Multiobjective Optimization using GAI Models

Jean-Philippe Dubus **Christophe Gonzales** **Patrice Perny**
LIP6 - UPMC
104 avenue du président Kennedy, F-75016 Paris
firstname.lastname@lip6.fr

Abstract

This paper deals with multiobjective optimization in the context of multiattribute utility theory. The alternatives (feasible solutions) are seen as elements of a product set of attributes and preferences over solutions are represented by generalized additive decomposable (GAI) utility functions modeling individual preferences or criteria. Due to decomposability, utility vectors attached to solutions can be compiled into a graphical structure closely related to junction trees, the so-called GAI net. We first show how the structure of the GAI net can be used to determine efficiently the exact set of Pareto-optimal solutions in a product set and provide numerical tests on random instances. Since the exact determination of the Pareto set is intractable in worst case, we propose a near admissible algorithm with performance guarantee, exploiting the GAI structure to approximate the set of Pareto optimal solutions. We present numerical experimentations, showing that both utility decomposition and approximation significantly improve resolution times in multiobjective search problems.

1 Introduction

The complexity of human decision making in organizations, the importance of the issues raised in decision problems and the increasing need to explain or justify any decision has led decision makers to seek a scientific support in the preparation of their decisions. During many years, rational decision making was understood as solving a single-objective optimization problem, the optimal decision being implicitly defined as a feasible solution minimizing a cost function under some technical constraints. However, the practice of decision making in organizations has shown the limits of such formulations. First, there is some diversity and subjectivity in human preferences that requires distinguishing between the objective description of the alternatives of a choice problem and their value as perceived by individuals. In decision theory, alternatives are often seen as multiattribute items characterized by a tuple in a product set of attributes domains, the value system of each individual being encoded by a utility function defined on the multiattribute space and measuring the relative attractiveness of each tuple. Hence objectives of individuals take the form of multiattribute utilities to be optimized. Typically, in a multiagent decision problem, we have to deal with several such utilities that must be optimized simultaneously. Since individual utilities are generally not commensurate, this is not always possible to construct an overall utility that might simplify the optimization task and we have to solve a multiobjective problem. Moreover, even when there is a single decision maker, several conflicting points of views may be considered in the preference analysis, leading to the definition of several criteria. All these observations have motivated the emergence of multicriteria methodologies for preference modeling and human decision support, an entire stream of research that steadily developed for 40 years [Keeney and Raiffa, 1993].

In human decision problems, alternatives are often characterized by a combination of local decisions providing the feasible set with a combinatorial structure. This explains the growing interest for multiobjective combinatorial optimization [Ehrgott, 1999], with applications in various contexts such as planning actions of autonomous agents, organizing production workflows, solving resource allocation problems. Besides the existence of several criteria, the combinatorial nature of multiattribute spaces is a significant source of complexity. This has motivated the development of preference representation languages aiming at simplifying preference handling and decision making on combinatorial domains.

As far as utility functions are concerned, the works on compact representation aim at exploiting preference independence among some attributes so as to decompose the utility of a tuple into a sum of smaller utility factors. Different decomposition models of utilities have been developed to model preferences. The most widely used assumes a special kind of independence among attributes called "mutual preferential independence". It ensures that preferences are representable by an additively decomposable utility [Krantz *et al.*, 1971; Bacchus and Grove, 1995]. Such decomposability makes both the elicitation process and the query optimizations very fast and simple. However, in practice, it may fail to hold as it rules out any interaction among attributes. Generalizations have thus been proposed in the literature to significantly increase the descriptive power of additive utilities. Among them, *multilinear utilities* [Keeney and Raiffa, 1993] and GAI (generalized additive independence) decompositions [Brazi-

unas and Boutilier, 2005; Gonzales and Perny, 2004] allow quite general interactions between attributes [Bacchus and Grove, 1995] while preserving some decomposability. The latter has been used to endow CP-nets with utilities (UCP-nets) [Boutilier *et al.*, 2001; Brafman *et al.*, 2004]. GAI decomposable utilities can be compiled into graphical structures closely related to junction trees, the so-called GAI networks. They can be exploited to perform classical optimization tasks (e.g. find a tuple with maximal utility) using a simple collect/distribute scheme essentially similar to that used in the Bayes net community. The next step is to address multiobjective optimization problems with such graphical models.

The aim of this paper is to show the potential of GAI models in representing and solving multiobjective combinatorial optimization problems. Assuming each objective is represented by a GAI decomposable utility function defined on the multiattribute space, we investigate the determination of Pareto-optimal elements. More precisely, in Section 2, we introduce exact and approximated optimality concepts linked to the notion of Pareto dominance. In Section 3, we show how both exact and approximated Pareto sets can be determined efficiently using GAI nets. Finally, in Section 4, we present numerical experimentations showing the impact of decomposition and approximation on solution times.

2 Non-dominated Solutions and their Approximation

We assume that alternatives are characterized by n variables $x_1, \ldots, x_n$ taking their values in finite domains $X_1, \ldots, X_n$ respectively. Hence alternatives can be seen as elements of the product set of these domains $\mathcal{X} = X_1 \times \ldots \times X_n$. By abuse of notation, for any set $\mathbf{Y} \subseteq \{1, ..., n\}$, $x_{\mathbf{Y}}$ will refer to the projection of $x \in \mathcal{X}$ on $\prod_{i \in \mathbf{Y}} X_i$. Considering a finite set of objectives $M = \{1, ..., m\}$, any solution $x \in \mathcal{X}$ can be characterized by a utility vector $(u^1(x), ..., u^m(x)) \in \mathbb{Z}_+^m$ where $u^i : \mathcal{X} \to \mathbb{Z}_+$ is the i^{th} utility function. It measures the relative utility of alternatives with respect to the i^{th} point of view (criterion or agent) considered in the problem. Hence, the comparison of alternatives reduces to that of their utility vectors. The set of all utility vectors attached to solutions in $\mathcal{X}$ is denoted by $\mathcal{U}$. We recall now some definitions related to dominance and optimality in multiobjective optimization.

