Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008)

Preference Aggregation with Graphical Utility Models

Christophe Gonzales and **Patrice Perny** and **Sergio Queiroz**
LIP6 – UPMC – Paris 6
104, avenue du président Kennedy, F-75016 Paris, France

Abstract

This paper deals with preference representation and aggregation in the context of multiattribute utility theory. We consider a set of alternatives having a combinatorial structure. We assume that preferences are compactly represented by graphical utility models derived from generalized additive decomposable (GAI) utility functions. Such functions enable to model interactions between attributes while preserving some decomposability property. We address the problem of finding a compromise solution from several GAI utilities representing different points of view on the alternatives. This scheme can be applied both to multicriteria decision problems and to collective decision making problems over combinatorial domains. We propose a procedure using graphical models for the fast determination of a Pareto-optimal solution achieving a good compromise between the conflicting utilities. The procedure relies on a ranking algorithm enumerating solutions according to the sum of all the GAI utilities until a boundary condition is reached. Numerical experiments are provided to highlight the practical efficiency of our procedure.

Introduction

The development of decision support systems and web recommender systems has stressed the need for models that can handle users preferences and perform preference-based recommendation tasks. Thus, current works in preference modeling and decision theory aim at developing compact preference models achieving a good trade-off between two conflicting aspects: i) the need for models flexible enough to describe sophisticated decision behaviors; and ii) the practical necessity of keeping the elicitation effort at a low level as well as the need for fast procedures to solve preference-based optimization problems. As an example, let us mention interactive decision support systems on the web where the preferred solution must be found among a combinatorial set of possibilities. This kind of application motivates the current interest for qualitative preference models and compact representations like CP-nets (Boutilier et al. 2004a; 2004b) and mCP-nets (Rossi, Venable, and Walsh 2004), their multiagent extension. Such models are deliberately simple and flexible enough to be integrated efficiently in interactive recommendation systems; agents preferences must be captured using only a few questions so as to perform a fast preference-based search over the possible items.

In other AI applications (e.g. configuration system, fair allocation of resources, combinatorial auctions), more time can be spent in the elicitation stage in order to get a finer description of preferences. In such cases, compact utility models can be used advantageously to handle preferences and improve discrimination between feasible solutions (Boutilier, Bacchus, and Brafman 2001; Chevaleyre, Endriss, and Lang 2007; Engel and Wellman 2007). Moreover, cardinal utility models allow us to escape from the framework of Arrow's impossibility theorem which considerably restricts aggregation possibilities (Pini et al. 2005).

In the literature, different quantitative models based on utilities have been developed to take into account different preference structures. The most widely used model assumes a special kind of independence among attributes called "mutual preferential independence" which ensures that preferences are representable by an additive utility (Krantz et al. 1971). Such decomposability makes the elicitation process very fast and simple. However, in practice, preferential independence may fail to hold as it rules out any interaction among attributes. Some generalizations have thus been investigated. For instance, *utility independence* on every attribute leads to a more sophisticated form called *multilinear utilities* (Bacchus and Grove 1995). The latter are more general than additive utilities but they still cannot cope with many interactions between attributes. To increase the descriptive power of such models, GAI (generalized additive independence) decompositions have been introduced by Fishburn in 1970, that allow more general interactions between attributes (Bacchus and Grove 1995) while preserving some decomposability. Such a decomposition has been used to endow CP-nets with utility functions (UCP-nets) both under uncertainty (Boutilier, Bacchus, and Brafman 2001) and under certainty (Brafman, Domshlak, and Kogan 2004).

In the same direction, general procedures have been developed to assess GAI utilities in decision under risk (Gonzales and Perny 2004; Braziunas and Boutilier 2005). These are directed by the structure of a graphical model called a GAI-network and consist in a sequence of questions involving simple lotteries that capture efficiently the basic features of the agent's attitude towards risk. Recently the elicitation of GAI models has been investigated in the context of deci-

This work was supported by ANR grant ANR-05-BLAN-0384.

sion making under certainty. In 2005, Gonzales and Perny proposed elicitation procedures relying on simple comparisons of outcomes (e.g. questions involve only a subset of attributes or outcomes varying in only few attributes). They also suggest efficient choice procedures to find GAI-optimal elements in a product set, using classical propagation techniques used in the Bayesian network literature.

In this paper, we address group decision making problems in similar contexts and we focus on the determination of a good compromise solution between agents. As usual in works on preferences, we assume that the alternatives to be compared belong to a product set the size of which prevents exhaustive enumeration. We assume that a GAI utility has been elicited for each agent and we tackle the problem of performing efficient choices for the group of agents. The paper is organized as follows: In the next section, we discuss individual and collective preference representation. After recalling some elements about GAI models we consider various aggregation criteria to define the notion of compromise between agents. Then, we present efficient algorithms to determine the best compromise solution for the group of agents and we provide results of numerical experiments performed on multiagent aggregation problems.

