Utility Functions for Ceteris Paribus Preferences

Michael McGeachie
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
mmcgeach@mit.edu

Jon Doyle
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535
Jon_Doyle@ncsu.edu

Abstract

Although *ceteris paribus* preference statements concisely represent one natural class of preferences over outcomes or goals, many applications of such preferences require numeric utility function representations to achieve computational efficiency. We provide algorithms, complete for finite universes of binary features, for converting a set of qualitative *ceteris paribus* preferences into quantitative utility functions.

Introduction

Although researchers have developed several logical representations of preference information (Doyle, Shoham, & Wellman 1991; Mura & Shoham 1999; Bacchus & Grove 1996), development of adequate means for automated reasoning within and computational use of these representations remains an underexplored topic. To improve the utility of these representations for practical applications, we consider the task of constructing or compiling numeric utility functions to fit logical preference specifications, and present algorithms for this task appropriate to the *ceteris paribus* logic of Doyle, Shoham, and Wellman (1991).

Wellman and Doyle (1991) observed that *ceteris paribus* preference provides a natural interpretation of a common type of human preference, and built on this to develop logical and mathematical formalizations of such information (Doyle, Shoham, & Wellman 1991; Doyle & Wellman 1994). *Ceteris paribus* preferences express preference comparisons over sets of possible worlds or outcomes characterized in terms of a set of binary features F. Then each *ceteris paribus* rule specifies some features of outcomes, and a preference over them, while ignoring the remaining features, the intent being that comparisons hold the ignored features constant.

Consider, for example, "Prefer programming tutors receiving an A in Software Engineering to tutors not receiving an A, other things being equal." We imagine a universe of computer science tutors described by binary features as indicated in Table 1. The preference about tutors stated previously means that a particular computer science tutor is more desirable if the tutor received an A in the Software Engineering course, all other features being equal. In particular,

Feature \ Tutor	p	q	r
Graduated	false	false	true
A in Software Engineering	true	false	false
A in Computer Systems	true	true	false
Cambridge resident	true	true	true
Will work Tuesdays	false	false	true
⋮	⋮	⋮	⋮

Table 1: Properties of possible computer science tutors

the tutor p described in Table 1 is preferred to the tutor q in that same table, assuming the elided features for each are identical. In contrast, the *ceteris paribus* preference makes no statement about the relative desirability of tutors p and r. Tutors p and q differ only in the feature mentioned in the preference statement, getting an A in Software Engineering. The stated preference does not apply to tutors p and r because they differ with regard to other features.

Although one can reason from statements expressed in this logic of preference to determine which of two outcomes is preferable, if either is, many applications of decision theory require the use of numerical utility functions. This paper describes means for constructing utility functions that represent preferences stated in a logic of preference *ceteris paribus* .

Ceteris paribus preference and utility

We employ a restricted logical language $\mathcal{L}$, patterned after (Doyle, Shoham, & Wellman 1991) but using only the standard logical operators $\neg$ and $\wedge$, over a set of atoms F corresponding to propositional features mentioned in preference statements. By $literals(F)$ we denote the atoms of F and their negations; $literals(F) = F \cup \{\neg f \mid f \in F\}$. We call a complete consistent set m of literals a *model*. That is, a set of features m is a model iff m contains, for each $f \in F$, exactly one of f and $\neg f$. We use $\mathcal{M}$ for the set of all models of $\mathcal{L}$.

A model of $\mathcal{L}$ assigns truth values to all atoms of $\mathcal{L}$, and therefore to all formulae in $\mathcal{L}$. We write $f(m)$ for the truth value assigned to feature f by model m. A model *satisfies* a sentence p of $\mathcal{L}$ if the truth values m assigns to the atoms

of p make p true. We write $m \models p$ when m satisfies p. We define the *proposition* expressed by a sentence p, denoted $[p]$ by $[p] = \{m \in \mathcal{M} \mid m \models p\}$.

A *preference order* is a complete preorder (reflexive and transitive relation) $\succsim$ over $\mathcal{M}$. When $m \succsim m'$, we say that m is *weakly preferred* to m'. If $m \succsim m'$ and $m' \not\succsim m$, we write $m \succ m'$ and say that m is *strictly preferred* to m'. If $m \succsim m'$ and $m' \succsim m$, then we say m is *indifferent* to m', written $m \sim m'$. A *utility function* $u : \mathcal{M} \to \mathbb{R}$ maps each model to a real number. A utility function u represents a preference order $\succsim$ just in case $u(m) \geq u(m')$ whenever $m \succsim m'$.

