), where \(Y = V - X\). We say that \(X\) is preferentially independent of \(Y\) iff, for all \(x_1, x_2, y_1, y_2\) we have that \(x_1 y_1 \succ x_2 y_1 \iff x_1 y_2 \succ x_2 y_2\).

In reality, customers’ preferences are much more complex. In most cases, the preferential independence relies on a certain value of other features, hence we call it conditionally preferentially independent and express it through the following definition.

**Definition 2** [Brafman et al., 2006] Let \(X, Y\) and \(Z\) be a partition of \(V\) and let \(z \in D(Z)\). \(X\) is conditionally preferentially independent of \(Y\) given \(z\) iff, for all \(x_1, x_2, y_1, y_2\) we have that \(x_1 y_1 z \succ x_2 y_1 z \iff x_1 y_2 z \succ x_2 y_2 z\).

Moreover, we can say that \(X\) is conditionally preferentially independent of \(Y\) under \(Z\) if the above formula is satisfied for every value of \(Z\). In order to illustrate the different CP-net components, let us consider the following example.

**Example 1**
We have an online shopping system where the goal is to purchase a laptop according to the buyer preferences. We assume here that the buyer is interested in only five attributes: Color, Brand, Weight, RAM and Price. According to sellers’ offers, the following range of possible values (domains) are deduced for each attribute:

\[
\begin{align*}
D_{\text{Brand}} &= \{\text{Dell, Sony, Toshiba}\} \\
D_{\text{Weight}}(\text{lb}) &= \{2.2, 3.5, 4\} \\
D_{\text{RAM}}(\text{GB}) &= \{1, 2, 4\} \\
D_{\text{Price}}(\$) &= \{680, 750, 890, 1100\} \\
D_{\text{Color}} &= \{\text{Black, White, Silver}\}
\end{align*}
\]

Next, the buyer submits the following preferences for the five attributes. (p1) The buyer prefers the highest RAM size. (p2) The buyer prefers the lightest laptop. (p3) The buyer prefers the cheapest laptop. (p4) The buyer prefers White more than Silver and Silver more than Black if Brand is Dell. Otherwise, he prefers Black more than Silver and Silver more than White. (p5) The buyer prefers Dell more than Sony and Sony more than Toshiba if Price is more than $900 and RAM is 2GB. Otherwise, he prefers Sony more than Toshiba and Toshiba more than Dell.

Figure 1 illustrates the representation of the problem in Example 1 with a CP-net. The three attributes, Weight, RAM and Price, do not depend on any other attribute. Therefore, they have an unconditional order of their values. The preferences for Brand and Color values depend on the values assigned to their respective parents. For example, if Price is
$1100 and RAM is 2 GB, the buyer’s order over Brand values is Dell > Sony > Toshiba while for other Price and RAM values, the preference order is Sony > Toshiba > Dell. Dependencies are represented with arrows going from parents to children.

Given a CP-net, a sweep forward procedure [Boutilier et al., 2004] can be used to find the optimal outcome. In our example 1, the optimal outcome is: [680, 4, Sony, 2.2, Black] (shown in underlined bold in Figure 1).

![Figure 1: The CP-net and the TCP-net for Examples 1 and 2 respectively (here, the TCP-net extends the CP-net with an arc from Brand to Weight).](image)

**Definition 4** [Brafman et al., 2006]

Let $X$ and $Y$ be a pair of variables from $V$, and let $Z \subseteq W = V - X, Y$. We say that $X$ is more important than $Y$ given $z \in D(Z)$ if, for every assignment $w'$ on $W'$ with $W' = V - (\{X, Y\} \cup Z)$ we have: $x_i y_z w' > x_j y_z w'$, whenever $x_i > x_j$ given $z w'$. We denote this relation by $X \succ_Z Y$. Finally, if for some $z \in D(Z)$ we have either $X \succ_Z Y$, or $Y \succ_Z X$, then we say that the relative importance of $X$ and $Y$ is conditioned on $Z$, and write $RI(X, Y | Z)$.

