Preference aggregation in combinatorial domains using GAI-nets

Christophe Gonzales, Patrice Perny, Sergio Queiroz

Preference aggregation in combinatorial domains using GAI-nets

Christophe Gonzales*, Patrice Perny*, Sergio Queiroz*

Abstract

This paper deals with preference representation and aggregation in combinatorial domains. We assume that the set of alternatives is defined as the cartesian product of finite domains and that agents' preferences are represented by generalized additive decomposable (GAI) utility functions. GAI functions allow an efficient representation of interactions between attributes while preserving some decomposability of the model. We address the preference aggregation problem and consider several criteria to define the notion of compromise solution (maxmin, minmaxregret, weighted Tchebycheff distance). For each of them, we propose a fast procedure for the exact determination of the optimal compromise solution in the product set. This procedure relies on a ranking algorithm enumerating solutions according to the sum of the agents individual utilities until a boundary condition is reached. We provide results of numerical experiments to highlight the practical efficiency of our procedure.

Key words : GAI-nets, Preference Modelling, Preference Aggregation

1 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. In this respect, 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 an

*Laboratoire d'informatique de Paris VI, Université Paris VI, 8 rue du capitaine Scott, 75015 Paris, France. forname.name@lip6.fr

admissible level as well as the need for fast procedures to solve preference-based optimization problems on large sets of alternatives. 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 [3, 4] and mCP-nets [18], their extension to the multiagent case. Such models are deliberately simple and flexible enough to be integrated efficiently in interactive recommendation systems; the preferences of the agents must be captured using only a few questions so as to perform a fast preference-based search over the possible items.

In other 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, utilities can significantly outperform qualitative models due to their higher descriptive power [2]. Moreover, the use of cardinal utilities in the multiagent setting allows to escape the framework of Arrow's impossibility theorem which considerably restricts aggregation possibilities [17].

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 the preferences are representable by an additive utility [14]. 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* [1]. 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 [10], that allow more general interactions between attributes [1] while preserving some decomposability. Such a decomposition has been used to endow CP-nets with utility functions (UCP-nets) both under uncertainty [2] and under certainty [5].

In the same direction [11, 6] propose a general procedure to assess GAI utilities in decision under risk. This one is directed by the structure of a graphical model called a GAI-network and consists 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 decision making under certainty. [12] thus proposes 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 the GAI-optimal element in a product set, using classical propagation techniques used in the Bayesian net 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 Section 2, we discuss individual and collective preference representation. After recalling some elements about GAI models and their graphical representation we consider various aggregation criteria to define the notion of compromise between agents. In Section 3, we provide efficient algorithms to determine the best compromise solution for the group of agents. Finally Section 4 reports numerical results obtained on multiagent aggregation problems.

2 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, (possibly superscripted) 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). Subsection 2.1 concerns the representation of the preference of a single agent. We will consider the multiagent decision problem in Subsection 2.2.

2.1 Individual Preference Modeling

Under mild hypotheses [8], it can be shown that $\succsim$ is representable by a utility, i.e., there exists a function $u : \mathcal{X} \mapsto \mathbb{R}$ such that $x \succsim y \Leftrightarrow u(x) \geq u(y)$ 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 $u_i(\alpha)$ for any $\alpha \in X_i$, $i = 1, \ldots, n$. 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:*

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

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 [1]. 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:*

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

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 we call *GAI networks* [11] which are essentially similar to the junction graphs used for Bayesian networks [13, 7]:

Definition 2 (GAI network) *Let* $\mathcal{X} = \prod_{i=1}^{n} X_i$*. Let* $Z_1, \ldots, Z_k$ *be a covering of* $N = \{1, \ldots, n\}$*. 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 network 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 networks. As mentioned in [11], this is not restrictive as general GAI nets can always be compiled into GAI trees. For any GAI decomposition, by Definition 2 the cliques of the GAI network 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 Figure 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 [7]).

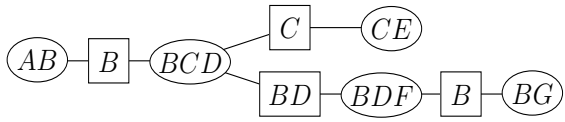

Figure 1: A GAI tree

2.2 Collective Decision Making

We consider now a multiagent decision problem involving a set $\mathcal{A}=\{1,\ldots,m\}$ of agents. We assume that, for each agent $j \in \mathcal{A}$, a GAI utility u^j representing her preferences over $\mathcal{X}$ has been elicited. For simplicity, we assume that each agent use the same absolute utility scale, so as to have commensurability. Then, a classical way of defining the best compromise solution for the group of agents is to define an overall utility $u(x)$ which gives, for any $x \in \mathcal{X}$, the value of x for the group. Thus we consider $u(x) = h(u^1(x),\ldots,u^m(x))$ where h is an aggregation function implicitly defining the type of compromise seek in $\mathcal{X}$. The best compromise solution is obtained by optimizing u over $\mathcal{X}$. When h is non-decreasing in each component, u-optimal solutions are weakly Pareto-optimal (i.e. there is no other solution improving the satisfaction of all the agents). Moreover, if h is strictly increasing in each component then u-optimal solutions are Pareto-optimal (i.e. there is no other solution improving the satisfaction of an agent without decreasing the satisfaction of another).