We define a new language $\mathcal{L}_{\mathrm{r}}$ of *preference rules* or *preference constraints* to consist of statements of the form, for $p, q \in \mathcal{L}$, of $p \trianglerighteq q$, meaning p is desired at least as much as q, and $p \rhd q$, meaning p is desired more than q. Models of $\mathcal{L}_{\mathrm{r}}$ consist of preference orders over models of $\mathcal{L}$. We define the meaning of preference rules in terms of the notions of $\mathcal{L}$-model equivalence and modification.

The *support* of a sentence $p \in \mathcal{L}$ is the minimal set of atoms determining the truth of p, denoted $s(p)$. The support of p is the same as the set of atoms appearing in an irredundant sum-of-products sentence logically equivalent to p. Two models m and m' are *equivalent modulo* p if they are the same outside the support of p. Formally, $m \equiv m' \bmod p$ iff

$$m \backslash (literals(s(p))) = m' \backslash (literals(s(p)))$$

Model modification is defined as follows. A set of *model modifications of* m *making* p *true*, written $m[p]$, are those models satisfying p which assign the same truth values to atoms outside the support of p as m does. That is,

$$m[p] = \{m' \in [p] \mid m \equiv m' \bmod p\}.$$

Formally, we say that a preference order $\succsim$ *satisfies* $p \trianglerighteq q$ if and only if for all m in $\mathcal{M}$, $m' \in m[p \wedge \neg q]$ and $m'' \in m[\neg p \wedge q]$, we have $m' \succsim m''$. This means that when two models assign the same truth values to all features not in the support of either p or q, one making p true and q false is weakly preferred to one making p false and q true. The preference order satisfies a strict *ceteris paribus* preference $p \rhd q$ if and only if in addition some case has a model makes p true and q false strictly preferred to one making p false and q true.

For a preference rule c, we write $[c]$ to denote the set of preference orders over $\mathcal{M}$ that satisfy c, that is, consistent with the constraint expressed by c. We write $[C]$ for a set of preference rules to denote the set of orders consistent with each $c \in C$, that is, $[C] = \bigcap_{c \in C} [c]$. Consistent rules and rule sets admit at least one consistent preference order. If a preference rule c implies that $m' \succ m''$, for $m', m'' \in \mathcal{M}$, we write $m' \succ_c m''$. For a set C of preference rules, we write $m' \succ_C m''$ to mean that $m' \succ_c m''$ for each $c \in C$.

We define the support of C, denoted $F(C)$, to be the set of features in $\mathcal{L}$ present in the support of statements of $\mathcal{L}$ appearing in constraints in C. Formally, $F(C)$ contains those features f such that either f or $\neg f$ appears in $\bigcup_{c \in C} s(c)$.

The following examines the problem of constructing, for a finite set C of *ceteris paribus* preference rules, a utility function u that represents some order in $[C]$.

Intermediate representation

The utility construction methods developed here employ an intermediate representation in terms of simple rules that relate paired patterns of specified and "don't care" feature values.

Let C be a finite set of preference rules. Because each preference rule mentions only finitely many features, $F(C)$ is also finite, and we write N to mean $|F(C)|$.

We construct utility functions representing the constraints in C in terms of model features. Features not specified in any rule in C are not relevant to compute the utility of a model, since there is no preference information about them in the set C. Accordingly, we focus our attention on $F(C)$.

We define the intermediate representation relative to an enumeration $\mathcal{V} = (f_1, \ldots, f_N)$ of $F(C)$.

We define the language $\mathcal{L}_{\mathrm{r}}(\mathcal{V})$ of intermediate rules in terms of a language $\mathcal{L}(\mathcal{V})$ of intermediate propositions over the ternary alphabet $\Gamma = \{0, 1, *\}$.