Accordingly, if for some $z \in Z$, we can also find $Y \succ_Z X$, then we can conclude that the relative importance of $X$ and $Y$ relies on $Z$, denoted as $RI(X, Y | Z)$.

### 3 Constrained TCP-nets (CTCP-nets)

#### 3.1 Definitions

**Definition 5** A conditional constraint $cc_i$ is defined as follows.

$$Y \ rel_{m+1} b \iff cc_i = X_1 \ rel_1 a_1 \ and/or \ ... \ X_m \ rel_m a_m$$

(1)

where $rel_i \in \{=, \neq, <, >, \leq, \geq\}$ and $m \geq 1$.

We denote by $vars Cd(cc_i)$ the set of variables in the condition of $cc_i$ and by $Conclusion(cc_i)$ the set of variables in the conclusion part (in our case this set is reduced to one element).

**Definition 6** Following the definition of the TCP-nets in [Brafman et al., 2006] we define the CTCP-net $T$ as a tuple $< G, CC, CP, I, CI, CP,T, CIT >$, where:

1. $G$ is the set of nodes corresponding to a set of problem variables $\{X_1, X_2, \ldots, X_n\}$. Each variable $X_i$ is defined over a domain $D(X_i)$ of discrete values.

2. $CC$ is the set of conditional constraint arcs (denoted as $cc-\text{arcs}$). A $cc-\text{arc} [X_i, X_j]$ expresses the fact that $X_j$ is restricted by a given condition on $X_i$'s values. This basically means that there exists a conditional constraint $cc$ where $X_i \in vars Cd(cc_i)$ and $X_j$ is the variable present in the conclusion of $cc$.

3. $CP$ is the set of conditional preference arcs (denoted as $cp-\text{arcs}$) corresponding to conditional preferences. A $cp-\text{arc} < X_i, X_j >$ means that the preferences over the values of $X_j$ depend on the actual value of $X_i$.

4. $I$ is the set of non-conditional relative importance arcs (denoted as $i-\text{arcs}$) corresponding to non-conditional relative importance relations. An $i-\text{arc} [X_i, X_j]$ corresponds to the following relative importance: $X_i \succ X_j$, denoting that $X_i$ is more important than $X_j$. This basically means that $X_i$ can be assigned a value if and only if $X_i$ has already been assigned a value.

5. $CI$ is the set of undirected conditional relative importance arcs (denoted as $ci-\text{arcs}$) corresponding to conditional relative importance relations. A $ci-\text{arc} [X_i, X_j]$ is in $T$ iff there is $RI(X_i, X_j | Z)$ for some $Z \subseteq G - \{X_i, X_j\}$. This basically means that $X_i$ and $X_j$ can be assigned a value if and only if $Z$ has already been assigned a value. Specifically, every $i-\text{arc} [X_i, X_j]$ can
be represented as $\mathcal{RT}(X_i, X_j, \phi)$. Here, $Z$ is called the selector set of $(X_i, X_j)$ and is denoted by $S(X_i, X_j)$.

7 CPT associates a Conditional Preference Table (CPT) with every node $X \in G$. CPT($X$) is a mapping from $D(Pa(X))$ (ie., assignments to $X$’s parents nodes) to a partial order over $D(X)$. Pa($X$) is $X$’s conditional (dependent) variable.

8 CIT associates with every $ci$–arc $\gamma = (\bar{X_i}, \bar{X_j})$, a (possibly partial) mapping CIT($\gamma$) from $D(S(X_i, X_j))$ to an order over the set $\{X_i, X_j\}$.

Note that the sub tuple $< CP, I, CI, CPT, CIT >$ corresponds to a TCP-net. Moreover, if the sets $I$ and $CI$ are empty then we have a CP-Net. Let us illustrate the different components of our CTCP-net through the following example.

**Example 3**

Let us consider our Example 2 and assume the buyer submits the following constraints.

(1) If Weight < 3 lb, the buyer does not buy Sony.

(2) If RAM > 2 GB then Price should be higher than $700.