If h is linear, then u is the sum of GAI utilities and, as such, is itself a GAI utility. Then the problem reduces to a monoagent decision problem with a GAI utility. However, linear aggregation functions are not good candidates as they may lead to choose a solution having a very ill-balanced utility profile. For example, consider a problem with 3 agents and assume that $\mathcal{X} = \{x, y, z, w\}$ with $u^1(x) = 0, u^2(x) = u^3(x) = 100$, $u^2(y) = 0, u^1(y) = u^3(y) = 100$, $u^3(z) = 0, u^1(z) = u^2(z) = 100$, $u^1(w) = u^2(w) = u^3(w) = 65$. All solutions except w are unacceptable for at least one agent. Thus w is the only possible compromise solution; yet it cannot be obtained by maximizing a linear combination (with positive coefficients) of agent utilities. To find better compromise solutions, the following non-linear criteria seem more adequate:

The maximin criterion: $u(x) = \min_{j\in\mathcal{A}} u^j(x)$, to be maximized over $\mathcal{X}$. This criterion amounts to maximize the satisfaction of the least satisfied agent.

The minimax Regret criterion: $u(x) = \max_{j\in\mathcal{A}} r^j(x)$, where $r^j = u(x^j) - u^j(x)$, and $x^j = \text{Argmax}_{x\in\mathcal{X}} u^j(x)$ (i.e. x^j denotes the optimal solution for agent j). This criterion is to be minimized over $\mathcal{X}$. It represents the maximal regret among agents, the regret r^j of agent j being defined as the utility gap between x and x^j.

The Tchebycheff criterion: $u(x) = \max_{j\in\mathcal{A}} w_j r^j(x)$ where w_j are positive coefficients. This criterion is to be minimized over $\mathcal{X}$. It represents the distance (w.r.t. a weighted

Tchebycheff norm) between two utility profiles: $(u^1(x), \ldots, u^n(x))$ obtained with solution x, and the ideal utility profile $(u^1(x^1), \ldots, u^m(x^m))$ corresponding to a fictitious ideal situation (generally not feasible) in which all agents are optimally satisfied simultaneously. This ideal point is an upper bound of the set of Pareto non-dominated solutions. Tchebycheff criterion is a standard tool for generating compromise solutions in multiobjective optimization [19]. It can be seen as a sophistication of the minimax regret criterion using coefficient defined by $w_j = \omega_j/(u(x^j) - u^j_*)$ where ω_j is the weight attached to agent j and $u^j_* = \min_{k \in \mathcal{A}} u^k(x^j)$. The term $1/(u(x^j) - u^j_*)$ is a normalization factor. It can be used as a correcting factor when the range of utility values attached to solutions significantly varies from an agent to another. It can also be useful when the agents'utilities are not commensurate. The weights allow to modulate the importance of agents and to control the type of compromise. As shown in [20], any Pareto optimal solution can be obtained by optimizing the Tchebycheff criterion with appropriate choices of ω_j.

The above aggregation functions are weakly increasing w.r.t. each component, which only ensures weak-Pareto optimality. However, they might be slightly modified to become strictly increasing (so as to generate Pareto-optimal solutions). For example, it is sufficient to add (resp. subtract) $\epsilon \sum_{i \in \mathcal{A}} u^i(x)$ for maximization (resp. minimization), ϵ being an arbitrarily small positive value. We discuss in the next section the optimization of each of the above functions to determine the better compromise solutions between agents.

