Journal of Artificial Intelligence Research 31 (2008) 83-112 Submitted 06/07; published 01/08

CUI Networks: A Graphical Representation for Conditional Utility Independence

Yagil Engel YAGIL@UMICH.EDU
Michael P. Wellman WELLMAN@UMICH.EDU
University of Michigan, Computer Science & Engineering
2260 Hayward St, Ann Arbor, MI 48109-2121, USA

Abstract

We introduce CUI networks, a compact graphical representation of utility functions over multiple attributes. CUI networks model multiattribute utility functions using the well-studied and widely applicable utility independence concept. We show how conditional utility independence leads to an effective functional decomposition that can be exhibited graphically, and how local, compact data at the graph nodes can be used to calculate joint utility. We discuss aspects of elicitation, network construction, and optimization, and contrast our new representation with previous graphical preference modeling.

1. Introduction

Modern AI decision making is based on the notion of *expected utility*, in which probability distributions are used to weigh the utility values for each of the possible outcomes. The representation of probability distribution functions by Markov or Bayesian networks (Pearl, 1988)—exploiting conditional independence to achieve compactness and computational efficiency—has led to a plethora of new techniques and applications. Despite their equal importance to decision making, preferences and utilities have generally not received the level of attention AI researchers have devoted to beliefs and probabilities. Nor have the (increasing) efforts to develop representations and inference methods for utility achieved a degree of success comparable to the impact of graphical models on probabilistic reasoning. Recognizing that utility functions over multidimensional domains may also be amenable to factoring based on independence (Keeney & Raiffa, 1976), several have aimed to develop models with analogous benefits (Bacchus & Grove, 1995; Boutilier, Bacchus, & Brafman, 2001; La Mura & Shoham, 1999; Wellman & Doyle, 1992). This is our goal as well, and we compare our approach to these and other methods in the Related Work section (2.2).

The development of compact representations for multiattribute utility begins with the notion of *preferential independence* (PI), or *separability* of subdomains in the outcome space. A subdomain of outcomes is separable in the PI sense if the preference order over this subdomain does not depend on the rest of the domain. When all subsets of attributes induce separable subdomains, the ordinal utility (value) function decomposes additively over its variables (Debreu, 1959; Fishburn, 1965; Gorman, 1968). A *cardinal utility function* represents not only preferences over outcomes but also a notion of *strength* of preferences, most notably to represent preferences over actions with uncertain outcomes, or *lotteries*. Direct adaptation of the PI concept to cardinal utility requires a generalization of this notion: a set of attributes is *utility independent* (UI) if the preference order over lotteries on the induced subdomain does not depend on the values of the rest of the attributes. A stronger judgement is to assert that the preference order over the joint domain depends only on its margins over some attribute subsets. The latter leads to powerful additive decompositions, either *fully additive* (when the subsets of attributes are disjoint), or *generalized*, which is an additive decomposition over overlapping subsets (Fishburn, 1967; Bacchus & Grove, 1995). Utility independence leads to less convenient decompositions, such as multilinear (Keeney & Raiffa, 1976) or hierarchical (Von Stengel, 1988; Wellman & Doyle, 1992). Most previous efforts in the AI community to adapt modern graphical modeling to utility functions employ the generalized additive decomposition (Bacchus & Grove, 1995; Boutilier et al., 2001; Gonzales & Perny, 2004). In contrast, our work continues the other thread, based on the weaker utility independence assumption. We elaborate on the difference between the types of independence following the presentation of formal definitions.

2. Background

Our utility-theoretic terminology follows the definitive text by Keeney and Raiffa (1976). In the multiattribute utility framework, an *outcome* is represented by a vector of values for n variables, called *attributes*. The decision maker's preferences are represented by a total pre-order, $\succeq$, over the set of outcomes. In common applications decision makers do not have the ability to choose a certain outcome, but rather an *action* that results in a probability distribution over outcomes, also called a *lottery*. The decision maker is hence assumed to have a preference order $\tilde{\succeq}$ over the set of possible lotteries. Given a standard set of axioms, $\tilde{\succeq}$ can be represented by a real-valued *utility function* over outcomes, $U(\cdot)$, such that numeric ranking of probabilistic outcomes by expected utility respects the ordering by $\tilde{\succeq}$. The utility function is unique up to positive affine transformations. A positive linear transform of $U(\cdot)$ represents the same preferences, and is thus *strategically equivalent*.

The ability to represent utility over probability distributions by a function over outcomes provides some structure, but in multiattribute settings the outcome space is n-dimensional. Unless n is quite small, therefore, an explicit (e.g., tabular) representation of $U(\cdot)$ will generally not be practical. Much of the research in multiattribute utility theory aims to identify structural patterns that enable more compact representations. In particular, when subsets of attributes respect various independence relationships, the utility function may be decomposed into combinations of modular subutility functions of smaller dimension.

Let $S = \{x_1, \ldots, x_n\}$ be a set of attributes. In the following definitions (and the rest of the work) capital letters denote subsets of attributes, small letters (with or without numeric subscripts) denote specific attributes, and $\overline{X}$ denotes the complement of X with respect to S. We denote the (joint) domain of X by $D(X)$, and indicate specific attribute assignments with prime signs or superscripts. To represent an instantiation of subsets X and Y at the same time we use a sequence of instantiation symbols, as in $X'Y'$.

In order to meaningfully discuss preferences over subsets of attributes, we need a notion of preferences over a subset given fixed values for the rest of the attributes.

Definition 1. Outcome Y' is conditionally preferred to outcome Y'' given $\overline{Y}'$, if $Y'\overline{Y}' \succeq Y''\overline{Y}'$. We denote the conditional preference order over Y given $\overline{Y}'$ by $\succeq_{\overline{Y}'}$.

Similarly we define conditional preference order over lotteries. The preference order $\tilde{\succeq}_{\overline{Y}'}$ over lotteries on Y is represented by a *conditional utility function*, $U(Y, \overline{Y}')$.

Definition 2. Y is *Preferential Independent (PI)* of $\overline{Y}$ if $\succeq_{\overline{Y}'}$ does not depend on the value chosen for $\overline{Y}'$.

Preferential independence can be very useful for qualitative preference assessment. First-order preferential independence (i.e., independence of a single attribute from the rest) is a natural assumption in many domains. For example, in typical purchase decisions greater quantity or higher quality is more desirable regardless of the values of other attributes. Preferential independence of higher order, however, requires invariance of the *tradeoffs among some attributes* with respect to variation in others, a more stringent—though still often satisfiable—independence condition. The standard PI condition applies to a subset with respect to the full complement of remaining attributes. The *conditional* version of PI specifies independence with respect to a subset of the complement, holding the remaining attributes fixed.

Definition 3. Y is *Conditionally Preferential Independent (CPI)* of X given Z ($Z = \overline{XY}$), if for any Z', $\succeq_{X'Z'}$ does not depend on the value chosen for X'. We denote this relationship by $\mathrm{CPI}(Y, X \mid Z)$.

The counterpart of preferential independence that considers probability distributions over outcomes is called *utility independence*.

Definition 4. Y is *Utility Independent* (UI) of $\overline{Y}$, if the conditional preference order for lotteries over Y, $\tilde{\succeq}_{\overline{Y}'}$, does not depend on value chosen for $\overline{Y}'$.

In our notations, we apply UI and the conditions defined below to sets of attributes or to specific attributes.

Given $\mathrm{UI}(Y, X)$, taking $X = \overline{Y}$, the conditional utility function over Y given X' is invariant up to positive affine transformations, for any fixed value X'. This fact can be expressed by the decomposition

$$U(X, Y) = f(X) + g(X)U(X', Y), \quad g(\cdot) > 0.$$

Note that the functions $f(\cdot)$ and $g(\cdot)$ may be different for each particular choice of X'. Since $U(X', Y)$ is a function only of Y, we sometimes use the notation $U_{X'}(Y)$.

Utility independence has a conditional version as well.

Definition 5. Y is *Conditionally Utility Independent (CUI)* of X given Z ($Z = \overline{XY}$) if for any Z', $\tilde{\succeq}_{X'Z'}$ does not depend on the value chosen for X'. We denote this relationship by $\mathrm{CUI}(Y, X \mid Z)$.

CUI also supports functional decomposition. For any Z', the conditional utility function over Y given $X'Z'$ is strategically equivalent to this function given a different instantiation of X. However, the transformation depends not only on X, but also on Z'. Hence we can write:

$$U(X, Y, Z) = f(X, Z) + g(X, Z)U(X', Y, Z), \quad g(\cdot) > 0. \tag{1}$$

That is, we can fix X on some arbitrary level X' and use two transformation functions f and g to get the value of $U(\cdot)$ for other levels of X. A stronger, symmetric form of independence which leads to additive decomposition of the utility function is called *additive independence*. We provide the definition for its conditional version.

Definition 6. X and Y are *Conditionally Additive Independent* given Z, $\mathrm{CAI}(X, Y \mid Z)$, if for any instantiation Z', $\tilde{\succeq}_{Z'}$ depends only on the marginal conditional probability distributions over XZ' and YZ'.

This means that for any value Z', and for any two probability distributions p, q such that $p(X, \cdot, Z') \equiv q(X, \cdot, Z')$, and $p(\cdot, Y, Z') \equiv q(\cdot, Y, Z')$, the decision maker is indifferent between p and q. A necessary (but not always sufficient) condition for this to hold is that the utility differences $U(X', Y, Z') - U(X'', Y, Z')$ (for any X', X'') do not depend on the value of Y.

CAI leads to the following decomposition (Keeney & Raiffa, 1976):

$$U(X, Y, Z) = f(X, Z) + g(Y, Z).$$

Other variations of utility independence were considered in the theoretical literature, leading to various decomposition results (Fishburn, 1975; Krantz, Luce, Suppes, & Tversky, 1971; Fuhrken & Richter, 1991).

2.1 Motivation

The most obvious benefit of a model based on (conditional) utility independence is the generality admitted by a weaker independence condition, in comparison to additive independence. Whereas additivity practically excludes any interaction between utility of one attribute or subset (X in Definition 6) to the value of another (Y), utility independence allows substitutivity and complementarity relationships, as long as the risk attitude towards one variable is not affected by the value of another. One could also argue that UI is particularly intuitive, based as it is on an invariance condition on the preference order. In contrast, (conditional) additive independence requires a judgment about the effects of joint versus marginal probability distributions. Moreover, additive independence is symmetric, whereas the condition $UI(X, Y)$ does allow the preference order over Y to depend on X.

Bacchus and Grove exemplify the difference between additive and utility independence on a simple state space of two boolean attributes: Health and Wealth. In their example, shown in Table 1, the attributes are not additive independent (it can be immediately seen using preference differences), because H and W are complements: having both is worth more than the sum of having each one without the other. We would have considered the two attributes substitutes if, for example, $U(W, H) = 4$ and $U(W, \neg H) = 3$. In both cases H and W are nonetheless preferential independent, since we always prefer to be richer (all else being equal) and healthier (all else being equal). For boolean variables, preferential and utility independence are equivalent (we always prefer lotteries that give higher probability to the preferred level) and therefore Health and Wealth are also UI of each other.