Figure 2 illustrates the representation of the problem in Example 3 with the constrained TCP-net we propose. Constraints restrict the values that some attributes can simultaneously take and are represented by edges between the nodes sharing the constraints. There are two Pareto optimal outcomes induced by the constrained TCP-net of our example: [680, 2, Sony, 3, 5, Black] and [750, 4, Sony, 3, 5, Black]. Finding the Pareto optimal outcomes can be achieved using the backtrack search algorithm we will present in the next Section. This algorithm has the ability to return the optimal solutions in an efficient way thanks to the constraint propagation techniques and variable ordering heuristics we have used.

3.2 Constraint Propagation for CTCP-nets

The following is the general procedure we use to manage conditional constraints as defined in the previous Section. Here, $m$ is the number of relations in the condition of the given conditional constraint.

- **m=1.** This is a conditional constraint involving one variable in the premise of the condition. We process it as a form of binary constraint using Algorithm 3 in the preprocessing step to remove the inconsistent values. Note that Algorithm 3 enforces directional arc consistency between the variable in the premise and the one in the conclusion of the conditional constraint. For instance, if the conditional constraint is $X \neq a \implies Y \neq b$ and value $a$ has been removed from the domain of $X$ then $b$ has to be removed from the domain of $Y$. As well, the conditional constraint is also used to propagate the effect of an assignment during the backtrack search following the look ahead strategy. For instance, if the conditional constraint is $X = a \implies Y \geq b$ and the current assignment is $X = a$ then all values from the domain of $Y$ that are less than $b$ should be removed.

- **m > 1.** This is a conditional constraint involving more than one relation in the condition of the conditional constraint. We process it similarly to the case where $m = 1$ but using generalized directional arc consistency as we are dealing with a form of n-ary constraint in this particular case. For instance, let us assume we have the following conditional constraint: $A \neq a$ or $B \neq b \implies C < c$ then if $a$ is removed from the domain of $A$ or $b$ is removed from the domain of $B$ then all the values from $C$’s domain that are greater or equal to $c$ should be removed. This propagation can happen in the preprocessing stage as well as the search phase. For example, if during the backtrack search $A$ (or $B$) is assigned a value other than $a$ (or $b$) then $C$ cannot be assigned a value that is greater or equal to $c$.
Arc consistency is enforced with an arc consistency algorithm [Mackworth, 1977; Dechter, 2003]. Since we are dealing with n-ary constraints, we use an adapted version of the Generalized Arc Consistency (GAC) algorithm presented in [Mouhoub and Feng, 2009]. This latter is a revised version of the original GAC algorithm proposed in [Lecoutre and Radoslaw, 2006] as well as a modified version of the bound consistency algorithm for discrete CSPs in the case of inequality relations [Lecoutre and Vion, 2005]. More precisely, bounds consistency is first used through inequality relations to reduce the bounds of the different domains of variables. The adapted GAC is then used to further reduce the domains of the variables. Let us describe now the details of our method. The modified GAC algorithm that we call CTCP-GAC is described in figure 3. This algorithm enforces arc consistency on all variables domains. CTCP-GAC starts with all possible pairs \((i, j)\) where \(j\) is a variable involved by the constraint \(i\). Each pair is then processed, through the function \(\text{REVISE}\) as follows. Each value \(v\) of the domain of \(j\) should have a value supporting it (such that the constraint \(j\) is satisfied) on the domain on every variable involved by \(i\) otherwise \(v\) will be removed. If there is a change in the domain of \(j\) (after removing values without support) after calling the function \(\text{REVISE}\) then this change should be propagated to all the other variables sharing a constraint with \(j\). When used as a bound consistency algorithm, \(cc\) involves inequality relations and the \(\text{REVISE}\) function (the function that does the actual revision of the domains) is defined as shown in figure 3 [Lecoutre and Vion, 2005]. In the other case, the \(\text{REVISE}\) function is defined as shown in the bottom right of figure 3 [Lecoutre and Radoslaw, 2006]. In the function \(\text{REVISE}\) (for bound consistency) of figure 3, the function \(\text{seekSupportArc}\) (respectively the function \(\text{seekSupport}\) of \(\text{REVISE}\) for semantic constraints in figure 3) is called to find a support for a given variable with a particular value. For instance when called in line 2 of the function \(\text{REVISE}\) for bound consistency, the function \(\text{seekSupportArc}\) looks, starting from the lower bound of \(j\)'s domain, for the first value that has a support in \(i\)'s domain. When doing so, any value not supported will be removed.