GAI Utilities: From Individual Preference Modeling to Collective Decision Making

Before describing GAI models, we shall introduce some notations. Throughout the paper, $\succsim$ denotes an agent's preference relation (a weak order) over some set $\mathcal{X}$. $x \succsim y$ means that x is at least as good as y. $\succ$ refers to the asymmetric part of $\succsim$ and $\sim$ to the symmetric one. In practice, $\mathcal{X}$ is often described by a set of attributes. For simplicity, we assume that $\mathcal{X}$ is the product set of their domains, although extensions to general subsets are possible. In the rest of the paper, uppercase letters (possibly subscripted) such as A, B, X_1 denote attributes as well as their domains. Unless otherwise mentioned, lowercase letters denote values of the attribute with the same uppercase letters: x, x^1 (resp. x_i, x_i^1) are thus values of X (resp. X_i). We shall now only consider the representation of the preferences of a single agent, the multiagent case being dealt with in a next subsection.

Individual Preference Modeling

Under mild hypotheses (Debreu 1964), it can be shown that $\succsim$ is representable by a utility, i.e., by a function $u : \mathcal{X} \mapsto \mathbb{R}$ s.t. $x \succsim y \Leftrightarrow u(x) \geq u(y)$ for all $x, y \in \mathcal{X}$. As preferences are specific to each individual, utilities must be elicited for each agent, which is impossible due to the combinatorial nature of $\mathcal{X}$. Moreover, in a recommendation system with multiple regular users, storing explicitly for each user the utility of every element of $\mathcal{X}$ is prohibitive. Fortunately, agent's preferences usually have an underlying structure induced by independencies among attributes that substantially decreases the elicitation effort and the memory needed to store preferences. The simplest case is obtained when preferences over $\mathcal{X} = X_1 \times \cdots \times X_n$ are representable by an additive utility $u(x) = \sum_{i=1}^{n} u_i(x_i)$ for any $x = (x_1, \ldots, x_n) \in \mathcal{X}$. This model only requires to store $u_i(x_i)$ for any $x_i \in X_i$, $i = 1, \ldots, n$. However, such decomposition in not always convenient because it rules out interactions between attributes. When agents preferences are more complex, a more elaborate model is needed as shown below:

Example 1 *Consider a set $\mathcal{X}$ of menus $x = (x_1, x_2, x_3)$, with main course $x_1 \in X_1 = \{$meat(M), fish$(F)\}$, drink $x_2 \in X_2 = \{$red wine (R), white wine$(W)\}$ and dessert $x_3 \in X_3 = \{$cake(C), sorbet$(S)\}$.*

First case. *Assume the agent's preferences are well represented by an additive utility u characterized by the following marginal utilities: $u_1(M) = 4$; $u_1(F) = 0$; $u_2(R) = 2$; $u_2(W) = 0$; $u_3(C) = 1$; $u_3(S) = 0$. Then the utilities of the 2^3 possible menus $x^{(i)}$ follow:*

$u(x^{(1)}) = u(M, R, C) = 7$; $u(x^{(2)}) = u(M, R, S) = 6$;
$u(x^{(3)}) = u(M, W, C) = 5$; $u(x^{(4)}) = u(M, W, S) = 4$;
$u(x^{(5)}) = u(F, R, C) = 3$; $u(x^{(6)}) = u(F, R, S) = 2$;
$u(x^{(7)}) = u(F, W, C) = 1$; $u(x^{(8)}) = u(F, W, S) = 0$;

which yields the following ordering:

$x^{(1)} \succ x^{(2)} \succ x^{(3)} \succ x^{(5)} \succ x^{(4)} \succ x^{(6)} \succ x^{(7)} \succ x^{(8)}$.

Second case. *Assume that another agent's ranking is: $x^{(1)} \succ x^{(2)} \succ x^{(7)} \succ x^{(8)} \succ x^{(3)} \succ x^{(4)} \succ x^{(5)} \succ x^{(6)}$. This can be explained by: i) a high priority granted to matching wine with main course (red wine for meat, white one for fish); ii) at a lower level of priority, meat is preferred to fish; and iii) cake is preferred to sorbet (ceteris paribus).*

Although not irrational, such preferences are not representable by an additive utility because $x^{(1)} \succ x^{(5)} \Rightarrow u_1(M) > u_1(F)$ whereas $x^{(7)} \succ x^{(3)} \Rightarrow u_1(F) > u_1(M)$. However, this does not rule out less disaggregated forms of additive decompositions such as: $u(x) = u_{1,2}(x_1, x_2) + u_3(x_3)$. For example, $u_{1,2}(M, R) = 6$, $u_{1,2}(F, W) = 4$, $u_{1,2}(M, W) = 2$, $u_{1,2}(F, R) = 0$, $u_3(C) = 1$, $u_3(S) = 0$ would represent $\succsim$.