(Conditional) additive independence and its resulting additive decomposition can be generalized to multiple subsets that are not necessarily disjoint. This condition is called

	H	$\neg H$
W	5	1
$\neg W$	2	0

Table 1: Utility values for the Health and Wealth example (Bacchus & Grove, 1995).

generalized additive independence (GAI). If GAI holds, $U(\cdot)$ decomposes to a sum of independent functions $f_i(\cdot)$ over the GAI subsets X_i. As shown by Bacchus and Grove, CAI conditions can be accumulated to a global GAI decomposition (see Section 2.2). The latter may also exist without any CAI conditions leading to it, but such a GAI condition is hard to identify: whereas each CAI condition corresponds to the independence of two attributes or two subsets, a global GAI condition does not have such an intuitive interpretation.

In the next example, no cardinal independence condition exists, except for a non symmetric CUI. The example also shows the difference between PI and UI, and hence requires the domains of H and W to include at least three values each. We also add a third attribute to the outcome space, location (L), indicating whether we live in the city or in the countryside (Table 2). In order to show that $UI(H, \{W, L\})$ does not hold it is enough to find that it is violated for one pair of lotteries. Given the partial outcome W_r, L_{ci} we prefer the equal chance lottery over $< H_f, H_s >$, whose expected utility is $\frac{12+5}{2}$, to the sure outcome H_g (value 8), whereas given W_p, L_{ci} we are indifferent (expected utility of 2 to both lotteries). Intuitively, it may be the case that the additional value we get from fitness (over good health) is higher if we are also rich, making it more significant than the value H_g adds over H_s. Similarly, $UI(W, \{H, L\})$ does not hold, by comparing the even-chance gamble over $< W_r, W_p >$ and the sure outcome W_m, first given H_f, L_{ci} and then given H_s, L_{ci}.

W and H are therefore not utility independent, but they are preferential independent. L, however, is not: when we are rich we would rather live in the city, and the other way round when we are poor, except for the case of being poor and sick under which we prefer the city.

	L_{ci}			L_{co}		
	H_f	H_g	H_s	H_f	H_g	H_s
W_r	12	8	5	10	6	4
W_m	6	4	3	6	3	2
W_p	3	2	1	4	1.5	0

Table 2: Utility values for the Health, Wealth and Location example. W_r means rich, W_m is medium income, W_p means poor. H_f is healthy and at top fitness, H_g means good health, and H_s means sick. L_{ci} stands for city location, and L_{co} means countryside location.

Therefore, no symmetric independence condition exists here, and that rules out any additive or multiplicative independence, conditional or not, between any subsets of attributes. Also, since no single variable is unconditionally UI, then no subset can be unconditionally UI. Further, the fact that preferences over L depend on the combination of H and W rules out a GAI decomposition of the form $\{W, L\}, \{W, H\}, \{H, L\}$.

We can, however, achieve decomposition using CUI. It is the case that CUI$(W, L|H)$, since each column on the left matrix (L_{ci}) is an affine transformation of its counterpart on the right side (L_{co}). For example, to transform the first column (H_f), multiply by $\frac{2}{3}$ and add 2.

This example illustrates the subtlety of utility independence. In particular, whereas preferences over L depend on W, W may still be (conditionally) UI of L. A CAI assumption for the same attributes must inevitably ignore the reversal of preferences over L for different values of W, hence a decision maker that will be queried for preferences under this assumption may not be able to provide meaningful answers.

The interaction with a system that requires preference representation normally requires the identification of structure, and then the population of the utility values that are required by the compact representation. It is therefore most important that these two aspects are simplified as possible, whereas the functional form handled by the system may be more sophisticated. This is exactly the tradeoff made by CUI nets, compared to a GAI-based representation: if the GAI condition is based on CAI, CUI nets achieve lower dimensionality (Section 7), and therefore easier elicitation. If a GAI condition is not based on a collection of CAI conditions, it is hard to identify. CUI nets simplify these bottleneck aspects, by driving the complexity into the algorithms and into the functional form that is handled behind the scenes.

2.2 Related Work

Perhaps the earliest effort to exploit separable preferences in a graphical model was the extension of influence diagrams by Tatman and Shachter (1990) to decompose value functions into sums and products of multiple value nodes. This structure provided computational advantages, enabling the use of dynamic programming techniques exploiting value separability.

Bacchus and Grove (1995) were first to develop a graphical model based on *conditional* independence structure. In particular, they establish that the CAI condition has a *perfect map* (Pearl & Paz, 1989); that is, a graph with attribute nodes S such that node separation reflects exactly the set of CAI conditions on S. More specifically, for any two sets of nodes $X, Y \subseteq S$, CAI$(X, Y|\overline{XY})$ holds if and only if there is no direct edge between a node in X and a node in Y. We use the term *CAI map* of S when referring to the graph which reflects the perfect map of CAI conditions, in the context of a preference order over $D(S)$. Bacchus and Grove go on to show that the utility function has a GAI decomposition over the set of maximal cliques of the CAI map. As we show in Section 7, the CUI network representation developed here achieves weakly better dimensionality than CAI maps due to the greater generality of the independence assumption.

Initiating another important line of work, Boutilier et al. (1999) introduced *CP networks*, an efficient representation for ordinal preferences over multiple attributes. In a CP network, each variable is conditionally PI of the rest given its parents. Ordinal multiattribute preference representation schemes (for decision making under certainty), and especially CP networks, can dramatically simplify the preference elicitation process, based as they are on intuitive relative preference statements that avoid magnitude considerations. However, the limited expressive power of CP networks may not suffice for complex decision problems, in which tradeoff resolution may hinge in a complicated way on attribute settings over rich domains. This problem is particularly acute when continuous or almost continuous attributes are involved, such as money or time.

Boutilier et al. (2001) subsequently extended this approach to numeric, cardinal utility in *UCP networks*, a graphical model that utilizes the GAI decomposition combined with a CP-net topology. This requires dominance relations between parents and their children, somewhat limiting the applicability of the representation. The GAI structure was also applied for graphical models by Gonzales and Perny (2004), who employ the clique graph of the CAI map (the *GAI network*) for elicitation purposes.

In earlier work, La Mura and Shoham (1999) redefine utility independence as a symmetric multiplicative condition, taking it closer to its probability analog, and supporting a Bayes-net like representation. Although multiplicative independence is different from additive independence, it is not necessarily weaker. Recent work by Abbas (2005) defines a subclass of utility functions on which a multiplicative notion of UI obeys an analog of Bayes's rule.

The only graphical decomposition suggested in the past for utility functions that is based on the original, non-symmetric notion of utility independence is the utility tree (Von Stengel, 1988, see also Wellman and Doyle, 1992, for discussion in an AI context). The utility tree decomposes the utility function using multilinear or multiplicative decomposition (Keeney & Raiffa, 1976), and then further tries to decompose each subset similarly. Using these hierarchical steps the utility function becomes a nested expression over functions of its smallest separable subsets and their complements.

2.3 Graphical Models of CUI

In their concluding remarks, Bacchus and Grove (1995) suggest investigating graphical models of other independence concepts, in particular utility independence. Founding a graphical model on UI is more difficult, however, as utility independence does not decompose as effectively as does additive independence. In particular, the condition $UI(Y, X)$ ensures that Y has a subutility function, but since X does not have one it is harder to carry on the decomposition into X. Hence in the case that X is large the dimensionality of the representation may remain too high. Our approach therefore employs CUI conditions on large subsets Y, in which case the decomposition can be driven further by decomposing the conditional utility function of Y using more CUI conditions.

In the sequel we show how serial application of CUI leads to functional decomposition. The corresponding graphical model, a CUI network, provides a lower-dimension representation of the utility function in which the function for any vertex depends only on the node and its parents. We demonstrate the use of CUI networks by constructing an example for a relatively complex domain. Next we elaborate on the technical and semantic properties of the model and knowledge required to construct it. Subsequent technical sections present optimization algorithms and techniques for further reducing the complexity of the representation.

3. CUI Networks

We begin by constructing a DAG representing a set of CUI conditions, followed by a derivation of the functional decomposition over the nodes of the DAG.

3.1 CUI DAG

Suppose that we obtain a set σ of CUI conditions on the variable set $S = \{x_1, \dots, x_n\}$, such that for each $x \in S$, σ contains a condition of the form

$$\mathrm{CUI}(S \setminus (\{x\} \cup P(x)) \ , \ x \mid P(x)).$$

In other words, there exists a set $P(x)$ that separates the rest of the variables from x. Such $P(x)$ always exists, because for $P(x) = S \setminus \{x\}$ the condition above trivially holds. The set σ can be represented graphically by the following procedure, which we name *procedure $\mathcal{C}$*.

1. Define an order on the set S (for convenience we assume the ordering $x_1, \dots, x_n$).
2. Define the set of parents of x_1 as $Pa(x_1) = P(x_1)$.
3. For each $i = 2, \dots, n$
   - Define the set of intermediate descendants of x_i, $\tilde{Dn}(x_i)$, as the set of nodes in $x_1, \dots, x_{i-1}$ that turned out to be descendants of x_i, that is those for which x_i is a parent or another descendant of x_i is a parent. Formally, $\tilde{Dn}(x_i)$ is the smallest set that satisfies the following condition:

     $$\forall j \in \{1, \dots, i-1\}, \ [x_i \in Pa(x_j), \text{ or } \ \exists k \in \{1, \dots, i-1\}. x_k \in \tilde{Dn}(x_i) \cap Pa(x_j) \Rightarrow x_j \in \tilde{Dn}(x_i).] \quad (2)$$

   - Define the parents of x_i to be the nodes in $P(x_i)$ which are not already descendants of x_i,

     $$Pa(x_i) = P(x_i) \setminus \tilde{Dn}(x_i).$$

This procedure defines a DAG. We denote $Dn(x)$ as the final set of descendants of x. It is the set defined by Equation (2), when replacing $\{1, \dots, i-1\}$ with $\{1, \dots, n\}$). By their definitions, $Dn(x) \supseteq \tilde{Dn}(x)$, hence

$$Pa(x) \cup Dn(x) \supseteq Pa(x) \cup \tilde{Dn}(x) = P(x). \quad (3)$$

Proposition 1. *Consider the DAG defined by procedure $\mathcal{C}$ for a set of attributes S. For any $x \in S$,*

$$\mathit{CUI}(S \setminus (\{x\} \cup Pa(x) \cup Dn(x)) \ , \ x \mid Pa(x) \cup Dn(x)). \quad (4)$$

Proof. By the definitions of $Pa(x)$ and $P(x)$, (4) holds when replacing $Dn(x)$ with $\tilde{Dn}(x)$. From the definition of CUI, it is straightforward that

$$\mathrm{CUI}(S \setminus (Y \cup W) \ , \ Y \mid W) \Rightarrow \mathrm{CUI}(S \setminus (Y \cup W \cup Z) \ , \ Y \mid W \cup Z),$$

because invariance of preference order over $S \setminus (Y \cup W)$ implies invariance of preference order over its subset $S \setminus (Y \cup W \cup Z)$, when the difference set Z is fixed. Given (3), and taking $W = Pa(x) \cup \tilde{Dn}(x)$ and $Z = Dn(x) \setminus \tilde{Dn}(x)$, we get (4). $\square$

As an example, we show the construction of the structure on a small set of variables $S = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, for which we are given the following set of CUI conditions:

$$\sigma = \{\mathrm{CUI}(\{x_4, x_5, x_6\}, x_1 \mid \{x_2, x_3\}), \mathrm{CUI}(\{x_4, x_3, x_6\}, x_2 \mid \{x_1, x_5\}), \mathrm{CUI}(\{x_2, x_4, x_6\}, x_3 \mid \{x_1, x_5\}), \mathrm{CUI}(\{x_1, x_3, x_5\}, x_4 \mid \{x_2, x_6\}), \mathrm{CUI}(x_6, x_5 \mid \{x_1, x_2, x_3, x_4\}), \mathrm{CUI}(\{x_1, x_2, x_3, x_5\}, x_6 \mid x_4)\}.$$

Construction of the network using the order implied by the indices results in the CUI DAG illustrated in Figure 1. The minimal separating set for x_1 is $\{x_2, x_3\}$. For x_2, we get $\tilde{Dn}(x_2) = \{x_1\}$, and the only non-descendant variable that is required to separate it from the rest is x_5, which is therefore its only parent. The rest of the graph is constructed in a similar way. When x_4 is placed, we find that $P(x_4) = \{x_2, x_6\}$. Therefore, x_4 becomes descendant of x_2 after x_2 is placed, in other words $Dn(x_2) = \tilde{Dn}(x_2) \cup \{x_4\} = \{x_1, x_4\}$.