After enforcing arc consistency in the preprocessing stage of our proposed solving method, we run a backtrack search algorithm with a look ahead strategy [Dechter, 2003] to find the Pareto optimal solutions of a given CTCP-net. In order to improve the time performance of the backtrack search, variables are first ordered following the most constrained variables first heuristic [Mouhoub and Jashmi, 2011]. Some of these variables will then be reordered according to the dependencies imposed by the CTCP-net. In this regard, variables need to be sorted after their respective parents in the corresponding conditional constraint, conditional preference or unconditional relative importance relation. In addition to this static variable ordering that occurs before the backtrack search, some variables are rearranged dynamically during the backtrack search according to the conditional relative importance relations (anytime the variable, the relative importance relies on, is assigned a particular value). Variables values are ordered according to the CPTs. Note that, like for variable ordering, some of these orders depend on values assigned to some other variables and this is done dynamically during the backtrack search. We adopt the Forward Check strategy [Har- alick and Elliott, 1980] as the constraint propagation technique during the backtrack search. Anytime a variable (that we call current variable) is assigned a value during the search, we propagate this decision to the non assigned variables using our CTCP-GAC algorithm as described above in our general procedure. In addition to reducing the size of the search space, this propagation will also detect later failure earlier. For instance, if one of the domains of the non assigned variables becomes empty then we assign another value to the current variable or backtrack to the previously assigned variable if there are no more values to assign to the current one. This backtrack search method will continue until all the variables are assigned in which case we obtain a complete assignment (consistent solution). We then test if the obtained solution is dominated by any other solution found so far. If it is not the case, we add it to the current set of Pareto optimal solutions. The algorithm stops when the search is exhausted (there are no more Pareto solutions).

### 4 Experimentation

In order to evaluate the time performance of our solving method, we conducted several experiments on real data selected from Kijiji.ca\(^3\). These data correspond to cars sale information. We assume that the goal here it to purchase a vehicle online. 100 products are used for the experiments and for each product the following attributes are considered: brand, model, year, engine size, color, milage, price, transmission, body type and seller name. The constraints and preferences are represented in our model as shown below.

- **Non-conditional constraints (NCC).**
  - Saleby ≠ capital
  - Saleby ≠ roadway

---

\(^3\)http://www.kijiji.ca/b-cars-vehicles/regina-area/c271700194
- Kilometers $\leq 150000$
- Brand $\neq$ Kia
- Year $\geq 2002$

- **Conditional constraints (CC):**

  (cc1) Saleby = nelson $\rightarrow$ Bodytype $\neq$ hatchback

  (cc2) Bodytype = hatchback $\rightarrow$ Saleby $\neq$ owner

  (cc3) $(Brand = Honda) \lor (Bodytype = suv) \rightarrow$ Color $\neq$ red

  (cc4) $(Kilometer \geq 13) \land (Transmission = manual) \rightarrow$ Brand $\neq$ Ford