Definition 1 *The weak Pareto dominance relation is defined on utility vectors of $\mathbb{Z}_+^m$ as: $u \succsim_P v \Leftrightarrow [\forall i \in M, u^i \geq v^i]$.*

Definition 2 *Any utility vector $u \in \mathcal{U}$ is said to be non-dominated in $\mathcal{U}$ (or Pareto-optimal) if, for all $v \in \mathcal{U}, v \succsim_P u \Rightarrow u \succsim_P v$. The set of non-dominated vectors in $\mathcal{U}$ is denoted ND($\mathcal{U}$) and is referred to as the "Pareto set".*

The Pareto set can be very large as $\mathcal{U}$ is the image of a combinatorial set $\mathcal{X}$ in $\mathbb{Z}_+^m$ and, in the worst case, all elements in $\mathcal{U}$ are non-dominated as shown by the following example:

Example 1 *Consider a decision problem with two objectives on a set $\mathcal{X} = \prod_{k=1}^n X_k$, where $X_k = \{0, 1\}, k = 1, \ldots, n$. Assume the objectives are additive utility functions defined, for any Boolean vector $x = (x_1, \ldots, x_n) \in \mathcal{X}$, by $u^i(x) = \sum_{k=1}^n u_k^i(x_k)$, $i = 1, 2$, where u_k^i is a marginal utility function defined on X_k by $u_k^1(x_k) = 2^{k-1}x_k$ and $u_k^2(x_k) = 2^{k-1}(1 - x_k)$. Then for all $x \in \{0, 1\}^n$, $u^1(x) = \sum_{k=1}^n 2^{k-1}x_k$ and $u^2(x) = \sum_{k=1}^n 2^{k-1}(1 - x_k)$ and therefore $u^1(x) + u^2(x) = \sum_{k=1}^n 2^{k-1} = 2^n - 1$. So there exist 2^n different Boolean vectors in $\mathcal{X}$ with distinct images in the utility space. Actually, all feasible utility vectors of type $(u_1, u_2) \in \mathcal{U}$ are on the same line characterized by equation $u_1 + u_2 = 2^n - 1$. This line is orthogonal to vector $(1, 1)$ which proves that all these vectors are Pareto-optimal. Here ND($\mathcal{U}$) $= \mathcal{U}$. In such a case, the size of the Pareto set grows exponentially with the number of attributes.*

Although pathological, this example shows that the determination of Pareto-optimal vectors may be intractable in practice on large size instances. Numerical tests presented in Section 4 will confirm this point. For this reason, relaxing the notion of Pareto dominance to approximate the Pareto set with performance guarantee is a good alternative in practice. In this perspective, we consider the notion of ε-dominance defined as follows [Papadimitriou and Yannakakis, 2000; Laumanns *et al.*, 2002; Perny and Spanjaard, 2008]:

Definition 3 *For any $\varepsilon > 0$, the ε-dominance relation is defined on utility vectors of $\mathbb{Z}_+^m$ as follows:*

$$u \succsim_\varepsilon v \Leftrightarrow [\forall i \in M, (1 + \varepsilon)u^i \geq v^i].$$

For instance, on the left part of Fig. 1, the black point in cone C ε-dominates any other point in the cone. Hence we can define the notion of approximation of the Pareto set as follows:

Definition 4 *For any $\varepsilon > 0$ and any set $\mathcal{V} \subseteq \mathcal{U}$ of bounded utility vectors, a subset $\mathcal{W} \subseteq \mathcal{V}$ is said to be an ε-covering of ND($\mathcal{V}$) if $\forall v \in$ ND($\mathcal{V}$), $\exists w \in \mathcal{W} : w \succsim_\varepsilon v$.*

In general, multiple ε-coverings of ND($\mathcal{V}$) exist, with different sizes, the most interesting being minimal w.r.t. set inclusion. For instance, on the left part of Fig. 1, the gray and the four black points form an ε-covering of the Pareto set since the union of cones A,B,C,D,E covers all feasible utility vectors. But the five points do not form a minimal covering since, excluding the gray point, the four remaining points still cover the entire feasible set as can be seen on Fig. 1.

The strength of the ε-covering concept derives from the following result due to [Papadimitriou and Yannakakis, 2000]: for any fixed number $m > 1$, for any finite $\epsilon > 0$ and any set $\mathcal{U}$ of bounded utility vectors such that $0 < u^i(x) \leq K$ for all $i \in M$, there exists in $\mathcal{U}$ an ε-covering of the Pareto set ND($\mathcal{U}$) the size of which is polynomial in $\log K$ and $1/\log(1 + \epsilon)$. The result can be simply explained as follows: to any utility vector $u \in \mathbb{Z}_+^m$, we can assign vector $\varphi(u)$ the components of which are $\varphi(u^i) = \lceil \frac{\log u^i}{\log(1+\varepsilon)} \rceil$. By definition, the following property holds:

Proposition 1 $\forall u, v \in \mathcal{U}, \varphi(u) \succsim_P \varphi(v) \Rightarrow u \succsim_\varepsilon v$.