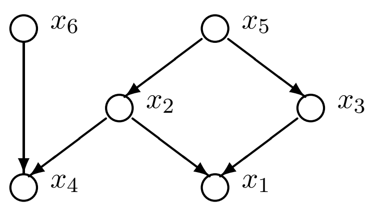


Figure 1: CUI DAG given σ and order x_1,. . . ,x_6.

Definition 7. Let $U(S)$ be a utility function representing cardinal preferences over $D(S)$. A *CUI DAG* for $U(\cdot)$ is a DAG, such that for any $x \in S$, (4) holds.

Procedure $\mathcal{C}$ yields a CUI DAG by Proposition 1. As the other direction, any given CUI DAG G (in which parents and descendants are denoted by $Pa_G(\cdot), Dn_G(\cdot)$, respectively) can be constructed using $\mathcal{C}$, as follows. Define $P(x) = Pa_G(x) \cup Dn_G(x)$ and a variable ordering according to the reverse topological order of G, and complete the execution of $\mathcal{C}$. It is straightforward to show that the set of parents selected for each x_i is exactly $Pa_G(x_i)$, hence the result is a DAG which is identical to G.

3.2 CUI Decomposition

We now show how the CUI conditions, guaranteed by Proposition 1, can be applied iteratively to decompose $U(\cdot)$ to lower dimensional functions. We first pick a variable ordering that agrees with the reverse topological order of the CUI DAG. To simplify the presentation, we rename the variables so the ordering is again $x_1, \ldots, x_n$. The CUI condition (4) on x_1 implies the following decomposition, according to (1):

$$U(S) = f_1(x_1, Pa(x_1), Dn(x_1)) + g_1(x_1, Pa(x_1), Dn(x_1)) U_{x_1^0}(S \setminus \{x_1\}). \tag{5}$$

Note that $Dn(x_1) = \emptyset$.

We assume that we have specified a reference point S^0, which is an arbitrary value chosen for each attribute $x \in S$, denoted by x^0. $U_{x_1^0}(\cdot)$ on the right hand side is the conditional utility function on S given that x_1 is fixed at a reference point x_1^0. For convenience we omit attributes whose values are fixed from the list of arguments.

By applying the decomposition based on the CUI condition of x_2 on the conditional utility function $U_{x_1^0}(\cdot)$, we get

$$U_{x_1^0}(S \setminus \{x_1\}) = f_2(x, Pa(x_2), Dn(x_2)) + g_2(x_2, Pa(x_2), Dn(x_2))U_{x_1^0,x_2^0}(S \setminus \{x_1, x_2\}). \quad (6)$$

Note that $Dn(x_2) \subseteq \{x_1\}$, and x_1 is fixed to x_1^0, hence f_2 and g_2 effectively depend only on x_2 and $Pa(x_2)$. This point is exploited below.

Substituting $U_{x_1^0(\cdot)}$ in (5) according to (6) yields:

$$U(S) = f_1 + g_1(f_2 + g_2U_{x_1^0,x_2^0}(S \setminus \{x_1, x_2\})) = f_1 + g_1f_2 + g_1g_2U_{x_1^0,x_2^0}(S \setminus \{x_1, x_2\}).$$

The list of arguments to the functions f_j, g_j are always $(x_j, Pa(x_j), Dn(x_j))$, and we omit it for readability.

We continue in this fashion and get

$$U(S) = \sum_{k=1}^{i-1}(f_k \prod_{j=1}^{k-1} g_j) + \prod_{j=1}^{i} g_j U_{x_1^0,\ldots,x_{i-1}^0}(x_i, \ldots, x_n),$$

and apply the CUI condition of x_i,

$$U_{x_1^0,\ldots,x_{i-1}^0}(x_i, x_{i+1}, \ldots x_n) = \\ f_i(x_i, Pa(x_i), Dn(x_i)) + g_i(x_i, Pa(x_i), Dn(x_i))U_{x_1^0,\ldots,x_i^0}(x_{i+1}, \ldots, x_n). \quad (7)$$

For convenience, we define the constant function $f_{n+1} \equiv U_{x_1^0,\ldots,x_n^0}()$. Ultimately we obtain

$$U(S) = \sum_{i=1}^{n+1}(f_i(x_i, Pa(x_i), Dn(x_i)) \prod_{j=1}^{i-1} g_j(x_j, Pa(x_j)), Dn(x_j)). \quad (8)$$

The variable ordering is restricted to agree with the reverse topological order of the graph, hence in (7), $Dn(x_i) \subseteq \{x_1, \ldots, x_{i-1}\}$. Therefore, all the variables in $Dn(x_i)$ on the right-hand side of (7) are fixed on their reference points, so f_i and g_i only depend on x_i and $Pa(x_i)$. Formally, let $y_1, \ldots, y_k$ be the variables in $Dn(x_i)$. With some abuse of notation, we define:

$$\begin{aligned} f_i(x_i, Pa(x_i)) &= f_i(x_i, Pa(x_i), y_1^0, \ldots, y_k^0), \\ g_i(x_i, Pa(x_i)) &= g_i(x_i, Pa(x_i), y_1^0, \ldots, y_k^0). \end{aligned} \quad (9)$$

Now (8) becomes

$$U(S) = \sum_{i=1}^{n+1}(f_i(x_i, Pa(x_i)) \prod_{j=1}^{i-1} g_j(x_j, Pa(x_j))). \quad (10)$$

This term is a decomposition of the multiattribute utility function to lower dimensional functions, whose dimensions depend on the number of variables of $Pa(x)$. As a result, the dimensionality of the representation is reduced (as in Bayesian networks) to the maximal number of parents of a node plus one.

We illustrate how the utility function is decomposed in the example of Figure 1. We pick the ordering x_4, x_1, x_6, x_3, x_2, x_5 that agrees with the reverse topological order of the graph (note that we are not renaming the variables here). To simplify notation we denote the conditional utility function in which x_i is fixed on the reference point by adding a subscript i to $U(\cdot)$.

$$\begin{aligned}
U(S) &= f_4(x_4x_2x_6) + g_4(x_4x_2x_6)U_4(S \setminus \{x_4\}) \\
U_4(S \setminus \{x_4\}) &= f_1(x_1x_2x_3) + g_1(x_1x_2x_3)U_{1,4}(S \setminus \{x_4x_1\}) \\
U_{1,4}(S \setminus \{x_4x_1\}) &= f_6(x_6) + g_6(x_6)U_{1,4,6}(x_2x_3x_5) \\
U_{1,4,6}(x_2x_3x_5) &= f_3(x_3x_5) + g_3(x_3x_5)U_{1,3,4,6}(x_2x_5) \\
U_{1,3,4,6}(x_2x_5) &= f_2(x_2x_5) + g_2(x_2x_5)U_{1,2,3,4,6}(x_5) \\
U_{1,2,3,4,6}(x_5) &= f_5(x_5) + g_5(x_5)U_{1,2,3,4,5,6}()
\end{aligned}$$

Note that each f_i and g_i depends on x_i and its parents. Merging the above equations, and using the definition $f_7 \equiv U_{1,2,3,4,5,6}()$ produces

$$U(S) = f_4 + g_4f_1 + g_4g_1f_6 + g_4g_1g_6f_3 + g_4g_1g_6g_3f_2 + g_4g_1g_6g_3g_2f_5 + g_4g_1g_6g_3g_2g_5f_7. \quad (11)$$

We established that $U(S)$ can be represented using a set of functions $\mathcal{F}$, that includes, for any $x \in S$, the functions (f_x, g_x) resulting from the decomposition (1) based on the CUI condition (4). This means that to fully specify $U(S)$ it is sufficient to obtain the data for functions in $\mathcal{F}$ (this aspect is discussed in Section 5).

Definition 8. Let $U(S)$ be a utility function representing cardinal preferences over $D(S)$. A *CUI network* for $U(\cdot)$ is a triplet $(G, \mathcal{F}, S^0)$. $G = (S, E)$ is a CUI DAG for $U(S)$, S^0 is a reference point, and $\mathcal{F}$ is the set of functions $\{f_i(x_i, Pa(x_i)), g_i(x_i, Pa(x_i)) \mid i = 1 \dots, n\}$ defined above.

The utility value for any assignment to S can be calculated from the CUI network according to (10), using any variable ordering that agrees with the reverse topological order of the DAG. In our example, we can choose a different variable ordering than the one used above, such as x_1, x_3, x_4, x_2, x_5, x_6, leading to the following expression.

$$U(S) = f_1 + g_1f_3 + g_1g_3f_4 + g_1g_3g_4f_2 + g_1g_3g_4g_2f_5 + g_1g_3g_4g_2g_5f_6 + g_1g_3g_4g_2g_5g_6f_7.$$

This sum of product is different than the one in (11). However, it is based on the same CUI decompositions and therefore the same functions (f_i, g_i).

3.3 Properties of CUI Networks

Based on Procedure $\mathcal{C}$ and the decomposition following it, we conclude the following.

Proposition 2. *Let S be a set of attributes, and σ a set of CUI conditions on S. If σ includes a condition of the form $CUI(S \setminus (x \cup Z_x), x \mid Z_x)$ for each $x \in S$, then σ can be represented by a CUI network whose dimensionality does not exceed* $\max_x(|Z_x| + 1)$.

Note that Z_x denotes here a minimal set of attributes (variables) that renders the rest CUI of x. This bound on the dimensionality will be obtained regardless of the variable ordering. We can expect the maximal dimension to be lower if the network is constructed using a good variable ordering. A good heuristic in determining the ordering would be to use attributes with smaller dependent sets first, so that the attributes with more dependents would have some of them as descendants. Based on such an ordering we would expect the less important attributes to be lower in the topology, while the more crucial attributes would either be present higher or have a larger number of parents.

