Artificial Intelligence 175 (2011) 1153–1179

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Decision making with multiple objectives using GAI networks

C. Gonzales, P. Perny *, J.Ph. Dubus

LIP6 – Université Pierre et Marie Curie, case 169, 4 place jussieu, 75005 Paris, France

ARTICLE INFO

Article history:
Received 28 February 2009
Received in revised form 20 July 2010
Accepted 15 September 2010
Available online 2 December 2010

Keywords:
Graphical models
GAI decomposable utility
Preference representation
Multiobjective optimization
Multiagent decision making
Compromise search
Fairness

ABSTRACT

This paper deals with preference representation on combinatorial domains and preference-based recommendation in the context of multicriteria or multiagent decision making. The alternatives of the decision problem 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. Thanks to decomposability, utility vectors attached to solutions can be compiled into a graphical structure closely related to junction trees, the so-called GAI network. Using this structure, we present preference-based search algorithms for multicriteria or multiagent decision making. Although such models are often non-decomposable over attributes, we actually show that GAI networks are still useful to determine the most preferred alternatives provided preferences are compatible with Pareto dominance. We first present two algorithms for the determination of Pareto-optimal elements. Then the second of these algorithms is adapted so as to directly focus on the preferred solutions. We also provide results of numerical tests showing the practical efficiency of our procedures in various contexts such as compromise search and fair optimization in multicriteria or multiagent problems.

1. Introduction

The complexity of decision problems in organizations, the importance of the issues raised 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. For 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 preferences of each individual being encoded by a utility function defined on the multiattribute space measuring the relative attractiveness of each tuple. Hence the objectives of individuals take the form of multiattribute utility functions to be maximized. Typically, in a multiagent decision problem, we have to deal with several such utility functions that must be optimized simultaneously. Since individual utilities are generally not commensurate, constructing an overall utility function gathering all relevant aspects is not always possible. Hence the problem does not reduce to a classical single-objective optimization task; we have to solve a multiobjective problem.

Moreover, even when there is a single decision maker, several points of views may be considered in the preference analysis, leading to the definition of several criteria. Rationality in decision making is generally not only a matter of costs reduction. In practice, other significant aspects that are not reducible to costs must be included in the analysis; the outcomes

* Corresponding author.
E-mail addresses: kaveh@river-valley.com (C. Gonzales), patrice.perny@lip6.fr (P. Perny), cvr@river-valley.com (J.Ph. Dubus).

0004-3702/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2010.11.020

of alternatives must be thought in a multidimensional space. This is the case in the elaboration of public policies where different aspects such as ecology and environment, education, health, security, public acceptability are considered in the evaluation process. This is also the case for individual decision of consumers. For example, when choosing a new car for a family, an individual will look at the cost, but will also consider several multiattribute utility functions concerning security in the car (brake system, airbags,...), velocity (speed, acceleration,...), space (boot size,...), environmental aspects (pollution) and aesthetics (color, shape, brand,...). All these observations have motivated the emergence of multicriteria methodologies for preference modeling and human decision support [1–4], an entire stream of research that steadily developed for forty years.

As for human decision making, automated decision making in complex environment requires optimization procedures involving multiple objectives. This is the case when computers are used for planning actions of autonomous agents or for organizing the workflow in production chains. Various other examples can be mentioned such as web search [5], *e*-commerce and resource allocation problems. In many of them, however, a decision is actually characterized by a combination of local decisions, thus providing the set of alternatives with a combinatorial structure. This explains the growing interest for multiobjective combinatorial optimization. Besides the explicit introduction of several possibly conflicting objectives in the evaluation process, the necessity of exploring large size solution spaces is an additional source of complexity. This has motivated the development in the AI community 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 [6,7]. Such decomposability makes both the elicitation process and the query optimizations very fast and simple. However, in practice, preferential independence 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* [2] and GAI (generalized additive independence) decompositions [8,9] allow quite general interactions between attributes [7] while preserving some decomposability. The latter has been used to endow CP nets with utilities (UCP nets) both under uncertainty [10] and under certainty [11]. 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 or to variable elimination algorithms in CSP [12–15]. In order to extend the use of GAI nets to multiobjective optimization tasks, we investigate the potential of GAI models for representing and solving multiobjective optimization problems.

As soon as multiple criteria or utility functions are considered in the evaluation of a solution, the notion of optimality is not straightforward. Among the various optimality criteria, the concept of Pareto optimality or efficiency is the most widely used. A solution is said to be Pareto-optimal or efficient if it cannot be improved on one criterion without being depreciated on another one. Pareto optimality is natural because it does not require any information about the relative importance of criteria and can be used as a preliminary filter to circumscribe the set of reasonable solutions in multiobjective problems. However, in combinatorial optimization problems, the complete enumeration of Pareto-optimal solutions is often infeasible in practice [16–18]. For this reason, in many real applications, people facing such complexity resort to artificial simplifications of the problem, either by focusing on the most important criterion (as in route planning assistants), or by performing a prior linear aggregation of the criteria to get a single objective version of the problem, or by generating samples of good solutions using heuristics, which in any case does not provide formal guarantees on the quality of the solutions.

In this paper, we assume that each objective is represented by a GAI decomposable utility function defined on the multiattribute space describing items. In Section 2, after recalling basic definitions related to GAI nets, we show how they make it possible to represent vector-valued utility functions in a compact form, thus facilitating preference handling in multiobjective decision-making problems. In Section 3, we present two exact algorithms exploiting the structure of the GAI net for the determination of Pareto-optimal elements. In Section 4 we propose a refinement of the second algorithm aiming at focusing the search on specific compromise solutions within the Pareto set. We provide exact algorithms for preference-based search with various preference models. The potential of this approach is illustrated in the context of fair multiagent optimization or in the context of compromise search in multicriteria optimization. Finally, in Section 5, we present numerical tests showing the practical feasibility of the proposed approach on various instances of multiobjective combinatorial problems.

2. Multidimensional GAI nets

We assume that alternatives are characterized by n attributes $x_1, \dots, x_n$ taking their values in finite domains $X_1, \dots, X_n$ respectively. Hence alternatives can be seen as elements of the product set of these domains $\mathcal{X} = X_1 \times \cdots \times X_n$. In the sequel, $\mathbf{N} = \{1, \dots, n\}$ will denote the set of all the attributes' indices. By abuse of notation, for any set $\mathbf{Y} \subseteq \mathbf{N}$, $X_\mathbf{Y}$ will refer to the Cartesian product of the X_i, $i \in \mathbf{Y}$, i.e., $X_\mathbf{Y} = \prod_{i \in \mathbf{Y}} X_i$, and $x_\mathbf{Y}$ will refer to the projection of $x \in \mathcal{X}$ on $X_\mathbf{Y}$, that is, the tuple formed by the x_i, $i \in \mathbf{Y}$. We also consider a binary relation $\succsim$ over $\mathcal{X}$ (actually this is a weak order). Essentially, $x \succsim y$ means that x is at least as good as y. Symbol $\succ$ refers to the asymmetric part of $\succsim$ and $\sim$ to the symmetric one.

Under mild hypotheses [19], it can be shown that $\succsim$ is representable by a utility function, i.e., by a function $u : \mathcal{X} \mapsto \mathbb{Z}_+$ s.t. $x \succsim y \Leftrightarrow u(x) \geqslant u(y)$ for all $x, y \in \mathcal{X}$. Actually, all the algorithms proposed in this paper also work with real-valued utility functions. Our assumption that utilities are integer-valued is only exploited in the proofs of complexity results. As preferences are specific to each individual, utilities must be elicited for each agent, which is often 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 [6] 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 $x_i \in X_i$, $i \in \mathbf{N}$, and it can be effortlessly elicited. However, such a decomposition is not always convenient because it inevitably rules out any interaction between attributes, which is far from being realistic. When preferences are complex, more elaborate models are thus needed. Some generalizations of additive utilities have thus been investigated. For instance *utility independence* on every X_i leads to a more sophisticated form called a *multilinear utility* [7]. Such utilities are more general than additive ones but still cannot cope with many kinds of interactions among attributes. To increase the descriptive power of such models, GAI (generalized additive independence) decompositions have been introduced by [20], that allow more general interactions between attributes [7,9,8,21] while still preserving some decomposability.

2.1. GAI models and GAI nets

GAI decomposition is a generalization of the additive decomposition in which subutilities u_i are allowed to be defined over overlapping factors. As such, they include additive and multilinear decompositions as special cases. They can be more formally defined as follows:

Definition 1. Let $\mathbf{C}_1, \ldots, \mathbf{C}_k$ be subsets of $\mathbf{N}$ such that $\mathbf{N} = \bigcup_{i=1}^{k} \mathbf{C}_i$. A utility $u(\cdot)$ representing $\succsim$ over $\mathcal{X}$ is GAI-decomposable w.r.t. the $X_{\mathbf{C}_i}$ iff there exist functions $u_i : X_{\mathbf{C}_i} \mapsto \mathbb{Z}_+$ such that:

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

Example 1. Utility function $u(a, b, c, d, e, f, g) = u_1(a, b) + u_2(b, c, d) + u_3(c, e) + u_4(b, d, f) + u_5(b, g)$ defined on $A \times B \times C \times D \times E \times F \times G$ is a GAI-decomposable utility, with $X_{\mathbf{C}_1} = A \times B$, $X_{\mathbf{C}_2} = B \times C \times D$, $X_{\mathbf{C}_3} = C \times E$, $X_{\mathbf{C}_4} = B \times D \times F$ and $X_{\mathbf{C}_5} = B \times G$.

GAI decompositions can be represented by graphical structures we call *GAI networks* [9,21] which are essentially similar to junction graphs used in the Bayesian network literature [22,23]:

Definition 2. Let $u(x_1, \ldots, x_n) = \sum_{i=1}^{k} u_i(x_{\mathbf{C}_i})$ be a GAI utility function over $\mathcal{X}$. A GAI network representing $u(\cdot)$ is an undirected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ satisfying the following three properties:

Property 1: $\mathcal{C} = \{X_{\mathbf{C}_1}, \ldots, X_{\mathbf{C}_k}\}$. Vertices $X_{\mathbf{C}_i}$ are called *cliques*. To each vertex $X_{\mathbf{C}_i}$ is associated the corresponding subutility factor u_i from the utility function u;

Property 2: $(X_{\mathbf{C}_i}, X_{\mathbf{C}_j}) \in \mathcal{E} \Rightarrow \mathbf{C}_i \cap \mathbf{C}_j \neq \emptyset$. Edges $(X_{\mathbf{C}_i}, X_{\mathbf{C}_j})$ are labeled by $X_{\mathbf{S}_{ij}}$, where $\mathbf{S}_{ij} = \mathbf{C}_i \cap \mathbf{C}_j$. $X_{\mathbf{S}_{ij}}$ is called a *separator*. Separator $X_{\mathbf{S}_{ij}}$ thus corresponds to the attributes that the two cliques $X_{\mathbf{C}_i}$ and $X_{\mathbf{C}_j}$ have in common;

Property 3: for all $X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$ such that $\mathbf{C}_i \cap \mathbf{C}_j = \mathbf{S}_{ij} \neq \emptyset$, there exists a path between $X_{\mathbf{C}_i}$ and $X_{\mathbf{C}_j}$ in $\mathcal{G}$ such that for every clique $X_{\mathbf{C}_h}$ in this path $\mathbf{S}_{ij} \subseteq \mathbf{C}_h$ (running intersection property).

In the rest of the paper, the $X_{\mathbf{C}_i}$ will always denote cliques of a GAI network and the $X_{\mathbf{S}_{ij}}$ will always denote the separator, i.e., the intersection, between cliques $X_{\mathbf{C}_i}$ and $X_{\mathbf{C}_j}$. By abuse of notation, $X_{\mathbf{S}_{ii}}$ will refer to clique $X_{\mathbf{C}_i}$ itself. Cliques are usually drawn as ellipses and separators as rectangles. For any GAI decomposition, by Definition 2, the cliques of the GAI network should be the sets of attributes of the subutilities. The edges in the network represent the intersections between subsets of attributes. As the intersections are commutative, the GAI network is an undirected graph. Note that this contrasts with UCP nets, where the relationships between vertices in the network correspond to conditional dependencies, thus justifying the use of directed graphs for UCP nets. For any clique $X_{\mathbf{C}_i}$, $\mathrm{Adj}(X_{\mathbf{C}_i})$ will refer to the set of cliques adjacent to $X_{\mathbf{C}_i}$.

In this paper, we shall only be interested in GAI trees. As we shall see, this is not restrictive as general GAI networks can always be compiled into GAI trees. The set of edges of a GAI network can be determined by any algorithm preserving the running intersection property (see the Bayesian network literature on this matter [23]). Fig. 1 shows one possible GAI network representing the GAI utility of Example 1. Note that this network is not the unique representation of the utility function.

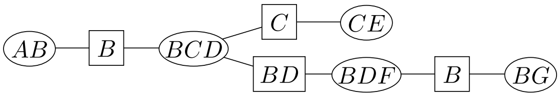


Fig. 1. A GAI tree.

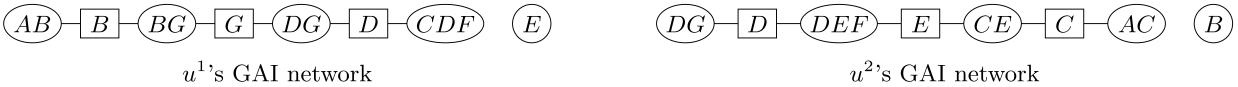


Fig. 2. The GAI trees representing u^1 and u^2.

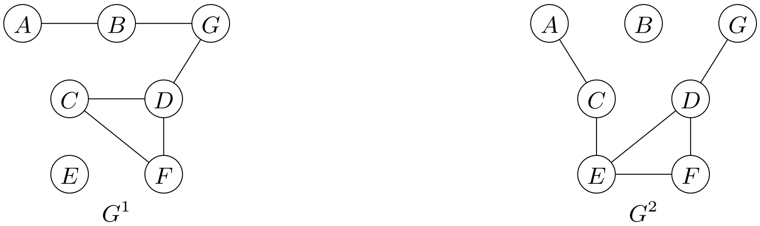


Fig. 3. The Markov networks G^1 of u^1 and G^2 of u^2.

2.2. Handling multiple objectives

Consider now a finite set of objectives $M = \{1, \dots, m\}$ and assume that any solution $x \in \mathcal{X}$ is characterized by a utility vector $(u^1(x), \dots, u^m(x)) \in \mathbb{Z}_+^m$ where $u^i : \mathcal{X} \to \mathbb{Z}_+$ is the ith utility function. This function measures the relative utility of alternatives with respect to the ith point of view (criterion or agent) considered in the problem. Hence, the comparison of alternatives reduces to that of their utility vectors, i.e., instead of comparing alternatives x and y through the numbers assigned to them by utility u as in the preceding subsection, we now compare them through vectors $(u^1(x), \dots, u^m(x))$ and $(u^1(y), \dots, u^m(y))$.

The u^i are functions $\mathcal{X} \to \mathbb{Z}_+$. Hence, separately, they can be considered as single utility functions. Assuming that each objective corresponds to a given agent, each u^i corresponds to the utility function representing the agent's preferences and vectors $(u^1(x), \dots, u^m(x))$ correspond to the utility of the group of agents. Now, if a u^i is the utility function of a given agent, by the preceding subsection, it may be GAI decomposable. Thus, assume that all the u^i are decomposable according to the same GAI net given in Fig. 1. Then, for any $i \in M$,

$$u^i(a, b, c, d, e, f, g) = u^i_1(a, b) + u^i_2(b, c, d) + u^i_3(c, e) + u^i_4(b, d, f) + u^i_5(b, g).$$

Note that, even if the values of the u^i_j differ from one agent to another, all these utilities can be stored in the GAI net of Fig. 1 as follows: store all functions u^i_1, $i \in M$, in clique AB, store all functions u^i_2, $i \in M$, in clique BCD, and so on. Hence the GAI networks described in Section 2.1 can be easily adapted to the multiobjective case. The key property that enables this generalization to the multiobjective case is the fact that all the subutilities u^i_j are defined on attribute sets (here $X_{\mathbf{C}_j}$) included in at least one clique of the GAI net. For instance, the u^i_1 are defined on $A \times B$ and can thus be stored in any clique containing both attributes A and B (here clique AB is the only possible choice).

Let us now consider the case where the agents have preferences that are not decomposable according to the same GAI network. For simplicity, we will illustrate our point for the two agent/objective case: $m = 2$. So suppose that u^1 and u^2 can be decomposed as follows:

$$\begin{aligned} u^1(a, b, c, d, e, f, g) &= u^1_1(a, b) + u^1_2(b, g) + u^1_3(d, g) + u^1_4(c, d, f) + u^1_5(e), \\ u^2(a, b, c, d, e, f, g) &= u^2_1(a, c) + u^2_2(b) + u^2_3(d, g) + u^2_4(d, e, f) + u^2_5(c, e). \end{aligned} \tag{1}$$

These decompositions correspond to the GAI networks of Fig. 2. Note that none of these trees can be used to store both u^1 and u^2, the left graph being unable to store $u^2_4(d, e, f)$, and the right one being unable to store $u^1_1(a, b)$. We thus need to construct another GAI tree that can contain both u^1 and u^2. To construct this new GAI network, we will first create another graph per u^i, called a Markov network. In this graph, each node corresponds to an attribute X_i, and two nodes X_i, X_j are connected by an edge if and only if there exists a subutility $u_h : X_{\mathbf{C}_h} \mapsto \mathbb{Z}_+$ such that $i, j \in \mathbf{C}_h$. In other words, the set of attributes over which u_h is defined contains both X_i and X_j. Hence, in the Markov network, to each subutility u_h corresponds a clique (a complete subgraph). Fig. 3 displays the Markov networks of u^1 and u^2 as described in Eq. (1).