3 Algorithms

3.1 The ranking approach for compromise search

Determining the optimal compromise solution among agents using a function of type $u(x) = h(u^1(x), \ldots, u^m(x))$ is not trivial because we face two difficulties: the combinatorial nature of $\mathcal{X}$ and the possible non-decomposability of function u (h is not linear). For example, if $h = \min$ (maximin utility), the determination of the best compromise solution is NP-hard as soon as there are $n \geq 3$ attributes, $m \geq 2$ agents, each 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 $u^j(x, y, z) = 1 - (1 - x)(1 - y)z$. Hence, the optimal value of the maximin optimization problem over $\mathcal{X} = X_1 \times \cdots \times X_n$ with functions $u^1, \ldots, u^m$ is 1 if and only if the initial 3-SAT problem is feasible. Similar reductions might be proposed for minimax regret and Tchebycheff criteria which establish the complexity of the search of a good compromise solution. To overcome the problem and be able to optimize a non-decomposable function f on $\mathcal{X}$, we suggest resorting to a

ranking approach based on the following 3-stage procedure:

Step 1: scalarization. we reformulate the problem as a monoagent problem, using an overall criterion $g(x)$ defined as a linear combination of individual utilities. Such a function is easier to optimize than f since, as the sum of GAI functions, it is also a GAI function.

Step 2: ranking. we enumerate the solutions of $\mathcal{X}$ from best to worst using function $g(x)$. Here, an efficient ranking algorithm exploiting the GAI structure of g to speed-up enumeration is needed. This point will be the core of subsection 3.2.

Step 3: stopping condition. we need to stop enumeration as early as possible due to the size of set $\mathcal{X}$. This can be done efficiently using the following proposition.

Proposition 1 *Consider two functions f and g defined from $\mathbb{R}^n$ into $\mathbb{R}$, to be minimized over $\mathcal{X}$, and such that $f(x) \geq g(x)$ for all $x \in \mathcal{X}$. Let $x^1, ..., x^k$ be the ordered sequence of k-best solutions generated during Step 2 with function g, if $g(x^k) \geq f(x^*)$ with $x^* = Argmin_{i=1,...,k} f(x^k)$ then x^* is optimal for f, i.e. $f(x^*) = \min_{x \in \mathcal{X}} f(x)$.*

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

This simple result can be applied to criteria introduced in the previous section. The following table gives g, the GAI approximation to be used, for any criterion f considered:

criterion	$f(x)$	$g(x)$
maximin	$-\min_j(u^j(x))$	$-1/m \sum_j u^j(x)$
minimax regret	$\max_j(r^j(x))$	$1/m \sum_j r_j(x)$
Tchebycheff	$\max_j(w_j r^j(x))$	$1/m \sum_j w_j r_j(x)$

Since $1/m \sum_j r_j(x)$ only differs from $-1/m \sum_j u^j(x)$ by a constant term, the same ranking algorithm can be used for maximin and minimax regret criteria. Tchebycheff is just an extension using weights w_j and does not raise additional issues. Let us now explain how to proceed to rank elements of $\mathcal{X}$ with a GAI function g (step 2).

3.2 Ranking using a GAI function

The ranking procedure we present in this subsection 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 network 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 first briefly present the choice algorithm and, then, derive the general ranking procedure. The former corresponds to solving:

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

The optimum can be found efficiently by exploiting the following properties:

1. the 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 following 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 (by 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:

$u_1^*(B) = \max_A u_1(A,B)$, $M_1(B) = \text{Argmax}_A u_1(A,B)$
$u_2^*(C) = \max_E u_2(C,E)$, $M_2(C) = \text{Argmax}_E u_2(C,E)$
$u_5^*(B) = \max_G u_5(B,G)$, $M_5(B) = \text{Argmax}_G u_5(B,G)$
$u_4'(B,D,F) = u_4(B,D,F) + u_5^*(B)$
$u_4^*(B,D) = \max_F u_4'(B,D,F)$, $M_4(B,D) = \text{Argmax}_F u_4'(B,D,F)$

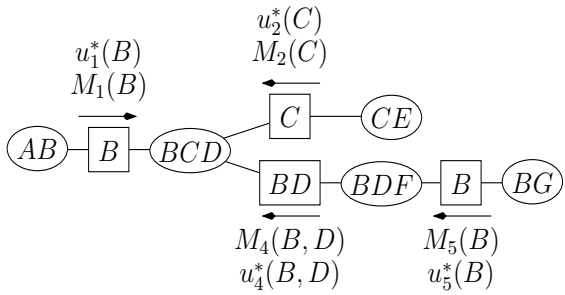

Figure 2: The Collect phase

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 function. 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:

Function `Instantiate`(C_i, F, x_F)
01 if $F = \emptyset$ then
02 let $x^*_{C_i} = \text{Argmax}\{u_i(x_{C_i}) : x_{C_i} \in \prod_{X_k \in C_i} X_k\}$
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 $x^*_{C_i} = \text{Argmax}\{u_i(x^*_{S_{ij}}, y_{D_{ij}})\}$
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} be tuple x^{i_j} without the values of the attributes in S_{ij}
12 return tuple $(x^*_{C_i}, y^{i_1}, \ldots, y^{i_k})$