From this point we usually omit the third argument when referring to a CUI condition, as in CUI(X,Y), which is taken equivalent to CUI$(X, Y \mid S \setminus (X \cup Y))$.

In order to achieve a low dimensional CUI networks, we are required to detect CUI conditions over large sets. This may be a difficult task, and we address it through an example in Section 4. The task is made somewhat easier by the fact that the set has to be CUI of a single variable; note that the condition $CUI(Y,x)$ is weaker than the condition $CUI(Y,X)$ when $x \in X$. Furthermore, Section 7 shows how the dimensionality can be reduced if the initial CUI decomposition is not sufficiently effective.

Based on properties of CUI, we can read additional independence conditions off the graph. First, we observe that CUI has a composition property at the second argument.

Lemma 3. *Let CUI(X,Y) $[X,Y \subset S]$, and CUI(A,B) $[A,B \subset S]$. Then*

$$\mathit{CUI}(A \cap X\ ,\ Y \cup B).$$

This property leads to the following claim, which allows us to derive additional CUI conditions once the graph is constructed.

Proposition 4. *Consider a CUI network for a set of attributes S. Define $Pa(X) = \bigcup_{x \in X} Pa(x)$ and $Dn(X) = \bigcup_{x \in X} Dn(x)$. Then for any $X \subset S$,*

$$\mathit{CUI}(S \setminus (X \cup Pa(X) \cup Dn(X))\ ,\ X).$$

Proof. By recursion on X, using Lemma 3 and Proposition 1. □

We also consider the other direction, by defining the set of nodes that renders a set CUI of the rest. This dual perspective becomes particularly useful for optimization (Section 6), because optimization based on the preference order over an attribute is meaningful only when holding enough other attributes fixed to make it CPI or CUI of the rest. Let $Ch(X)$ denote the union of children of nodes in X, and let $An(X)$ denote all of the ancestors of nodes in X, in both cases excluding nodes which are themselves in X.

Proposition 5. *Consider a CUI network for a set of attributes S. CUI$(X, S \setminus (X \cup An(X) \cup Ch(X)))$ for any $X \subseteq S$.*

Proof. Let $y \notin X \cup Ch(X) \cup An(X)$. Then clearly $\forall x \in X, x \notin Pa(y) \cup Dn(y)$. Hence from Proposition 1, CUI(X,y). We apply Lemma 3 iteratively for each $y \notin X \cup Ch(X) \cup An(X)$ (note that the first argument is X for each CUI condition, so it is X in the result as well), and get the desired result. □

We conclude this section by relating CUI networks to CAI maps.

Proposition 6. *Let $G = (X, E)$ be a CAI map, and $x_1, \ldots, x_n$ an ordering over the nodes in X. Let $G' = (X, E')$ be the DAG such that there is a directed arc (x_i, x_j) in E' iff $i < j$ and $(x_i, x_j) \in E$. Then G' is a CUI network.*

We note, however, that CAI maps decompose the utility function over the maximal cliques, whereas CUI networks decompose over nodes and their parents. Section 7 bridges this gap. In addition, this result is used in Section 6.3.

4. CUI Modeling Example

To demonstrate the potential representational advantage of CUI networks we require a domain that is difficult to simplify otherwise. The example we use is the choice of a software package by an enterprise that wishes to automate its sourcing (strategic procurement) process. We focus on the software's facilities for running auction or RFQ (request for quotes) events, and tools to select winning suppliers either manually or automatically.

We identified nine key features of these kinds of software packages. In our choice scenario, the buyer evaluates each package on these nine features, graded on a discrete scale (e.g., one to five). The features are, in brief:

Interactive Negotiations (*IN*) allows a separate bargaining procedure with each supplier.

Multi-Stage (*MS*) allows a procurement event to be comprised of separate stages of different types.

Cost Formula (*CF*) buyers can formulate their total cost of doing business with each supplier.

Supplier Tracking (*ST*) allows long-term tracking of supplier performance.

MultiAttribute (*MA*) bidding over multiattribute items, potentially using a scoring function.[1]

Event Monitoring (*EM*) provides an interface to running events and real-time graphical views.

Bundle Bidding (*BB*) bidding for bundles of goods.

Grid Bidding (*GB*) adds a bidding dimension corresponding to an aspect such as time or region.

Decision Support (*DS*) tools for optimization and for aiding in the choice of the best supplier(s).

We observe first that additive independence does not widely apply in this domain. For example, Multi-Stage makes several other features more useful or important: Interactive Negotiations (often useful as a last stage), Decision Support (to choose which suppliers

1. We hope the fact that the software itself may include facilities for multiattribute decision making does not cause undue confusion. Naturally, we consider this an important feature.

proceed to the next stage), and Event Monitoring (helps keep track of how useful was each stage in reducing costs). Conversely, in some circumstances Multi-Stage can substitute for the functionality of other features: MultiAttribute (by bidding on different attributes in different stages), Bundle Bidding (bidding on separate items in different stages), Grid Bidding (bidding on different time/regions in different stages) and Supplier Tracking (by extracting supplier information in a "Request for Information" stage). The potential dependencies for each attribute are shown in Table 3.

Attr	Complements	Substitutes	CUI set
EM	CF ST MS		IN,DS,MA,GB,BB
IN	ST MS MA		EM,CF,ST,DS,GB,BB
CF	EM MS DS MA GB BB	DS	MA,GB,BB
ST	EM MS IN DS	MA	IN,CF,GB,BB
MA	IN DS CF	MS BB ST GB	GB,BB
MS	DS EM IN ST	GB BB MA CF	MA,GB,BB
DS	CF MA GB ST BB MS		IN,EM
GB	CF DS	MA MS BB	MA,BB
BB	CF DS	MA MS GB	MA,GB

Table 3: Dependent and independent sets for each attribute.

The presence of a complement or substitute relation precludes additive independence. From this fact we can identify a set of six attributes that must be mutually (additive) dependent: $\{BB, GB, DS, MA, MS, CF\}$. In consequence, the best-case dimensionality achieved by a CAI map (and other CAI-based representations, see Section 2.2), for this domain would be six, the size of the largest maximal clique.

In order to construct a CUI network we first identify, for each attribute x, a set Y that is CUI of it. We can first guess such a set according to the complement/substitute information in Table 3; typically, the set of attributes that are neither complements nor substitutes would be CUI. This is the approach taken for the attributes EM and DS. However, attributes that are complements or substitutes may still be CUI of each other, and we therefore attempt to detect and verify potentially larger CUI sets. Keeney and Raiffa (1976) provide several useful results that can help in detection of UI, and those results can be generalized to CUI. In particular they show that we can first detect a *conditional preferential independence* (CPI) condition in which one element is also CUI. Based on this result, in order to verify for example that

$$\mathrm{CUI}(\{BB, GB, MA\}, CF \mid S \setminus \{BB, GB, MA, CF\}), \tag{12}$$

the following two conditions are sufficient:

$$\mathrm{CPI}(\{BB, GB, MA\}, CF \mid S \setminus \{BB, GB, MA, CF\}), \tag{13}$$

$$\mathrm{CUI}(BB, \{GB, MA, CF\} \mid S \setminus \{BB, GB, MA, CF\}). \tag{14}$$

Detection and verification of these conditions are also discussed by Keeney and Raiffa (1976). For our example, we observe that the features BB, GB, and MA each add a qualitative element to the bidding. Each bidding element is best exploited when cost formulation is available, so complements CF. The complementarity is similar for each feature, thus implying (13). Moreover, BB is a crucial feature and therefore the risk attitude towards it is not expected to vary with the level of CF, MA, and GB, and that implies (14), together leading to (12).

In a similar fashion, we observe that the nature of the substitutivity of the three mechanisms BB, GB, MA in MS is similar: each can be simulated using multiple stages. That means that the tradeoffs among the three do not depend on MS, meaning that CPI($\{BB, GB, MA\}, MS$) holds. Next, the dependency among the triplet $\{BB, GB, MA\}$ is also a result of the option to substitute one by another. As a result, each pair is CPI of the third. Finally, we find that the complementarity of ST and IN is marginal and does not affect the tradeoffs with other attributes. We can therefore verify the following conditions: CUI($\{BB, GB, MA\}, MS$), CUI($\{BB, GB\}, MA$), CUI($\{ST, EM, CF, DS, GB, BB\}, IN$), and CUI($\{GB, BB, CF, IN\}, ST$). The resulting maximal CUI sets for each attribute are shown in Table 3.

To construct the network we start with the variable with the largest CUI set, IN, which needs only MS and MA as parents, after which it is EM that gets CF, MS, and ST as parents. Next, we consider ST which needs four attributes in its conditional set, but EM is a descendant, therefore only DS, MS, and MA are needed as parents. The next variable to choose is MS, which needs only CF and DS as parents since the other dependant variables are descendants. Had we chosen CF before MS it would have needed four parents: IN, MS, ST, and DS (note that although IN is CUI of CF and so is the set $\{BB, GB, MA\}$, this is not the case for the union $\{BB, GB, MA, IN\}$). Now that we choose CF after MS it has MS, ST, and IN as descendants and therefore only DS is a parent. The complete variable ordering is IN, EM, ST, MS, CF, DS, MA, GB, BB, and the resulting CUI network is depicted in Figure 2. The maximal dimension is four.

The structure we obtained over the utility function in the above example is based largely on objective domain knowledge, and may be common to various sourcing departments. This demonstrates an important aspect of graphical modeling captured by CUI networks: encoding qualitative information about the domain, thus making the process of extracting the numeric information easier. This structure in some cases differs among decision makers, but in other cases (as above) it makes sense to extract such data from domain experts and reuse this structure across decision makers.

5. Representation and Elicitation

In this section, we derive an expression for local node data in terms of conditional utility functions, and discuss how to elicit utility information from judgments about relative preference differences.

5.1 Node Data Representation

Representing U by a CUI network requires that we determine the f and g functions for each CUI condition. At any node y the functions f, g represent the affine transformation of the conditional utility function $U(x^0, Y, Z)$ (here $Z = Pa(x)$) to strategically equivalent utility functions for other values of x. Like the transformation functions for UI (Keeney & Raiffa,

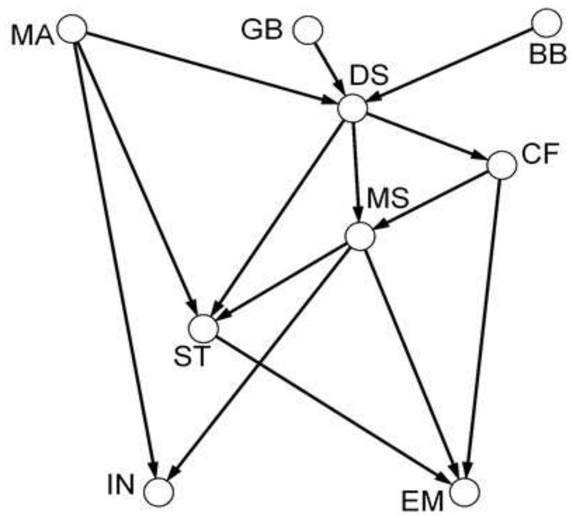


Figure 2: CUI network for the example. The maximal number of parents is 3, leading to dimension 4.