Now, create the union of both graphs, i.e., the Markov network containing an edge between two nodes if and only if G^1 and/or G^2 contains the same edge. The union of G^1 and G^2 is represented in Fig. 4(a). Next, this graph is triangulated

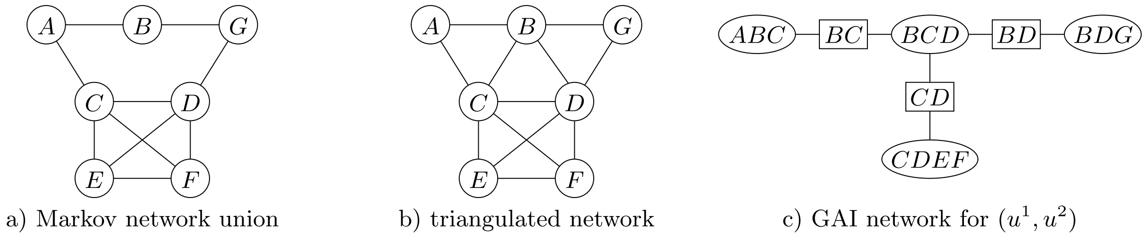


Fig. 4. The creation of the GAI net for (u^1, u^2).

using any triangulation technique [24–26]. Finally, the triangulated graph is mapped into a GAI network: each maximal complete subgraph corresponds to a clique of the GAI network. In the latter, edges are added by any algorithm preserving the running intersection property [23]. In [27], Rose guarantees that whenever the cliques of a GAI net correspond to the maximal complete subgraphs of a triangulated Markov network, then the GAI net is a tree.

Hence, given a set $u^1, \dots, u^m$ of utilities, each one having its own GAI decomposition over $\mathcal{X}$, a global GAI tree can be constructed to store all these utilities. In the sequel, instead of storing m utilities $u^i_j : X_{\mathbf{C}_j} \mapsto \mathbb{Z}_+$, $i \in \{1, \dots, m\}$, in each clique $X_{\mathbf{C}_j}$, we chose an alternative but equivalent representation: we just store one vector-valued utility $u_j : X_{\mathbf{C}_j} \mapsto \mathbb{Z}^m_+$ such that $u_j(x_{\mathbf{C}_j}) = (u^1_j(x_{\mathbf{C}_j}), \dots, u^m_j(x_{\mathbf{C}_j}))$.

3. Pareto search

3.1. Problem formulation

The set of all utility vectors attached to solutions in $\mathcal{X}$ is denoted by $\mathbb{U}$. We recall now some definitions related to dominance and optimality in multiobjective optimization.

Definition 3. The weak Pareto dominance relation is defined on utility vectors of $\mathbb{Z}^m_+$ as: $u \succsim_P v \Leftrightarrow [\forall i \in M,\ u^i \geqslant v^i]$.

Definition 4. The Pareto dominance relation $\succ_P$ is defined as the asymmetric part of $\succsim_P$: $u \succ_P v \Leftrightarrow [u \succsim_P v$ and $\mathrm{not}(v \succsim_P u)]$.

Definition 5. Any utility vector $u \in \mathbb{U}$ is said to be non-dominated in $\mathbb{U}$ (or Pareto-optimal) if $\nexists v \in \mathbb{U}$ such that $v \succ_P u$. The set of non-dominated vectors in $\mathbb{U}$ is denoted $ND(\mathbb{U})$ and is referred to as the “Pareto set”.

The problem of determining the Pareto set in $\mathcal{X}$ can be stated as follows:

Pareto-optimal elements (PO)

Input: a product set of finite domains $\mathcal{X} = X_1 \times \cdots \times X_n$ (n finite), m GAI utility functions $u^i : \mathcal{X} \to \mathbb{Z}_+$, $i = 1, \dots, m$ (m finite),

Goal: determine the entire set of non-dominated vectors in $\mathbb{U}$, and for each utility vector $u \in ND(\mathbb{U})$ a corresponding tuple $x_u \in \mathcal{X}$.

This problem is generally intractable on large size instances. Even when $m = 2$, it may happen that the size of the Pareto set grows exponentially with the number of attributes, as shown by the following example:

Example 2. Consider an instance of PO with two objectives ($m = 2$) on a set $\mathcal{X} = \prod_{j=1}^n X_j$, where $X_j = \{0, 1\}$, $j = 1, \dots, n$. Assume that the objectives are additive utility functions defined, for any Boolean vector $x = (x_1, \dots, x_n) \in \mathcal{X}$, by $u^i(x) = \sum_{j=1}^n u^i_j(x_j)$, $i = 1, 2$, where u^i_j is a marginal utility function defined on X_j by $u^1_j(x_j) = 2^{j-1}x_j$ and $u^2_j(x_j) = 2^{j-1}(1 - x_j)$. Then for all $x \in \{0, 1\}^n$, $u^1(x) = \sum_{j=1}^n 2^{j-1}x_j$ and $u^2(x) = \sum_{j=1}^n 2^{j-1}(1 - x_j)$ and therefore $u^1(x) + u^2(x) = \sum_{j=1}^n 2^{j-1} = 2^n - 1$. So there exists 2^n different Boolean vectors in $\mathcal{X}$, with distinct images in the utility space, all being located 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 $ND(\mathbb{U}) = \mathbb{U}$.

Although pathological, this example shows that the determination of the entire Pareto set may induce prohibitive run times in practice on large size instances with two criteria or more. Numerical tests presented in Section 5 will confirm this point. We establish now a complexity result concerning problem PO (proofs are given in Appendix A).

Proposition 1. *As soon as $|X_i| \geqslant 2$, $i \in \mathbf{N}$, and $m \geqslant 2$, deciding whether there exists a tuple in $\mathcal{X}$ the utility vector of which weakly Pareto dominates a given utility vector $u \in \mathbb{Z}^m_+$ is a NP-complete decision problem (referred to as problem P_u in the sequel).*

3.2. Multiobjective optimization algorithms

Despite the worst case complexity of problem PO, we may expect to solve real instances of reasonable size in admissible times. To this end, we introduce below solution algorithms for PO based on message propagation schemes within the GAI network. For clarity reasons, we first present a variable elimination PO algorithm that processes all the vectors of a given clique before removing it from the GAI network. In the next subsections, we will consider best-first variations of this algorithm which will be more efficient for preference-based search.

3.2.1. A variable elimination algorithm

The algorithm described below is a direct application of variable elimination to determine the Pareto set. Its principle has already been used for CSP in [28]. The algorithm extensively relies on the following proposition and its corollaries:

Proposition 2. *Let* $(\mathbf{D}, \mathbf{E})$ *be a partition of* $\mathbf{N}$. *Assume that utility* $u : \mathcal{X} \mapsto \mathbb{Z}_+^m$ *is additively decomposable as* $u = u_1 + u_2$, *with* $u_1 : X_{\mathbf{D}} \mapsto \mathbb{Z}_+^m$ *and* $u_2 : X_{\mathbf{E}} \mapsto \mathbb{Z}_+^m$ *(here,* $+$ *unambiguously refers to the pointwise addition over vectors). Then*:

$$ND(\mathbb{U}) \subseteq ND\big(\big\{u_1(x_{\mathbf{D}}),\ x_{\mathbf{D}} \in X_{\mathbf{D}}\big\}\big) \boxtimes ND\big(\big\{u_2(x_{\mathbf{E}}),\ x_{\mathbf{E}} \in X_{\mathbf{E}}\big\}\big), \tag{2}$$

where, for any sets $\mathcal{V}, \mathcal{W}$ *of vectors of* $\mathbb{Z}_+^m$, $\mathcal{V} \boxtimes \mathcal{W}$ *is defined as* $\mathcal{V} \boxtimes \mathcal{W} = \{v + w,\ v \in \mathcal{V},\ w \in \mathcal{W}\}$.

In other words, undominated utility vectors of $\mathbb{U}$ can only result from the addition of one undominated utility vector from subset $X_{\mathbf{D}}$ and one undominated utility vector from subset $X_{\mathbf{E}}$. For instance, if $u : A \times B \mapsto \mathbb{Z}_+$ is decomposable as $u_1(A) + u_2(B)$ where u_1 and u_2 are defined as:

$$u_1 = \begin{array}{|c|c|c|}\hline a_1 & a_2 & a_3 \\ \hline (3,4) & (4,2) & (2,3) \\ \hline \end{array} \qquad u_2 = \begin{array}{|c|c|c|}\hline b_1 & b_2 & b_3 \\ \hline (3,5) & (6,3) & (3,3) \\ \hline \end{array}$$

Then $ND(\{u_1(a_i)\}) = \{u_1(a_1) = (3,4),\ u_1(a_2) = (4,2)\}$ and $ND(\{u_2(b_i)\}) = \{u_2(b_1) = (3,5),\ u_2(b_2) = (6,3)\}$ which, composed with operator $\boxtimes$, produce the following set:

$$\big\{u(a_1,b_1) = (6,9),\ u(a_1,b_2) = (9,7),\ u(a_2,b_1) = (7,7),\ u(a_2,b_2) = (10,5)\big\}.$$

Therefore $ND(\mathbb{U}) = \{u(a_1,b_1), u(a_1,b_2), u(a_2,b_2)\}$. Note that no Pareto element involves $u_1(a_3)$ or $u_2(b_3)$, which are dominated in $ND(\{u_1(a_i)\})$ and $ND(\{u_2(b_i)\})$ respectively.

As a consequence, if u is additively decomposable, an efficient procedure to determine $ND(\mathbb{U})$ can be to first determine independently $ND(\{u(x_{\mathbf{D}}),\ x_{\mathbf{D}} \in X_{\mathbf{D}}\})$ and $ND(\{u(x_{\mathbf{E}}),\ x_{\mathbf{E}} \in X_{\mathbf{E}}\})$, then sum-up all these vectors, and finally keep only the undominated resulting vectors.

Corollary 1. *Let* $\mathcal{G}$ *be a GAI tree with only two cliques* $X_{\mathbf{C}_1}$ *and* $X_{\mathbf{C}_2}$ *and their separator* $X_{\mathbf{S}_{12}}$. *Let* $\mathbf{D}_1 = \mathbf{C}_1 \backslash \mathbf{S}_{12}$ *and* $\mathbf{D}_2 = \mathbf{C}_2 \backslash \mathbf{S}_{12}$, *i.e., the* $\mathbf{D}_i$ *are the indices of the attributes that appear in* $\mathbf{C}_i$ *but not in* $\mathbf{C}_{3-i}$ *(or in other words, they appear only on one side of the separator). Then*:

$$ND(\mathbb{U}) \subseteq \bigcup_{x_{\mathbf{S}_{12}} \in X_{\mathbf{S}_{12}}} \big(ND\big(\big\{u_1(x_{\mathbf{D}_1}, x_{\mathbf{S}_{12}}),\ x_{\mathbf{D}_1} \in X_{\mathbf{D}_1}\big\}\big) \boxtimes ND\big(\big\{u_2(x_{\mathbf{D}_2}, x_{\mathbf{S}_{12}}),\ x_{\mathbf{D}_2} \in X_{\mathbf{D}_2}\big\}\big)\big).$$

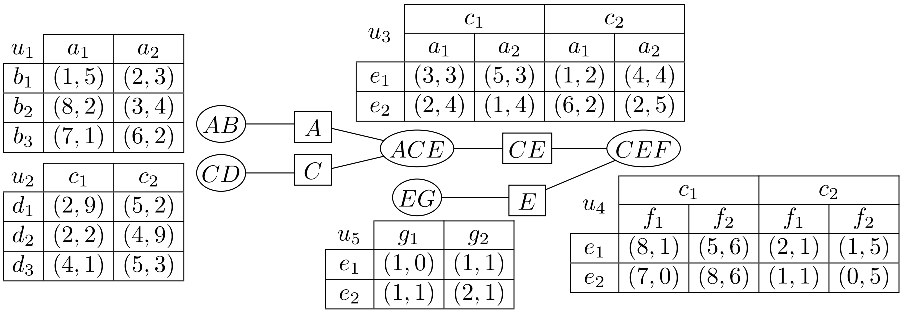
In other words, for each fixed value $x_{\mathbf{S}_{12}}$ of $X_{\mathbf{S}_{12}}$, if a utility vector $u_1(y_{\mathbf{D}_1}, x_{\mathbf{S}_{12}})$ is Pareto dominated by another vector $u_1(x_{\mathbf{D}_1}, x_{\mathbf{S}_{12}})$ defined for the same value of the separator, no combination of $u_1(y_{\mathbf{D}_1}, x_{\mathbf{S}_{12}})$ with another vector $u_2(x_{\mathbf{D}_2}, x_{\mathbf{S}_{12}})$ can result in a vector of $ND(\mathbb{U})$. Hence, to determine $ND(\mathbb{U})$, first determine the undominated vectors of type $u_1(x_{\mathbf{D}_1}, x_{\mathbf{S}_{12}})$ and $u_2(x_{\mathbf{D}_2}, x_{\mathbf{S}_{12}})$ and sum them up for any fixed value $x_{\mathbf{S}_{12}}$, then keep only those that are undominated.

Corollary 2. *Let* $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ *be any GAI network, with* $\mathcal{C} = \{X_{\mathbf{C}_1}, \ldots, X_{\mathbf{C}_k}\}$, *and let* $X_{\mathbf{S}_{ij}}$ *be any separator. Let* $\{X_{\mathbf{C}_{i_1}}, \ldots, X_{\mathbf{C}_{i_r}}\}$ *and* $\{X_{\mathbf{C}_{i_{r+1}}}, \ldots, X_{\mathbf{C}_{i_k}}\}$ *be the sets of cliques on each side of separator* $X_{\mathbf{S}_{ij}}$ *and let* $\mathbf{D} = \bigcup_{t=1}^{r} \mathbf{C}_{i_t} \backslash \mathbf{S}_{ij}$ *and* $\mathbf{E} = \bigcup_{t=r+1}^{k} \mathbf{C}_{i_t} \backslash \mathbf{S}_{ij}$. *Then*:

$$ND(\mathbb{U}) \subseteq \bigcup_{x_{\mathbf{S}_{ij}} \in X_{\mathbf{S}_{ij}}} \left(ND\left(\left\{\sum_{t=1}^{r} u_t(x_{\mathbf{C}_{i_t}}),\ x_{\mathbf{D}} \in X_{\mathbf{D}}\right\}\right) \boxtimes ND\left(\left\{\sum_{t=r+1}^{k} u_t(x_{\mathbf{C}_{i_t}}),\ x_{\mathbf{E}} \in X_{\mathbf{E}}\right\}\right)\right).$$

In other words, to determine $ND(\mathbb{U})$, it is sufficient to select any separator, then compute for each fixed value $x_{\mathbf{S}_{ij}}$ of this separator the undominated utility vectors on each side of the separator and sum-up all these vectors. Finally gather all these vectors for all the values $x_{\mathbf{S}_{ij}}$ and keep only the undominated ones.

Corollary 2 can now be exploited recursively to compute the Pareto set over $\mathbb{U}$: consider the GAI network of Fig. 5, in which subutility tables are displayed next to their corresponding clique. The overall utility function u over $A \times B \times C \times D \times E \times F \times G$ is thus decomposable as: $u_1(A,B) + u_2(C,D) + u_3(A,C,E) + u_4(C,E,F) + u_5(E,G)$. Using Corollary 2



u_1	a_1	a_2
b_1	$(1,5)$	$(2,3)$
b_2	$(8,2)$	$(3,4)$
b_3	$(7,1)$	$(6,2)$

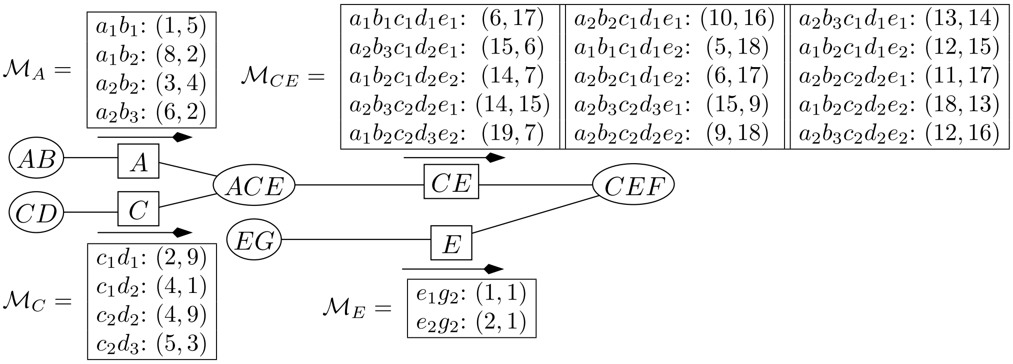
u_2	c_1	c_2
d_1	$(2,9)$	$(5,2)$
d_2	$(2,2)$	$(4,9)$
d_3	$(4,1)$	$(5,3)$

u_3	c_1		c_2	
	a_1	a_2	a_1	a_2
e_1	$(3,3)$	$(5,3)$	$(1,2)$	$(4,4)$
e_2	$(2,4)$	$(1,4)$	$(6,2)$	$(2,5)$

u_4	c_1		c_2	
	f_1	f_2	f_1	f_2
e_1	$(8,1)$	$(5,6)$	$(2,1)$	$(1,5)$
e_2	$(7,0)$	$(8,6)$	$(1,1)$	$(0,5)$

u_5	g_1	g_2
e_1	$(1,0)$	$(1,1)$
e_2	$(1,1)$	$(2,1)$

Fig. 5. The subutility tables of our example.