Thanks to the scaling and rounding operation, the number of different possible values for φ is bounded on each axis by $\lceil \log K / \log(1 + \varepsilon) \rceil$. Hence the cardinality of set $\varphi(\mathcal{U}) = \{\varphi(u), u \in \mathcal{U}\}$ is upper bounded by $B(K, m, \varepsilon) = \lceil \log K / \log(1 + \varepsilon) \rceil^m$. This can easily be explained using the right part of Fig. 1 representing a logarithmic grid in the space of criteria. Any square of the grid represents a different class

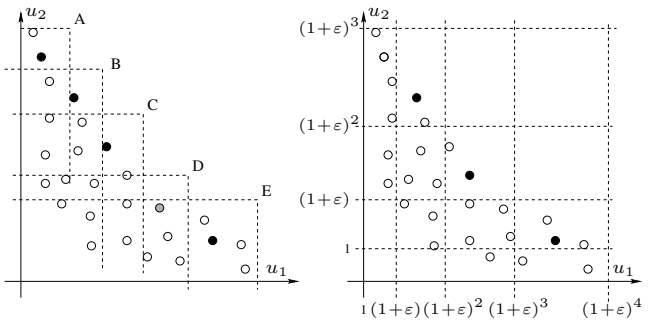


Figure 1: ε-coverings of the Pareto set

of utility vectors having the same image through φ. Thanks to Proposition 1, we can see that any vector belonging to a given square covers any other element of the square in terms of $\succsim_\varepsilon$. Hence, choosing one representative in each square, we cover the entire set $\mathcal{U}$. The size of the covering is bounded by the number of squares in the grid, i.e., $B(K, m, \varepsilon)$. The covering can easily be refined by keeping only the elements of $\text{ND}(\varphi(\mathcal{U}))$ in the covering (plotted in black on Fig. 1). Since we have no more than one Pareto-optimal element for each different value of $\varphi(u)$ such a covering will include at most $\lceil \log K / \log(1+\varepsilon) \rceil^{m-1}$ elements. In Example 1, if items are described by 20 Boolean attributes, then the Pareto set contains more than one million of elements (2^{20}) whereas $\lceil \log 2^{20} / \log 1.1 \rceil = 146$ elements are sufficient to cover this set with a tolerance threshold of 10% ($\varepsilon = 0.1$).

It is now natural to wonder how GAI decomposability of utility functions can be used to efficiently compute i) the exact set of Pareto-optimal utility vectors (when it is not too large) and ii) an ε-covering of the Pareto set when it is too large for complete enumeration. In both cases, we would like to be able to recover a feasible solution for each returned vector. These questions are discussed in the following sections.

3 Pareto Set Computation using GAI nets

GAI decompositions are extensions of additive and multilinear decompositions [Bacchus and Grove, 1995]. They encode utilities as a sum of subutilities with overlapping factors:

Definition 5 (GAI decomposition) *Let $\mathcal{X} = \prod_{i=1}^n X_i$. Let $\mathbf{Z}_1, \ldots, \mathbf{Z}_r \subseteq \mathbf{N} = \{1, \ldots, n\}$ be such that $\mathbf{N} = \cup_{i=1}^r \mathbf{Z}_i$. For every i, let $X_{\mathbf{Z}_i} = \prod_{j \in \mathbf{Z}_i} X_j$. Utility $u(\cdot)$ is GAI-decomposable w.r.t. the $X_{\mathbf{Z}_i}$'s iff there exist functions $u_i : X_{\mathbf{Z}_i} \mapsto \mathbb{R}$ such that $u(x) = \sum_{i=1}^r u_i(x_{\mathbf{Z}_i})$, for all $x \in \mathcal{X}$, where $x_{\mathbf{Z}_i}$ denotes the tuple constituted by the x_j's, $j \in \mathbf{Z}_i$.*

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 6 (GAI network) *Let $\mathcal{X} = \prod_{i=1}^n X_i$. Let $\mathbf{Z}_1, \ldots, \mathbf{Z}_r$ be some subsets of $\mathbf{N} = \{1, \ldots, n\}$ such that $\bigcup_{i=1}^r \mathbf{Z}_i = \mathbf{N}$. Let $u(x) = \sum_{i=1}^r u_i(x_{\mathbf{Z}_i})$ for all $x \in \mathcal{X}$. Then a GAI network representing $u(\cdot)$ is an undirected graph $G = (V, E)$, satisfying the following two properties:*

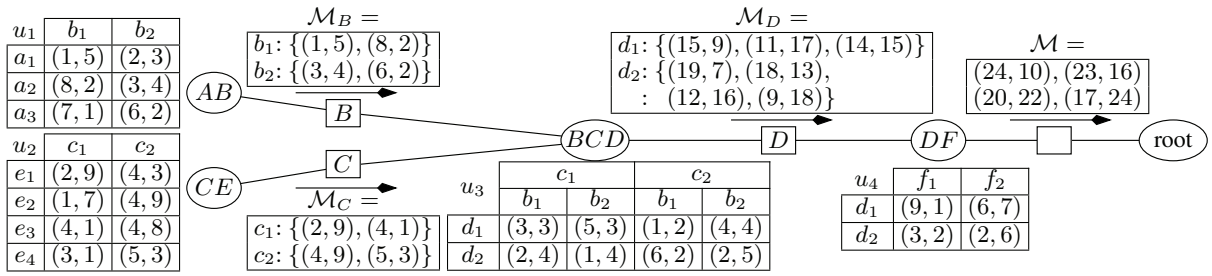
1. *$V = \{X_{\mathbf{Z}_1}, \ldots, X_{\mathbf{Z}_r}\}$;*
2. *For every $(X_{\mathbf{Z}_i}, X_{\mathbf{Z}_j}) \in E$, $\mathbf{Z}_i \cap \mathbf{Z}_j \neq \emptyset$. For every pair of nodes $X_{\mathbf{Z}_i}, X_{\mathbf{Z}_j}$ such that $\mathbf{Z}_i \cap \mathbf{Z}_j = \mathbf{T}_{ij} \neq \emptyset$, there exists a path in G linking $X_{\mathbf{Z}_i}$ and $X_{\mathbf{Z}_j}$ such that all of its nodes contain all the indices of $\mathbf{T}_{ij}$ (Running intersection property).*

Nodes of V are called cliques. *Every edge $(X_{\mathbf{Z}_i}, X_{\mathbf{Z}_j}) \in E$ is labeled by $X_{\mathbf{T}_{ij}} = X_{\mathbf{Z}_i \cap \mathbf{Z}_j}$ and is called a* separator.