1976), the transformation functions for CUI can be represented in terms of the conditional utility functions $U(x, Y^1, Z)$ and $U(x, Y^2, Z)$ for suitable values Y^1 and Y^2 (see below). We can determine f and g by solving the system of two equations below, both based on applying (1) for these specific values of Y:

$$U(x, Y^1, Z) = f(x, Z) + g(x, Z)U(x^0, Y^1, Z),$$
$$U(x, Y^2, Z) = f(x, Z) + g(x, Z)U(x^0, Y^2, Z),$$

yielding

$$g(x, Z) = \frac{U(x, Y^2, Z) - U(x, Y^1, Z)}{U(x^0, Y^2, Z) - U(x^0, Y^1, Z)}, \tag{15}$$
$$f(x, Z) = U(x, Y^1, Z) - g(x, Z)U(x^0, Y^1, Z). \tag{16}$$

The only restriction on the choice of Y^1, Y^2 is that the decision maker must not be indifferent between them given x^0 and the current assignment to Z. For example, Y^1, Y^2 may differ on any single attribute $y \in Y$ that is *strictly essential*.

5.2 Elicitation of Measurable Value Functions

A utility function that is a used for choosing an action that leads to a known probability distribution over the outcomes, should be obtained through elicitation of preferences over lotteries, for example using even-chance gambles and their *certainty equivalents* (Keeney & Raiffa, 1976). Based on the preceding discussion, to fully specify $U(\cdot)$ via a CUI network, we need to obtain the numeric values for the conditional utility functions $U(x, Y^1, Pa(x))$ and $U(x, Y^2, Pa(x))$ for each node x. This is significantly easier than obtaining the full n-dimensional function, and in general can be done using methods described in preference elicitation literature (Keeney & Raiffa, 1976). In this section we show how elicitation can be conducted in cases when the choice is assumed to be done over certain outcomes, but a cardinal representation is nevertheless useful.

For particular applications we can point out specific attributes that can be used as a measurement for others. The most common example is preferences that are quasi-linear in a special attribute such as money or time. These kind of preferences can be represented by a *measurable value function*, or *MVF* (Krantz et al., 1971; Dyer & Sarin, 1979). An MVF is a cardinal utility function defined under certainty and represents *preference differences*. It has been shown (Dyer & Sarin, 1979) that UI has an analogous interpretation for MVF with similar resulting decomposition. The extension to CUI is straightforward.

For the case of monetary scaling, the preference difference over a pair of outcomes represents the difference in *willingness to pay* (wtp) for each. A potential way to elicit the MVF is by asking the decision maker to provide her wtp to improve from one outcome to another, particularly when these outcomes differ over a single attribute.

Under this interpretation, we first observe from (15) that $g(x, Z)$ can be elicited in terms of preference differences, between outcomes that possibly differ over a single attribute. The result can convey qualitative preference information. Assume $Y^2 \succeq Y^1$ and that $\forall x . x^0 \preceq x$. Then $g(x, Z)$ is the ratio of the preference difference between Y^1 and Y^2 given x to the same difference given x^0 (Z is fixed in all outcomes). Hence, if Y and x are complements then $g(x, Z) > 1$ and increasing in x. If Y and x are substitutes, $g(x, Z) < 1$ and decreasing in x. This holds regardless of the choice for Y^1, Y^2, since by CUI($Y, x \mid Z$) all attributes in Y maintain the same complementarity or substitutivity relationship to x. Note also that $g(x, Z) = 1$ iff $CAI(Y, x \mid Z)$. Another important observation is that though both Y and x may depend on Z, in practice we do not expect the *level of dependency* between Y and x to depend on the particular value of Z. In that case g becomes a single-dimensional function, independent of Z.

$f(x, Z)$, intuitively speaking, is a measurement of *wtp* to improve from x^0 to x. The value $U(x^0, Y^1, Z)$ is multiplied by $g(x, Z)$ to compensate for the interaction between Y and x, allowing $f(\cdot)$ to be independent of Y. If we perform the elicitation obeying the topological order of the graph, the function $U(x^0, Y^1, Z)$ can be readily calculated for each new node from data stored at its predecessors. Choose $Y^1 = Y^0$, and let $Z = \{z_1, \dots, z_k\}$, ordered such that children precede parents. Since Y, x are fixed on the reference point,

$$U(x^0, Y^0, Z) = \sum_{i=1}^{k} (f_{z_i} \prod_{j=1}^{i-1} g_{z_j}) f_{n+1}().$$

Now we can obtain $f(x, Z)$ as follows: first we elicit the preference difference function $e(x, Z) = U(x, Y^1, Z) - U(x^0, Y^1, Z)$. Then, assuming $g(x, Z)$ was already obtained, calculate:

$$f(x, Z) = e(x, Z) - (g(x, Z) - 1) U(x^0, Y^1, Z).$$

6. Optimization

One of the primary uses of utility functions is to support optimal choices, as in selecting an outcome or action. The complexity of the choice depends on the specific properties of the environment. When the choice is among a limited set of definite outcomes, we can recover the utility of each outcome using the compact representation and choose the one with the highest value. For instance, in the software example of Section 4 we would normally choose among an enumerated set of vendors or packages. In this procurement scenario we assume the utility is an MVF, and we usually choose the outcome that yields the highest utility net of price. In case of decision under uncertainty, when the choice is among actions that lead to probability distributions over outcomes, the optimal choice is selected by computing the expected utility of each action. If each action involves a reasonably bounded number of outcomes with non-zero probability, this again can be done by exhaustive computation.

Nevertheless, it is often useful to directly identify the maximal utility outcome given a quantitative representation of utility. In case of a direct choice over a constrained outcome space, the optimization algorithm serves as a subroutine for systematic optimization procedures, and such can be adapted from the probabilistic reasoning literature (Nilsson, 1998). The algorithm may also be useful as a heuristic aid for optimization of expected utility or net utility mentioned above, when the set of possible outcomes is too large for an explicit, exhaustive choice.

In this section, we develop optimization algorithms for discrete domains, and show how in many cases CUI networks can provide leverage for optimization of CAI maps. As is typical for graphical models, our optimization algorithm is particularly efficient when the graph is restricted to a tree.

6.1 Optimization Over CUI Trees

Definition 9. A *CUI tree* is a CUI network in which no node has more than one child.

Note that this type of graph corresponds to an upside-down version of a standard directed tree (or a forest).

Let T be a CUI tree. We assume WLOG that T is connected (a forest can be turned into a tree by adding arcs). As an upside-down sort of tree, it has any number of roots, and a single leaf. We denote the root nodes by $a_i \in \{a_1, \ldots, a_k\}$, the child of a_i by b_i, and so on. For each root node a_i, we define the function

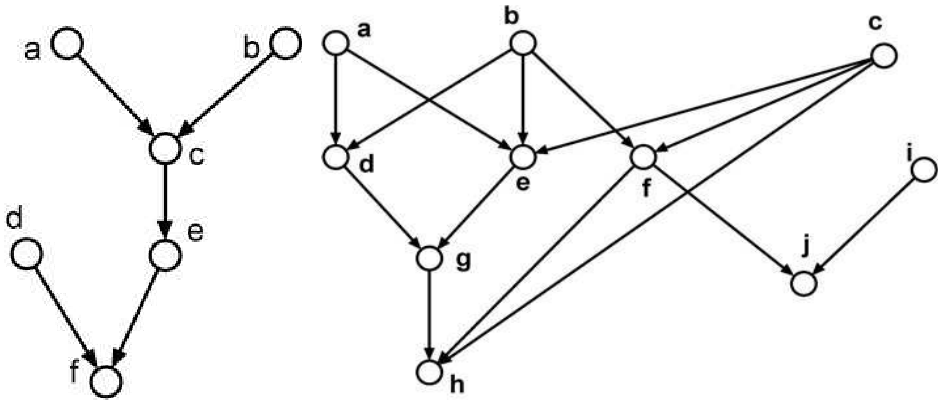
$$h_{a_i}(b_i) = \arg \max_{a_i' \in D(a_i)} U(b_i, a_i'),$$

denoting the selection of an optimal value of a_i corresponding to a given value of its child. From Proposition 5, h_{a_i} does not depend on the reference values chosen for $S \setminus \{a_i, b_i\}$. The function $h_{a_i}(\cdot)$, which we call the *optimal value function* (OVF) of a_i, is stored at node a_i since it is used by its descendants as described below.

Next, each b_i has no children or a single child c_i, and any number of parents. For simplicity of exposition we present the case that b_i has two parents, a_i and a_j. The maximization function for b_i is defined as

$$h_{b_i}(c_i) = \arg \max_{b_i' \in D(b_i)} U(c_i, b_i', h_{a_i}(b_i'), h_{a_j}(b_i')).$$

In words, we pick the optimal value of b_i for each assignment to its child and its parents. But since we already know the optimum of the parents for each value of b_i, we need only consider this optimum for each evaluation on the domain of b_i.



(a) (b)

Figure 3: CUI networks in optimization examples: (a) Tree (b) Non-tree

The only external child of the set $\{a_i, a_j, b_i\}$ is c_i, and it has no external ancestors, hence $\{a_i, a_j, b_i\}$ is CUI of the rest given c_i, therefore the maximization above again does not depend on the reference values of the rest of the attributes. Similarly, when computing $h_{c_i}(d_i)$ for the child c_i of b_i, each value for c_i fixes b_i (and any other parents of c_i), and that fixes a_i and a_j (and the other ancestors of c_i). The last computation, at the leaf x, evaluates each value of x. Each value x' causes this cascade of fixed values to all of the ancestors, meaning we finally get the optimal choice by comparing $|D(x)|$ complete assignments.

We illustrate the execution of the algorithm on the CUI tree of Figure 3a. We compute $h_a(c)$ which is the optimal value of a for each value of c, and similarly $h_b(c)$. Next, to compute $h_c(e)$, for each value e' of e we compare all outcomes $(e', c', h_a(c'), h_b(c'))$, $c' \in D(c)$. At node d we compute $h_d(f)$, which is independent of the other nodes. At node e we compute $h_e(f) = \arg\max_{e'} U(f, e', h_c(e'), h_b(h_c(e')), h_a(h_c(e'))$ (node d can be ignored here) and at node f it is

$$h_f() = \arg\max_{f' \in D(f)} U(f', h_e(f'), h_d(f'), h_c(h_e(f')), h_b(h_c(h_e(f'))), h_a(h_c(h_e(f'))).$$

Note how each candidate value of f causes the cascade of optimal values to all of its ancestors. The solution is then $h_f()$ and the resulting values of all the ancestors.

This optimization algorithm iterates over the nodes in topological order, and for each x_i it calculates the OVF $h_{x_i}(x_j)$, where x_j is the child of x_i. This calculation uses the values of the OVF stored for its parents, and therefore involves comparison of $|D(x_i)||D(x_j)|$ outcomes. In case the numeric data at the nodes is available, factoring in the time it takes to recover the utility value for each outcome (which is $O(n)$), the algorithm runs in time $O(n^2 \max_i |D(x_i)|^2)$.