Third case. *Assume that the ranking of a third agent is: $x^{(2)} \succ x^{(1)} \succ x^{(7)} \succ x^{(8)} \succ x^{(4)} \succ x^{(3)} \succ x^{(5)} \succ x^{(6)}$. Her preference system is actually a refinement of the previous one. She prefers cake to sorbet when the main course is fish but the opposite obtains when the main course is meat.*

In this case, using similar arguments, it can be shown that the previous decomposition does not fit anymore due to the interaction between attributes X_1 and X_3. However it is interesting to remark that preferences can still be represented by a decomposable utility of the form: $u(x) = u_{1,2}(x_1, x_2) + u_{1,3}(x_1, x_3)$, setting for instance:

$u_{1,2}(M,R)=6$; $u_{1,2}(F,W)=4$; $u_{1,2}(M,W)=2$; $u_{1,2}(F,R)=0$;
$u_{1,2}(M,C)=0$; $u_{1,2}(M,S)=1$; $u_{1,2}(F,C)=1$; $u_{1,2}(F,S)=0$.

Such a decomposition over overlapping factors is called a GAI decomposition (Bacchus and Grove 1995). It includes additive and multilinear decompositions as special cases, but it is much more flexible since it does not make any assumption on the kind of interactions between attributes. GAI decompositions can be defined more formally as follows:

Definition 1 (GAI decomposition) *Let $\mathcal{X} = \prod_{i=1}^{n} X_i$. Let $Z_1, \ldots, Z_k$ be some subsets of $N = \{1, \ldots, n\}$ such that $N = \cup_{i=1}^{k} Z_i$. For every i, let $X_{Z_i} = \prod_{j \in Z_i} X_j$. Utility $u(\cdot)$ representing $\succsim$ is GAI-decomposable w.r.t. the X_{Z_i}'s iff*

there exist functions $u_i : X_{Z_i} \mapsto \mathbb{R}$ *such that:*

$u(x) = \sum_{i=1}^{k} u_i(x_{Z_i})$, *for all* $x = (x_1, \ldots, x_n) \in \mathcal{X}$,

where x_{Z_i} *denotes the tuple constituted by the* x_j*'s,* $j \in Z_i$.

GAI decompositions can be represented by graphical structures called *GAI networks* (Gonzales and Perny 2004) which are essentially similar to the junction graphs used for Bayesian networks (Jensen 1996; Cowell et al. 1999):

Definition 2 (GAI network) *Let* $\mathcal{X} = \prod_{i=1}^{n} X_i$. *Let* $Z_1, \ldots, Z_k$ *be a covering of* $\{1, \ldots, n\}$. *Assume that* $\succsim$ *is representable by a GAI utility* $u(x) = \sum_{i=1}^{k} u_i(x_{Z_i})$ *for all* $x \in \mathcal{X}$. *Then a GAI net representing* $u(\cdot)$ *is an undirected graph* $G = (V, E)$, *satisfying the following properties:*

1. $V = \{X_{Z_1}, \ldots, X_{Z_k}\}$;
2. *For every* $(X_{Z_i}, X_{Z_j}) \in E$, $Z_i \cap Z_j \neq \emptyset$. *For every* X_{Z_i}, X_{Z_j} *s.t.* $Z_i \cap Z_j = T_{ij} \neq \emptyset$, *there exists a path in* G *linking* X_{Z_i} *and* X_{Z_j} *s.t. all of its nodes contain all the indices of* T_{ij} *(Running intersection property).*

Nodes of V *are called* cliques. *Every edge* $(X_{Z_i}, X_{Z_j}) \in E$ *is labeled by* $X_{T_{ij}} = X_{Z_i \cap Z_j}$ *and is called a* separator.

Cliques are drawn as ellipses and separators as rectangles. In this paper, we shall only be interested in GAI trees, i.e., in singly-connected GAI nets. Actually, this is not restrictive as any GAI network can be compiled into a GAI tree (Gonzales and Perny 2004). For any GAI decomposition, by Definition 2, the cliques of the GAI net should be the sets of variables of the subutilities. For instance, if $u(a, b, c, d, e, f, g) = u_1(a, b) + u_2(c, e) + u_3(b, c, d) + u_4(b, d, f) + u_5(b, g)$ then, as shown in Fig.1, the cliques are: AB, CE, BCD, BDF, BG. By property 2 of Definition 2 the set of edges of a GAI network can be determined by any algorithm preserving the running intersection property (see the Bayesian network literature (Cowell et al. 1999)).

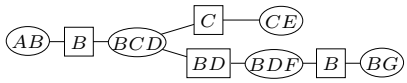

Figure 1: A GAI tree

Collective Decision Making

We consider now a collective decision problem involving a set $\mathcal{A} = \{1, \ldots, m\}$ of agents. We assume that, for each agent $i \in \mathcal{A}$, a GAI utility $u^i : \mathcal{X} \to \mathbb{R}$, has been elicited on $\mathcal{X}$. For any feasible solution x and any agent $i \in \mathcal{A}$, the utility index $u^i(x)$ measures the satisfaction of agent i with solution x. The utility vector $(u^1(x), \ldots, u^m(x))$ represents the agents' satisfaction profile. Pareto and strict Pareto dominance relations are respectively defined by: $x >_P y \Leftrightarrow [\forall i \in \mathcal{A}, u^i(x) \geq u^i(y) \text{ and } \exists k \in \mathcal{A}, u^k(x) > u^k(y)]$, and $x \succcurlyeq_P y \Leftrightarrow [\forall i \in \mathcal{A}, u^i(x) > u^i(y)]$.