Cliques are drawn as ellipses and separators as rectangles. Here, we shall only consider GAI trees. This is not restrictive as general GAI nets can always be compiled into GAI trees [Gonzales and Perny, 2004]. For any GAI decomposition, by Definition 6, the cliques of the GAI net should be the sets of variables of the subutilities. For instance, the GAI net of Fig. 2 represents the GAI decomposition: $u(a, b, c, d, e, f) = u_1(a, b) + u_2(c, e) + u_3(b, c, d) + u_4(d, f)$.

3.1 Exact Pareto Set Computation

To understand how GAI nets can be exploited to compute Pareto sets, assume that two agents have preferences over a set $\mathcal{X} = A \times B \times C \times D \times E \times F$. Both agents have utilities decomposable according to the GAI net of Fig. 2, i.e., $u^i(a, b, c, d, e, f) = u_1^i(a, b) + u_2^i(c, e) + u_3^i(b, c, d) + u_4^i(d, f)$, $i = 1, 2$, the tables of Fig. 2 showing pairs (u_j^1, u_j^2). Denote by $\mathcal{U}$ the set of all possible values of u. The key property of our algorithm is that, for any additive utility vector $v^i(x, y) = v_1^i(x) + v_2^i(y)$, $i = 1, 2$, the Pareto set w.r.t. v can only be constituted by non-dominated vectors $v_1(x)$'s and $v_2(y)$'s. Indeed, if $v_1(x) \succ_P v_1(x')$, then, for any y, $v_1(x) + v_2(y) \succ_P v_1(x') + v_2(y)$, hence $v_1(x')$ cannot be part of the Pareto set w.r.t. v. Now, let us come back to the GAI net of Fig. 2. For a fixed value, say b_1, of separator B, $u^i(a, b, c, d, e, f)$ becomes an additive utility: $u_1^i(a, b_1) + v^i(b_1, c, d, e, f)$. Consequently, for $B = b_1$, vectors in $ND(\mathcal{U})$ can only be constituted with non-dominated $u_1(a, b_1)$'s and, for instance, it cannot contain $u_1(a_3, b_1) = (7, 1)$ as $u_1(a_2, b_1) = (8, 2) \succ_P u_1(a_3, b_1)$. Hence, if clique AB sends to BCD message $\mathcal{M}_B$ containing, for each value b_j of B, non-dominated vectors $u_1(a, b_j)$, then only these vectors need be considered for determining $ND(\mathcal{U})$. For the same reason, only non-dominated vectors $\mathcal{M}_C$ of u_2 need be considered. Now, for each value d_j of D, $u(a, b, c, d_j, e, f)$ can be decomposed additively as $w(a, b, c, d_j, e) + u_4(d_j, f)$. Hence only non-dominated vectors $w(a, b, c, d_j, e)$ need be considered for determining $ND(\mathcal{U})$. But, as shown previously, $w(a, b, c, d_j, e)$ must be equal to the sum of one vector of $\mathcal{M}_B$, one vector of $\mathcal{M}_C$ and one compatible vector of u_3. By compatible, we mean that if the vector chosen in $\mathcal{M}_B$ (resp. $\mathcal{M}_C$) corresponds to $B = b_i$ (resp. $C = c_k$), then only vector $u_3(b_i, c_k, d_j)$ is allowed. For determining $ND(\mathcal{U})$, only non-dominated such sums need be considered. These are precisely stored in message $\mathcal{M}_D$ sent from clique BCD to clique DF. Finally, $ND(\mathcal{U})$ can be exactly determined by computing all the sums of one vector of $\mathcal{M}_D$ with a compatible vector in u_4 and keeping only those non-dominated. This corresponds to message $\mathcal{M}$. Note the efficiency of the algorithm: in our example $|\mathcal{X}| = 192$, hence a naive approach would make dominance tests between 192 different vectors. Here, for each b_i (resp. c_j, d_k), message $\mathcal{M}_B$ (resp. $\mathcal{M}_C$, $\mathcal{M}_D$) requires dominance tests between 3 (resp. 4, 16) vectors, and $\mathcal{M}$ needs dominance tests between 14 vectors.



u_1	b_1	b_2
a_1	(1, 5)	(2, 3)
a_2	(8, 2)	(3, 4)
a_3	(7, 1)	(6, 2)

u_2	c_1	c_2
e_1	(2, 9)	(4, 3)
e_2	(1, 7)	(4, 9)
e_3	(4, 1)	(4, 8)
e_4	(3, 1)	(5, 3)

u_3	c_1		c_2	
	b_1	b_2	b_1	b_2
d_1	(3, 3)	(5, 3)	(1, 2)	(4, 4)
d_2	(2, 4)	(1, 4)	(6, 2)	(2, 5)

u_4	f_1	f_2
d_1	(9, 1)	(6, 7)
d_2	(3, 2)	(2, 6)