6.2 Optimization Over General DAGs

A common way in graphical models to apply tree algorithms to non-trees is by using the *junction graph*. However, the common notion of a junction graph for DAG is a polytree, whereas our algorithm above is specialized to a (unit) tree. Instead, we optimize the CUI network directly by generalizing the tree algorithm.

In the tree case, fixing the value of the child of a node x is sufficient in order to separate x from the rest of the graph, excluding ancestors. We consider each value of the child at a time, so it also determines the values for all the ancestors. In a general DAG it is no longer sufficient for the OVF to depend on the children, because they do not provide sufficient information to determine the values of $An(x)$. Hence we generalize this notion to be the *scope* of x ($Sc(x)$, defined below), which is a set of nodes on which the OVF of x must depend, in order for an iterative computation of the OVF to be sound.

With this generalization, the DAG algorithm is similar to the tree algorithm. Let G be a CUI network, and $x_1, \ldots, x_n$ a variable ordering that agrees with the topological order of G (parents precede children). For each x_i (according to the ordering), compute $h_{x_i}(Sc(x_i))$ for any instantiation of $Sc(x_i)$. The optimal instantiation can now be selected backwards from $h_{x_n}()$, since for each node x_i that is reached the values for $Sc(x_i)$ are already selected.

$Sc(x_i)$ is computed as follows: scan variables $x_{i+1}, \ldots, x_n$ in this order. When scanning x_j, add x_j to $Sc(x_i)$ if the following conditions hold:

1. There is an undirected path between x_j and x_i.

2. The path is not blocked by a node already in $Sc(x_i)$.

By these conditions, $Sc(x_i)$ includes all the children of x_i, but non of x_i's ancestor since they precede x_i in the ordering. In addition, $Sc(x_i)$ includes all nodes that are needed to block the paths that reach x_i through its ancestors. For example, if x_k, x_j are children of an ancestor x_a of x_i, and $k < i < j$, then x_j must be in $Sc(x_i)$, because of the path through x_a. The children of x_j are blocked by x_j, so unless they have another path to x_i they will not be in $Sc(x_i)$. The children of x_k, if ordered later than x_i, will be in $Sc(x_i)$ (but their children will not), and so on.

Figure 3b is an example of a CUI network that is not a tree. We consider the scopes under the variable ordering $a, b, \ldots, j$. The scope of roots always equals their set of children (because there is no other path reaching them), meaning $Sc(a) = \{d, e\}$, $Sc(b) = \{d, e, f\}$, $Sc(c) = \{e, f, h\}$, $Sc(i) = \{j\}$. The scope of d must include its child g and its siblings e and f. All paths of h, j, and i to d are blocked by g, e, f therefore $Sc(d) = \{g, e, f\}$. For e, we must include its child g, and its "younger" sibling f. h has a blocked path to e through $f \in Sc(e)$, but also a non-blocked one through $c \notin Sc(e)$, therefore $Sc(e) = \{g, f, h\}$. Similarly, g and h are in the scope of f due to paths through b and c respectively, hence $Sc(f) = \{g, h, j\}$. For g, in addition to its child h we add j whose path to g through f, b, e is not blocked ($Sc(g) = \{h, j\}$) and finally $Sc(h) = Sc(i) = \{j\}$ and $Sc(j) = \{\}$.

The next step, computing the OVF, requires that we compare a set of outcomes that differ on $x_i \cup Co(x_i)$, where $Co(x_i)$ is a set of nodes whose OVF can be determined by $x_i \cup Sc(x_i)$ (hence they are *covered* by x_i). For this maximization to be valid, the condition $\mathrm{CUI}(x_i \cup Co(x_i), S \setminus (x_i \cup Co(x_i) \cup Sc(x_i)))$ must hold. We formally define $Co(x_i)$, and establish this result which is proved in the appendix.

Definition 10. $Co(x_i)$ is the smallest set of nodes that satisfied the following condition

$$\forall j < i, \;\; Sc(x_j) \subseteq (\{x_i\} \cup Sc(x_i) \cup Co(x_i)) \rightarrow x_j \in Co(x_i). \tag{17}$$

Intuitively, x_j is covered by x_i if each node $x_k \neq x_i$ in its scope, is either in the scope of x_i or was determined (according to its own scope) to be covered by x_i. In Figure 3b, $f \in Co(g)$ because $Sc(f) = \{g\} \cup Sc(g)$. $e \in Co(g)$ because $Sc(e) \subset \{g\} \cup Sc(g) \cup \{f\}$. Moreover, $Sc(d) = \{g, e, f\}$ hence $d \in Co(g)$ as well, and similarly we find that $a, b \in Co(g)$. In this example all the nodes preceding g in the ordering are covered, but this is not necessarily always the case.

Lemma 7. *An assignment to x_i and $Sc(x_i)$ is sufficient to determine $h_{x_j}(\cdot)$ for each $x_j \in Co(x_i)$.*

Lemma 8. *For any node x_i, CUI$(\{x_i\} \cup Co(x_i), S \setminus (\{x_i\} \cup Co(x_i) \cup Sc(x_i)))$. Meaning that x_i and the nodes it covers are CUI of the rest given $Sc(x_i)$.*

When the algorithm reaches node x_i, every choice of assignment to $Sc(x_i) \cup \{x_i\}$ determines optimal values for $Co(x_i)$ (Lemma 7). We compare the $|D(x_i)|$ assignments which differ over the values of x_i and $Co(x_i)$, and select an optimal one as the value of $h_{x_i}(Sc(x_i))$. This optimum does not depend on the nodes in $S \setminus (\{x_i\} \cup Co(x_i) \cup Sc(x_i)))$ due to Lemma 8.

To illustrate, we examine what happens when the algorithm reaches node g in Figure 3b. At this point $h_x(Sc(x))$ is known for any x that precedes g. As showed, all these nodes are in $Co(g)$. Indeed, the assignment to $Sc(g) = \{g, h, j\}$ directly determines the value for $h_f(\cdot)$, and then together with $h_f(\cdot)$ it determines the value for $h_e(\cdot)$, and further it cascades to the rest of the nodes. The CUI network shows that CUI$(\{a, b, c, d, e, f, g\}, \{i\})$ (given $\{h, j\}$) and therefore the maximization operation (over the choice of value for g) is valid regardless of the value of i.

The performance of the optimization algorithm is exponential in the size of the largest scope (plus one). Note that this would be seriously affected by the choice of variable ordering. Also note, that in the case of a tree this algorithm specializes to the tree optimization above, since there is no node that has a path to an ancestor of x_i, except for other ancestors of x_i which must precede x_i in the ordering. Therefore it is always the case that $Sc(x_i) = Ch(x_i)$, meaning that $h_{x_i}(\cdot)$ is a function of its single child. Based on that, we expect the algorithm to perform better the more similar the CUI network is to a tree.

6.3 CUI Tree for Optimization of CAI Maps

The optimization procedure for CUI trees is particularly attractive due to the relatively low amount of preference information it requires. In some cases the comparison can be done directly, without even having the data that comprises the utility function. Aside from the direct benefit to CUI networks, we are interested in applying this structure to the optimization of CAI maps. In some domains a CAI map is a simple and effective way to decompose the utility function. However, the optimization of CAI maps is exponential in the size of its tree width, and it requires the full data in terms of utility functions over its maximal cliques. If a CAI map happens to have a simple structure, such as a tree, or the CP condition, faster optimization algorithms can be used. However, it could be the case that a CAI map is not a tree, but more subtle CUI conditions might exist which cannot be captured by CAI conditions. If enough such conditions could be detected to turn the CAI map into a CUI tree (or close enough to a tree), we could take advantage of our simple optimization procedure.

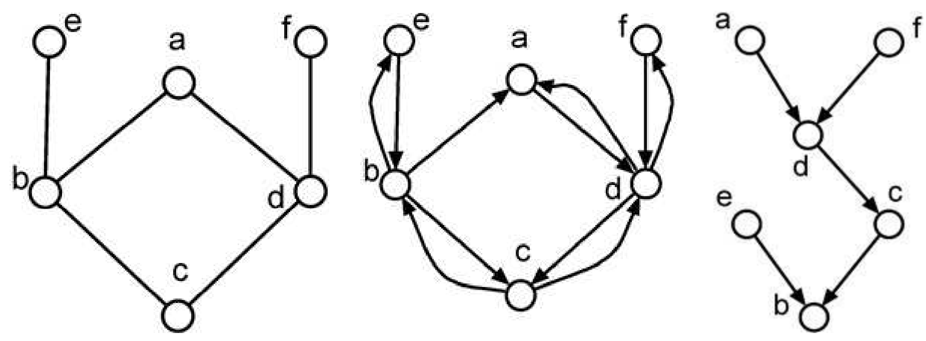


Figure 4: (a) A CAI map containing a cycle. (b) Enhanced CAI map, expressing CUI of $\{a, d, f\}$ in b. (c) An equivalent CUI tree.

Definition 11. Let $G = (V, E)$ be a CAI map. An *enhanced CAI map* is a directed graph $G' = (V, A)$, in which a pair of arcs $(u, v), (v, u) \in A$ implies the same dependency as an edge $(u, v) \in E$, and in addition for any node x, $\text{CUI}(S \setminus (\{x\} \cup \text{In}(x)), x)$ ($\text{In}(x)$ denoting the set of nodes y for which $(y, x) \in A$). We call the pair of arcs $(u, v), (v, u) \in A$ a *hard link* and an arc $(u, v) \in A$ s.t. $(v, u) \notin A$ a *weak link*.

For any CAI map, an enhanced CAI map can be generated by replacing each edge (u, v) with the arcs (u, v) and (v, u). This does not require any additional CUI conditions because these are entailed by the CAI map. However, if additional CUI conditions as above can be detected, we might be able to remove one (or both) of the directions. Figure 4a shows a CAI map which contains a cycle. If we could detect that $\text{CUI}(\{a, d, f\}, b)$, we could remove the direction (a, b) and get the enhanced CAI map in Figure 4b. The set of CUI conditions implied by the enhanced CAI map can now be expressed by a CUI tree, as in Figure 4c.

Proposition 9. *Consider an enhanced CAI map G. Let ω be an ordering on the nodes of G, and G' a DAG which is the result of removing all arcs (u, v) whose direction does not agree with ω. If for any such removed arc, v is an ancestor of u in G', then G' is a CUI network.*

For hard links, the removal of (u, v) leaves v as a parent of u, so the condition trivially holds. To obtain a CUI tree, the key is therefore to find a variable ordering under which enough weak links can be removed to turn the graph into a tree, maintaining the condition of Proposition 9. For a large number of variables, an exhaustive search over variable orderings may not be feasible. However in many cases it can be effectively constrained, restricting the number of orderings that we need to consider. For example, in order to break the cycle in Figure 4b it is clear that the weak link (b, a) must be implied by the ordering, so that a could be an ancestor of b. The only way for this to happen (given the existing hard links), is that c is a parent of b, d is a parent of c, and a a parent of d.