A statement in $\mathcal{L}(\mathcal{V})$ consists of a sequence of N letters drawn from the alphabet Γ, so that $\mathcal{L}(\mathcal{V})$ consists of words of length N over Γ. For example, if $\mathcal{V} = (f_1, f_2, f_3)$, we have $*10 \in \mathcal{L}(\mathcal{V})$. Given a statement $p \in \mathcal{L}(\mathcal{V})$ and a feature $f \in F(C)$, we write $f(p)$ for the value in Γ assigned to f in p. In particular, if $f = \mathcal{V}_i$, then $f(p) = p_i$.

An intermediate rule in $\mathcal{L}_{\mathrm{r}}(\mathcal{V})$ consists of a triple $p \succ q$ in which $p, q \in \mathcal{L}(\mathcal{V})$ have matching $*$ values. That is, $p \succ q$ is in $\mathcal{L}_{\mathrm{r}}(\mathcal{V})$ just in case $p_i = *$ if and only if $q_i = *$ for all $1 \leq i \leq N$. For example, if $\mathcal{V} = (f_1, f_2, f_3)$, $\mathcal{L}_{\mathrm{r}}(\mathcal{V})$ contains the expression $*10 \succ *00$ but not the expression $*10 \succ 0*0$. We refer to the statement in $\mathcal{L}(\mathcal{V})$ left of the $\succ$ symbol in a rule r as the left-hand side of r, and denote it $LHS(r)$. We define right-hand side $RHS(r)$ analogously. Thus $p = LHS(p \succ q)$ and $q = RHS(p \succ q)$.

We regard statements of $\mathcal{L}(\mathcal{V})$ containing no $*$ letters as *models* of $\mathcal{L}(\mathcal{V})$, and write $\mathcal{M}(\mathcal{V})$ to denote the set of all such models. We say a model m *satisfies* s, written $m \models s$, just in case m assigns the same truth value to each feature as s does for each non $*$ feature in s. That is, $m \models s$ iff $f(m) = f(s)$ for each $f \in F(C)$ such that $f(s) \neq *$. For example, 0011 satisfies both $*0*1$ and $00**$.

We project models in $\mathcal{M}$ to models in $\mathcal{M}(\mathcal{V})$ by a mapping $\alpha : \mathcal{M} \to \mathcal{M}(\mathcal{V})$ defined, for each $m \in \mathcal{M}$ and $f \in F(C)$, so that $f(\alpha(m)) = 1$ if $f \in m$ and $f(\alpha(m)) = 0$ if $\neg f \in m$. This projection induces an equivalence relation on $\mathcal{M}$, and we write $[m]$ to mean the set of models in $\mathcal{M}$ mapped to the same model in $\mathcal{M}(\mathcal{V})$ as m, namely $[m] = \{m' \in \mathcal{M} \mid \alpha(m') = \alpha(m)\}$.

We say that a pair of models (m, m') of $\mathcal{L}(\mathcal{V})$ *satisfies* a rule r in $\mathcal{L}_{\mathrm{r}}(\mathcal{V})$, and write $(m, m') \models r$, if m satisfies $LHS(r)$, m' satisfies $RHS(r)$, and m, m' have the same value for those features represented by $*$ in r, that is, $m_i = m'_i$ for each $1 \leq i \leq N$ such that $LHS(r)_i = *$. For example, $(100, 010) \models 10* \succ 01*$, but $(101, 010) \not\models 10* \succ 01*$.

The meaning $[r]$ of a rule r in $\mathcal{L}_{\mathrm{r}}(\mathcal{V})$ is the set of all preference orders $\succ$ over $\mathcal{M}$ such that for each $m, m' \in \mathcal{M}$, if $(\alpha(m), \alpha(m')) \models r$, then $m \succ m'$. The meaning of a set R of rules consists of the set of preference orders consistent with each rule in the set, that is, $[R] = \bigcap_{r \in R} [r]$. Thus a rule

$**01 \succ **10$ represents four specific preferences

$$0001 \succ 0010$$
$$0101 \succ 0110$$
$$1001 \succ 1010$$
$$1101 \succ 1110$$

Note that this says nothing at all about the preference relationship between, *e.g.*, 0101 and 1010.

To use the intermediate representation, we must translate sets C of *ceteris paribus* rules into sets R of intermediate representation rules in a way that guarantees compatibility of meaning in the sense that $[R] \subseteq [C]$. We do this as follows.