$\mathcal{M}_A =$

a_1b_1: $(1,5)$
a_1b_2: $(8,2)$
a_2b_2: $(3,4)$
a_2b_3: $(6,2)$

$\mathcal{M}_{CE} =$

$a_1b_1c_1d_1e_1$: $(6,17)$	$a_2b_2c_1d_1e_1$: $(10,16)$	$a_2b_3c_1d_1e_1$: $(13,14)$
$a_2b_3c_1d_2e_1$: $(15,6)$	$a_1b_1c_1d_1e_2$: $(5,18)$	$a_1b_2c_1d_1e_2$: $(12,15)$
$a_1b_2c_1d_2e_2$: $(14,7)$	$a_2b_2c_1d_1e_2$: $(6,17)$	$a_2b_2c_2d_2e_1$: $(11,17)$
$a_2b_3c_2d_2e_1$: $(14,15)$	$a_2b_3c_2d_3e_1$: $(15,9)$	$a_1b_2c_2d_2e_2$: $(18,13)$
$a_1b_2c_2d_3e_2$: $(19,7)$	$a_2b_2c_2d_2e_2$: $(9,18)$	$a_2b_3c_2d_2e_2$: $(12,16)$

$\mathcal{M}_C =$

c_1d_1: $(2,9)$
c_1d_2: $(4,1)$
c_2d_2: $(4,9)$
c_2d_3: $(5,3)$

$\mathcal{M}_E =$

e_1g_2: $(1,1)$
e_2g_2: $(2,1)$

Fig. 6. The messages $\mathcal{M}_i$.

with separator A, we can conclude that vectors in $ND(\mathbb{U})$ can only result from the sum of vectors $u_2, u_3, \ldots,$ to vectors $u_1(A,B)$ that are *undominated* for fixed values of A. Send the latter as message $\mathcal{M}_A$ on separator A (see Fig. 6). Similarly, by Corollary 2 with separator C (resp. E), only vectors $u_2(C,D)$ (resp. $u_5(E,G)$) that are undominated for fixed values of C (resp. E) can lead to vectors in $ND(\mathbb{U})$. Send these vectors as message $\mathcal{M}_C$ on separator C and message $\mathcal{M}_E$ on separator E respectively. Now, apply Corollary 2 with separator CE: vectors in $ND(\mathbb{U})$ can only derive from undominated vectors $u_1 + u_2 + u_3$ for fixed values of CE. But we already know that vectors u_1 (resp. u_2) that are dominated for fixed values of A (resp. C) cannot be part of a vector in $ND(\mathbb{U})$. As a consequence, the undominated vectors $u_1 + u_2 + u_3$ for fixed values of CE can be determined by first computing all the possible vectors $u_1 + u_2 + u_3$ with vectors u_1 and u_2 restricted to $\mathcal{M}_A$ and $\mathcal{M}_C$ respectively, and, then, keeping for each value of CE the undominated ones. In other words, we should compute $\mathcal{M}_{CE}(c,e) = ND(\bigcup_{a\in A} \mathcal{M}_A(a) \boxtimes \{u_2(a,c,e)\} \boxtimes \mathcal{M}_C(c))$ for every $c, e \in C \times E$. This results in an overall separator's message $\mathcal{M}_{CE} = \{\mathcal{M}_{CE}(c,e)\colon\ c,e \in C \times E\}$. Now there just remains to combine the vectors of $\mathcal{M}_{CE}$, $\mathcal{M}_E$ and $u_4(C,E,F)$ and keep the undominated ones: these are the Pareto set $ND(\mathbb{U}) = ND(\bigcup_{c,e,f\in C\times E\times F} \mathcal{M}_{CE}(c,e) \boxtimes u_4(c,e,f) \boxtimes \mathcal{M}_E(e))$. Indeed, this combination corresponds to the combination of all the possible vectors u_i except those that are known not to be part of $ND(\mathbb{U})$. In the end, $ND(\mathbb{U}) = \{u(a_1b_1c_1d_1e_2f_2g_2) = (15,25);\ u(a_1b_2c_1d_1e_2f_2g_2) = (22,22);\ u(a_1b_2c_1d_2e_2f_2g_2) = (24,14);\ u(a_2b_2c_1d_1e_2f_2g_2) = (16,24)\}$.

Note the gain resulting from the application of Corollary 2: for instance, for computing message $\mathcal{M}_{CE}$, we needed only 40 additions (16 additions to compute $\mathcal{M}_A \boxtimes \mathcal{M}_C$, out of which 4 vectors were removed because they were dominated, and then 24 additions to combine the 12 remaining vectors with u_3) instead of 72 additions if we had computed $u_1 + u_2 + u_3$ over $A \times B \times C \times D \times E$. Similarly, computing $ND(\mathbb{U}) = ND(\bigcup_{c,e,f\in C\times E\times F} \mathcal{M}_{CE}(c,e) \boxtimes u_4(c,e,f) \boxtimes \mathcal{M}_E(e))$ required 8 additions to compute $u_4(c,e,f) \boxtimes \mathcal{M}_E(e)$, and then 30 additions to compute the addition of the result with message $\mathcal{M}_{CE}$. Overall, we computed 78 additions instead of the 298 needed if we had computed u over the whole Cartesian product $\mathcal{X}$.

The procedure described above justifies a "collect" algorithm where a clique (here CEF) collects all the information from its neighbors (via messages $\mathcal{M}_i$) to compute the Pareto set. To produce these messages the neighbors also collect the necessary information from their other neighbors, and so on. This results in function `Pareto` described below. However, to define this algorithm and the next ones more conveniently, we will not work directly with subutility vectors as we did above but rather with labels:

Definition 6. A label is a triple $\langle v, x_{\mathbf{C}}, X_{\mathbf{S}_{ij}}\rangle$, where v is a vector of $\mathbb{Z}_+^m$, $x_{\mathbf{C}} \in X_{\mathbf{C}}$ is an instantiation of a set $X_{\mathbf{C}}$ of attributes, and $X_{\mathbf{S}_{ij}}$ is a separator of a GAI network.

Intuitively, a label $\langle v, x_{\mathbf{C}}, X_{\mathbf{S}_{ij}}\rangle$ corresponds to subutility vector v, with the additional information of the partial instantiation $x_{\mathbf{C}}$ of the attributes that were involved in its construction and the separator $X_{\mathbf{S}_{ij}}$ on which the message containing v has been transmitted. Given a clique $X_{\mathbf{C}_i}$ the subutility of which is $u_i : X_{\mathbf{C}_i} \mapsto \mathbb{Z}_+^m$, we define the set of labels corresponding to u_i as $\mathit{Labels}(X_{\mathbf{C}_i}) = \{\langle u_i(x_{\mathbf{C}_i}), x_{\mathbf{C}_i}, X_{\mathbf{C}_i}\rangle\colon\ x_{\mathbf{C}_i} \in X_{\mathbf{C}_i}\}$. In addition, to handle labels easily, we define for any set of labels $\mathcal{V}$, $\mathcal{W}$, any set of attributes $X_{\mathbf{E}}$ and any instantiation $x_{\mathbf{E}} \in X_{\mathbf{E}}$, the following operators:

- $\mathit{ND}(\mathcal{V})$ denotes a set of labels of $\mathcal{V}$ the utility vectors of which are undominated (actually, we keep in $\mathit{ND}(\mathcal{V})$ only one label per undominated vector, that is, we are interested only in one instantiation for each utility vector).
- $\mathcal{V} \otimes \mathcal{W} = \{\langle v + w, x_{\mathbf{C}\cup\mathbf{D}}, X_{\mathbf{E}\cup\mathbf{F}}\rangle\colon\ \langle v, x_{\mathbf{C}}, X_{\mathbf{E}}\rangle \in \mathcal{V}$ and $\langle w, x_{\mathbf{D}}, X_{\mathbf{F}}\rangle \in \mathcal{W}\}$, i.e., operator $\otimes$ aggregates additively labels of $\mathcal{V}$ and $\mathcal{W}$ the partial instantiations of which agree on attributes $X_{\mathbf{C}\cap\mathbf{D}}$. This operator will be used extensively to combine appropriately the aforementioned messages.
- $\mathcal{V}_{\to X_{\mathbf{E}}} = \{\langle v, x_{\mathbf{D}}, X_{\mathbf{E}}\rangle\colon\ \langle v, x_{\mathbf{D}}, X_{\mathbf{F}}\rangle \in \mathcal{V}\}$, i.e., $\mathcal{V}_{\to X_{\mathbf{E}}}$ contains the same labels as $\mathcal{V}$ except that their third component is substituted by $X_{\mathbf{E}}$. This will be used to "move" messages from one separator to another one.
- $\mathcal{V}[x_{\mathbf{E}}] = \{\langle v, y_{\mathbf{D}}, X_{\mathbf{F}}\rangle \in \mathcal{V}\colon\ y_{\mathbf{E}} = x_{\mathbf{E}}\}$, i.e., $\mathcal{V}[x_{\mathbf{E}}]$ is the subset of labels of $\mathcal{V}$ that "agree" with partial instantiation $x_{\mathbf{E}}$.
- $\mathcal{V}_{\Downarrow X_{\mathbf{E}}} = \bigcup_{x_{\mathbf{E}} \in X_{\mathbf{E}}} \mathit{ND}(\mathcal{V}[x_{\mathbf{E}}])_{\to X_{\mathbf{E}}}$, i.e., $\mathcal{V}_{\Downarrow X_{\mathbf{E}}}$ contains the set of all the labels of $\mathcal{V}$ that are undominated by any other label of $\mathcal{V}$ with the same partial instantiation $x_{\mathbf{E}}$. This operator will be used to discard all the labels the combination of which cannot lead to undominated solutions due to Corollary 2.

Given these operators, we can now express the basic message-passing algorithm we described above for computing the Pareto set[1]:

Algorithm 1: A variable elimination algorithm for computing the Pareto set.

```
Function Pareto_Collect($X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$)
01 message $\mathcal{M}_{ij} \leftarrow \mathit{Labels}(X_{\mathbf{C}_i})$
02 for all cliques $X_{\mathbf{C}_k} \in \mathrm{Adj}(X_{\mathbf{C}_i}) \backslash \{X_{\mathbf{C}_j}\}$ do
03     call Pareto_Collect($X_{\mathbf{C}_k}, X_{\mathbf{C}_i}$)
04     $\mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \otimes \mathcal{M}_{ki}$
05 done
06 $\mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij\Downarrow X_{\mathbf{S}_{ij}}}$

Function Pareto()
01 Let root $= X_{\mathbf{C}_p}$ be any clique
02 call Pareto_Collect($X_{\mathbf{C}_p}, X_{\mathbf{C}_p}$)
03 return $\mathit{ND}(\mathcal{M}_{pp})$
```

Proposition 3. *Given a GAI tree $\mathcal{G}$, function* Pareto() *returns precisely the Pareto set.*

Proposition 4. *(See Rollon and Larrosa [28].)* Pareto() *requires space $O(km \times \prod_{i=1}^m K_i \times d^{w^*})$ and time $O(km \times \prod_{i=1}^m K_i^2 \times d^{w^*+1})$, where k is the number of cliques in the GAI network, d is the largest attribute's domain size, w^* is the network's induced width* (*i.e., the number of variables in the largest clique minus one*) *and K_i is a bound on utility u^i.*

Note that function Pareto_Collect, as described above, is generic and does not impose any ordering on messages $\mathcal{M}_{ki}$'s combinations. In practice, the number of operations performed during the for loop of lines 02–05 depends on how combinations are performed. A simple yet very effective strategy consists in, first, computing all the $\mathcal{M}_{ki}$ by calling the appropriate Pareto_Collect($X_{\mathbf{C}_k}, X_{\mathbf{C}_i}$) and, only then, perform the combinations. The latter can be computed iteratively by always selecting the pair of messages that produces a message with the smallest dimension. For instance, assume that we wish to compute $\mathcal{M}_{1i} \otimes \mathcal{M}_{2i} \otimes \mathcal{M}_{3i}$, with messages $\mathcal{M}_{1i}$, $\mathcal{M}_{2i}$, $\mathcal{M}_{3i}$ defined on $A \times B$, $A \times C$ and $C \times D$ respectively. Then first computing $\mathcal{M}_1 = \mathcal{M}_{1i} \otimes \mathcal{M}_{2i}$, and then $\mathcal{M}_1 \otimes \mathcal{M}_{3i}$ produces the same result as computing $\mathcal{M}_2 = \mathcal{M}_{1i} \otimes \mathcal{M}_{3i}$, and then $\mathcal{M}_2 \otimes \mathcal{M}_{2i}$ but the former is faster than the latter since $\mathcal{M}_1$ is defined on $A \times B \times C$ whereas $\mathcal{M}_2$ is defined on $A \times B \times C \times D$.

3.2.2. A best-first Pareto search algorithm

In function Pareto() described previously, sending all the subutility vectors that are undominated for fixed separator values in one single message $\mathcal{M}_{ij}$ prevents applying prunings that can significantly speed-up the algorithm. For this reason, we now propose an alternative algorithm that sends the undominated subutility vectors (labels actually) one by one on the separators. When such vector reaches the root clique, this vector produces new knowledge that can be used to prune those vectors that have not reached the root yet and that we now know for sure cannot be part of the solution. This algorithm is very similar in spirit to the variant of the *MOA** algorithm by Mandow and de la Cruz [29] that improves the standard *MOA** algorithm [30–32]. It favors the early detection of partial solutions that will lead to suboptimal solutions and can therefore discard the corresponding labels hence limiting the combinatorial blowup. The main difference between our approach and *MOA** lies in the exploitation of an explicit junction tree structure instead of an implicit state space graph.

[1] Recall that, by abuse of notation, $X_{\mathbf{S}_{ii}} = X_{\mathbf{C}_i}$ for all $i = 1, \ldots, k$, so that $\langle v, x_{\mathbf{C}}, X_{\mathbf{S}_{ii}}\rangle$ corresponds to a label transmitted to clique $X_{\mathbf{C}_i}$.

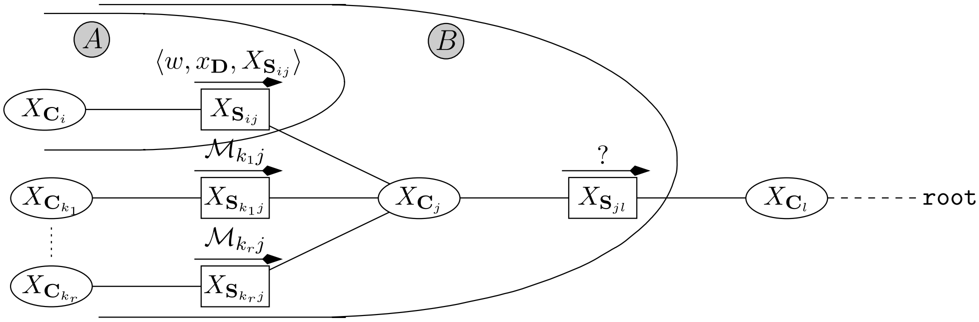


Fig. 7. Making a label move toward `root`.

On one hand, our search algorithm requires satisfying specific constraints imposed by the junction tree. Actually, whenever a label is moved from one separator to the next one, it is combined with other labels stored into adjacent separators; to ensure that this combination is meaningful, the partial instantiations of these labels must necessarily be compatible. Thus, our approach needs more information about how labels were generated than *MOA** usually does. On the other hand, as our best-first search algorithm is based on the tree structure of the GAI network and proceeds from leaves toward the `root` clique, a given label can never be generated more than once. Thus, unlike *MOA**, our algorithm does not need to keep track of a list of *closed* labels and, actually, it never stores such a list. In a sense, this feature is close to *frontier search* algorithms [33].