Proposition 10. *Let $c = (y_1, \ldots, y_k)$ be a cycle in an enhanced CAI map G. Assume that c contains exactly one weak link: (y_i, y_{i+1}) for some $i < k$, or (y_k, y_1). Let ω be a variable* *ordering that does not agree with the order of the path* $p = (y_{i+1}, y_{i+2}, \ldots, y_k, y_1, \ldots, y_i)$. *Then any CUI network constructed from* G *and* ω *(by Proposition 9), is not a tree.*

Therefore any cycle that contains one weak link leads to a constraint on the variable ordering. Cycles with more than one weak link also lead to constraints. If c above has another weak link (y_j, y_{j+1}), one of the two links must be removed, and the ordering must agree with either the path p above or the path $p' = (y_{j+1}, y_{j+2}, \ldots, y_k, y_1, \ldots, y_j)$. Assuming WLOG that $j > i$, the paths $(y_{i+1}, \ldots, y_j)$ and $(y_{j+1}, \ldots, y_i)$ are required for both p and p', and therefore can be used as constraints. Similarly we can find the intersection of the paths implied by any number of weak links in a cycle.

Sometimes the constraint set can lead to an immediate contradiction, and in such case search is redundant. If it does not, it can significantly reduce the search space. However, the major bottleneck in preference handling is usually elicitation, rather than computation. Therefore, given that a good variable ordering may lead to the reduction of the optimization problem to a simpler, qualitative task, eliminating the need for a full utility elicitation, it would be worthwhile to invest the required computation time.

7. Nested Representation

From Section 5.1 we conclude that node data can be represented by conditional utility functions depending on the node and its parents. But this may not be the best dimensionality that can be achieved by a network. Perhaps the set $Z = Pa(x)$ has some internal structure, in the sense that the subgraph induced by Z has maximal dimension lower than $|Z|$. In such a case we could recursively apply CUI decomposition to the conditional utility functions for this subgraph. This approach somewhat resembles the hierarchical decomposition done for the utility trees (Keeney & Raiffa, 1976; Von Stengel, 1988). For example, to represent f_1 for the network of Figure 1, we require the conditional utility function $U(x_1, x_4^1, x_5^1, x_6^1, x_2, x_3)$. However from the network we can see that $\mathrm{CUI}(x_3, x_2 \mid x_1, x_4, x_5, x_6)$. Hence we can decompose this conditional utility:

$$U(x_1, x_4^1, x_5^1, x_6^1, x_2, x_3) = f'(x_1, x_2) + g'(x_1, x_2)U(x_1, x_3, x_2^0, x_4^1, x_5^1, x_6^1).$$

We use the notation f' and g' since these are not the same as the f and g functions of the top level decomposition.

A nested representation can be generated systematically (Algorithm 1), by decomposing each local function at node x (x's *utility factors*) whose argument set $Z \subseteq Pa(x)$ does not form a clique. We do that by performing a complete CUI decomposition over the subgraph induced by Z (keeping in mind that all the resulting factors depend also on x).

Proposition 11. *Let* G *be a CUI network for utility function* $U(S)$*. Then* $U(S)$ *can be represented by a set of conditional utility functions, each depending on a set of attributes corresponding to (undirected) cliques in* G.

7.1 Discussion

This result reduces the maximal dimensionality of the representation to the size of the largest maximal clique of the CUI network. For instance, applying it to the example in

Data: CUI Utility factors $U(x, Pa(x), \hat{Y}), U(x, Pa(x), \tilde{Y})$ at each node x
  `/* note:` $\hat{Y}, \tilde{Y} \in D(Y)$ `*/`

Determine order $x_1, \ldots, x_n$;

for $j = 1, \ldots, n$ **do** `/* initialization */`
  $K_j^1 = \{x_j\} \cup Pa(x_j)$ `/* scope of utility factors */` ;
  $Y_j^1 = S \setminus K_j^1$ `/* rest of variables */` ;
  $Q_j^1 = Pa(x_j)$;
  $A_j^1 = \emptyset,\ \ d_j = 1$;
end

for $j = 1, \ldots, n$ **do**
  for $i = 1, \ldots, d_j$ **do** `/* loop on factors in node` j`*/`
    if $Q_j^i \neq \emptyset$ *and* K_j^i *is not a clique* **then**
      Let G_j^i be the subgraph induced by Q_j^i;
      Decompose $U_j^i(K_j^i)$ according the CUI network on G_j^i;
      foreach $x_r \in Q_j^i$ **do**
        Let $d_r = d_r + 1$ (current num. of factors at x_r) and denote $d = d_r$;
        $A_r^d = A_j^i \cup \{x_j\}$, $Q_r^d = Pa(x_r) \cap Q_j^i$;
        $K_r^d = A_r^d \cup \{x_r\} \cup Q_r^d$, $Y_r^d = S \setminus K_r^d$;
        Store new CUI factors of x_r: $U(K_r^d, \hat{Y}_r^d), U(K_r^d, \tilde{Y}_r^d)$;
          `/*`$\hat{Y}_r^d, \tilde{Y}_r^d$ `are fixed assignments to` Y_r^d`*/`
      end
      Remove factors $U(K_j^i, \hat{Y}_j^i), (K_j^i, \tilde{Y}_j^i)$;
    end
  end
end

Algorithm 1: Recursive CUI decomposition. Process node in reverse topological order (outermost loop). Decompose each factor stored at current node, whose parents do not form a clique. At each such parent x_r (innermost loop) store resulting new factors. These are defined over x_r, those of x_r's parents that are also in $Pa(x_i)$ (this is Q_r^d), and a clique A_r^d on which the original factor depends. Each time a factor is decomposed its set Q shrinks. When it is empty, K is a clique.

Section 4 reduces dimensionality from four to three. An important implication is that we can somewhat relax the requirement to find very large CUI sets. If some variables end up with many parents, we can reduce dimensionality using this technique. As the example below illustrates, this technique aggregates lower order CUI conditions to a more effective decomposition.

The procedure may generate a complex functional form, decomposing a function multiple times before the factors become restricted to a clique. The ultimate number of factors required to represent $U(S)$ is exponential in the number of such nesting levels. However, each decomposition is based on a CUI network on a subgraph, and therefore typically reduces the number of entries that are maintained.

We expect the typical application of this technique to be for *composition* rather than *decomposition*. We execute Algorithm 1 without the actual data, resulting in a list of factors per node (that are conditional utility functions over cliques of the graph). That means that for elicitation purposes we can restrict attention to conditional utility functions over maximal cliques. Once these are obtained, we have sufficient data for all the factors. We can then recover the original, more convenient CUI-network representation of the function and store it as such (more on that in the example below). Therefore, the effective dimensionality for elicitation is that of the maximal cliques. The storage for efficient usage requires the potentially higher dimension of the original CUI network, but typically this is less of a concern.

With this result and Proposition 6, CUI networks are shown to always achieve weakly better dimensionality than CAI maps, since both representations reduce the dimensionality to the size of the maximal clique.

7.2 Example

We illustrate this result using a simple example. Consider a domain with four attributes (a, b, c, d), and the following CUI conditions:

$$\mathrm{CUI}(b, c), \mathrm{CUI}(c, b), \mathrm{CUI}(d, a)$$

The CUI network corresponding to the variable ordering a, b, c, d is depicted in Figure 5. Since the CUI sets are small (a single variable each), for any variable ordering there must be a node with two parents, meaning dimensionality of three. The nesting operation below combines these lower order conditions to reduce the dimensionality to two.

Initially, the utility function is represented using the conditional utility functions listed according to their corresponding nodes in the column "Level 0" in Table 4. To remove the three-dimensional factors, we need to decompose the functions of node a according to the CUI network on $\{b, c\}$, which contains no arcs. This proceeds as follows:

$$U(a, b, c, d^1) = \\ f_b^1(a, c) + g_b^1(a, c)U(a, b, c^0, d^1) = f_b^1(a, c) + g_b^1(a, c)(f_c^1(a, b) + g_c^1(a, b)U(a, b^0, c^1, d^1))$$

$$U(a, b, c, d^2) = \\ f_b^2(a, c) + g_b^2(a, c)U(abc^0d^2) = f_b^2(a, c) + g_b^2(a, c)(f_c^2(a, b) + g_c^2(a, b)U(a, b^0, c^0, d^2))$$

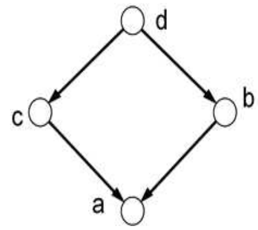


Figure 5: Nesting example

The resulting functions are $f_b^i(a,c)$, $g_b^i(a,c)$, $f_c^i(a,b)$, $g_c^i(a,b)$, $i = 1, 2$. The functions $f_b^i(a,c)$ and $g_b^i(a,c)$ can be represented using the conditional utility functions $U(a, b^1, c, d^i)$ and $U(a, b^2, c, d^i)$, and similarly the other two functions. We can delete the factors of a, $U(a, b, c, d^i)$, and add the new lower dimensional factors as a second column to a's parents b and c. Though we had to multiply the number of factors we store by four, all the new factors are conditional utility functions over subdomains of the (deleted) higher dimensional factors. The algorithm continues to node b, and loops over its six factors. If there are factors that are defined over a set of parents of b that are not a clique it decomposes them and store the new factors in the next table column. In case a factor from column "level 1" could be decomposed, we would add a "level 2" column to store the result. In our simple example no further decomposition is possible.

Attr	Level 0 (CUI net)	Level 1
a	$\cancel{U(a,b,c,d^1)}$, $\cancel{U(a,b,c,d^1)}$	
b	$U(a^0, b, c^1, d)$, $U(a^0, b, c^2, d)$	$U(a, b, c^1, d^1)$, $U(a, b, c^2, d^1)$ $U(a, b, c^1, d^2)$, $U(a, b, c^2, d^2)$
c	$U(a^0, b^1, c, d)$, $U(a^0, b^2, c, d)$	$U(a, b^1, c, d^1)$, $U(a, b^2, c, d^1)$ $U(a, b^1, c, d^2)$, $U(a, b^2, c, d^2)$
d	$U(a^0, b^0, c^0, d)$	

Table 4: Nested CUI decomposition

The reverse direction mentioned above is done as follows: we run Algorithm 1 without any data, resulting in a table such as Table 4 (without the actual utility values). We then elicit the data for the non deleted factors (all are limited to maximal cliques). Next, we recover the more convenient "level 0" CUI representation using the table, by computing each deleted factor (going from rightmost columns to the left) as a function of the factors stored at its parents.

8. Conclusions

We present a graphical representation for multiattribute utility functions, based on conditional utility independence. CUI networks provide a potentially compact representation of the multiattribute utility function, via functional decomposition to lower-dimensional functions that depend on a node and its parents. CUI is a weaker independence condition than those previously employed as a basis for graphical utility representations, allowing common patterns of complementarity and substitutivity relations disallowed by additive models.

We proposed techniques to obtain and verify structural information, and use it to construct the network and elicit the numeric data. In addition, we developed an optimization algorithm that performs particularly well for the special case of CUI trees. In some cases it can also be leveraged for efficient optimization of CAI maps. Finally, we show how functions can be further decomposed over the set of maximal cliques of the CUI network. With this technique, CUI networks can achieve the same dimensionality of graphical models based on CAI and GAI decompositions, yet with more broadly applicable independence conditions.