A solution $x \in \mathcal{X}$ such that $y >_P x$ for no $y \in \mathcal{X}$ is said to be *Pareto optimal*. Clearly, admissible solutions to the collective choice problem must be sought within the set of Pareto-optimal solutions. With such solutions indeed, there is no way of increasing the satisfaction of an agent without decreasing that of other agents. A milder optimality notion is given by *weak Pareto optimality* that concerns any solution $x \in \mathcal{X}$ such that $y \succcurlyeq_P x$ for no $y \in \mathcal{X}$. In order to explore the possible compromise solutions in the Pareto set, a classical approach in multiobjective optimization is to generate compromise solutions by minimizing the following scalarizing function (Wierzbicki 1986; Steuer and Choo 1983):

$$f_w(x) = \| w(\bar{u} - u(x)) \|_\infty = \max_{i \in \mathcal{A}} \{w_i | \bar{u}^i - u^i(x)|\}$$

where $\bar{u} = (\bar{u}^1, \ldots, \bar{u}^m)$ represents an ideal utility profile and w is a positive weighting vector. The choice of the Tchebycheff norm focuses on the worst component and therefore guarantees that only feasible solutions close to reference point $\bar{u}$ on *every* component will receive a good score. This promotes well-balanced solutions. Functions $f_w(x)$ fulfill two important properties (see (Wierzbicki 1986)):

Property 1: If $\forall i \in \mathcal{A}, w_i > 0$ then all solutions x minimizing $f_w(x)$ over the set $\mathcal{X}$ are weakly Pareto-optimal. Moreover at least one of them is Pareto-optimal.

Property 2. If $\forall i \in \mathcal{A}, \bar{u}^i > \sup_{x \in \mathcal{X}} u^i(x)$, then for any Pareto-optimal solution x, there exists a weighting vector w such that x is the unique solution minimizing $f_w(x)$ over $\mathcal{X}$.

Property 1 shows that minimizing $f_w(x)$ yields at least one Pareto-optimal solution. Property 2 shows that any Pareto-optimal solution can be obtained with the appropriate choice of parameters w. This second property is very important. It prevents excluding a priori good compromise solutions. Yet, it is not satisfied by usual linear aggregators:

Example 1 Consider a problem with 3 agents and assume that $\mathcal{X} = \{x, y, z, t\}$ with $u^1(x) = 0, u^2(x) = u^3(x) = 100, u^2(y) = 0, u^1(y) = u^3(y) = 100, u^3(z) = 0, u^1(z) = u^2(z) = 100, u^1(t) = u^2(t) = u^3(t) = 65$. All solutions except t are unacceptable for at least one agent. Thus t is the only possible compromise solution and it is Pareto-optimal; yet it cannot be obtained by maximizing a linear combination of individual utilities (with positive coefficients). ♦

This explains why $f_w(x)$, as a scalarizing function, is preferred to a weighted sum in multiobjective optimization on non-convex sets (Wierzbicki 1986; Steuer and Choo 1983).

Note that compromise search in a collective decision making problem can be seen as an interactive process. Standard techniques are known to set w so as to target on the center of the Pareto set and get a well-balanced compromise solution (Steuer and Choo 1983). Then parameter w might evolve during the interactive search to explore more specific areas of interest. To save space, we concentrate now on the main technical problem to solve at any step of the exploration: how to determine efficiently the optimal compromise solution within the product set $\mathcal{X}$.

Algorithms

The Ranking Approach for Compromise Search

Determining the optimal compromise solution among agents using function f_w introduced in the previous section is not straightforward because we face two difficulties: the combinatorial nature of $\mathcal{X}$ and the non-decomposability of function f_w (obviously it is not a GAI function). Even in simple cases where $u^i(x)$ can be computed in polynomial time

for every $x \in \mathcal{X}$ and every $i \in \mathcal{A}$, the problem of finding the solution minimizing $f_w(x)$ over $\mathcal{X}$ is NP-hard as soon as there are $n \geq 3$ attributes and $m \geq 2$ agents, each one having a GAI utility function including at least one factor of size greater than or equal to 3. This can be proved using a reduction from 3-SAT. Indeed, consider an instance of 3-SAT with n variables and m clauses. To each variable, we associate a Boolean attribute X_i and to any clause C_j over variables we associate an agent with Boolean function u^j. For instance $C_j = x \vee y \vee \neg z$ will be represented by function $u^j(x, y, z) = 1 - (1 - x)(1 - y)z$. Then we set $\bar{u}^i = 1$ for all $i \in \mathcal{A}$. Hence, the optimal value of f_w over $\mathcal{X} = X_1 \times \cdots \times X_n$ with functions $u^1, \ldots, u^m$ is 0 if and only if the initial 3-SAT problem is feasible.