More precisely, the idea is to maintain two lists of labels: $\mathcal{L}^{open}$, which are the labels that have not reached yet the `root` clique, and $\mathcal{L}^{Pareto}$, which are essentially those labels that have reached `root` and are still undominated. At the beginning of the algorithm, $\mathcal{L}^{Pareto}$ should be empty, and $\mathcal{L}^{open}$ should be basically filled with the labels of the leaves of the GAI network. A nonempty $\mathcal{L}^{open}$ set means that there still exist labels that can possibly be combined with other labels to produce in the end undominated labels that should belong to $\mathcal{L}^{Pareto}$. So, while $\mathcal{L}^{open}$ is nonempty, select one of its labels and make it move toward `root` (by combining it with other appropriate labels, see below). When a label reaches `root`, it is of course discarded from $\mathcal{L}^{open}$ and $\mathcal{L}^{Pareto}$ is updated. When all the labels of interest have reached `root`, then $\mathcal{L}^{open}$ becomes empty and the algorithm has computed the Pareto set.

To illustrate how labels move toward the root, consider an arbitrary label $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ which is currently located on separator $X_{\mathbf{S}_{ij}}$ (see Fig. 7). Like in function `Pareto()`, this label corresponds to the subutility of a partial instantiation of the attributes in the "A" area of the GAI net. Moving it to separator $X_{\mathbf{S}_{jl}}$ should thus logically produce a label corresponding to the instantiation of the attributes in the "B" area of the GAI net. Hence $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ should necessarily be added to compatible labels located on separators $X_{\mathbf{S}_{k_1 j}}, \ldots, X_{\mathbf{S}_{k_r j}}$ as well as to compatible labels stored in clique $X_{\mathbf{C}_j}$ (else some attributes in the "B" area would remain uninstantiated). Among all such possible labels, those that were sent on separators $X_{\mathbf{S}_{k_1 j}}, \ldots, X_{\mathbf{S}_{k_r j}}$ at earlier steps of the algorithm seem to be good candidates. So let us consider the set of labels $\mathcal{M}_{k_t j}$, $t = 1, \ldots, r$, sent on these separators at earlier steps. We propose to generate all the compatible combinations of these messages, i.e., $\{\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle\} \otimes \mathcal{M}_{k_1 j} \otimes \cdots \otimes \mathcal{M}_{k_r j} \otimes \mathit{Labels}(X_{\mathbf{C}_j})$, and then to project the resulting set of labels on separator $X_{\mathbf{S}_{jl}}$ (discarding of course all the dominated labels for fixed values of $X_{\mathbf{S}_{jl}}$) or, in other words, to compute:

$$\mathcal{V} = \left(\left\{\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle\right\} \otimes \mathcal{M}_{k_1 j} \otimes \cdots \otimes \mathcal{M}_{k_r j} \otimes \mathit{Labels}(X_{\mathbf{C}_j})\right)_{\Downarrow X_{\mathbf{S}_{jl}}}. \tag{3}$$

The labels in $\mathcal{V}$ thus correspond to a set of labels appropriate for separator $X_{\mathbf{S}_{jl}}$. As such they should be added to $\mathcal{L}^{open}$ since $\mathcal{L}^{open}$ represents the sets of labels that may potentially be part of the Pareto solutions we look for. In addition, label $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ can now be safely removed from $\mathcal{L}^{open}$ since it has been dealt with (i.e., it has been combined with other labels). The process just described informally can now be described algorithmically by the following function:

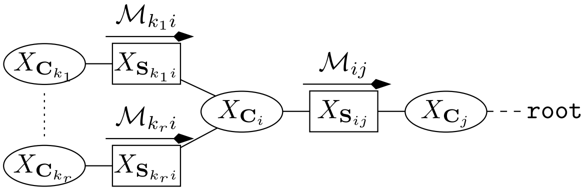
Algorithm 2: The function for moving labels within the Junction tree.

```
Function move_label($\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$)
01 $\mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \cup \{\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle\}$
02 $\mathcal{V} \leftarrow \mathit{Labels}(X_{\mathbf{C}_j}) \otimes \{\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle\}$
03 if $X_{\mathbf{C}_j} \neq$ root then let $X_{\mathbf{C}_l}$ be the clique $\in \mathrm{Adj}(X_{\mathbf{C}_j})$ s.t. $X_{\mathbf{C}_l}$ is on the path between $X_{\mathbf{C}_j}$ and root
04 for all cliques $X_{\mathbf{C}_k} \in \mathrm{Adj}(X_{\mathbf{C}_j})$, $X_{\mathbf{C}_k} \neq X_{\mathbf{C}_i}$ and $X_{\mathbf{C}_k} \neq X_{\mathbf{C}_l}$ (if $X_{\mathbf{C}_l}$ has been defined in line 02) do
05     $\mathcal{V} \leftarrow \mathcal{V} \otimes \mathcal{M}_{kj}$
06 done
07 if $X_{\mathbf{C}_j} \neq$ root then $\mathcal{V} \leftarrow \mathcal{V}_{\Downarrow X_{\mathbf{S}_{jl}}}$ else $\mathcal{V} \leftarrow ND(\mathcal{V})$
08 return $\mathcal{V}$
```

Of course, Eq. (3) produces a set of labels corresponding to instantiations of all the attributes of the "B" area of Fig. 7 if and only if all messages $\mathcal{M}_{k_t j}$ are nonempty. Hence we should enforce that whenever function `move_label` is called, the $\mathcal{M}_{k_t j}$ are actually nonempty. A simple way to achieve this is to initialize the Pareto search by function `initial_labels` below which fills each separator $X_{\mathbf{S}_{ij}}$ with exactly one label per value $x_{\mathbf{S}_{ij}}$. The basic idea of the function is to apply a collect scheme from `root` toward the leaves of the GAI tree and, each time a separator is encountered, to fill it with one label per value. More precisely, for separators on the leaves of the tree, we compute the set of undominated labels per value of the separator, say $\mathcal{V}_{\Downarrow x_{\mathbf{S}_{ij}}}$. As this set may contain several labels per value $x_{\mathbf{S}_{ij}}$, we just keep one label per $x_{\mathbf{S}_{ij}}$ (this produces a message $\mathcal{M}_{ij}$) and put the other ones into $\mathcal{L}^{open}$ to be processed later on. When we encounter a separator $X_{\mathbf{S}_{jl}}$ like in Fig. 7, the collect scheme first fills separators $X_{\mathbf{S}_{ij}}, X_{\mathbf{S}_{k_1 j}}, \ldots, X_{\mathbf{S}_{k_r j}}$ with messages $\mathcal{M}_{\mathbf{S}_{ij}}, \mathcal{M}_{\mathbf{S}_{k_1 j}}, \ldots, \mathcal{M}_{\mathbf{S}_{k_r j}}$ respectively. Now, these messages can be combined to produce a message that can be stored on $X_{\mathbf{S}_{jl}}$: $\mathcal{V} = (\mathcal{M}_{\mathbf{S}_{ij}} \otimes \mathcal{M}_{\mathbf{S}_{k_1 j}} \otimes \cdots \otimes \mathcal{M}_{\mathbf{S}_{k_r j}} \otimes \mathit{Labels}(X_{\mathbf{C}_j}))_{\Downarrow x_{\mathbf{S}_{jl}}}$. Again, $\mathcal{V}$ may contain several messages per value $x_{\mathbf{S}_{jl}}$, hence we store only one of them into message $\mathcal{M}_{jl}$ and put the other ones into $\mathcal{L}^{open}$ to be processed later on. Of course, as the label chosen to be stored into $\mathcal{M}_{jl}$ is processed immediately while the others (those added to $\mathcal{L}^{open}$) will be processed later, the former should be selected as the one that currently seems best fitted to produce a Pareto element when moved till the `root`. For this reason, we call this element a "*most promising*" label. Different strategies do exist to define what a most promising label should be. In our experiments, we defined it as being the label with the highest utility average (over the M objectives). When optimistic heuristics were available, we used the highest average of the sum of the utility vector and the heuristic vector. Of course, alternative characterizations could also have been used such as, e.g., the highest lexicographic utility value (using a lexicographic order over the objectives). Overall, the above algorithm leads to the following function `initial_labels`:

Algorithm 3: The function initializing separator messages with one label per separator's value.

```
Function initial_labels($X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$)
01 $\mathcal{V} \leftarrow \mathit{Labels}(X_{\mathbf{C}_i})$
02 for all cliques $X_{\mathbf{C}_k} \in \mathrm{Adj}(X_{\mathbf{C}_i}) \backslash \{X_{\mathbf{C}_j}\}$ do
03     call initial_labels($X_{\mathbf{C}_k}, X_{\mathbf{C}_i}$)
04     $\mathcal{V} \leftarrow \mathcal{V} \otimes \mathcal{M}_{ki}$
05 done
06 $\mathcal{V} \leftarrow \mathcal{V}_{\Downarrow x_{\mathbf{S}_{ij}}}$
07 $\mathcal{M}_{ij} = \bigcup_{x_{\mathbf{S}_{ij}} \in X_{\mathbf{S}_{ij}}}$ most promising label of $\mathcal{V}[x_{\mathbf{S}_{ij}}]$
08 $\mathcal{L}^{open} \leftarrow \mathcal{L}^{open} \cup (\mathcal{V} \backslash \mathcal{M}_{ij})$
```



Let us illustrate this algorithm on the GAI net of Fig. 5: a call to `initial_labels`(CEF, CEF) would call `initial_labels`(EG, CEF) and `initial_labels`(ACE, CEF) on line 03. The first call first creates set $\mathcal{V} = \{\langle(1,0), e_1 g_1, \text{EG}\rangle, \langle(1,1), e_1 g_2, \text{EG}\rangle, \langle(1,1), e_2 g_1, \text{EG}\rangle, \langle(2,1), e_2 g_2, \text{EG}\rangle\}$ on line 01 and $\mathcal{V}$ is reduced to $\mathcal{V} = \{\langle(1,1), e_1 g_2, \text{E}\rangle, \langle(2,1), e_2 g_2, \text{E}\rangle\}$ on line 06. As $\mathcal{V}$ contains only one label per separator's value e_1,e_2, message $\mathcal{M}_E = \mathcal{V}$ as shown in Fig. 8. Now `initial_labels`(ACE, CEF) calls `initial_labels`(AB, ACE) and `initial_labels`(CD, ACE). The first one will compute $\mathcal{V} = \{\langle(1,5), a_1 b_1, \text{A}\rangle, \langle(8,2), a_1 b_2, \text{A}\rangle, \langle(3,4), a_2 b_2, \text{A}\rangle, \langle(6,2), a_2 b_3, \text{A}\rangle\}$ on line 06 (discarding both labels $\langle(7,1), a_1 b_3, \text{A}\rangle$ and $\langle(2,3), a_2 b_1, \text{A}\rangle$ because they are dominated by $\langle(8,2), a_1 b_2, \text{A}\rangle$ and $\langle(3,4), a_2 b_2, \text{A}\rangle$ respectively). Here, $\mathcal{V}$ contains more than one element per a_i, so we need select only one element per a_i in $\mathcal{V}$ to create message $\mathcal{M}_A$. Say $\mathcal{M}_A = \{\langle(1,5), a_1 b_1, \text{A}\rangle, \langle(3,4), a_2 b_2, \text{A}\rangle\}$ (see Fig. 8). Note that, in this example, to reduce the number of iterations of the algorithm, the most promising label is not always chosen as the one with the highest utility average. The other elements of $\mathcal{V}$, i.e., $\{\langle(8,2), a_1 b_2, \text{A}\rangle, \langle(6,2), a_2 b_3, \text{A}\rangle\}$ are thus added to $\mathcal{L}^{open}$ to be processed later on. Similarly, `initial_labels`(CD, ACE) computes on line 06 label set $\mathcal{V} = \{\langle(2,9), c_1 d_1, \text{C}\rangle, \langle(4,1), c_1 d_3, \text{C}\rangle, \langle(4,9), c_2 d_2, \text{C}\rangle, \langle(5,3), c_2 d_3, \text{C}\rangle\}$. From $\mathcal{V}$ we extract message $\mathcal{M}_C = \{\langle(2,9), c_1 d_1, \text{C}\rangle, \langle(4,9), c_2 d_2, \text{C}\rangle\}$ and labels $\langle(4,1), c_1 d_3, \text{C}\rangle$ and $\langle(5,3), c_2 d_3, \text{C}\rangle$ are added to $\mathcal{L}^{open}$ to be processed later on. Now `initial_labels`(ACE, CEF) can compute label set $[\mathit{Labels}(\text{ACE}) \otimes \mathcal{M}_A \otimes \mathcal{M}_C]_{\Downarrow_{CE}}$, which leads to:

$$\mathcal{V} = \big\{\big\langle(6,17), a_1 b_1 c_1 d_1 e_1, \text{CE}\big\rangle, \big\langle(10,16), a_2 b_2 c_1 d_1 e_1, \text{CE}\big\rangle, \big\langle(5,18), a_1 b_1 c_1 d_1 e_2, \text{CE}\big\rangle,$$
$$\big\langle(6,17), a_2 b_2 c_1 d_1 e_2, \text{CE}\big\rangle, \big\langle(11,17), a_2 b_2 c_2 d_2 e_1, \text{CE}\big\rangle, \big\langle(9,18), a_2 b_2 c_2 d_2 e_2, \text{CE}\big\rangle\big\}$$

on line 06. From this set, we extract message $\mathcal{M}_{CE}$ by selecting one label per value (c_i, e_j):

$$\mathcal{M}_{CE} = \big\{\big\langle(10,16), a_2 b_2 c_1 d_1 e_1, \text{CE}\big\rangle, \big\langle(6,17), a_2 b_2 c_1 d_1 e_2, \text{CE}\big\rangle,$$
$$\big\langle(11,17), a_2 b_2 c_2 d_2 e_1, \text{CE}\big\rangle, \big\langle(9,18), a_2 b_2 c_2 d_2 e_2, \text{CE}\big\rangle\big\}$$

and add to $\mathcal{L}^{open}$ labels $\langle(6,17), a_1 b_1 c_1 d_1 e_1, \text{CE}\rangle$ and $\langle(5,18), a_1 b_1 c_1 d_1 e_2, \text{CE}\rangle$ that were not selected to be part of $\mathcal{M}_{CE}$. Finally, `initial_labels`(CEF, CEF) computes $\mathcal{V} = \mathit{Labels}(\text{CEF}) \otimes \mathcal{M}_{CE} \otimes \mathcal{M}_E$, i.e.:

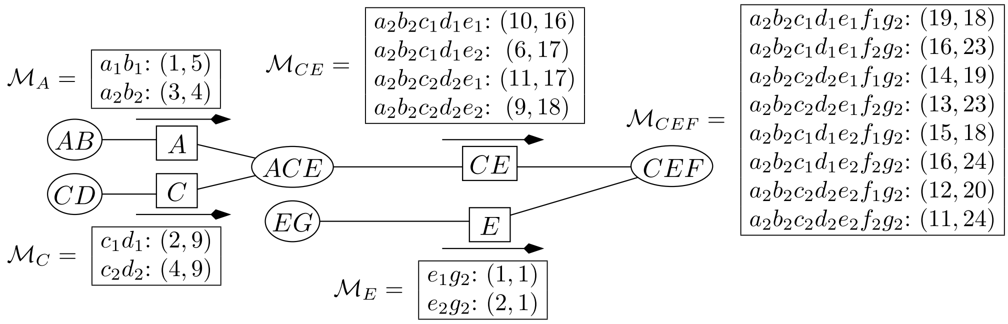

$\mathcal{M}_A =$
a_1b_1: $(1,5)$
a_2b_2: $(3,4)$
$\mathcal{M}_{CE} =$
$a_2b_2c_1d_1e_1$: $(10,16)$
$a_2b_2c_1d_1e_2$: $(6,17)$
$a_2b_2c_2d_2e_1$: $(11,17)$
$a_2b_2c_2d_2e_2$: $(9,18)$
$\mathcal{M}_{CEF} =$
$a_2b_2c_1d_1e_1f_1g_2$: $(19,18)$
$a_2b_2c_1d_1e_1f_2g_2$: $(16,23)$
$a_2b_2c_2d_2e_1f_1g_2$: $(14,19)$
$a_2b_2c_2d_2e_1f_2g_2$: $(13,23)$
$a_2b_2c_1d_1e_2f_1g_2$: $(15,18)$
$a_2b_2c_1d_1e_2f_2g_2$: $(16,24)$
$a_2b_2c_2d_2e_2f_1g_2$: $(12,20)$
$a_2b_2c_2d_2e_2f_2g_2$: $(11,24)$
AB
A
CD
C
ACE
CE
CEF
EG
E
$\mathcal{M}_C =$
c_1d_1: $(2,9)$
c_2d_2: $(4,9)$
$\mathcal{M}_E =$
e_1g_2: $(1,1)$
e_2g_2: $(2,1)$

Fig. 8. Messages computed by function `initial_labels`.