The translation involves considering models restricted to subsets of features. We write $\mathcal{M}[S]$ to denote the set of models over a feature set $S \subseteq F$, so that $\mathcal{M} = \mathcal{M}[F]$. If $m \in \mathcal{M}[S]$ and $S' \subseteq S$, we write $m \restriction S'$ to denote the *restriction* of m to S', that is, the model $m' \in \mathcal{M}[S']$ assigning the same values as m to all features in S'. We say that a model $m \in \mathcal{M}[S]$ satisfies a model $m' \in \mathcal{M}[S']$, written $m \models m'$ just in case $S' \subseteq S$ and $m' = m \restriction S'$.

A set of rules R of intermediate representation is compatible with a rule $c = p_c \rhd q_c$ in the *ceteris paribus* representation of the previous section just in case $[R] \subseteq [c]$. If $m_1 \succ_c m_2$, this means that for some $r \in R$ we have $(m_1', m_2') \models r$, where m_1, m_2 model $\mathcal{L}$, m_1', m_2' model $\mathcal{L}(\mathcal{V})$, such that $m_1 \in [m_1']$ and $m_2 \in [m_2']$. We give a construction for such an r from an arbitrary p.

We first define the *characteristic model* $\mu(p)$ of a statement p in $\mathcal{L}(\mathcal{V})$ to be the model in $\mathcal{M}[s(p)]$ defined by

$$\mu(p) = \{f \mid f(p) = 1\} \cup \{\neg f \mid f(p) = 0\}.$$

Note that for $m \in \mathcal{M}$ we have $m \models \mu(\alpha(m))$, that is, $\mu(\alpha(m)) = m \restriction F(C)$.

We translate a single *ceteris paribus* rule $c \in \mathcal{L}_{\rm r}$ into a set of intermediate representation rules R by the support of c. If c is of the form $p_c \rhd q_c$, where p_c, q_c are sentences in $\mathcal{L}$, then models that satisfy $p_c \wedge \neg q_c$ are preferred to models that satisfy $\neg p_c \wedge q_c$, other things being equal. For brevity, let $s_c = s(p_c \wedge \neg q_c) \cup s(\neg p_c \wedge q_c)$, so that $s_c \subseteq F(C)$ is the set of support features for each rule $r \in R$, and consider models in $\mathcal{M}[s_c]$. Let W_l be the set of such models satisfying $p_c \wedge \neg q_c$, that is

$$W_l = \{w \in \mathcal{M}[s_c] \mid w \models p_c \wedge \neg q_c\},$$

and define W_r as the corresponding set of models satisfying the right-hand side,

$$W_r = \{w \in \mathcal{M}[s_c] \mid w \models \neg p_c \wedge q_c\}.$$

We construct new sets W_l' and W_r' of statements in $\mathcal{L}(\mathcal{V})$ from W_l and W_r by augmenting each member and translating into $\mathcal{L}(\mathcal{V})$. We define these new sets by

$$\begin{aligned} W_l' &= \{w \in \mathcal{L}(\mathcal{V}) \mid (\mu(w) \restriction s_c) \in W_l\} \\ W_r' &= \{w \in \mathcal{L}(\mathcal{V}) \mid (\mu(w) \restriction s_c) \in W_r\}. \end{aligned}$$

Note that the members of W_l' and W_r' are of length $|F(C)|$, while those of W_l and W_r are of size $|s_c|$.

We now construct a set of rules in $\mathcal{L}_{\rm r}(\mathcal{V})$ to include a rule for each pair of augmented statements, that is, a set

$$R = \{w_l' \succ w_r' \mid w_l' \in W_l', w_r' \in W_r'\}.$$

This completes the translation.

Consider a simple example. In the following, we assume $\mathcal{V} = (f_1, f_2, f_3, f_4)$. A *ceteris paribus* rule might be of the form $f_2 \wedge \neg f_4 \rhd f_3$. This expresses the preference for models satisfying

$$(f_2 \wedge \neg f_4) \wedge \neg f_3$$

over models satisfying

$$\neg(f_2 \wedge \neg f_4) \wedge f_3$$

other things being equal. Note $s_c = \{f_2, f_3, f_4\}$. Then, following the above construction, $W_l = \{\{f_2, \neg f_3, \neg f_4\}\}$, and $W_r = \{\{f_2, f_3, f_4\}, \{\neg f_2, f_3, f_4\}, \{\neg f_2, f_3, \neg f_4\}\}$. In this case we translate these into three intermediate representation rules:

$$*100 \succ *111$$
$$*100 \succ *011$$
$$*100 \succ *010$$

The above translation can be used to convert a set C of *ceteris paribus* rules into a set C' of intermediate representation rules equivalent in the sense that both sets denote the same set of preference orders $[C] = [C']$. Furthermore, this translation from rules in $\mathcal{L}_{\rm r}$ to rules in $\mathcal{L}_{\rm r}(\mathcal{V})$ can be proven correct, by reference to the definitions of the notations used in each language. We omit the proof here because, although the ideas are intuitively simple, the details are cumbersome. A translation from $\mathcal{L}_{\rm r}(\mathcal{V})$ to $\mathcal{L}_{\rm r}$ is also possible, and follows in much the same way.

It is important to note that the richer language $\mathcal{L}_{\rm r}$ allows us to represent more complicated preferences than are possible in $\mathcal{L}_{\rm r}(\mathcal{V})$. Accordingly, the translation of a single *ceteris paribus* rule might produce many intermediate representation rules.

Some direct utility functions

We now define several direct utility functions consistent with a set of preferences in the intermediate representation. One can use these to define utility functions over models in $\mathcal{L}$ by composition with the model-projection mapping α. Specifically, given a set C of preference rules and a translation of C into a set C' of intermediate preference rules, one finds the utility of a model $m \in \mathcal{M}$ by computing the projection $\alpha(m) \in \mathcal{M}(\mathcal{V})$ and using one of the functions defined in the following, each of has the form $u : \mathcal{M}(\mathcal{V}) \to \mathbb{R}$.

To construct these utility functions, we use the rules in C' to define a directed graph $G(C')$ over $\mathcal{M}(\mathcal{V})$, called a *model graph*, that represents the preferences expressed in C'. Each node in the graph represents one of the 2^N possible models $\mathcal{M}(\mathcal{V})$. Algorithmic constructions can get by explicitly representing only those nodes of the graph linked by edges, which typically constitute a smaller subset.

The model graph $G(C')$ contains an edge $e(m_1, m_2)$ from source m_1 to sink m_2 if and only if $(m_1, m_2) \models r$ for some rule $r \in C'$. Each edge represents a preference for the source over the sink. If C' is consistent, then $G(C')$ is acyclic; a cycle would indicate the inconsistency of C'. We can determine whether m is preferred to m' by looking for a path from m to m' in $G(C')$. The existence of such a path means $m \succ m'$.

We define four utility functions over the model graph $G(C')$, as summarized in Table 2.

u_M	("Minimizing")	longest outgoing path length
u_D	("Descendant")	number of descendants
u_X	("Maximizing")	longest incoming path length
u_T	("Topological")	rank in topological-sort order

Table 2: Four model-graph utility functions

- u_M, the *minimizing* utility function, sets $u_M(m)$ equal to the number of nodes on the longest path originating from m in $G(C')$.
- u_D, the *descendant* utility function, sets $u_D(m)$ equal to the total number of nodes on all paths originating from m in $G(C')$.
- u_X, the *maximizing* utility function, sets $u_X(m)$ equal to the length of the longest path in $G(C')$, denoted $\max(G(C'))$, minus the number of nodes on the longest path originating at any node other than m and ending at m in $G(C')$.
- u_T, the *topological-sort* utility function, sets $u_T(m)$ equal to 2^N minus the rank of m in the order obtained by a topological sort T of $G(C')$.

Each of the ordinal utility functions so defined has somewhat different properties, particularly in how they assign values to preferentially-unrelated models m_1 and m_2 such that neither $m_1 \succ m_2$ or $m_2 \succ m_1$ according to C'. The ordering properties specified by C' say nothing about how to order these, so we may order them as convenient. In particular, a utility function u consistent with C' need not $u(m_1) = u(m_2)$, but instead can distinguish these utility values.

Before proving that these four types of functions represent the stipulated preference orders, we illustrate the differences among them by considering an example in which the rules in C' are such that only the pairs of models related in Table 3 satisfy any rule in C'. These orderings leave the relationship

$m_1 \succ m_2$
$m_3 \succ m_4$
$m_4 \succ m_5$
$m_4 \succ m_6$

Table 3: Orderings illustrating alternative utility functions

between m_1 and m_3 unspecified, as well as the relationship between any of these models and some other model m_7. A utility representation of these orderings may order these unrelated models in any way.