  (cc4) $(Price \geq 8000) \rightarrow$ Kilometers $\leq 120000$

- **Non-Conditional preferences (NCP) on variables domains:**

  (ncp1) Year: descending order

  (ncp2) Price: ascending order

  (ncp3) Kilometers: ascending order

  (ncp4) Transmission: auto $\succ$ manumatic $\succ$ manual

- **Conditional preferences (CP):**

  (cp1) $(Year \geq 2005) \land (Kilometers \leq 15) \rightarrow$
  Brand$(Toyota \succ Pontiac \succ Nissan \succ Mazda \succ$
  Kia $\succ$ Hyundai $\succ$ Honda $\succ$ Ford $\succ$ Dodge $\succ$
  Chrysler $\succ$ Chevrolet $\succ$ Buick $\succ$ Benz $\succ$ BMW $\succ$
  Audi); otherwise : Brand$(Audi \succ BMW \succ Benz \succ$
  Buick $\succ$ Chevrolet $\succ$ Chrysler $\succ$ Dodge $\succ$ Ford $\succ$
  Honda $\succ$ Hyundai $\succ$ Kia $\succ$ Mazda $\succ$ Nissan $\succ$
  Pontiac $\succ$ Toyota)

  (cp2) $(Saleby \neq owner) \lor (Transmission = manual) \rightarrow$
  Color$(yellow \succ white \succ silver \succ red \succ grey \succ$
  green $\succ$ gold $\succ$ brown $\succ$ blue $\succ$ black); otherwise : Color$(black \succ blue \succ brown \succ gold \succ green \succ grey \succ$
  red $\succ$ silver $\succ$ white $\succ$ yellow)

  (cp3) $Brand = Honda \rightarrow Bodytype(\succ convertible \succ$
  wagon $\succ$ suv $\succ$ coupe $\succ$ sedan $\succ$ truck $\succ$
  van $\succ$ hatchback); otherwise : Bodytype$(\succ$ hatchback $\succ$
  coupe $\succ$ sedan $\succ$ suv $\succ$ truck $\succ$ van $\succ$ wagon $\succ$
  convertible)

  (cp4) $Price \leq 10000 \rightarrow Saleby(owner \succ nelson \succ capital \succ$
  roadway); otherwise : Saleby(nelson $\succ$ capital $\succ$
  owner $\succ$ roadway)

- **Non-conditional relative importance(NCIR):**

  (ncr1) Price $\succ$ Year

  (ncr2) Price $\succ$ Saleby

  (ncr3) Year $\succ$ Brand

  (ncr4) Brand $\succ$ Bodytype

- **Conditional Relative importance(CIR):**

  (cri1) Saleby $\neq$ owner $\rightarrow$ Year $\succ$ Kilometers; otherwise : Kilometers $\succ$ Year

  (cri2) Transmission = auto $\rightarrow$ Color $\succ$
  Bodytype; otherwise : Bodytype $\succ$ Color

The experiments are conducted on a PC with the following specifications: Inter(R) Core(TM) i7-4500U CPU @ 1.8GHz
and 16GB RAM; and running Windows 8 64-bit operating system. The test platform is MyEclipse 8.5.

Figure 4 reports the running time required in seconds to return the optimal solution when varying the number of products from 20 to 100. For each experiment, 30 runs are conducted and the average running time is taken. As we can see from the figure our proposed method is capable of provide an answer in less than 2 seconds even when the number of products is 100.

5 Conclusion and Future Work

Constraints and preferences handling is a complex but interesting problem that is related to a wide variety of real world applications. Our proposed CTCP-nets has the ability to represent and solve these constraint problems under preferences by returning one or more solutions satisfying all the constraints and maximizing the preferences. This process can be done in a very efficient running time, thanks to the constraint propagation techniques that we propose. The proposed CTCP-net has been implemented with a generic design that offers the flexibility for future maintenance and extensibility. It will be therefore possible in the future to add other modules dealing with new features and properties such as the case of cyclic CTCP-nets.

In the near future we intend to consider dynamic CTCP-nets in the case of constraints and preferences addition and retraction. Adding constraints and preferences can be relevant when the number of Pareto optimal solutions is very large. In this particular situation, we need to add more constraints or preferences in order to bring this number down to a manageable size. On the other hand, the retraction of constraints can happen when the CTCP-net is inconsistent. In this case, we need to relax some constraints in order to restore the consistency of the network. We have previously proposed incremental constraint propagation techniques for managing constraints in a dynamic environment [Mouhoub, 2003; Mouhoub and Sukpan, 2012] and are planning to adapt these techniques for the dynamic CTCP-net.
References