$$\begin{aligned}\mathcal{V} = \big\{&\big\langle(19,18), a_2b_2c_1d_1e_1f_1g_2, \textsc{cef}\big\rangle, \big\langle(16,23), a_2b_2c_1d_1e_1f_2g_2, \textsc{cef}\big\rangle,\\ &\big\langle(14,19), a_2b_2c_2d_2e_1f_1g_2, \textsc{cef}\big\rangle, \big\langle(13,23), a_2b_2c_2d_2e_1f_2g_2, \textsc{cef}\big\rangle,\\ &\big\langle(15,18), a_2b_2c_1d_1e_2f_1g_2, \textsc{cef}\big\rangle, \big\langle(16,24), a_2b_2c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle,\\ &\big\langle(12,20), a_2b_2c_2d_2e_2f_1g_2, \textsc{cef}\big\rangle, \big\langle(11,24), a_2b_2c_2d_2e_2f_2g_2, \textsc{cef}\big\rangle\big\}\end{aligned}$$

and, as $\mathcal{V}$ contains only one label per triple (c,e,f), $\mathcal{M}_{CEF} = \mathcal{V}$. Overall, the created messages $\mathcal{M}_{ij}$ are those of Fig. 8 and $\mathcal{L}^{open}$ is initialized to:

$$\begin{aligned}\mathcal{L}^{open} = \big\{&\big\langle(8,2), a_1b_2, \textsc{a}\big\rangle, \big\langle(6,2), a_2b_3, \textsc{a}\big\rangle, \big\langle(4,1), c_1d_3, \textsc{c}\big\rangle,\\ &\big\langle(5,3), c_2d_3, \textsc{c}\big\rangle, \big\langle(6,17), a_1b_1c_1d_1e_1, \textsc{ce}\big\rangle, \big\langle(5,18), a_1b_1c_1d_1e_2, \textsc{ce}\big\rangle\big\}.\end{aligned}$$

In addition to filling separators with a single label per separator's value, which was its primary purpose, function `initial_labels` has an important feature:

Proposition 5. *Let $X_{\mathbf{C}_p}$ be any clique. Call* `initial_labels`$(X_{\mathbf{C}_p}, X_{\mathbf{C}_p})$*. Then, for any utility vector $u(x) = \sum_{j=1}^{k} u_j(x_{\mathbf{C}_j}) \in ND(\mathbb{U})$, one and only one of the following two assertions holds*:

1. *there exists a label $\langle u(x), x, X_{\mathbf{C}_p}\rangle$ in message $\mathcal{M}_{pp}$;*
2. *there exist some indices $j_1, \ldots, j_r$ such that $\mathcal{L}^{open}$ contains a label $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{j_1 l}}\rangle$ such that $w = \sum_{t=1}^{r} u_{j_t}(x_{\mathbf{C}_{j_t}})$, $\mathbf{D} = \bigcup_{t=1}^{r} \mathbf{C}_{j_t}$ and $X_{\mathbf{C}_{j_1}}$ is, among $X_{\mathbf{C}_{j_1}}, \ldots, X_{\mathbf{C}_{j_r}}$, the clique which is nearest to* `root`*.*

According to Proposition 5, after calling `initial_labels`$(X_{\mathbf{C}_p}, X_{\mathbf{C}_p})$, for any $u(x) \in ND(\mathbb{U})$, either there exists a label corresponding to $u(x)$ in $\mathcal{M}_{pp}$, or there exists a label in $\mathcal{L}^{open}$ corresponding to $u(x)$. The latter case can be interpreted as the fact that the propagation of the label corresponding to $u(x)$ toward root $X_{\mathbf{C}_p}$ has been temporarily stopped on a given separator because this label did not seem, at that time, to be a most promising label to belong to $ND(\mathbb{U})$. We should thus subsequently use function `move_label` to make it eventually reach root $X_{\mathbf{C}_p}$. This justifies algorithm `Pareto`* below, which we present for simplicity without pruning rules (those will be given later on):

Algorithm 4: Basic best-first Pareto search.

```
Function Pareto*()
01 let root $X_{\mathbf{C}_p}$ be any clique
02 $\mathcal{L}^{open} \leftarrow \emptyset$; call initial_labels($X_{\mathbf{C}_p}, X_{\mathbf{C}_p}$)
03 $\mathcal{L}^{Pareto} \leftarrow ND(\mathcal{M}_{pp})$
04 while $\mathcal{L}^{open} \neq \emptyset$ do
05    let $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}}\rangle$ be the most promising label in $\mathcal{L}^{open}$
06    remove $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}}\rangle$ from $\mathcal{L}^{open}$
07    $\mathcal{V} \leftarrow$ move_label($\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}}\rangle$)
08    if $X_{\mathbf{S}_{ij}}$ = root $X_{\mathbf{C}_p}$ then $\mathcal{L}^{Pareto} \leftarrow ND(\mathcal{L}^{Pareto} \cup \mathcal{V})$ else $\mathcal{L}^{open} \leftarrow \mathcal{L}^{open} \cup \mathcal{V}$
09 done
10 return $\mathcal{L}^{Pareto}$
```

In our experiments, we defined the "*most promising*" label of line 05 as being the nearest label to the `root` clique and, to break ties, that with the highest utility average (over the M objectives). Favoring labels that are nearest to the `root` clique is effective because it tends to quickly fill $\mathcal{L}^{Pareto}$ with feasible labels that can be used subsequently to prune as early as possible the labels of $\mathcal{L}^{open}$. When optimistic heuristics were available, we broke ties using the highest average of the sum of their utility vectors and their heuristic vectors.

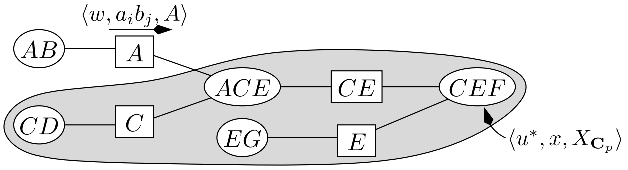
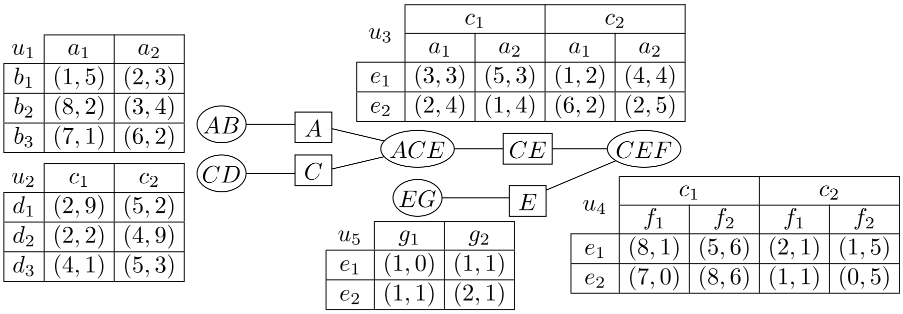

$\langle w, a_i b_j, A\rangle$
AB
A
ACE
CE
CEF
CD
C
EG
E
$\langle u^*, x, X_{\mathbf{C}_p}\rangle$

Fig. 9. An optimistic heuristic-based pruning rule.



u_1	a_1	a_2
b_1	$(1,5)$	$(2,3)$
b_2	$(8,2)$	$(3,4)$
b_3	$(7,1)$	$(6,2)$

u_2	c_1	c_2
d_1	$(2,9)$	$(5,2)$
d_2	$(2,2)$	$(4,9)$
d_3	$(4,1)$	$(5,3)$

u_3	c_1		c_2	
	a_1	a_2	a_1	a_2
e_1	$(3,3)$	$(5,3)$	$(1,2)$	$(4,4)$
e_2	$(2,4)$	$(1,4)$	$(6,2)$	$(2,5)$

u_5	g_1	g_2
e_1	$(1,0)$	$(1,1)$
e_2	$(1,1)$	$(2,1)$

u_4	c_1		c_2	
	f_1	f_2	f_1	f_2
e_1	$(8,1)$	$(5,6)$	$(2,1)$	$(1,5)$
e_2	$(7,0)$	$(8,6)$	$(1,1)$	$(0,5)$

Fig. 10. The subutility tables of our example.

Proposition 6. *Given a GAI tree $\mathcal{G}$, function* `Pareto*`() *returns precisely the Pareto set.*

Proposition 7. `Pareto*`() *requires space $O(km \times \prod_{i=1}^m K_i \times d^{w^*})$ and time $O(km \times \prod_{i=1}^m K_i^2 \times d^{w^*+1})$, where k is the number of cliques in the GAI network, d is the largest attribute's domain size, w^* is the network's induced width* (*i.e., the number of attributes in the largest clique minus one*) *and K_i is a bound on utility u^i.*

For the moment, function `Pareto*` moves labels one by one toward the root but it applies no pruning rule (except, of course, that given by Corollary 2) and, in the end, it sends precisely the same messages $\mathcal{M}_{ij}$ as those sent by function `Pareto` in Section 3.2.1. Hence we shall now introduce an additional pruning rule to improve the efficiency of the algorithm.

3.2.3. Pruning rule

Consider the graph of Fig. 9 and apply function `Pareto*`. At some step of the algorithm, select on line 05 a label, say $\langle w, a_i b_j, A\rangle$, from $\mathcal{L}^{open}$ to be moved toward separator CE. Assume that we know that there exists some vector v such that, for any completion $(c_k, d_l, e_r, f_s, g_t)$ of the attributes of the gray area, $v \succsim_P u_2(c_k, d_l) + u_3(a_i, c_k, e_r) + u_4(c_k, e_r, f_s) + u_5(e_r, g_t)$. If, in addition, it turns out that $\mathcal{L}^{Pareto}$ contains at that time a label $\langle u^*, x, X_{\mathbf{C}_p}\rangle$ such that $u^* \succsim_P v + w$, then label $\langle w, a_i b_j, A\rangle$ can be safely discarded because it cannot be part of a solution of $ND(\mathbb{U})$. Indeed, if there exists a vector $z \in \mathbb{U}$ such that $w' + w \succsim_P z$, with w' the utility vector of an instantiation $(c_k, d_l, e_r, f_s, g_t)$ of the attributes of the gray area, then $u^* \succsim_P v + w \succsim_P w' + w \succsim_P z$ and so u^* also Pareto dominates z. As a consequence, vector $w' + w$ should not be added to $ND(\mathbb{U})$.

This suggests defining an optimistic heuristic that, given a label like $\langle w, a_i b_j, A\rangle$, is able to return a vector or a set of vectors like the v vector mentioned in the preceding paragraph. For a vector set-valued optimistic heuristic see [30]. In this paper, for simplicity, we present a single-vector valued heuristic easily computable using the collect algorithm illustrated in Fig. 11: on separator E, send message $\mathcal{H}_E$ containing, for each value e_i of E, a utility vector constituted by the max over each criterion of $\{u_5(e_i, g_1), u_5(e_i, g_2)\}$, that is,

$$\begin{aligned}\mathcal{H}_E &= \big\{\big\langle\big(\max\{1,1\}, \max\{0,1\}\big), e_1, \textsc{e}\big\rangle, \big\langle\big(\max\{1,2\}, \max\{1,1\}\big), e_2, \textsc{e}\big\rangle\big\}\\ &= \big\{\big\langle(1,1), e_1, \textsc{e}\big\rangle, \big\langle(2,1), e_2, \textsc{e}\big\rangle\big\}.\end{aligned}$$

Clearly, by construction, vectors in $\mathcal{H}_E$ weakly Pareto dominate all possible vectors $u_5(e_r, g_t)$, $r, t = 1, 2$. As a consequence, the utility vectors in $\mathcal{V} = \mathit{Labels}(\textsc{cef}) \otimes \mathcal{H}_E$ Pareto dominate $\mathit{Labels}(\textsc{cef}) \otimes \mathit{Labels}(\textsc{eg})$. Now we can apply the same process with $\mathcal{V}$: construct a message $\mathcal{H}_{CE}$ containing, for each pair (c_k, e_r), a vector constituted by the max over each criterion of $\mathcal{V}[c_k, e_r]$. Here again, by construction, vectors in $\mathcal{H}_{CE}$ weakly Pareto dominate all possible vectors in $\mathcal{V}$ which, in turn, weakly Pareto dominate all possible vectors in $\mathit{Labels}(\textsc{cef}) \otimes \mathit{Labels}(\textsc{eg})$. Apply the same process for constructing message $\mathcal{H}_C$ and, finally, to construct $\mathcal{H}_A$ apply again this maximization per criterion scheme on $\mathcal{H}_C \otimes \mathit{Labels}(\textsc{ace}) \otimes \mathcal{H}_{CE}$. Clearly, by the process of construction, vectors in $\mathcal{H}_A$ weakly Pareto dominate all subutility vectors $u_2(c_k, d_l) + u_3(a_i, c_k, e_r) + u_4(c_k, e_r, f_s) + u_5(e_r, g_t)$, for any completion $(c_k, d_l, e_r, f_s, g_t)$, which was precisely what we were looking for. This justifies the following recursive algorithm:

Algorithm 5: Computation of the optimistic heuristic.

```
Function Heuristic_Collect($X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$)
01 $\mathcal{V} \leftarrow$ Labels($X_{\mathbf{C}_i}$)
02 for all cliques $X_{\mathbf{C}_r} \in \text{Adj}(X_{\mathbf{C}_i}) \backslash X_{\mathbf{C}_j}$ do
03    call Heuristic_Collect($X_{\mathbf{C}_r}, X_{\mathbf{C}_i}$)
04    $\mathcal{V} \leftarrow \mathcal{V} \otimes \mathcal{H}_{ir}$
05 done
06 $\mathcal{H}_{ji} \leftarrow \mathcal{M}\text{ax}_{\downarrow X_{\mathbf{S}_{ij}}} \mathcal{V}$
```

where, for any set of attributes $X_{\mathbf{E}}$ and any set of labels $\mathcal{V}$, $\mathcal{M}\text{ax}_{\downarrow X_{\mathbf{E}}}\mathcal{V}$ is defined as:

$$\mathcal{M}\text{ax}_{\downarrow X_{\mathbf{E}}}\mathcal{V} = \left\{\left\langle (v^1, \dots, v^m), x_{\mathbf{E}}, X_{\mathbf{S}_{ij}} \right\rangle: \text{ for all } i,\ v^i = \max\left\{w^i: \left\langle \left(w^1, \dots, w^m\right), y_{\mathbf{C}}, X_{\mathbf{D}} \right\rangle \in \mathcal{V}[x_{\mathbf{E}}]\right\}\right\}.$$

In other words, for each $x_{\mathbf{E}}$, v^i is the max for criterion i of the utilities of the labels agreeing with $x_{\mathbf{E}}$. Now it is clear that the following proposition holds:

Proposition 8. *Call* Heuristic_Collect($X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$). *Then, for any $x_{\mathbf{S}_{ij}} \in X_{\mathbf{S}_{ij}}$, $\mathcal{H}_{ji}[x_{\mathbf{S}_{ij}}]$ weakly Pareto dominates the sums of the subutilities obtained by instantiations of the attributes in cliques $X_{\mathbf{r}}$ such that $X_{\mathbf{C}_i}$ is not on the path between $X_{\mathbf{r}}$ and* root.

Note that, in Proposition 8, cliques $X_{\mathbf{r}}$ are precisely those located in the gray area of Fig. 9.