We now introduce an original method to determine the f_w-optimal tuple in $\mathcal{X}$. It is based on a 3-step procedure:

Step 1: scalarization. We consider the overall utility function $u_w(x) = 1/m \sum_{i=1}^{m} w_i u^i(x)$. Such a function provides a lower approximation of $f_w(x)$: considering $U = 1/m \sum_{i=1}^{m} w_i \bar{u}_i$, it is clear that $U - u_w(x) \leq f_w(x)$ for all $x \in \mathcal{X}$ (the average is lower than the maximum). Note that $u_w(x)$ is much easier to optimize than $f_w(x)$ since, as the weighted sum of GAI functions, it is also a GAI function that can be compiled into a GAI-net structure.

Step 2: ranking. we enumerate the solutions of $\mathcal{X}$ by decreasing utility $u_w(x)$. Here, an efficient ranking algorithm exploiting the GAI structure of u_w to speed-up enumeration is needed. This point will be the core of the next subsection.

Step 3: stopping condition. The ranking is stopped as soon as we reach a solution x^k such that $u_w(x^k) \leq U - f_w(x^*)$ where x^* minimizes f_w among the already detected solutions. This cut is justified by the following result:

Proposition 1 *Let $x^1, ..., x^k$ be the ordered sequence of k-best solutions generated during Step 2 with function u_w, if $u_w(x^k) \leq U - f_w(x^*)$ with $x^* = Argmin_{i=1,\ldots,k} f_w(x^i)$ then x^* is optimal for f_w, i.e. $f_w(x^*) = \min_{x \in \mathcal{X}} f_w(x)$.*

Proof. For any $i > k$, by construction, $f_w(x^i) \geq U - u_w(x^i) \geq U - u_w(x^k)$. Since $U - u_w(x^k) \geq f_w(x^*)$, by hypothesis, we get $f_w(x^i) \geq f_w(x^*)$, which shows that no solution found after step k in the ranking can improve the current best solution x^*. □

Ranking Using a GAI Function

Let u be a GAI-decomposable utility w.r.t. some X_{Z_i}'s. The procedure we present in this subsection for ranking elements w.r.t. u heavily relies on another one designed to answer *choice queries*, that is, to find the preferred tuple over $\mathcal{X}$. To avoid exhaustive pairwise comparisons which would be too prohibitive due to the combinatorial nature of $\mathcal{X}$, both procedures take advantage of the structure of the GAI net to decompose the query problem into a sequence of local optimizations, hence keeping the computational cost of the overall ranking task at a very admissible level. We briefly present the choice algorithm and, then, derive the general ranking procedure. The former corresponds to solving:

$$\max_{x \in \mathcal{X}} u(x_1, \ldots, x_n) = \max_{x \in \mathcal{X}} \sum_i u_i(x_{Z_i}). \qquad (1)$$

The optimum can be found efficiently by exploiting that:

1. max over $X_1, \ldots, X_n$ of $u(X_1, \ldots, X_n)$, can be decomposed as $\max_{X_1} \ldots \max_{X_n} u(X_1, \ldots, X_n)$, and the order in which the max's are performed is unimportant;
2. if $u(X_1, \ldots, X_n)$ can be decomposed as $f() + g()$ where $f()$ does not depend on X_i, then $\max_{X_i}[f() + g()] = f() + \max_{X_i} g()$;
3. in a GAI-net, the running intersection ensures that a variable contained in an outer clique C (i.e. a clique with at most one neighbor) but not contained in C's neighbor does not appear in the rest of the net.

Properties 2 and 3 suggest computing the max recursively by first maximizing over the variables contained only in the outer cliques as only one factor is involved in these computations, then adding the result to the factor of their adjacent clique, remove these outer cliques and iterate until all cliques have been removed. This leads to the algorithm below, where $F = \emptyset$ the first time we call `Collect` (F just avoids infinite loops due to line 03 of the algorithm).

Function `Collect`(clique C_i, F)
01 for all C_j in {cliques adjacent to C_i}$\backslash F$ in the GAI-net do
02 let $S_{ij} = C_i \cap C_j$ be the separator between C_i and C_j
03 let u_j^* be defined on S_{ij} by `Collect`(C_j, {C_i})
04 substitute $u_i(x_{C_i})$ by $u_i(x_{C_i}) + u_j^*(x_{S_{ij}})$ for all x_{C_i}'s
05 done
06 if $F \neq \emptyset$ then
07 let C_j be the only clique $\in F$ and let $S_{ij} = C_i \cap C_j$
08 let $M_i^*(x_{S_{ij}}) = \text{Argmax}\{u_i(y_{C_i}) : y_{S_{ij}} = x_{S_{ij}}\}$ and let $u_i^*(x_{S_{ij}}) = u_i(M_i^*(x_{S_{ij}}))$ for all $x_{S_{ij}}$ in $\prod_{X_k \in S_{ij}} X_k$
09 store matrix M_i^* in separator S_{ij} and return u_i^*
10 endif