Figure 2: Computing the Pareto set using a GAI network

This process can be automated as follows: first, we need a function that, given two sets of vectors, computes the non dominated vectors of their sums. `GetLabelsNonDom` defined below performs this operation. Actually, this function takes in argument some sets of labels $\mathcal{V}$ and $\mathcal{W}$ rather than sets of vectors. A label is simply a pair $\langle v, P\rangle$ such that v is a utility vector and P is some information enabling to get the instantiation of the attributes that led to v's utility value. In this function, operator $\oplus$ is defined as $\mathcal{V} \oplus \mathcal{W} \mapsto \mathcal{L} = \{\langle v + v', P''\rangle : \langle v, P\rangle \in \mathcal{V}$ and $\langle v', P'\rangle \in \mathcal{W}\}$, i.e., operator $\oplus$ combines all vectors in $\mathcal{V}$ with all vectors in $\mathcal{W}$. The rest of the function removes iteratively all dominated vectors and $\mathcal{L}_{\text{out}}$ contains the labels of all the non-dominated vectors.

Function `GetLabelsNonDom`(label sets $\mathcal{V}, \mathcal{W}$)
01 $\mathcal{L} \leftarrow \mathcal{V} \oplus \mathcal{W}$; $\mathcal{L}_{\text{out}} \leftarrow \emptyset$
02 **for all** labels $\langle v, P\rangle \in \mathcal{L}$ **do**
03 **if** $\nexists \langle v', P'\rangle \in \mathcal{L}_{\text{out}}$ s.t. $v' \gtrsim_P v$ **then**
04 remove from $\mathcal{L}_{\text{out}}$ all labels $\langle v', P'\rangle$ s.t. $v \gtrsim_P v'$
$\mathcal{L}_{\text{out}} \leftarrow \mathcal{L}_{\text{out}} \cup \{\langle v, P\rangle\}$
05 **done**
06 **return** $\mathcal{L}_{\text{out}}$

Given this function, it is easy to compute the $\mathcal{M}$'s messages mentioned previously. Consider for instance the generation of message $\mathcal{M}_a$ of Fig. 3. Let $X_{\mathbf{C}_{p_1}}, \ldots, X_{\mathbf{C}_{p_r}}$ be the cliques adjacent to $X_{\mathbf{C}_a}$ that already sent their messages $\mathcal{M}_{P_i}$'s to $X_{\mathbf{C}_a}$. Let $X_{\mathbf{S}_{ab}}$ be the separator between cliques $X_{\mathbf{C}_a}$ and $X_{\mathbf{C}_b}$ and let $X_{\mathbf{D}_a}$ be the attributes in clique $X_{\mathbf{C}_a}$ that do not belong to this separator. For any fixed value $x_{\mathbf{S}_{ab}}$ of $X_{\mathbf{S}_{ab}}$, let $\mathcal{M}_a[x_{\mathbf{S}_{ab}}]$ denote the non-dominated vectors given $x_{\mathbf{S}_{ab}}$ to be sent to clique $X_{\mathbf{C}_b}$. Then, as mentioned at the beginning of this section, $\mathcal{M}_a[x_{\mathbf{S}_{ab}}]$ is the set of the non-dominated vectors among $\cup_{x_{\mathbf{D}_a} \in X_{\mathbf{D}_a}} \{u_a(x_{\mathbf{D}_a}, x_{\mathbf{S}_{ab}}) + \sum_{j=1}^{r} v^{i_j} : v^{i_j} \in \mathcal{M}_{P_{i_j}}$ are compatible with $(x_{\mathbf{D}_a}, x_{\mathbf{S}_{ab}})\}$. To compute this, we exploit the transitivity of Pareto dominance and first add to a given $u_a(x_{\mathbf{D}_a}, x_{\mathbf{S}_{ab}})$ all its compatible v^{i_1}'s vectors. In function `GetLabelsMessage`, this is done on Line 07. Then combine the resulting vectors with all the compatible v^{i_2}'s and extract the non-dominated vectors.

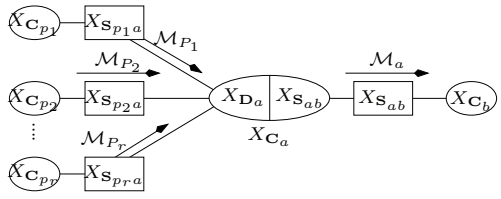


Figure 3: Illustration of function `GetLabelsMessage`

This is done on Line 09 by a call to `GetLabelsNonDom`. Iterate over all the $X_{\mathbf{C}_{p_i}}$'s neighbors (loop 08–10). As a result, we get all the non-dominated vectors compatible with a given value $x_{\mathbf{D}_a}$. There remains to make the unions of these sets of vectors for different instantiations (loop 06–12) and to extract from these unions the non-dominated vectors (Line 11). `GetLabelsMessage` thus computes correctly $\mathcal{M}_a$.