Proposition 9. Heuristic_Collect($X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$) *requires space $O(km \times d^{w^*})$ and time $O(km \times d^{w^*+1})$, where k is the number of cliques in the GAI network, d is the largest attribute's domain size, w^* is the network's induced width* (*i.e., the number of attributes in the largest clique minus one*) *and K_i is a bound on utility u^i.*

Importing this optimistic heuristic into function Pareto* essentially requires modifying its line 08 where $\mathcal{L}^{open}$ and $\mathcal{L}^{Pareto}$ were updated: now each time $\mathcal{L}^{Pareto}$ is updated, we should try to prune as well all the labels in $\mathcal{L}^{open}$ which, when combined with their optimistic heuristic value, are Pareto dominated by some element in $\mathcal{L}^{Pareto}$. So let us define the corresponding dominance operator: for any sets of labels $\mathcal{V}, \mathcal{W}$, where $\mathcal{W}$ are labels with complete instantiations,

$$ND_H(\mathcal{V}, \mathcal{W}) = \left\{\langle v, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle \in \mathcal{V}: \text{ there exists no } \langle w, y, X_{\mathbf{C}_p} \rangle \in \mathcal{W}:\ w \succsim_P v + h, \text{ where } h \\ \text{is the utility of } \mathcal{H}_{ij}[x_{\mathbf{S}_{ij}}] \text{ as returned by } \texttt{Heuristic\_Collect}(X_{\mathbf{C}_i}, X_{\mathbf{C}_j})\right\}.$$

Clearly, calling $ND_H(\mathcal{L}^{open}, \mathcal{L}^{Pareto})$ in function Pareto* will remove only the labels from $\mathcal{L}^{open}$ that, if moved until root, would produce labels weakly Pareto dominated by those of $ND(\mathbb{U})$. As a consequence, it is safe to discard such labels and doing it as early as possible reduces the run time of the algorithm. This leads to the following Pareto search algorithm:

Algorithm 6: Efficient best-first Pareto search algorithm.

```
Function Pareto$^*_{\mathcal{H}}$()
01 let root $X_{\mathbf{C}_p}$ be any clique
02 $\mathcal{L}^{open} \leftarrow \emptyset$; call initial_labels($X_{\mathbf{C}_p}, X_{\mathbf{C}_p}$)
03 for all Separators $X_{\mathbf{S}_{ji}}$ do call Heuristic_Collect($X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$) done
04 $\mathcal{L}^{Pareto} \leftarrow ND(\mathcal{M}_{pp})$;  $\mathcal{L}^{open} \leftarrow ND_H(\mathcal{L}^{open}, \mathcal{L}^{Pareto})$
05 while $\mathcal{L}^{open} \neq \emptyset$ do
06    let $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ be the most promising label in $\mathcal{L}^{open}$
07    remove $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ from $\mathcal{L}^{open}$
08    $\mathcal{V} \leftarrow$ move_label($\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$)
09    if $X_{\mathbf{S}_{ij}} =$ root $X_{\mathbf{C}_p}$ then $\mathcal{L}^{Pareto} \leftarrow ND(\mathcal{L}^{Pareto} \cup \mathcal{V})$;  $\mathcal{L}^{open} \leftarrow ND_H(\mathcal{L}^{open}, \mathcal{V})$
10    else $\mathcal{L}^{open} \leftarrow \mathcal{L}^{open} \cup ND_H(\mathcal{V}, \mathcal{L}^{Pareto})$
11 done
12 return $\mathcal{L}^{Pareto}$
```

Proposition 10. *Given a GAI tree $\mathcal{G}$, function* Pareto$^*_{\mathcal{H}}$() *returns precisely the Pareto set.*

Proposition 11. Pareto$^*_{\mathcal{H}}$() *requires space $O(km \times \prod_{i=1}^m K_i \times d^{w^*})$ and time $O(km \times \prod_{i=1}^m K_i^2 \times d^{w^*+1})$, where k is the number of cliques in the GAI network, d is the largest attribute's domain size, w^* is the network's induced width* (*i.e., the number of attributes in the largest clique minus one*) *and K_i is a bound on utility u^i.*

Note that, for simplicity of exposition, we chose to call Heuristic_Collect as many times as there are different separators. This is not optimally efficient as many messages $\mathcal{H}_{ri}$ computed by this function are actually computed several times. However, using techniques similar to inference in Bayesian networks [34], i.e., by using a collect/distribute algorithm,

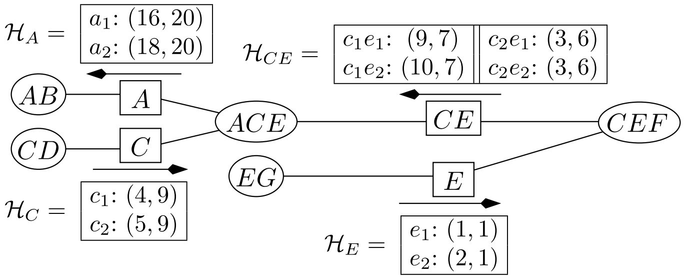


Fig. 11. Computing the optimistic heuristic for label $\langle w, a_ib_j, A\rangle$.

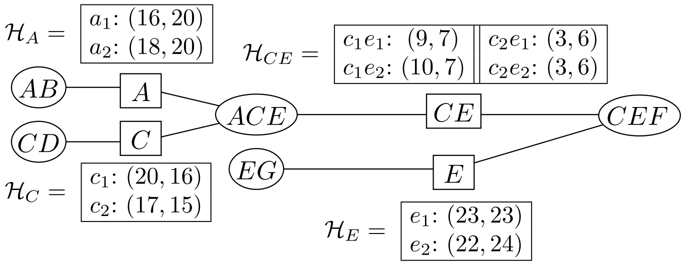


Fig. 12. The optimistic heuristic computed for every separator.

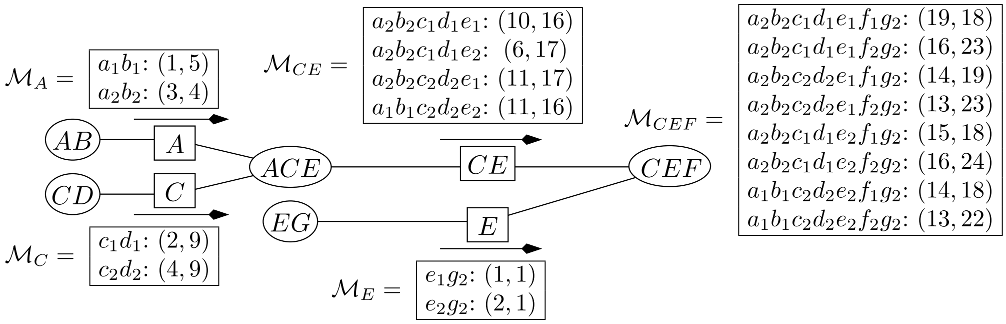

$\mathcal{M}_A =$
a_1b_1: (1, 5)
a_2b_2: (3, 4)
$\mathcal{M}_{CE} =$
$a_2b_2c_1d_1e_1$: (10, 16)
$a_2b_2c_1d_1e_2$: (6, 17)
$a_2b_2c_2d_2e_1$: (11, 17)
$a_1b_1c_2d_2e_2$: (11, 16)
$\mathcal{M}_{CEF} =$
$a_2b_2c_1d_1e_1f_1g_2$: (19, 18)
$a_2b_2c_1d_1e_1f_2g_2$: (16, 23)
$a_2b_2c_2d_2e_1f_1g_2$: (14, 19)
$a_2b_2c_2d_2e_1f_2g_2$: (13, 23)
$a_2b_2c_1d_1e_2f_1g_2$: (15, 18)
$a_2b_2c_1d_1e_2f_2g_2$: (16, 24)
$a_1b_1c_2d_2e_2f_1g_2$: (14, 18)
$a_1b_1c_2d_2e_2f_2g_2$: (13, 22)
AB
A
ACE
CE
CEF
CD
C
EG
E
$\mathcal{M}_C =$
c_1d_1: (2, 9)
c_2d_2: (4, 9)
$\mathcal{M}_E =$
e_1g_2: (1, 1)
e_2g_2: (2, 1)

Fig. 13. The content of messages $\mathcal{M}_{ij}$ after the completion of `initial_labels`(CEF, CEF).

all these redundancies can be removed and the overall computational burden to compute the heuristic for all the separators is only twice the time required to complete one call to `Heuristic_Collect`($X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$).

Let us now see on the GAI network of Fig. 10 how our new pruning rule can effectively improve the run time to compute the Pareto set. First, we display in Fig. 12 the values of the optimistic heuristic computed on every separator.

After the completion of `initial_labels`(CEF, CEF), messages $\mathcal{M}_{ij}$ are for instance those of Fig. 13 (defining "the most promising" vectors appropriately) and the content of $\mathcal{L}^{open}$ is described below:

$\mathcal{L}^{open} =$

separator	*inst*	*vect*	*vect* + $\mathcal{H}$	*separator*	*inst*	*vect*	*vect* + $\mathcal{H}$
A	a_1b_2	(8, 2)	(24, 22)	CE	$a_1b_1c_1d_1e_1$	(6, 17)	(15, 24)
A	a_2b_3	(6, 2)	(24, 22)	CE	$a_1b_1c_1d_1e_2$	(5, 18)	(15, 25)
C	c_1d_3	(4, 1)	(24, 17)	CE	$a_1b_1c_2d_2e_1$	(6, 16)	(9, 22)
C	c_2d_3	(5, 3)	(22, 18)	CE	$a_2b_2c_2d_2e_2$	(9, 18)	(12, 24)

Then $\mathcal{L}^{Pareto}$ is filled with the undominated labels of $\mathcal{M}_{CEF}$, i.e.:

$$\mathcal{L}^{Pareto} = \left\{\left\langle(19, 18), a_2b_2c_1d_1e_1f_1g_2, \textsc{cef}\right\rangle, \left\langle(16, 24), a_2b_2c_1d_1e_2f_2g_2, \textsc{cef}\right\rangle\right\}.$$

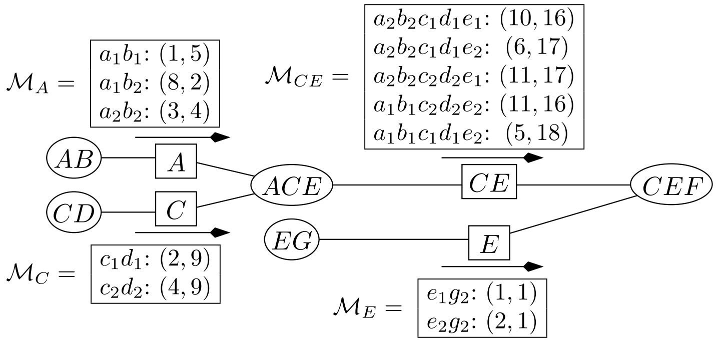


Fig. 14. The content of messages $\mathcal{M}_{ij}$ after moving label $\langle(8,2), a_1b_2, A\rangle$.

Now, the last instruction of line 04 reduces set $\mathcal{L}^{open}$: all the labels which, combined with the optimistic heuristic, are dominated by elements in $\mathcal{L}^{Pareto}$ are discarded.

$\mathcal{L}^{open} =$

separator	*inst*	*vect*	*vect* + $\mathcal{H}$	*separator*	*inst*	*vect*	*vect* + $\mathcal{H}$
A	a_1b_2	(8, 2)	(24, 22)	CE	$a_1b_1c_1d_1e_2$	(5, 18)	(15, 25)
A	a_2b_3	(6, 2)	(24, 22)				
C	c_1d_3	(4, 1)	(24, 17)				
C	c_2d_3	(5, 3)	(22, 18)				

Next we enter the while loop of lines 05–11. Let us assume that the most promising element is the label located on separator CE. Then we move this label up to the root, thus producing $\mathcal{V} = \{\langle(14,19), a_1b_1c_1d_1e_2f_1g_2, \textsc{cef}\rangle, \langle(15,25), a_1b_1c_1d_1e_2f_2g_2, \textsc{cef}\rangle\}$. The second element is thus added to $\mathcal{L}^{Pareto}$ on line 09, which becomes:

$$\mathcal{L}^{Pareto} = \big\{\big\langle(19,18), a_2b_2c_1d_1e_1f_1g_2, \textsc{cef}\big\rangle, \big\langle(16,24), a_2b_2c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle, \\ \big\langle(15,25), a_1b_1c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle\big\},$$

and $\mathcal{L}^{open}$ remains unchanged. In addition, label $\langle(5,18), a_1b_1c_1d_1e_2, \textsc{ce}\rangle$ is added to separator CE, thus resulting in message $\mathcal{M}_{CE}$ as described in Fig. 14. Assume the next label selected on line 06 is $\langle(8,2), a_1b_2, A\rangle$. Then this label is moved to separator CE. We thus compute $\mathcal{V} = \{\langle(8,2), a_1b_2, A\rangle\} \otimes \mathit{Labels}(ACE) \otimes \mathcal{M}_C$, where $\mathcal{M}_C$ is precisely the message described in Fig. 13, and the resulting label set $\mathcal{V}$ is equal to:

$$\mathcal{V} = \big\{\big\langle(13,14), a_1b_2c_1d_1e_1, \textsc{ace}\big\rangle, \big\langle(12,15), a_1b_2c_1d_1e_2, \textsc{ace}\big\rangle, \\ \big\langle(13,13), a_1b_2c_2d_2e_1, \textsc{ace}\big\rangle, \big\langle(18,13), a_1b_2c_2d_2e_2, \textsc{ace}\big\rangle\big\}.$$

Of course, $\mathcal{V}_{\Downarrow_{CE}} = \mathcal{V}$ and this set is appended to $\mathcal{L}^{open}$:

$\mathcal{L}^{open} =$

separator	*inst*	*vect*	*vect* + $\mathcal{H}$	*separator*	*inst*	*vect*	*vect* + $\mathcal{H}$
				CE	$a_1b_2c_1d_1e_1$	(13, 14)	(22, 21)
A	a_2b_3	(6, 2)	(24, 22)	CE	$a_1b_2c_1d_1e_2$	(12, 15)	(22, 22)
C	c_1d_3	(4, 1)	(24, 17)	CE	$a_1b_2c_2d_2e_1$	(13, 13)	(16, 19)
C	c_2d_3	(5, 3)	(22, 18)	CE	$a_1b_2c_2d_2e_2$	(18, 13)	(21, 19)

In addition, label $\langle(8,2), a_1b_2, A\rangle$ is added to message $\mathcal{M}_A$, and the contents of the messages are now those described in Fig. 14. Note that $\mathcal{L}^{Pareto}$ is unaffected.

And we execute again the while loop of lines 05–11. Let $\langle(12,15), a_1b_2c_1d_1e_2, \textsc{ce}\rangle$ be the next label selected on line 06. Moving this label to clique CEF produces a label set $\mathcal{V} = \{\langle(21,16), a_1b_2c_1d_1e_2f_1g_2, \textsc{cef}\rangle, \langle(22,22), a_1b_2c_1d_1e_2f_2g_2, \textsc{cef}\rangle\}$. Only the second element is thus added to $\mathcal{L}^{Pareto}$ on line 09, which becomes:

$$\mathcal{L}^{Pareto} = \big\{\big\langle(16,24), a_2b_2c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle, \big\langle(15,25), a_1b_1c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle, \\ \big\langle(22,22), a_1b_2c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle\big\}.$$

Note that label $\langle(19,18), a_2b_2c_1d_1e_1f_1g_2, \textsc{cef}\rangle$, which previously belonged to $\mathcal{L}^{Pareto}$ has been discarded from this set since it is dominated by $\langle(22,22), a_1b_2c_1d_1e_2f_2g_2, \textsc{cef}\rangle$. In addition, $\mathcal{L}^{open}$ is updated as follows (discarding elements with our new pruning rule):

$\mathcal{L}^{open} =$

separator	*inst*	*vect*	*vect* + $\mathcal{H}$	*separator*	*inst*	*vect*	*vect* + $\mathcal{H}$
A	a_2b_3	(6, 2)	(24, 22)	C	c_1d_3	(4, 1)	(24, 17)

Now, we enter the while loop again and select $\langle(6,2), a_2b_3, \textsc{a}\rangle$ to be moved. However, no element of set $\mathcal{V} = \{\langle(6,2), a_2b_3, \textsc{a}\rangle\} \otimes \mathit{Labels}(\textsc{ace}) \otimes \mathcal{M}_C$ is added to $\mathcal{L}^{open}$ because $ND_H(\mathcal{V}, \mathcal{L}^{Pareto}) = \emptyset$. Finally, there remains only label $\langle(4,1), c_1d_3, \textsc{c}\rangle$ to be moved. This creates a set $\mathcal{V}$ with 8 elements, out of which only 2 are kept due to our pruning rule:

$$\mathcal{L}^{open} = \begin{array}{|c|c|c|c||c|c|c|c|}\hline separator & inst & vect & vect+\mathcal{H} & separator & inst & vect & vect+\mathcal{H} \\ \hline CE & a_1b_2c_1d_3e_1 & (15,6) & (24,13) & CE & a_1b_2c_1d_3e_2 & (17,4) & (24,14) \\ \hline\end{array}$$

Moving $\langle(17,4), a_1b_2c_1d_3e_2, \textsc{ce}\rangle$ toward root will produce a new Pareto element of utility value (24, 14). As a consequence $\mathcal{L}^{open}$ becomes empty since label $\langle(15,6), a_1b_2c_1d_3e_1, \textsc{ce}\rangle$, when combined with its optimistic heuristic, is dominated by this new Pareto element. Therefore, $\mathcal{L}^{open}$ being empty, the execution of the algorithm is completed. The Pareto set thus computed is:

$$\mathcal{L}^{Pareto} = \big\{\big\langle(16,24), a_2b_2c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle, \big\langle(15,25), a_1b_1c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle, \\ \big\langle(22,22), a_1b_2c_1d_1e_2f_2g_2, \textsc{cef}\big\rangle, \big\langle(24,14), a_1b_2c_1d_2e_2f_2g_2, \textsc{cef}\big\rangle\big\}.$$

As we can see, the pruning rule is quite effective as it discards many labels that would have been combined without this rule: on overall, there were actually only 38 labels combinations instead of 78 for function `Pareto`.

4. Preference-based search

As mentioned above, in a multiagent or multicriteria problem, comparing feasible solutions in $\mathcal{X}$ amounts to comparing their respective utility profile. The basic preference model to compare solutions is Pareto dominance.

The results obtained in Section 3 show that the exact determination of the Pareto set requires, for some instances, prohibitive computation times (see, e.g., Proposition 1). Fortunately, determining the entire set of Pareto-optimal elements is not always necessary. For example, in multiagent problems, the value of a solution is often measured by a social welfare function assessing the overall utility of solutions for the society of agents. For example one can be interested in maximizing the sum of individual utilities (utilitarianism), or in maximizing the satisfaction of the least satisfied agent (egalitarianism) or any compromise between the two attitudes.

One major issue in multiagent decision making processes seeking approval of all agents is fairness of decision procedures. This normative principle generally refers to the idea of favoring solutions that fairly share happiness or utility among agents. More formally, when comparing two utility vectors u and v (one component per agent), claiming that "u is more fair than v" usually conveys the vague notion that the components of u are "less spread out" or "more nearly equal" than are the components of v. This intuitive notion leaves room for different definitions of fairness and various models have been proposed by mathematicians who developed a formal theory of majorization [35] and by economists who provided axiomatic foundations of inequality measures (for a synthesis see [36,37]). All these models provide the solution space with a transitive preference structure refining Pareto dominance, which is either a partial weak-order (e.g. Lorenz dominance) or a complete weak-order (e.g. ordered weighted averages). We will see now that the general algorithm $\mathtt{Pareto}^*_{\mathcal{H}}$ presented in Section 3.2.3 to determine the Pareto set can be further specialized to focus the search on fair compromise solutions.