This function recursively reduces Eq. (1) by removing one by one all the subutilities (extracting their max). Thus, calling function `Collect` on any clique returns the value of the utility of the most preferred element. For instance, on the example of Figure 1, applying `Collect`$(BCD, \emptyset)$ results in the message propagations described in Figure 2, where:
$u_1^*(B) = \max_A u_1(A, B)$, $M_1(B) = \text{Argmax}_A u_1(A, B)$
$u_2^*(C) = \max_E u_2(C, E)$, $M_2(C) = \text{Argmax}_E u_2(C, E)$
$u_5^*(B) = \max_G u_5(B, G)$, $M_5(B) = \text{Argmax}_G u_5(B, G)$
$u_4'(B, D, F) = u_4(B, D, F) + u_5^*(B)$
$u_4^*(B,D) = \max_F u_4'(B,D,F)$, $M_4(B,D) = \text{Argmax}_F u_4'(B,D,F)$

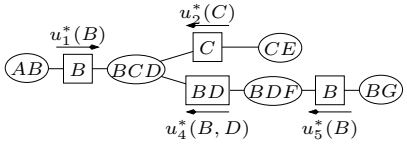

Figure 2: The Collect phase

At the end of the collect, $\max_{BCD} u_3^*(B, C, D) = u_3(B, C, D) + u_1^*(B) + u_2^*(C) + u_4(B, D)$ corresponds to the maximum value of the utility. Let $(\hat{b}, \hat{c}, \hat{d})$ be a solution to $\max_{BCD} u_3^*(B, C, D)$. Then $(\hat{b}, \hat{c}, \hat{d})$ is obviously a projection on $B \times C \times D$ of a most preferred element of $\mathcal{X}$. But the corresponding utility is $u_3(\hat{b}, \hat{c}, \hat{d}) + u_1^*(\hat{b}) + u_2^*(\hat{c}) + u_4(\hat{b}, \hat{d})$ which, in turn, is obtained at $M_1(\hat{b})$, $M_2(\hat{c})$ and $M_4(\hat{b}, \hat{d})$. Finally, $M_4(\hat{b}, \hat{d})$ corresponds to utility value

$u_4(\hat{b},\hat{c},\hat{d}) + u_5^*(\hat{b})$, obtained at $M_5(\hat{b})$. Consequently, the optimal tuple can be obtained by propagating recursively the attributes instantiations (the M_i's) from clique BCD toward the outer cliques, as shown on Figure 3. This leads to the following algorithm, where F is the last clique that called `Instantiate` and x_F is an instantiation of F's attributes:

Function `Instantiate`(C_i, F, x_F)
01 if $F = \emptyset$ then
02 let $x^*_{C_i}$ = Argmax$\{u_i(x_{C_i}) : x_{C_i} \in \prod_{X_k \in C_i} X_k\}$
03 else
04 let C_j be the only clique $\in F$ and let $x^*_{C_j} = x_F$
05 let $S_{ij} = C_i \cap C_j$ and $D_{ij} = C_i \backslash C_j$
06 let $x^*_{C_i}$ = Argmax$\{u_i(x^*_{S_{ij}}, y_{D_{ij}})\}$
07 endif
08 let $\{C_{i_1}, \ldots, C_{i_k}\}$ = {cliques adjacent to C_i}$\backslash F$
09 foreach j varying from 1 to k do
10 let x^{i_j} = `Instantiate` $(C_{i_j}, \{C_i\}, x^*_{C_i})$
11 let y^{i_j} = tuple x^{i_j} without the values of the attributes in S_{ij}
12 return tuple $(x^*_{C_i}, y^{i_1}, \ldots, y^{i_k})$

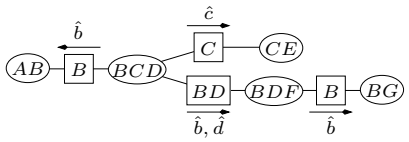

Figure 3: The Instantiation phase

Function `Optimal_choice`(GAI-net)
01 Let C_0 be any clique in the GAI-net
02 call `Collect`($C_0, \emptyset, \emptyset$) and let x^* = `Instantiate`($C_0, \emptyset, \emptyset$)
03 return the optimal choice x^*