The idea that the distance between nodes in an acyclic graph can indicate their relative utilities underlies the minimizing utility function u_M. That function assigns the following values to the models in the example.

$$\begin{array}{lll} u_M(m_1) = 1 & u_M(m_2) = 0 & u_M(m_3) = 2 \\ u_M(m_4) = 1 & u_M(m_5) = 0 & u_M(m_6) = 0 \\ u_M(m_7) = 0 & & \end{array}$$

We call this utility function minimizing because it assigns minimal (0) utility to models not explicitly preferred to other models according to the preference set. Thus the model m_7, about which no preferences were specified, is assigned the value 0.

The descendant utility function uses the relationships of the example to assign the following utilities to models.

$$\begin{array}{lll} u_D(m_1) = 1 & u_D(m_2) = 0 & u_D(m_3) = 3 \\ u_D(m_4) = 2 & u_D(m_5) = 0 & u_D(m_6) = 0 \\ u_D(m_7) = 0 & & \end{array}$$

Descendant utility gives slightly higher values to m_3 and m_4 than minimizing utility, since the former counts all descendants, while the latter counts only the longest path.

The maximizing utility function assigns the following values to the models of the example.

$$\begin{array}{lll} u_X(m_1) = 2 & u_X(m_2) = 1 & u_X(m_3) = 2 \\ u_X(m_4) = 1 & u_X(m_5) = 0 & u_X(m_6) = 0 \\ u_X(m_7) = 2 & & \end{array}$$

This function assigns $u_X(m_7) = 2$, the highest value among those assigned, to the node about which the preferences provide no information.

In general, a directed graph admits more than one topological sort. Each topological sort utility function therefore assigns values that depend on the ordering of nodes in some specific topological sort. The following assignment of values reflects one possible topological sort of the models in the example.

$$\begin{array}{lll} u_T(m_1) = 7 & u_T(m_2) = 6 & u_T(m_3) = 5 \\ u_T(m_4) = 4 & u_T(m_5) = 3 & u_T(m_6) = 2 \\ u_T(m_7) = 1 & & \end{array}$$

The class of topological sort utility functions thus illustrates by itself the ability to vary utility assignments consistent with the specificied preferences.

We now show that the minimizing and descendant utility functions defined over model graphs accurately represent the preferences defining the graphs. We state the corresponding results for the maximizing and topological sort utility functions, but omit the proofs here due to space limitations.

Theorem 1 *The minimizing utility function u_M defined over a model graph $G(C')$ represents some preference order in $[C']$.*

Proof. Let $u_M(m)$ be equal to the number of nodes on the longest path originating from m in $G(C')$. For u_M to be consistent with C' requires that $u_M(m_1) > u_M(m_2)$ whenever $m_1 \succ m_2$ according to C'. Choose a pair m_1, m_2 such that $m_1 \succ m_2$ according to C'. By construction of $G(C')$, there exists a path from m_1 to m_2 in $G(C')$. Since $G(C')$ is acyclic, no part of the path from m_1 to m_2 can be reached from m_2. This implies that the longest path originating at m_1 contains the longest path originating at m_2, but not vice versa. Therefore $u_M(m_1) > u_M(m_2)$. □

Theorem 2 *The descendant utility function u_D defined over a model graph $G(C')$ represents some preference order in $[C']$.*

Proof. From the proof of Theorem 1, if $m_1 \succ m_2$ according to C', then by construction of $G(C')$ there exists a path from m_1 to m_2 in $G(C')$. Since $G(C')$ is acyclic, and m_2 is on a path from m_1, therefore m_1 has at least one more descendent than m_2, namely, m_2. Therefore $u_D(m_1) > u_D(m_2)$. □

Theorem 3 *The maximizing utility function u_X defined over a model graph $G(C')$ represents some preference order in $[C']$.*

Proof omitted.

Theorem 4 *Every topological-sort utility function u_T defined over a model graph $G(C')$ represents some preference order in $[C']$.*

Proof omitted.

Complexity

The utility functions outlined in the previous section, while conceptually simple, have worst-case complexity exponential in the number of relevant features $N = |F(C)|$.