The need for refining Pareto dominance is also present in single-agent multicriteria decision making problems. In such problems, the most preferred solutions are usually those achieving a good compromise between the various conflicting objectives involved in the decision model. We are generally not interested in generating extreme solutions favoring a particular criterion to the detriment of the others. The standard way of generating compromise solutions within the Pareto set is to optimize a "scalarizing function" measuring the overall quality of solutions by aggregation of criteria or, more generally, to define an overall preference model refining Pareto dominance and narrowing the initial optimality concept. When preference information is not sufficient to formulate a stable overall preference model, iterative compromise search can still be used to explore the Pareto set. One starts with a "neutral" initial preference model used to generate a well-balanced compromise solution within the Pareto set and the model progressively evolves with feedbacks from the decision maker during the exploration to better meet its desiderata. Such an interactive process is used in multiobjective programming on continuous domains to scan the Pareto set which is infinite, see e.g. [38,1]. The same approach is worth investigating in combinatorial problems when complete enumeration of the Pareto set is not feasible. This will be discussed in Section 4.4.

The common problem in all these situations is to determine the most-preferred solutions with respect to a given preference model $\succsim$ refining Pareto dominance. Hence, the rest of this section is dedicated to this general problem. We propose a refinement of $\mathtt{Pareto}^*_{\mathcal{H}}$ that exploits the GAI structure of utility functions to determine the most preferred solutions without resorting to complete enumeration of the Pareto set. For the sake of illustration, we will consider here 3 different models: on one hand Lorenz dominance and ordered weighted averages for fair optimization in multiagent decision making problems, on the other hand Tchebycheff distances for compromise search in multicriteria decision making problems. We will report numerical tests and provide computation times obtained for these models in Section 5.

4.1. Lorenz dominance

Lorenz dominance is a refinement of Pareto dominance used in fair optimization problems when utility functions $u^1, \ldots, u^m$ represent the preferences of m agents. In addition to the initial objective aiming at maximizing individual utilities, fairness refers to the idea of favoring Pareto-optimal solutions having a well-balanced utility profile. For this reason, in fair optimization problems, we are interested in working with a preference relation $\succsim$ satisfying the two following axioms:

P-Monotonicity. For all $u, v \in \mathbb{Z}_+^m$, $u \succsim_P v \Rightarrow u \succsim v$ and $u \succ_P v \Rightarrow u \succ v$,

where $\succ$ is the strict preference relation defined as the asymmetric part of $\succsim$. P-Monotonicity is a natural unanimity principle enforcing consistency with P-dominance.

Transfer Principle. Let $u \in \mathbb{Z}_+^m$ be such that $u^i > u^j$ for some i, j. Then for all ε such that $0 < \varepsilon < u^i - u^j$, $u - \varepsilon e_i + \varepsilon e_j \succ u$ where e_i (resp. e_j) is the vector whose ith (resp. jth) component equals 1, all others being null.

This axiom captures the idea of fairness as follows: if $u^i > u^j$ for some utility vector $u \in \mathbb{Z}_+^m$, slightly increasing component u^j to the detriment of u^i while preserving the sum of individual utilities would produce a better distribution of utilities and consequently improve the fairness of the solution. For example vector $u = (11, 10, 11)$ should be preferred to $v = (12, 9, 11)$ because there exists a transfer of size $\epsilon = 1$ to pass from v to u. Note that using a similar transfer of size greater than $12 - 9 = 3$ would increase inequality. This explains why the transfers must have a size $\varepsilon < u^i - u^j$. Such transfers are said to be *admissible* in the sequel. They are known as *Pigou–Dalton transfers* in social choice theory, where they are used to reduce inequality in the income distribution over a population (see [37] for a survey).

Note that the Transfer Principle enables to discriminate between some pair of vectors having the same sum of utilities, but it does not apply in the comparison of utility vectors having different sums. This is the reason why Transfer Principle must be combined with P-Monotonicity. For example, to compare $w = (11, 11, 11)$ and $z = (12, 9, 10)$ we can use vectors u and v introduced above and observe that $w \succ u$ (P-Monotonicity), $u \succ v$ (Transfer Principle explained above) and $v \succ z$ (P-Monotonicity). Hence $w \succ z$ by transitivity. In order to better characterize those vectors that can be compared using such combinations of P-Monotonicity and Transfer Principle, we recall now the definition of Lorenz vectors and related concepts (for more details see e.g. [35]):

Definition 7. For all $u \in \mathbb{Z}_+^m$, the *generalized Lorenz vector* associated to u is the vector:

$$L(u) = \left(u^{(1)}, u^{(1)} + u^{(2)}, \ldots, u^{(1)} + u^{(2)} + \cdots + u^{(m)}\right)$$

where $u^{(1)} \leqslant u^{(2)} \leqslant \cdots \leqslant u^{(m)}$ represent the components of $u = (u^1, \ldots, u^m)$ sorted by increasing order. The jth component of $L(u)$ is $L_j(u) = \sum_{i=1}^{j} u^{(i)}$.

Definition 8. The generalized Lorenz (weak) dominance relation on $\mathbb{Z}_+^m$ is defined for all $u, v \in \mathbb{Z}_+^m$, by $u \succsim_L v \Leftrightarrow L(u) \succsim_P L(v)$ and its strict part (called L-dominance hereafter) is defined by $u \succ_L v \Leftrightarrow L(u) \succ_P L(v)$.

The notion of L-dominance was initially introduced to compare vectors with the same average cost. The generalized version of L-dominance considered here is a classical extension allowing vectors with different averages to be compared. Within a set $U \subset \mathbb{Z}_+^m$, any utility vector u is said to be *L-dominated* when $v \succ_L u$ for some v in U, and *L-non-dominated* when there is no v in U such that $v \succ_L u$. The set on L-non-dominated vectors in U is denoted ND^L. In order to establish the link between generalized Lorenz dominance and preferences satisfying combination of P-Monotonicity and Transfer Principle we recall a result of Chong [39]:

Theorem 1. *For any pair of distinct vectors $u, v \in \mathbb{Z}_+^m$, if $u \succ_P v$, or if u obtains from v by a Pigou–Dalton transfer, then $u \succ_L v$. Conversely, if $u \succ_L v$, then there exists a sequence of admissible transfers and/or Pareto improvements to transform v into u.*

For example we have: $L(w) = (11, 22, 33) \succ_P (9, 19, 31) = L(z)$ which directly proves the existence of a sequence of Pareto improvements and/or admissible transfers passing from z to w. This theorem establishes L-dominance as the minimal transitive relation (with respect to set inclusion) satisfying simultaneously P-Monotonicity and Transfer Principle. Hence, the subset of L-non-dominated elements defines the best candidates to optimality in fair optimization problems. This explains our interest in solving the following problem:

LORENZ-OPTIMAL ELEMENTS (LO)

Input: a product set of finite domains $\mathcal{X} = X_1 \times \cdots \times X_n$ (n finite), m GAI utility functions $u^i : \mathcal{X} \to \mathbb{Z}_+$, $i = 1, \ldots, m$ (m finite),

Goal: determine the entire set of L-non-dominated vectors in $\mathbb{U}$, and for each utility vector $u \in ND^L(\mathbb{U})$ a corresponding tuple $x_u \in \mathcal{X}$.

Unfortunately, although the set of L-non-dominated elements is a subset of the Pareto set, it can still be sufficiently large to prevent any efficient enumeration as shown by the following result:

Proposition 12. *Problem LO is intractable, even when $m = 2$; it requires a number of operations which grows, in worst case, exponentially with the number of attributes.*

Proposition 13. *As soon as $|X_i| \geqslant 2$ and $m \geqslant 2$, deciding whether there exists a tuple in $\mathcal{X}$ the utility of which weakly L-dominates a given utility vector u in $\mathbb{Z}_+^n$ is a NP-complete decision problem (referred to as problem L_u in the sequel).*

Despite the apparent negative results of Propositions 12 and 13, we will see in Section 5 that, in practice, the average size of ND^L is small as compared to that of ND. This suggests that there might exist algorithms, efficient on average, to determine the set of L-non-dominated elements. The following subsection is dedicated to such an algorithm.

4.2. A focused search algorithm for L-non-dominated elements

We introduce now a modification of $\mathtt{Pareto}^*_{\mathcal{H}}$ search algorithm introduced in Section 3.2.3 to determine L-non-dominated elements in $\mathcal{X}$. Since these elements are necessarily Pareto-optimal we might first determine Pareto-optimal elements and then, by pairwise comparisons, determine the L-non-dominated elements. This procedure would not be efficient due to the size of the Pareto set. Instead, we prefer using a nice feature of $\mathtt{Pareto}^*_{\mathcal{H}}$ that computes Pareto optimal solutions one by one, thus leaving room for pruning rules based on L-dominance. Using such rule, we are going to specialize $\mathtt{Pareto}^*_{\mathcal{H}}$ to focus directly on Lorenz-optimal solutions. However this cannot be done naively as shown in the following example:

Example 3. Consider a GAI network with three Boolean attributes A, B, C and two cliques AB and BC. Assume that there are two Pareto-optimal solutions on clique AB: $(1,1)$ and $(0,1)$ with utility $u_{AB}(1,1) = (2,2)$ and $u_{AB}(0,1) = (3,1)$. We have $L(2,2) = (2,4)$ and $L(3,1) = (1,4)$. Hence we might be tempted to eliminate vector $(3,1)$ which is L-dominated by $(2,2)$ on AB and to send message $(0,1)$ with utility $(2,2)$ to the other clique BC. However this would be a mistake. Assume indeed that the only compatible Pareto-optimal vector on clique BC is $(1,0)$ with $u_{BC}(1,0) = (1,3)$ we would output solution $(1,1,0)$ with utility $(2,2) + (1,3) = (3,5)$ with $L(3,5) = (3,8)$ whereas there exists a better solution: $(0,1,0)$ with utility $(3,1) + (1,3) = (4,4)$ with $L(4,4) = (4,8)$.

This example shows that we cannot simply substitute Pareto dominance by L-dominance everywhere in a Pareto search algorithm to get an admissible algorithm for determining L-non-dominated elements. Lorenz dominance cannot be used to compare two labels located on a given separator and having the same partial instantiation over this separator (as was suggested for Pareto dominance by Corollary 2). It can only be used to prune labels by comparison with other labels corresponding to complete tuples already evaluated. We explain now the exact management of labels for the determination of L-non-dominated elements.

For determining Lorenz non-dominated elements, we will define the counterparts of labels' functions ND and ND_H for the Lorenz dominance (instead of the Pareto dominance). More formally, let $\mathbb{L}$ denote the set of all labels. We define a function $ND^L : \mathbb{L} \mapsto \mathbb{L}$ which, for any set of labels $\mathcal{V}$, returns a set $ND^L(\mathcal{V}) \subseteq \mathcal{V}$ containing one label per set of labels of $\mathcal{V}$ having the same generalized Lorenz non-dominated vector, i.e., for any $V = \langle v, x_{\mathbf{D}}, X_{\mathbf{E}}\rangle \in ND^L(\mathcal{V})$, there exists no $V' = \langle w, y_{\mathbf{F}}, X_{\mathbf{G}}\rangle \in \mathcal{V}$ such that $w \succ_L v$. In addition, we define $ND^L_H : \mathbb{L} \times \mathbb{L} \mapsto \mathbb{L}$ which, for any pair of label sets $(\mathcal{V}, \mathcal{W})$, returns a set $ND^L_H(\mathcal{V}, \mathcal{W}) \subseteq \mathcal{V}$ containing one label per set of labels of $\mathcal{V}$ having the same generalized Lorenz vector heuristically undominated by any generalized Lorenz vector of a label in $\mathcal{W}$. In other words, for any $V = \langle v, x_{\mathbf{D}}, X_{\mathbf{E}}\rangle \in ND^L_H(\mathcal{V}, \mathcal{W})$, there exists no $V' = \langle w, y_{\mathbf{F}}, X_{\mathbf{G}}\rangle \in \mathcal{W}$ such that $w \succ_L v + h$, where $\langle h, x_{\mathbf{H}}, X_{\mathbf{E}}\rangle$ is the only label that agrees with $x_{\mathbf{D}}$ in $\mathcal{H}_E$ (as defined in function `Heuristic_Collect`). Using, ND^L and ND^L_H, we can now provide the counterpart of function $\mathtt{Pareto}^*_{\mathcal{H}}$ for the Lorenz dominance. Function $\mathtt{Lorenz}^*_{\mathcal{H}}$ is identical to $\mathtt{Pareto}^*_{\mathcal{H}}$ except on lines 04, 09 and 10 where Lorenz dominance is used to discard complete instantiated labels that are dominated. Note in particular that function `move_label` remains unchanged and still uses Pareto dominance and Corollary 2 to prune labels.

The pruning rules using function ND^L of lines 04, 09 and 10 can be illustrated on a simple example: assume that we wish to add on line 10 label $\langle w = (8,3), a_i b_j, A\rangle$ into $\mathcal{L}^{open}$. In addition, assume that there exists in $\mathcal{L}^{Lorenz}$ a label $\langle u^* = (10,20), x, X_{\mathbf{C}_p}\rangle$ as shown in Fig. 9. Finally, assume that, for all the instantiations of the attributes of the gray area of Fig. 9 compatible with a_i and b_j, the corresponding utility vectors are Pareto dominated by $v = (4,4)$. Then operator ND, as used in the preceding section, cannot be exploited to prune w because $u^* = (10,20) \not\succsim_P w + v = (12,7)$. However, Lorenz dominance operator ND^L can be used to prune w because $L(u^*) = (10,30) \succsim_P L(v+w) = (7,19)$. This explains why, in practice, function $\mathtt{Lorenz}^*_{\mathcal{H}}$ is much faster than function $\mathtt{Pareto}^*_{\mathcal{H}}$. It is important to note that pruning a vector w using L-dominance can only be achieved through a comparison with a vector u^* corresponding to a complete assignment of the attributes. As shown in Example 3, we cannot extend this pruning rule to utility vectors u^* corresponding to only partial assignments of the attributes. As a consequence, in Algorithm 7, functions `initial_labels` and `move_label` must be the same as in Algorithm 4, i.e., they *must* use Pareto dominance, not Lorenz dominance. As in $\mathtt{Pareto}^*_{\mathcal{H}}$, there are different possible strategies to define what the most promising label should be. In our experiments, we simply defined it as nearest label to the `root` clique and, to break ties, that with the highest sum over the M objectives of the Lorenz vector of the sum of its utility vector and its heuristic vector.

Algorithm 7: Efficient best-first Lorenz search algorithm.

```
**Function** Lorenz$^*_{\mathcal{H}}$()
let root $X_{\mathbf{C}_p}$ be any clique
$\mathcal{L}^{open} \leftarrow \emptyset$; call initial_labels($X_{\mathbf{C}_p}, X_{\mathbf{C}_p}$)
**for all** Separators $X_{\mathbf{S}_{ji}}$ **do** call Heuristic_Collect($X_{\mathbf{C}_i}, X_{\mathbf{C}_j}$) **done**
$\mathcal{L}^{Lorenz} \leftarrow ND^L(\mathcal{M}_{pp})$; $\mathcal{L}^{open} \leftarrow ND^L_H(\mathcal{L}^{open}, \mathcal{L}^{Lorenz})$
**while** $\mathcal{L}^{open} \neq \emptyset$ **do**
    let $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}}\rangle$ be the most promising label in $\mathcal{L}^{open}$
    remove $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}}\rangle$ from $\mathcal{L}^{open}$
    $\mathcal{V} \leftarrow$ move_label($\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}}\rangle$)
    **if** $X_{\mathbf{S}_{ij}} =$ root $X_{\mathbf{C}_p}$ **then** $\mathcal{L}^{Lorenz} \leftarrow ND^L(\mathcal{L}^{Lorenz} \cup \mathcal{V})$; $\mathcal{L}^{open} \leftarrow ND^L_H(\mathcal{L}^{open}, \mathcal{V})$
    **else** $\mathcal{L}^{open} \leftarrow \mathcal{L}^{open} \cup ND^L_H(\mathcal{V}, \mathcal{L}^{Lorenz})$
**done**
return $\mathcal{L}^{Lorenz}$
```

Proposition 14. *Given a GAI tree $\mathcal{G}$, function* Lorenz$^*_{\mathcal{H}}$() *returns the Lorenz-optimal set.*

Proposition 15. Lorenz$^*_{\mathcal{H}}$() *requires space $O(km \times \prod_{i=1}^m K_i \times d^{w^*})$ and time $O(km \log m \times \prod_{i=1}^m K_i^2 \times d^{w^*+1})$, where k is the number of cliques in the GAI network, d is the largest attribute's domain size, w^* is the network's induced width* (*i.e., the number of attributes in the largest clique minus one*) *and K_i is a bound on utility u^i.*

4.3. Ordered weighted averages

Although L-dominance is a refinement of Pareto dominance used to capture an idea of fairness in comparisons, it is still a partial relation and as such, not always sufficient to discriminate between multiple feasible solutions, as shown by Proposition 12. This is the reason why several inequality measures have been proposed in the literature to refine Lorenz dominance. Among them, preference weak-orders induced by Ordered Weighted Averages (OWA) [40] appear as natural extensions of L-dominance. As shown in [41], under reasonable axioms such as compatibility with L-dominance, completeness of preferences, continuity and comonotonic independence, the only possible model is an ordered weighted average used with decreasing weights. This result is consistent with those of Ogryczak [42] that justify the use of OWA operators in equitable optimization. OWA operators are formally defined as follows:

Definition 9. The family of Ordered Weighted Averages (OWA) is a class of aggregators that assign weights to ranks and that perform a linear combination of scores, once they have been ranked. More formally, for any utility vector $u \in \mathbb{Z}_+^m$, the OWA is defined by:

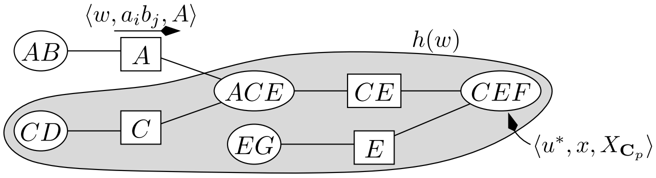
$$OWA(u) = \sum_{i=1}^m w_i u^{(i)} = \sum_{i=1}^m (w_i - w_{i+1}) L_i(u)$$

where $w_1 > w_2 > \cdots > w_m > w_{m+1} = 0$ and $u^{(1)} \leqslant u^{(2)} \leqslant \cdots \leqslant u^{(m)}$ represent the components of $u = (u^1, \ldots, u^m)$ sorted by increasing order.