Acknowledgments

A preliminary version of this paper was published in the proceedings of AAAI-06. The work was supported in part by NSF grant IIS-0205435, and the STIET program under NSF IGERT grant 0114368. We are grateful to the thorough work of the anonymous reviewers, whose suggestions provided valued help in finalizing this paper.

Appendix A. Proofs

A.1 Lemma 3

Proof. Let $Z = S \setminus (X \cup Y)$ and $C = S \setminus (A \cup B)$. We simply apply the two independence conditions consequentially, and we can define $\hat{f}, \hat{g}$ such that:

$$\begin{aligned} U(S) = U(XYZ) = f(YZ) + g(YZ)U_Y(S \setminus Y) = f(YZ) + g(YZ)(f'((BC) \setminus Y) \\ + g'((BC) \setminus Y)U_{YB}(S \setminus (YB))) = \hat{f}(ZBYC) + \hat{g}(ZBYC)U(S \setminus (YB)). \end{aligned}$$

Since $Z \cup Y \cup B \cup C = S \setminus (A \cap X)$, the last decomposition is equivalent to the decomposition (1) for the condition CUI($A \cap X$, $Y \cup B$). $\square$

A.2 Proposition 6

Proof. A CAI condition is stronger than a CUI condition, in that $CAI(x, y) \Rightarrow$ CUI$(x, y) \wedge$ CUI(y, x). To be a CUI network, for each node x_i it must be the case that all other nodes are CUI of it given its parents and descendants. This is obvious since x_i is CAI of all other nodes given its parents and children. $\square$

A.3 Lemma 7

Proof. To determine $h_{x_j}(\cdot)$, any $y \in Sc(x_j)$ needs to be determined. If $y \in \{x_i\} \cup Sc(x_i)$ we are done, if not its own scope is covered and therefore recursively determined by the assignment to $\{x_i\} \cup Sc(x_i)$. $\square$

A.4 Lemma 8

We first introduce two additional lemmas.

Lemma 12. $Ch(\{x_i\} \cup Co(x_i)) \subseteq (\{x_i\} \cup Sc(x_i) \cup Co(x_i))$

Proof. Let $x_j \in \{x_i\} \cup Co(x_i)$, and $y \in Ch(x_j)$. If $x_j = x_i$ the proof is immediate because $Ch(x_i) \subseteq Sc(x_i)$. Assume $x_j \in Co(x_i)$. We know from Definition 10 that $Ch(x_j) \subseteq Sc(x_j) \subseteq (\{x_i\} \cup Sc(x_i) \cup Co(x_i))$, and this proves the lemma. □

Lemma 13. $An(\{x_i\} \cup Co(x_i)) \subseteq Co(x_i)$

Proof. Let $x_j \in An(x_i)$ (clearly $j < i$, therefore $x_j \notin Sc(x_i)$). Let $x_{j_1} \in Sc(x_j)$. Then $j_1 > j$ and there is an undirected path from x_{j_1} to x_j, not blocked by $Sc(x_j)$. If $j_1 \geq i$, then $x_{j_1} \in Sc(x_i) \cup \{x_i\}$ because it has an unblocked path to x_j (and from there to x_i). Otherwise, let $x_{j_2} \in Sc(x_{j_1})$, and apply the same argument to x_{j_2}. We continue until x_{j_k} such that $\forall x_y \in Sc(x_{j_k})$, $y > i$ at which point $x_y \in Sc(x_i) \cup \{x_i\}$ by the path $x_y, x_{j_k}, \ldots, x_{j_1}, x_j, x_i$ and the recursion halts (note that it includes empty scopes), proving that $x_j \in Co(x_i)$.

It is left to prove that $An(Co(x_i)) \subseteq Co(x_i)$. Let $x_j \in Co(x_i), y \in An(x_j)$. Applying the first part of the proof on x_j, we get that $y \in Co(x_j)$. From Definition of $Co(x_j)$, we get $y < j$ and $\forall w \in Sc(y)$, either $w = x_j$, $w \in Sc(x_j)$ or $w \in Co(x_j)$. To show that $y \in Co(x_i)$, we need to prove for each of the cases that $w \in \{x_i\} \cup Sc(x_i) \cup Co(x_i)$.

1. If $w = x_j$ immediately $w \in Co(x_i)$.
2. If $w \in Sc(x_j)$, from $x_j \in Co(x_i)$ we get that either $w = x_i$, $w \in Sc(x_i)$ or $w \in Co(x_i)$.
3. If $w \in Co(x_j)$, we repeat the argument recursively $\forall z \in Sc(w)$. Note that z precedes w therefore the recursion will halt at some point.

□

Lemma 8. Let $X = \{x_i\} \cup Co(x_i)$. From Lemma 13, X has no external ancestors. From Lemma 12, all external children of X are in $Sc(x_i)$. Therefore $S \setminus (X \cup An(X) \cup Ch(X)) = S \setminus (X \cup Sc(x_i))$ and the result is immediate from Proposition 5. □

A.5 Proposition 9

Proof. Let x be a node in G'. Let $Y = S \setminus (x \cup In(x))$ in G and $\hat{Y} = S \setminus (x \cup Pa(x) \cup Dn(x))$ in G'. By definition of G, we know that CAI(Y, x), so also CUI(Y, x). Let $y \notin Y, y \neq x$ (so $y \in In(x)$ in G). If $y \in \hat{Y}$, then $y \notin Pa(x) = In(x)$ in G'. Then the arc (y, x) was removed, meaning that $y \in Dn(x)$. It therefore must be the case that $y \notin \hat{Y}$. Therefore $\hat{Y} \subseteq Y$ hence CUI$(\hat{Y}, x)$. □

A.6 Proposition 10

Proof. For G to become a CUI tree, for each cycle at least one weak link must be removed. Since (y_i, y_{i+1}) is the only weak link for c, it must be removed. By Proposition 9, the variable ordering must ensure that y_{i+1} is an ancestor of y_i. This can be done through the path according to the order of p, or there might be another path from y_{i+1} to y_i. Let p_1 be such path. Then the combination of p_1 and p is another cycle c_1, which therefore must be broken. Since p comprises of strong links, there must be at least one weak link (u, v) in p_1. For (u, v) to be removed, v must be an ancestor of u. This can be done through the path in the cycle c_1, and this path includes p, or through another path if such exists, for which we can repeat the argument. At each stage we get a larger cycle c_i, and a larger path

$p_i \supset p_{i-1}$. Therefore at some point there will be just one path p_i that must be guaranteed by the variable ordering, and this path includes p. □

A.7 Proposition 11

Proof. We show that Algorithm 1 leads to a functional decomposition over cliques. The outer loop in the algorithm maintains the following iteration properties:

1. $\forall a \in A^i_j, Q^i_j \subseteq Pa(a)$
2. U^j_i is defined over K^i_j
3. $A^i_j \cup x_j$ is a clique

These properties hold trivially after initialization. Assume they are valid for all factors stored in the network until outer iteration j and inner iteration i, we next show that they remain valid for each factor U^d_r that is created in iteration j, i:

1. By definition $A^d_r = A^i_j \cup x_j$. From previous iteration and definition of Q^d_r, $\forall a \in A^i_j, Q^d_r \subseteq Q^i_j \subseteq Pa(a)$. From definitions of Q^i_j and Q^d_r we get $Pa(x_j) \supseteq Q^i_j \supseteq Q^d_r$, and together it yields the result.
2. U^d_r is a factor in the CUI decomposition of $U^i_j(K^i_j)$ over G^i_j. Its scope contains: (i) the nodes that are not affected by the last CUI decomposition, i.e. in $K^i_j \setminus Q^i_j = A^i_j \cup x_j = A^d_r$, (ii) its node x_r, and (iii) the parents $Pa(x_r)$ which were not fixed in U^i_j (i.e. $Pa(x_r \cap K^i_j)$). We know $K^i_j = A^i_j \cup x_j \cup Q^i_j$, and $x_j \notin Pa(x_r)$ (because $x_r \in Pa(x_j)$), and also $Pa(x_r) \cap A^i_j = \emptyset$ (using a similar argument and property 1). Therefore $(Pa(x_r) \cap K^i_j) \subset Q^i_j$, and from (i),(ii),(iii) we get that $K^d_r = A^d_r \cup x_r \cup Q^d_r$.
3. A^d_r is a clique by its definition and the same property of previous iteration. $x_r \in Q^i_j$, therefore from property 1 of previous iteration $x_r \in Pa(a)$ for each $a \in A^i_j$. Also $x_r \in Q^i_j \subseteq Pa(x_j)$ (the last containment is immediate from definition of Q^i_j). Therefore x_r is a parent of all members of A^d_r, and as a result $A^d_r \cup x_r$ is a clique.

By the iteration properties, either K^d_r is a clique, or Q^d_r is non empty and decomposition can be applied once we reach node r in the outer loop. At the end of the process all factors which are not defined on cliques where removed. All the factors that remained are defined on cliques. $U(S)$ can be still represented by the new set of factors since we only applied valid decompositions to its factors. □

References

Abbas, A. (2005). Attribute dominance utility. *Decision Analysis*, *2*, 185–206.

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

Boutilier, C., Bacchus, F., & Brafman, R. I. (2001). UCP-networks: A directed graphical representation of conditional utilities. In *Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 56–64, Seattle.

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

Debreu, G. (1959). Topological methods in cardinal utility theory. In Arrow, K., Karlin, S., & Suppes, P. (Eds.), *Mathematical Methods in the Social Sciences*. Stanford University Press.

Dyer, J. S., & Sarin, R. K. (1979). Measurable multiattribute value functions. *Operations Research*, *27*, 810–822.

Fishburn, P. C. (1965). Independence in utility theory with whole product sets. *Operations Research*, *13*, 28–45.

Fishburn, P. C. (1967). Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review*, *8*, 335–342.

Fishburn, P. C. (1975). Nondecomposable conjoint measurement for bisymmetric structures. *Journal of Mathematical Psychology*, *12*, 75–89.

Fuhrken, G., & Richter, M. K. (1991). Polynomial utility. *Economic Theory*, *1*(3), 231–249.

Gonzales, C., & Perny, P. (2004). GAI networks for utility elicitation. In *Ninth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 224–234, Whistler, BC, Canada.

Gorman, W. M. (1968). The structure of utility functions. *Review of Economic Studies*, *35*, 367–390.

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

Krantz, D. H., Luce, R. D., Suppes, P., & Tversky, A. (1971). *Foundations of Measurement*, Vol. 1. Academic Press, New York.

La Mura, P., & Shoham, Y. (1999). Expected utility networks. In *Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 366–373, Stockholm.

Nilsson, D. (1998). An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, *8*(2), 159–173.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Pearl, J., & Paz, A. (1989). Graphoids: A graph based logic for reasoning about relevance relations. In Du Boulay, B. (Ed.), *Advances in Artificial Intelligence II*. North-Holland, New York.

Tatman, J. A., & Shachter, R. D. (1990). Dynamic programming and influence diagrams.. *20*, 365–379.

Von Stengel, B. (1988). Decomposition of multiattribute expected utility functions. *Annals of Operations Research*, *16*, 161–184.

Wellman, M. P., & Doyle, J. (1992). Modular utility representation for decision-theoretic planning. In *First International Conference on Artificial Intelligence Planning Systems*, pp. 236–242, College Park, MD.