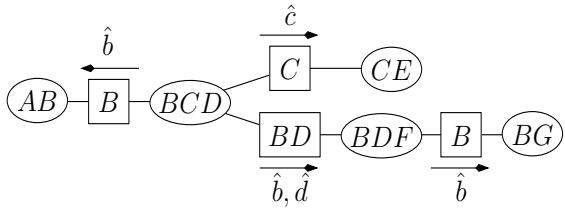

Figure 3: The Instantiation phase

Function `Optimal_choice`(GAI-net)
01 Let C_0 be any clique in the GAI-net
02 call `Collect`(C_0, $\emptyset$) and let x^* = `Instantiate`(C_0,$\emptyset$,$\emptyset$,$\emptyset$)
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 by [16]: 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 $\not\supseteq \{x^i\}$ as described above
10 return $kbest$

4 Numerical Tests

To evaluate our approach in practice, we have performed experiments on various instances of the multiattribute multiagent search problem. We have recorded computation times and the number of solutions generated before returning the optimal compromise solution for each of the three criteria discussed in the paper. The experiments were performed on a 3.2GHz PC with a Java program.

4.1 Test data

To run the experiments, we generated synthetic data for GAI-decomposable preferences. All GAI decompositions involved 20 variables, with 10 subutilities $u_i(x_{Z_i})$ of domain size $|x_{Z_i}|$ randomly drawn between 2 to 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 variables. For variables that were not selected in any subutility function, we created unary subutilities. Next, we created 5 different utility functions for the structure previously generated, representing the preferences of 5 agents. For each subutility function 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 5 different GAI-decomposable utility

functions 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} possible configurations.

4.2 Results

The average results (t: times in ms and $\#gen$: number of generated solutions) over 100 runs are summarized below:

Domain	maximin		minimax Regret		Tchebycheff	
size	t(ms)	$\#gen$	t(ms)	$\#gen$	t(ms)	$\#gen$
2	371	9145	118	2437	96	1916
5	2784	60020	1100	22364	1052	21640
10	4703	99709	3934	78518	3104	60479

As it can be seen, we obtained average times ranging from 0.1 to 4.7 seconds, depending on the compromise criterion and the attributes domain size. Finding the maximin compromise solution required the enumeration of more solutions than minimax Regret and Tchebycheff. However, we can see that the most prominent factor is the attributes domain size. Fortunately, the number of elements that need be enumerated before returning the solution increases at a much lower rate than the problem size. For instance, from 20 attributes of domain size 5 to 20 attributes of domain size 10, the number of configurations is multiplied by over 10^6 while, at the same time, the average number of solutions enumerated increased by a factor less than 3. 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.

5 Conclusion

In this paper we have shown how GAI-networks could be used not only to efficiently perform individual recommendations (choice and ranking) on combinatorial sets, but also to solve collective recommendation requests for multiagent decision problems. The proposed procedure allows the determination of various types of compromise solutions and remains very efficient provided the number of agents is not too important. It might be used in many real-world situations like preference-based design of an holidays-trip for a

group, preference-based configuration of a car for the family, or for content-based movie recommendation tasks for a group of friends.

Further sophistication of our approach are possible, for instance using AND/OR trees [9, 15] within the GAI structure instead of computing the whole u_i^*'s during the collect phases and performing substitutions of whole utility tables. The ranking algorithm can therefore be improved using a dynamic selection of the clique passed in argument to the collect/instantiation phases (depending on the tuples ranked so far). This is left for further research.

References

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

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

[3] C Boutilier, R Brafman, C Domshlak, H Hoos, and D Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. A.I. Research*, 21:135–191, 2004.

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

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

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

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

[8] G Debreu. Continuity properties of paretian utility. *Int. Econ. Review*, 5:285–293, 1964.

[9] R Dechter. AND/OR search spaces for graphical models. Technical report, ICS, 2004.

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

[11] C Gonzales and P Perny. GAI networks for utility elicitation. In *KR'04*, 2004.

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

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

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

[15] R Marinescu and R Dechter. AND/OR tree search for constraint optimization. In *CP'04 – Workshop on Pref. and Soft Constraints*, 2004.

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

[17] Pini, Rossi, Venable, and Walsh. Aggregating partially ordered preferences: Impossibility and possibility results. *TARK*, 10, 2005.

[18] Francesca Rossi, Kristen Brent Venable, and Toby Walsh. mCP nets: Representing and reasoning with preferences of multiple agents. pages 729–734, 2004.

[19] R.E. Steuer and E.-U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Math. Prog.*, 26:326–344, 1983.

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