Function `GetLabelsMessage`($X_{\mathbf{C}_a}, X_{\mathbf{C}_b}$)
01 let $X_{\mathbf{C}_{p_1}}, ..., X_{\mathbf{C}_{p_r}}$ be the cliques adjacent to $X_{\mathbf{C}_a}$ except $X_{\mathbf{C}_b}$
02 let $\mathcal{M}_{p_i}$, $i = 1, ..., r$, be the vectors sets sent by $X_{\mathbf{C}_{p_i}}$ to $X_{\mathbf{C}_a}$
03 let $\mathbf{S}_{ab} = \mathbf{C}_a \cap \mathbf{C}_b$, $\mathbf{D}_a = \mathbf{C}_a \backslash \mathbf{S}_{ab}$ and $\mathbf{S}_{p_i a} = \mathbf{C}_{p_i} \cap \mathbf{C}_a$
04 **for all** $x_{\mathbf{S}_{ab}} \in X_{\mathbf{S}_{ab}}$ **do**
05 $\mathcal{M}_a[x_{\mathbf{S}_{ab}}] \leftarrow \emptyset$
06 **for all** $x_{\mathbf{D}_a} \in X_{\mathbf{D}_a}$ **do**
07 $\mathcal{V} \leftarrow \{\langle u_a(x_{\mathbf{C}_a}), x_{\mathbf{C}_a}\rangle\} \oplus \mathcal{M}_{p_1}[x_{\mathbf{S}_{p_1 a}}]$
08 **for all** $i \in \{2, \ldots, r\}$ **do**
09 $\mathcal{V} \leftarrow$ `GetLabelsNonDom`$(\mathcal{V}, \mathcal{M}_{p_i}[x_{\mathbf{S}_{p_i a}}])$
10 **done**
11 $\mathcal{M}_a[x_{\mathbf{S}_{ab}}] \leftarrow$ `GetLabelsNonDom`$(\mathcal{V}, \mathcal{M}_a[x_{\mathbf{S}_{ab}}])$
12 **done**
13 **done**
14 **return** $\mathcal{M}_a = \{\mathcal{M}_a[x_{\mathbf{S}_{ab}}] : x_{\mathbf{S}_{ab}} \in X_{\mathbf{S}_{ab}}\}$

Now, to complete the computation of the Pareto set, there remains to organize the calls to `GetLabelsMessage`. This can be done by a classical "collect" algorithm as described below. Create a dummy attribute A of domain size 1 as well as a dummy clique `root` containing only A. Select an arbitrary clique $X_{\mathbf{C}_a}$, add A to it, and add an edge between $X_{\mathbf{C}_a}$ and `root`. Then calling `Collect`($X_{\mathbf{C}_a}$,`root`) clearly results in message $\mathcal{M}_a$ sent by $X_{\mathbf{C}_a}$ to `root` being the Pareto set. Now, for any $v \in \mathcal{M}_a$, it is possible to determine which instantiations of the attributes have utility v because the collect phase returned labels which, by definition, contain enough informations to get back the instantiations.

Function `Collect`(clique $X_{\mathbf{C}_a}$ by clique $X_{\mathbf{C}_b}$)
01 **for all** cliques $X_{\mathbf{C}_{p_i}}$ adjacent to $X_{\mathbf{C}_a}$ except $X_{\mathbf{C}_b}$ **do**
02 call `Collect` ($X_{\mathbf{C}_{p_i}}, X_{\mathbf{C}_a}$)
03 $\mathcal{M}_a \leftarrow$ `GetLabelsMessage`$(X_{\mathbf{C}_a}, X_{\mathbf{C}_b})$
04 send to $X_{\mathbf{C}_b}$ label set $\mathcal{M}_a$
05 **done**

For a fixed number of objectives m, function `Collect` is *pseudopolynomial*, i.e., it returns the Pareto set in time and space bounded by a polynomial in K, the largest of the u^i's over $\mathcal{X}$, and $L|X_{\mathbf{C}}|$, the size of the instance, where L is the number of cliques in the GAI net and $|X_{\mathbf{C}}|$ is the size of the biggest clique (i.e., the product of the domain sizes of its attributes). Actually, label sets $\mathcal{V}$ and $\mathcal{W}$ passed to `GetLabelsNonDom` have less than K^m elements as each

utility value is an integer between 1 and K. Hence, on line 01, $\mathcal{L}$ is computed in $O(mK^{2m})$. But $|\mathcal{L}| \leq K^m$ as $\mathcal{L}$ contains sums of compatible subutilities. So, the for loop of lines 02–05 is executed at most K^m times. As $\mathcal{L}_{\text{out}} \subseteq \mathcal{L}$, lines 03 and 04 can be executed in $O(mK^m)$. Hence `GetLabelsNonDom` is in $O(mK^{2m})$. In function `GetLabelsMessage`, the for loops of lines 04 and 06 parse all the elements of $X_{\mathbf{C}_a}$. Within these loops, Line 07 is performed in $O(mK^m)$ and lines 09 and 11 in $O(mK^{2m})$. Hence the whole function is in $O(|X_{\mathbf{C}_a}|rmK^{2m})$, hence within $O(|X_{\mathbf{C}}|rmK^{2m})$. Finally, function `Collect` calls `GetLabelsMessage` over all the cliques, hence all edges should be parsed and thus `Collect` is in $O(|X_{\mathbf{C}}|LmK^{2m})$. Note that $|X_C|$ is exponential in w where w is the treewidth of the GAI net (i.e. the number of attributes in its largest clique). The algorithm is hence exponential in w but remains pseudo-polynomial for bounded w. In practice, w is small as subutility factors usually involve only a few attributes.

3.2 A FPTAS for the Pareto Set

Computing an ε-covering is in essence similar to computing a Pareto set. However, care must be taken when using ε-dominance. Let $X_{\mathbf{C}_a}$ and $X_{\mathbf{C}_b}$ be two adjacent cliques. Let v, v', v'' be three vectors on clique $X_{\mathbf{C}_a}$ such that $v \succsim_\epsilon v'$ and v, v'' are non-dominated on clique $X_{\mathbf{C}_a}$. Let w, w'' be two vectors on clique $X_{\mathbf{C}_b}$ such that $v''+w'' \succsim_\epsilon v+w$. Since $v+w$ is dominated, it should be deleted. However, to preserve a covering of the Pareto set, we need to be sure that $v''+w'' \succsim_\epsilon v'+w$. Unfortunately, although $v+w \succsim_\epsilon v'+w$ by additivity, we only have $(1+\epsilon)^2(v''+w'') \succsim_P v'+w$ instead of $(1+\varepsilon)(v''+w'') \succsim_P v'+w$. For this reason, we should have used an $\varepsilon/2$-dominance to ensure that the message sent by the second clique is actually an ε-covering. More generally, we need a finer dominance notion:

Definition 7 *For any utility vectors $u, v \in \mathbb{Z}_+^m$, and any $w > 0$, $u \succsim_\epsilon^w v \Leftrightarrow (1+\varepsilon)^w u \succsim_P v$. Then u is said to (ε, w)-dominate v, and a set of (ε, w)-non-dominated vectors is called an (ε, w)-covering.*

Lemma 1 *Let x, x_i, y, y_i, z be some positive vectors and w, w_i be positive numbers. Then the following properties hold:*
P1: $x \succsim_\epsilon^w y \Longrightarrow x+z \succsim_\epsilon^w y+z$.
P2: $[x \succsim_\epsilon^w y$ and $y \succsim_\epsilon^{w'} z] \Longrightarrow x \succsim_\epsilon^{w+w'} z$.

Hence as the collect over k cliques requires chaining k (ε, w_i)-dominance tests, $i = 1, ..., k$, a sufficient condition to get an ε-covering is to choose the w_i's summing to 1. Weights w_i's can be chosen equal to $1/L$, where L is the number of cliques or, better, they can be proportional to the sizes of the cliques. To implement this idea (with all the w_i's equal to $1/L$), we substitute function `GetLabelsNonDom` by the one below which computes $(\epsilon, 1/L)$-covering labels.

Function `GetLabelsNonDom`(label sets $\mathcal{V}, \mathcal{W}$)
01 $\mathcal{L} \leftarrow \mathcal{V} \oplus \mathcal{W}$; $\mathcal{L}_{\text{grid}} \leftarrow \emptyset$; $\mathcal{L}_{\text{out}} \leftarrow \emptyset$
02 **for all** labels $\langle v, P\rangle \in \mathcal{L}$ **do**
04 **if** $\varphi(v) \notin \mathcal{L}_{\text{grid}}$ and $\nexists \langle v', P'\rangle \in \mathcal{L}_{\text{out}}$ s.t. $v' \succsim_\epsilon^{1/L} v$ **then**
05 remove from $\mathcal{L}_{\text{out}}$ all labels $\langle v', P'\rangle$ s.t. $v \succsim_P v'$
06 $\mathcal{L}_{\text{out}} \leftarrow \mathcal{L}_{\text{out}} \cup \{\langle v, P\rangle\}$; $\mathcal{L}_{\text{grid}} \leftarrow \mathcal{L}_{\text{grid}} \cup \{\varphi(v)\}$
07 **done**
08 **return** $\mathcal{L}_{\text{out}}$

Proposition 2 *Let $X_{\mathbf{C}_a}$ be any clique of the GAI net. Add to $X_{\mathbf{C}_a}$ a dummy attribute A with a domain size of 1. Add to the GAI net a dummy clique* `root` *containing only A and an edge between $X_{\mathbf{C}_a}$ and* `root`*. Call* `Collect`*($X_{\mathbf{C}_a}$,*`root`*). Then the message sent by $X_{\mathbf{C}_a}$ to* `root` *is an ε-covering.*

Provided $1 \leq K \leq 2^{p(L|X_{\mathbf{C}}|)}$, where p denotes some polynomial, function `Collect` using the newly defined `GetLabelsNonDom` is a *Fully Polynomial Time Approximation Scheme* (FPTAS), i.e., it returns the ε-covering in time and space bounded by a polynomial in $1/\varepsilon$ and the instance size $L|X_{\mathbf{C}}|$. Indeed, vector sets $\mathcal{V}$ and $\mathcal{W}$ passed to `GetLabelsNonDom` are some (ε, w)-coverings, so they contain at most $\lceil L \log K / \log(1+\varepsilon) \rceil^m$ elements. Here, we only consider small ε's, hence $1/\log(1+\varepsilon) \leq 2/\varepsilon$. So, $\mathcal{V}$ and $\mathcal{W}$ contain less than $(2L \log K/\varepsilon)^m$ elements and $\mathcal{L}$ is computed in $O(m(L \log K/\varepsilon)^{2m})$. By definition, $\mathcal{L}_{\text{grid}}$ cannot contain more than $\lceil L \log K / \log(1+\varepsilon) \rceil^m$ elements and checking whether $\varphi(v) \in \mathcal{L}_{\text{grid}}$ is in $O(m)$. We thus check whether $v' \succsim_\epsilon^{1/L} v$ at most $(2L \log K/\varepsilon)^m$ times. For the same reason, $\mathcal{L}_{\text{out}}$ cannot have more than $(2L \log K/\varepsilon)^m$ elements, hence line 05 can be completed in $O((L \log K/\varepsilon)^m)$. Hence `GetLabelsNonDom` is within $O(m(L \log K/\varepsilon)^{2m})$. Therefore, the overall complexity of `Collect` is in $O(L|X_{\mathbf{C}}|m(L \log K/\varepsilon)^{2m})$.