As for ranking, consider the example of Fig. 2. Assume that `Optimal_choice` returned $x^* = (\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{e}, \hat{f}, \hat{g})$. Then, the next best tuple, say x^2, differs from x^* by at least one attribute, i.e. there exists a clique C_i such that the projection of x^2 on C_i differs from that of x^*. As we do not know on which C_i the difference occurs, we can test all the possibilities and partition the feasible space into:
Set 1: $(B, C, D) \neq (\hat{b}, \hat{c}, \hat{d})$
Set 2: $(B, C, D) = (\hat{b}, \hat{c}, \hat{d})$ and $(B, D, F) \neq (\hat{b}, \hat{d}, \hat{f})$
Set 3: $(B, C, D, F) = (\hat{b}, \hat{c}, \hat{d}, \hat{f})$ and $(B, G) \neq (\hat{b}, \hat{g})$
Set 4: $(B, C, D, F, G) = (\hat{b}, \hat{c}, \hat{d}, \hat{f}, \hat{g})$ and $(C, E) \neq (\hat{c}, \hat{e})$
Set 5: $(B, C, D, E, F, G) = (\hat{b}, \hat{c}, \hat{d}, \hat{e}, \hat{f}, \hat{g})$ and $(A, B) \neq (\hat{a}, \hat{b})$

The construction of the above sets follows the decomposition advocated in (Nilsson 1998): the cliques in which the attributes are constrained to be different from those of x^* are enumerated in the order in which the cliques are called by function `Collect` within the call to `Optimal_choice`. Sets 1 to 5 above thus correspond to a collect phase encountering successively cliques (B, C, D), (B, D, F), (B, G), (C, E) and (A, B). Finding the best element in a given Set is essentially similar to finding the optimal choice except that lines 02 and 06 in function `Instantiate` need be modified to avoid some instantiations (like $(\hat{b}, \hat{c}, \hat{d})$).

Assume now that the second best tuple, say $x^2 = (a^2, b^2, \hat{c}, d^2, \hat{e}, \hat{f}, g^2)$, is the optimal choice of Set 1. Then the next tuple, x^3, is the best tuple that is different from both x^* and x^2. It can be retrieved using the same process. As x^2 is in Set 1, we should substitute Set 1 by the sets below to exclude x^2 and, then, iterate the same process:
Set 1.1: $(B, C, D) \notin \{(\hat{b}, \hat{c}, \hat{d}), (b^2, \hat{c}, d^2)\}$
Set 1.2: $(B, C, D) = (b^2, \hat{c}, d^2)$ and $(B, D, F) \neq (b^2, d^2, \hat{f})$
Set 1.3: $(B, C, D, F) = (b^2, \hat{c}, d^2, \hat{f})$ and $(B, G) \neq (b^2, g^2)$
Set 1.4: $(B, C, D, F, G) = (b^2, \hat{c}, d^2, \hat{f}, g^2)$ and $(C, E) \neq (\hat{c}, \hat{e})$
Set 1.5: $(B, C, D, E, F, G) = (b^2, \hat{c}, d^2, \hat{e}, \hat{f}, g^2)$ and $(A, B) \neq (a^2, b^2)$
This justifies the following algorithm:

Function `k-best`(GAI-net, k)
01 let x^* be the tuple resulting from `Optimal_choice`
02 let $\mathcal{S} = \{\text{Sets } i\}$ as described above and let $kbest = \emptyset$
03 for each Set i, let opt(Set i) be the optimal choice in Set i
04 for i = 2 to k do
05 let Set j be an arbitrary element of
$\{\text{Set } j \in \mathcal{S} : opt(\text{Set } j) \geq opt(\text{Set } p), \text{Set } p \in \mathcal{S}\}$
06 let x^i, the ith best element, be opt(Set j)
07 add x^i to $kbest$ and remove Set j from $\mathcal{S}$
08 substitute Set j in $\mathcal{S}$ by sets $\not\ni \{x^i\}$ as described above
10 return $kbest$

Numerical Tests

To evaluate our approach in practice, we have performed experiments on various instances of multiattribute multiagent search problems, using a Java program running on a 2.1GHz PC. We have recorded computation times and the number of solutions generated before returning the optimal compromise solution according to a weighted Tchebycheff norm.

Test Data

To run the experiments, we generated synthetic data for GAI-decomposable preferences. All GAI decompositions involved 20 attributes, with 10 subutilities $u_i(x_{Z_i})$ involving a number of attributes chosen randomly between 2 and 4. It does not seem realistic to consider higher-order interactions as far as human preference modeling is concerned (such complex interaction might actually be very difficult to assess in practice). Each u_i's domain variables were randomly selected from the set of all attributes. For variables that were not selected in any subutility, we created unary subutilities. Next, we created m different utilities for the structure previously generated, representing the preferences of m agents. We performed the tests for $m \in \{2, 5, 7\}$. For each subutility u_i^j of an agent j, we first generated its maximum value $\max(u_i^j)$, in the interval $[0, 1]$. Then we uniformly generated the utility values for all configurations of u_i^j in the interval $[0, \max(u_i^j)]$. This gave us m different GAI-decomposable utilities with the same structure. We generated test data for variables of domain sizes 2, 5 and 10, resp. giving problems with 2^{20}, 5^{20} and 10^{20} total possible configurations. This means that we have subutility terms with tables containing up to 10000 elements (4 variables with domain size 10).