As noted earlier, the model graph $G(C')$ has 2^N nodes, but this exponential size does not in itself imply exponential cost in computing utility functions because the utility functions derive from graph edges rather than graph nodes. The descendant utility function u_D, for example, requires counting the number of descendants of nodes, a number at worst linear in the number of edges. The other utility functions measure the number of ancestors, or the longest path from or to a node. Clearly counting the number of ancestors is the same computational burden as counting the number of descendants. Computing the longest path originating at a node and ending elsewhere has the same bound, since searching all descendants can determine the longest path. Accordingly, the number of edges in the model graph provides a basic complexity measure for these utility computations.

In fact, a simple and familiar example shows that the model graph can contain a number of edges exponential in the size of $F(C)$. Suppose, for instance, that $F(C)$ consists of four features and that the derived intermediate preference rules C' consist of those displayed in Table 4. These rules

$***1 \succ ***0$
$**10 \succ **01$
$*100 \succ *011$
$1000 \succ 0111$

Table 4: Lexicographic preference rules

order all models lexicographically in a in a complete linear order, the same ordering we give models if we interpret them as binary representations of the integers from 0 to 15. The longest path through $G(C')$ has length $2^{|F(C)|}$, so the number of edges is exponential in $|C'| = |F(C)|$. One should note that this example does not imply utility dependence among the features, but it does imply that the preferences over some features dominate the preferences over other features. Moreover, the example does not show that derivation of a utility function must take exponential time, because lexicographic utility functions can be expressed in much simpler ways than counting path length. The true complexity of this problem remains an open question.

In fact, one can trade computation cost between construction and evaluation of the utility function. The evaluation of specific utility values can be reduced by significant preprocessing in the function-construction stage. Clearly the utility value of $m \in \mathcal{M}(\mathcal{V})$ could be cached at the corresponding node in $G(C')$, using, for example, Dijkstra's all-paths algorithm. Alternatively, values for a smaller number of nodes might be cached. The computation of a utility value could then proceed by traversing $G(C')$ until each branch of the search from the starting node reaches a node with a cached utility value, and then computing the desired node value from these cached values and the graph portion traversed in reaching them.

For instance, one might compute a utility function akin to the descendant utility function by keeping a static set of nodes with cached utility values. If one finds k separate branches of the search starting from a node m and terminating in k nodes with cached values $u(m_i)$ for $i = 1, ..., k$, and traverses a total of t nodes along these branches, we assign

$$u(m) = 1 + t + \sum_{i=1}^{k} u(m_i).$$

This value can differ from the descendant utility value because two cached nodes might share some descendants. Reducing evaluation cost from exponential levels can be achieved by cutting the number of nodes searched before reaching cached nodes to $\log(|G(C')|)$, but as the lexicographic example shows, this can require caching $|G(C')|/\log|G(C')|$ nodes, implying an exponential level of preprocessing cost.

Alternatively, one might calculate utility values by repetitively computing $G(C')$. One might do this by searching for a path from a model m_0 by finding rules in $r \in C'$ such that $(m_0, m_1) \models r$, where m_1 is arbitrary. The proof of Theorem 1 implies that an edge $e(m_0, m_1)$ in $G(C')$ exists if and only if there exists some $(m_0, m_1) \models r$ for any $r \in C'$. Thus, searching the list of rules in C' for a pair $(m_0, m_1) \models r$ for some $r \in C'$ is equivalent to following an edge $e(m_0, m_1)$ in $G(C')$. Then one recursively looks for rules r such that $(m_1, m_2) \models r$, and then (m_2, m_3), and so on, such that the search becomes a traversal of the graph $G(C')$. Each branch of the search terminates when $(m_{k-1}, m_k) \models r$ for some rule $r \in C'$, but $(m_k, m_{k+1}) \not\models s$ for all rules s in C. We know that there exists a path from m_0 with length k; if k is the length of the longest such path, one can then assign $u(m_0) = k$. Note that the heuristics presented by (Boutilier *et al.* 1999) do not apply because our preference rules are of a quite different form, as seen in Table 4.