4 Experimentations

In order to evaluate the performance of our algorithm, we performed numerical tests on a Core 2 Duo 2.66 GHz with 4 Gb of RAM. The first part of the experiments aims at illustrating the potential of GAI-decompositions to reduce computation times for the exact Pareto set. To this end, we have considered a search space with 17 attributes of size 5 and a GAI-net having a chain structure with 16 cliques $X_{\mathbf{C}_i} = \{X_i, X_{i+1}\}, i = 1, ..., 16$, so as to have $X_{\mathbf{C}_i} \cap X_{\mathbf{C}_{i+1}} = \{X_{i+1}\}$ as separators. Then we drew utility tables randomly and computed the exact Pareto set using this GAI-net. Next, we merged cliques by pairs to form 8 cliques $X_{\mathbf{C}'_i} = X_{\mathbf{C}_{2i-1}} \cup X_{\mathbf{C}_{2i}}, i = 1, ..., 8$, so as to represent the same utility in a less decomposed form; we computed the exact Pareto set again and recorded the computation time. Then we iterated the process and merged again cliques by pairs. We repeated the process until a single clique of size 17 was obtained. The next table provides computation times (in seconds) as the size of cliques varies from 2 to 17 (a timeout was set to 1 hour); the times represent averages over 20 different instances with different utility tables, for problems involving 2 and 5 objectives respectively. The results show that exploiting utility decompositions makes the multiobjective optimization significantly faster.

	Treewidth (number of attributes per clique)				
	2	3	5	9	17
2 objs	0.055	0.141	1.486	895.556	> 3600
5 objs	31.316	125.726	2467.78	> 3600	> 3600

The second set of experimentations consists of evaluating the potential of a joint use of utility decomposition and ϵ-dominance to reduce computation times. To this end we have generated the pathological bi-objective instances of Example 1 with different sizes (from 10 to 20 attributes). For each instance, the covering of the Pareto set is computed with the

FPTAS for different values of the accuracy level ε varying from 0 to 10%. Note that when $\varepsilon = 0$ we get the exact Pareto set. The table below provides the cardinality of the output set (# sol) and computation times (in seconds). We can observe significant reduction of the output set as ε increases. Computation times are almost negligible.

	Number of attributes					
	10		15		20	
ϵ	# sol	time	# sol	time	# sol	time
0	1024	0.008	32768	0.257	10^6	8.9
0.01	407	0.008	644	0.014	782	0.206
0.05	117	0.002	149	0.004	193	0.010
0.1	58	0.002	72	0.002	109	0.005

The next table provides results on randomly generated bi-objective problems of different sizes (from 15 to 30 attributes). We give the cardinality of the output set (# sol) as well as the average computation times (in seconds) over 100 experimentations. Instances are generated as follows: for a given number of attributes, we generate a complete subgraph in the Markov graph containing the node representing the kth variable, and 2 other variables chosen randomly. Each GAI-network is then obtained by triangulation of the generated Markov graph, and each utility table is filled with values between 1 and 100. We get the following results:

	Number of attributes							
	15		20		25		30	
ϵ	# sol	time	# sol	time	# sol	time	# sol	time
0	94	0.21	390	5.05	537	53.87	1595	103.02
0.01	37	0.029	102	0.064	123	0.30	147	0.64
0.05	9	0.012	14	0.017	17	0.060	18	0.13
0.1	3	0.006	3	0.009	4	0.027	4	0.059

Finally, we have investigated how the number of objectives impacts on solution times. The table below provides the running times of the FPTAS for $\varepsilon = 10\%$, on problems with between 5 and 15 attributes, and from 2 to 10 objectives.

	Number of attributes		
# obj	5	10	15
2	0.001	0.002	0.003
5	0.003	1.07	32.9
10	0.084	189	589

5 Conclusion

We have shown that GAI models can be used efficiently to generate Pareto sets using utility decompositions. The techniques introduced here bear some similarity with those involved in multiobjective state-space search [Stewart and White III, 1991] or in multiobjective CSPs [Rollon and Larrosa, 2007], with some additional specificities due to the use of junction trees and the aim of generating approximations. Our FPTAS reveals very fast, even on pathological instances, provided the number of objectives (not to be confused with the number of attributes) and the sizes of cliques are not too large (this is usually the case in reality). Besides utility decompositions, our tests confirm the practical efficiency of ε-dominance on multiobjective problems, as in [Laumanns *et al.*, 2002; Bazgan *et al.*, 2007; Perny and Spanjaard, 2008]. Knowing that the size of the Pareto set can be huge in real-world multiobjective combinatorial problems, utility decompositions and ε-coverings appear as key concepts to enhance efficiency of multiobjective optimization procedures while keeping performance guarantees. The approach proposed here also enables to go beyond Pareto optimality, even for non-decomposable preference models, provided they refine Pareto dominance. It can indeed be used to devise preference-based search algorithms where Pareto dominance is used to prune dominated solutions or subsolutions during the search.

References

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

[Bazgan *et al.*, 2007] C Bazgan, H Hugot, and D Vanderpooten. A practical efficient FPTAS for the 0-1 multiobjective knapsack problem. In *ESA*, 2007.

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

[Brafman *et al.*, 2004] R Brafman, C Domshlak, and T Kogan. On generalized additive value-function decomposition. In *UAI*, 2004.

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

[Cowell *et al.*, 1999] R Cowell, A Dawid, S Lauritzen, and D Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer, 1999.

[Ehrgott, 1999] M Ehrgott. *Multicriteria Optimization*. Springer, 1999.

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

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

[Keeney and Raiffa, 1993] R L Keeney and H Raiffa. *Decisions with Multiple Objectives - Preferences and Value Tradeoffs*. Cambridge University Press, 1993.

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

[Laumanns *et al.*, 2002] M Laumanns, L Thiele, K Deb, and E Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[Papadimitriou and Yannakakis, 2000] C H Papadimitriou and Y Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *FOCS*, 2000.

[Perny and Spanjaard, 2008] P Perny and O Spanjaard. Near admissible algorithms for multiobjective search. In *ECAI*, 2008.

[Rollon and Larrosa, 2007] E Rollon and J Larrosa. Multiobjective russian doll search. In *AAAI*, 2007.

[Stewart and White III, 1991] B Stewart and C White III. Multiobjective A*. *J. of ACM*, 38(4):775–814, 1991.