Results

The observed results are summarized in Table 1. As can be seen, average times range from 0.02 to 44.65 seconds. It can be seen that the attributes domain sizes and the number of agents are decisive to the efficiency of the proposed algorithm. For groups of 3 and 5 agents, the average times were below 1 second, indicating that the algorithm can be used to

Table 1: Average results (t: time in seconds–values in parentheses are the sample standard deviation; $\#gen$: number of generated solutions) over 30 runs, for 3, 5 and 7 agents

Domain	$m = 3$			$m = 5$			$m = 7$		
size	t(s)		$\#gen$	t(s)		$\#gen$	t(s)		$\#gen$
2	0.02	(0.02)	155	0.04	(0.03)	1447	0.21	(0.16)	8327
5	0.04	(0.03)	397	0.63	(0.58)	23552	10.03	(9.77)	342216
10	0.15	(0.06)	672	1.28	(1.24)	43961	44.65	(43.53)	1507528

give instant responses to small groups, even when the solution space is big (note that for 20 attributes of domain size 5 to 20 attributes of domain size 10, the number of possible configurations is multiplied by over 10^6). For $m = 7$ performance in response time is about 45 seconds which begins to be critical for online recommendations. However, it is not so frequent to engage a large group into a collective decision making session using sophisticated preference models.

We also ran experiments where each agent had a different GAI decomposable preference structure. In these cases, to generate the aggregated GAI network we triangulated the Markov graph induced by the subutilities of all the agents. The more the discrepancy between the agents structures, the larger the cliques, and the less efficient our algorithm. Whenever the GAI network structures were very different, it turned out to be impossible to conduct the ranking procedure due to the too large amount of memory required to fill the cliques. However, there are many practical situations where interacting attributes are almost identical for all agents, the difference between individual utilities being mainly due to discrepancies in utility values.

Conclusion

In this paper we have shown how GAI-nets could be used not only to efficiently perform individual recommendations (choice and ranking) on combinatorial sets, but also to solve collective recommendation queries in multiagent decision problems. Our procedure allows the determination of various types of compromise solutions and remains very efficient provided the number of agents is not too large. It might be used in many real-world situations like preference-based design of a collective holidays-trip, or for content-based movie recommendations for a group, and fair allocation in combinatorial auctions problems. Finally, further sophistications of our approach are possible. For instance, the ranking algorithm can be improved using a dynamic selection of the clique passed in argument to the collect/instantiation phases (depending on the tuples ranked so far).

References

Bacchus, F., and Grove, A. 1995. Graphical models for preference and utility. In *UAI'95*.

Boutilier, C.; Bacchus, F.; and Brafman, R. 2001. UCP-networks; a directed graphical representation of conditional utilities. In *UAI'01*.

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004a. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. of Artificial Intelligence Research* 21:135–191.

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004b. Preference-based constraint optimization with CP-nets. *Computational Intelligence* 20.

Brafman, R.; Domshlak, C.; and Kogan, T. 2004. On generalized additive value-function decomposition. In *UAI'04*.

Braziunas, D., and Boutilier, C. 2005. Local utility elicitation in GAI models. In *UAI'05*.

Chevaleyre, Y.; Endriss, U.; and Lang, J. 2007. Expressive power of weighted propositional formulas for cardinal preference modeling. In *Proceedings of KR'07*, 145–152.

Cowell, R.; Dawid, A.; Lauritzen, S.; and Spiegelhalter, D. 1999. *Probabilistic Networks and Expert Systems*. Stat. for Engineering and Information Science. Springer-Verlag.

Debreu, G. 1964. Continuity properties of Paretian utility. *International Economic Review* 5:285–293.

Engel, Y., and Wellman, M. P. 2007. Generalized value decomposition and structured multiattribute auctions. In *EC'07*, 227–236.

Fishburn, P. C. 1970. *Utility Theory for Decision Making*. Wiley.

Gonzales, C., and Perny, P. 2004. GAI networks for utility elicitation. In *KR'04*, 224–234.

Gonzales, C., and Perny, P. 2005. GAI networks for decision making under certainty. In *IJCAI'05 – Workshop on Advances in Preference Handling*.

Jensen, F. 1996. *An introduction to Bayesian Networks*. Taylor and Francis.

Krantz, D.; Luce, R. D.; Suppes, P.; and Tversky, A. 1971. *Foundations of Measurement (Additive and Polynomial Representations)*, volume 1. Academic Press.

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

Pini, M. S.; Rossi, F.; Venable, K. B.; and Walsh, T. 2005. Aggregating partially ordered preferences: Impossibility and possibility results. *TARK* 10.

Rossi, F.; Venable, K. B.; and Walsh, T. 2004. mCP nets: Representing and reasoning with preferences of multiple agents. In *AAAI'04*, 729–734.

Steuer, R., and Choo, E.-U. 1983. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming* 26:326–344.

Wierzbicki, A. 1986. On the completeness and constructiveness of parametric characterizations to vector optimization problems. *OR Spektrum* 8:73–87.