Improvements

Elsewhere we improve on the performance of these direct constructions of utility functions by using *utility independence* to decompose the utility-construction task into a prob-

lem of finding an appropriate combination of subutility functions. We provide methods for partitioning the set of features into clusters that are mutually utility-independent. We use the direct utility function constructions described in the preceding to construct subutility functions over each of these clusters. We then use the initial *ceteris paribus* statements to identify constraints on the relations of these subutility function values, and apply standard constraint-satisfaction methods to determine subfunction-combination parameters that yield overall utility functions satisfying the given *ceteris paribus* rules.

Related work

Many researchers have defined related logics of desire (van der Torre & Weydert 1998; Mura & Shoham 1999; Shoham 1997) or *ceteris paribus* preference (Tan & Pearl 1994a; 1994b; Bacchus & Grove 1996; 1995; Boutilier *et al.* 1999).

Bacchus and Grove (1995; 1996) have presented a somewhat different conception of *ceteris paribus* preference specification that incorporates numerical representations from the start and algorithms for computing with it. Similar to La Mura and Shoham (1999), their computation paradigm is an adaptation of Bayesian Networks to utility. Both systems start with probabilistic actions with known probabilities and outcomes with known numeric utility values, and then discuss optimizations resulting from utility independence.

Tan and Pearl (1994a; 1994b) use conditional *ceteris paribus* comparatives in a manner similar to the *ceteris paribus* preferences used in this work. Their work does not address computation, instead concentrating on specificity of preferences and when one preference supersedes another preference.

Boutilier et al. (1999; 1997), also propose a system of conditional *ceteris paribus* preference statements. They construct a chain of "flipping feature values" to conduct dominance queries similar in spirit to the algorithm proposed here. The *ceteris paribus* preference representation they employ is quite different from that of (Doyle, Shoham, & Wellman 1991), and the methods of (Boutilier *et al.* 1999) are not directly applicable to constructing utility functions.

Acknowledgements

This work was supported in part by DARPA under contract F30602-99-1-0509. Michael McGeachie is supported in part by a training grant from the National Library of Medicine, and a grant from the Pfizer corporation.

References

Bacchus, F., and Grove, A. 1995. Graphical models for preference and utility. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 3–19. Morgan Kaufmann.

Bacchus, F., and Grove, A. 1996. Utility independence in a qualitative decision theory. In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning*, 542–552. Morgan Kaufmann.

Boutilier, C.; Brafman, R.; Geib, C.; and Poole, D. 1997. A constraint-based approach to preference elicitation and decision making. In Doyle, J., and Thomason, R. H., eds., *Working Papers of the AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, 19–28. Menlo Park, California: AAAI.

Boutilier, C.; Brafman, R. I.; Hoos, H. H.; and Poole, D. 1999. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of Uncertainty in Artificial Intelligence 1999 (UAI-99)*.

Doyle, J., and Wellman, M. P. 1994. Representing preferences as *ceteris paribus* comparatives. In *Working Notes of the AAAI Symposium on Decision-Theoretic Planning*. AAAI.

Doyle, J.; Shoham, Y.; and Wellman, M. P. 1991. A logic of relative desire (preliminary report). In Ras, Z., ed., *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems*, Lecture Notes in Computer Science. Berlin: Springer-Verlag.

Mura, P. L., and Shoham, Y. 1999. Expected utility networks. In *Proc. of 15th conference on Uncertainty in Artificial Intelligence*, 366–373.

Shoham, Y. 1997. A mechanism for reasoning about utilities (and probabilities): Preliminary report. In Doyle, J., and Thomason, R. H., eds., *Working Papers of the AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, 85–93. Menlo Park, California: AAAI.

Tan, S.-W., and Pearl, J. 1994a. Qualitative decision theory. In *AAAI94*. Menlo Park, CA: AAAI Press.

Tan, S.-W., and Pearl, J. 1994b. Specification and evaluation of preferences for planning under uncertainty. In Doyle, J.; Sandewall, E.; and Torasso, P., eds., *KR94*. San Francisco, CA: Morgan Kaufmann.

van der Torre, L., and Weydert, E. 1998. Goals, desires, utilities and preferences. In *Proceedings of the ECAI'98 Workshop on Decision Theory meets Artificial Intelligence*.

Wellman, M., and Doyle, J. 1991. Preferential semantics for goals. In Dean, T., and McKeown, K., eds., *Proceedings of the Ninth National Conference on Artificial Intelligence*, 698–703. Menlo Park, California: AAAI Press.