Note that the most important weights are attached to least satisfied agents, consistently with the intuitive idea of egalitarianism. Note also that $OWA(u)$ can be expressed as a linear combination of Lorenz components $L_i(u)$, and the coefficients involved in the combination are strictly positive (the weights w_i strictly decrease as i increases). In this case, the OWA function obviously provides a weak-order refining L-dominance. Unfortunately, the determination of an OWA-optimal solution in $\mathcal{X}$ is NP-hard:

Proposition 16. *As soon as $|X_i| \geqslant 2$ and $m \geqslant 2$, the problem P_α consisting of deciding whether there exists an element $x \in \mathcal{X}$ of utility vector $u(x)$ such that $OWA(u(x)) \geqslant \alpha$, for a fixed positive integer α, is a NP-complete decision problem.*

The procedure used for Lorenz can easily be adapted to compute optimal OWA elements within $\mathcal{X}$. For computing the best element according to OWA, we just need to redefine two functions ND^{OWA} and ND^{OWA}_H, as we did for Lorenz. We thus define a function $ND^{OWA}: \mathbb{L} \mapsto \mathbb{L}$ which, for any set of labels $\mathcal{V}$, returns a set $ND^{OWA}(\mathcal{V}) \subseteq \mathcal{V}$ containing one label per set of labels of $\mathcal{V}$ having the same OWA, i.e., for any $V = \langle v, x_{\mathbf{D}}, X_{\mathbf{E}}\rangle \in ND^{OWA}(\mathcal{V})$, there exists no $V' = \langle w, y_{\mathbf{F}}, X_{\mathbf{G}}\rangle \in \mathcal{V}$ such that $w \succ_{OWA} v$, i.e., such that $OWA(w) > OWA(v)$. In addition, we define $ND^{OWA}_H: \mathbb{L} \times \mathbb{L} \mapsto \mathbb{L}$ which, for any pair of label sets $(\mathcal{V}, \mathcal{W})$, returns a set $ND^{OWA}_H(\mathcal{V}, \mathcal{W}) \subseteq \mathcal{V}$ containing one label per set of labels of $\mathcal{V}$ having the same OWA non-dominated by the OWA of any label's vector of $\mathcal{W}$. In other words, for any $V = \langle v, x_{\mathbf{D}}, X_{\mathbf{E}}\rangle \in ND^{OWA}_H(\mathcal{V}, \mathcal{W})$, there exists no $V' = \langle w, y_{\mathbf{F}}, X_{\mathbf{G}}\rangle \in \mathcal{W}$ such that $w \succ_{OWA} v + h$, where $\langle h, x_{\mathbf{H}}, X_{\mathbf{E}}\rangle$ is the only label that agrees with $x_{\mathbf{D}}$ in $\mathcal{H}_E$ (as defined in function Heuristic_Collect). Now, replace in function Lorenz$^*_{\mathcal{H}}$ the ND^L and ND^L_H by ND^{OWA} and ND^{OWA}_H respectively



$\langle w, a_ib_j, A\rangle$
$h(w)$
AB
A
ACE
CE
CEF
CD
C
EG
E
$\langle u^*, x, X_{\mathbf{C}_p}\rangle$

Fig. 15. Pruning rule for *OWA*: prune w whenever $OWA(u^*) \geqslant OWA(w + h(w))$.

and the resulting algorithm determines the optimal element w.r.t. OWA. The notion of a most promising label must be updated as well. We simply defined it as the nearest label to the `root` clique with the highest OWA value of the sum of its utility vector and its heuristic vector. The pruning rule with ND_H^{OWA} is illustrated in Fig. 15: for any label $\langle w, a_ib_j, A\rangle$, let $h(w)$ be a vector that Pareto dominates all the utility vectors corresponding to the instantiations of the attributes in the gray area; if $OWA(u^*) \geqslant OWA(w + h(w))$, then w can be safely discarded because it cannot be part of a solution of $ND^{OWA}(\mathbb{U})$.

4.4. Weighted Tchebycheff distances

The previous procedures were proposed in the context of fair multiagent decision making. We can easily adapt such procedures to single-agent but multicriteria decision problems. In this case, GAI utility functions $u^1, \ldots, u^m$ represent criteria defined from different subsets of attributes referring to different viewpoints about the solutions (e.g., security, velocity, space, aesthetics for a car, as already mentioned in the introduction). In this context, the notion of fairness is replaced by the notion of well-balanced compromise solution. In order to explore the possible compromise solutions in the Pareto set, a classical approach in multicriteria optimization is to generate Pareto-optimal solutions by minimizing the following scalarizing function (Wierzbicki [43]; Steuer and Choo [38]):

$$f_w(x) = \left\| w\big(\bar{u} - u(x)\big)\right\|_\infty = \max_{i\in M}\big\{w_i\big|\bar{u}^i - u^i(x)\big|\big\}$$

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 utility vector $\bar{u}$ on every component will receive a good score. This promotes well-balanced solutions. Function f_w fulfills two important properties [43]:

Property 1. If $\forall i \in M, w_i > 0$ then all solutions x minimizing f_w over the set $\mathcal{X}$ are weakly Pareto-optimal (i.e. no feasible solution can perform better on all criteria simultaneously). Moreover at least one of them is Pareto-optimal.

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

Property 1 shows that minimizing f_w yields at least one Pareto-optimal solution. Property 2 shows that any Pareto-optimal solution can be obtained with the appropriate choice of parameter 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 4. Consider a problem with 3 criteria 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 very bad with respect to at least one criterion. Thus t is the only reasonable compromise solution and it is Pareto-optimal; yet it cannot be obtained by maximizing a linear combination of individual utilities (with positive coefficients) because it does not belong to the boundary of the convex hull of feasible utility vectors.

Fig. 16 represents a feasible area and different Pareto-optimal compromise solutions that can be obtained by minimizing a weighted Tchebycheff distance, for different weights. Among them, only the filled points can be obtained by maximization of a linear combination of criteria.

Example 4 and Fig. 16 explain why f_w, as a scalarizing function, is preferred to a weighted sum in multiobjective optimization on non-convex sets [43,38]. This remark is important because the optimization of a weighted sum of criteria would have been an easier problem. Indeed, a weighted sum of GAI decomposable functions is still GAI decomposable. Hence optimizing a weighted sum of criteria amounts to finding the optimal tuple in a GAI network. As mentioned in [21] this problem can be solved with standard non-serial dynamic programming [12] as for the computation of the most plausible explanation (MPE) in Bayesian networks [13]. On the contrary, weighted Tchebycheff distances, as introduced above, are not GAI decomposable. In [44] we show that finding a solution x in $\mathcal{X}$ that minimizes the Tchebycheff criterion is a NP-hard problem and we propose a solution procedure. It relies on a ranking algorithm enumerating solutions according

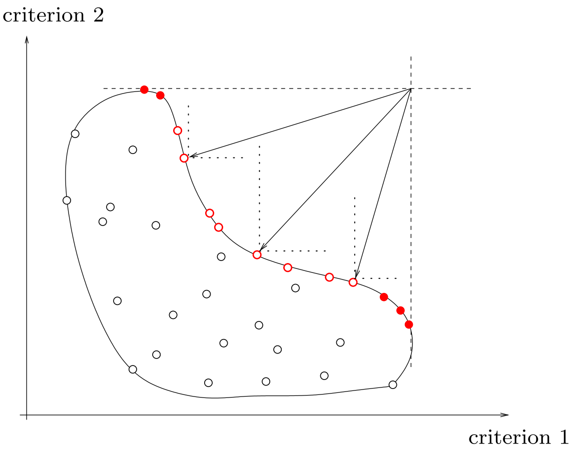


Fig. 16. Compromise solutions minimizing weighted Tchebycheff norms.

Table 1
Performance of our algorithms on biobjective problems derived from classical benchmarks.

File	n	w^*	$\text{Par}^*_{\mathcal{H}}$	$\text{Lor}^*_{\mathcal{H}}$	*OWA*	#L	#Par	File	n	w^*	$\text{Par}^*_{\mathcal{H}}$	$\text{Lor}^*_{\mathcal{H}}$	*OWA*	#L	#Par
GEOM30a_4	30	6	0.315	0.317	0.286	6	16	kbtree5_2_4_5_10_1	62	5	4.335	3.863	3.196	3	46
GEOM40_2	40	5	0.019	0.008	0.007	2	18	kbtree5_2_4_5_30_1	62	5	5.109	4.607	4.203	3	47
dubois30	90	3	0.054	0.046	0.041	1	38	kbtree5_2_4_5_50_1	62	5	3.530	3.136	2.783	5	42
dubois50	150	3	0.152	0.121	0.106	4	66	kbtree5_2_4_5_70_1	62	5	3.026	2.717	2.393	3	42
dubois100	300	3	0.751	0.637	0.568	1	115	kbtree5_2_4_5_90_1	62	5	3.692	3.322	3.072	4	49
pret150_25	150	8	1.032	0.998	0.823	1	55	cnf2.40.100.730621	40	11	0.746	0.730	0.672	3	16
pret150_40	150	8	0.750	0.705	0.597	6	56	cnf2.40.100.730623	40	12	1.630	1.622	1.433	5	10
pret150_75	150	9	2.048	1.842	1.644	2	57	cnf2.80.100.735545	80	6	0.050	0.044	0.038	1	16
hailfinder	56	4	29.302	28.553	27.808	34	169	cnf2.80.100.735549	80	6	0.038	0.034	0.030	2	17
insurance	27	8	28.279	28.271	28.151	1	58	alarm	37	4	0.172	0.150	0.128	23	92

to the weighted sum of criteria until a boundary condition is reached that guarantees that the optimal solution is found. We propose here an alternative approach based on our $\texttt{Pareto}^*_{\mathcal{H}}$ procedure. We first fix the components of the ideal point as $\bar{u}^i = \sup_{x\in\mathcal{X}} u^i(x) + 1$, $i = 1, \ldots, m$, each value $\sup_{x\in\mathcal{X}} u^i(x)$ being obtained by a monocriterion optimization using the GAI net. Then we implement a focused search procedure as for OWA, just by replacing OWA by the Tchebycheff criterion. This approach has been implemented and tested on random instances using a weighting vector w fixed so as to generate well-balanced compromise solutions within the Pareto set (see [1,45]). The experiments are presented in Section 5. Note that this approach easily generalizes to any aggregation function, provided it is monotone with respect to Pareto dominance.

5. Numerical tests

In order to evaluate the performance of our algorithms, we performed experiments on a 2 GB PC equipped with a 3.6 GHz Pentium 4 running the aGrUM[2] graphical model library. For the first set of experiments, we showed that our algorithms perform well on network structures found in practice. We thus used classical benchmarks available on http://carlit.toulouse.inra.fr/cflibtars. As our algorithms return exact – not approximate – solutions, we limited the experiments on networks with induced width $w^* \leqslant 15$. In the repository, benchmarks are mono-objective problems, hence we mapped them into multiobjective ones. To this end, for each objective we generated a utility decomposable according to the GAI network of the mono-objective problem. More precisely, for each utility u_i of the mono-objective problem, we computed its minimal and maximal values u_* and u^*, and we generated for each objective j a corresponding utility u_i^j by drawing random values between u_* and u^*. In the end, the multiobjective GAI network had thus the same structure as the mono-objective one. For biobjective problems, we evaluated the run times of $\texttt{Pareto}^*_{\mathcal{H}}$, $\texttt{Lorenz}^*_{\mathcal{H}}$ and *OWA* (performances for Tchebycheff distance are similar to *OWA*). The results are displayed in Table 1. In this table, n refers to the number of attributes, w^* to the induced width of the GAI network (as computed by our triangulation algorithm), columns #L and #Par show the number of Lorenz-optimal and Pareto-optimal elements respectively. All the other columns report average run times in seconds over 100 experiments.

As could be expected, $\texttt{Lorenz}^*_{\mathcal{H}}$ and *OWA* usually outperform $\texttt{Pareto}^*_{\mathcal{H}}$, essentially because the number of Lorenz undominated elements is much smaller than that of Pareto undominated ones. Note however that most of the attributes of

[2] See http://agrum.lip6.fr.

Table 2
Performance on biobjective problems when all attributes' domain sizes are 4.

File	*n*	w^*	$\text{Par}^*_{\mathcal{H}}$	$\text{Lor}^*_{\mathcal{H}}$	*OWA*	#L	#Par	File	*n*	w^*	$\text{Par}^*_{\mathcal{H}}$	$\text{Lor}^*_{\mathcal{H}}$	*OWA*	#L	#Par
GEOM30a_4	30	6	1.001	1.037	0.936	4	65	kbtree5_2_4_5_10_1	62	5	11.955	10.188	9.471	8	269
GEOM40_2	40	5	0.394	0.423	0.352	4	64	kbtree5_2_4_5_30_1	62	5	11.781	10.274	7.575	5	199
dubois30	90	3	11.616	10.911	9.904	4	310	kbtree5_2_4_5_50_1	62	5	9.099	7.516	7.035	8	304
dubois50	150	3	54.627	51.974	48.046	15	530	kbtree5_2_4_5_70_1	62	5	12.299	10.502	9.492	8	243
dubois100	300	3	1047.16	949.368	905.818	10	1451	kbtree5_2_4_5_90_1	62	5	9.955	8.169	7.074	3	254
pret150_25	150	8	–	–	–	–	–	cnf2.40.100.730621	40	11	–	–	–	–	–
pret150_40	150	8	–	–	–	–	–	cnf2.40.100.730623	40	12	–	–	–	–	–
pret150_75	150	9	–	–	–	–	–	cnf2.80.100.735545	80	6	–	–	–	–	–
hailfinder	56	4	25.888	23.067	21.095	5	178	cnf2.80.100.735549	80	6	–	–	–	–	–
insurance	27	8	–	–	–	–	–	alarm	37	4	2.734	2.643	2.416	7	103

Table 3
Run times for random problems with 5 criteria.

n	Par	$\text{Par}^*_{\mathcal{H}}$	$\text{Lor}^*_{\mathcal{H}}$	*OWA*	#L	#Par	Tcheb
10	1.519	0.451	0.237	0.196	7	2957	0.203
11	8.199	3.927	0.398	0.301	21	7134	0.326
12	31.512	11.995	7.506	7.143	5	8891	7.258
13	55.833	23.389	7.903	7.322	9	11 484	7.473
14	162.425	44.526	6.055	5.022	22	16 928	5.231
15	427.137	104.028	76.941	73.707	9	22 676	74.419
16	2050.512	105.577	60.467	56.931	5	33 334	58.092
19	–	1620.304	392.702	359.253	11	42 655	367.835
20	–	–	512.344	484.233	13	45 245	497.613

these instances are Boolean, which is seldom the case in decision problems. Significant exceptions are problems hailfinder and insurance, which are actually Bayesian networks with attributes of domain sizes up to 11. For such problems, we can see that run times are much higher than for the other experiments.

To test the behavior of our algorithms in a more decision-theoretic framework, we performed a second round of experiments using the same network structures as in Table 1 but now with all attributes of domain size 4. For each instance, we filled the utility tables with numbers drawn randomly between 0 and 20. The run times for biobjective problems are summarized in Table 2 ("–" indicate when the program failed due to the lack of memory space available). Note that, in such context, the Pareto sets and the run times are much bigger than those of Table 1.

Finally, for the last set of experiments, we studied how the algorithms behaved in the presence of multiple conflicting criteria (actually, we chose 5 criteria/objectives). The experiments of Tables 1 and 2 are not appropriate for this purpose because such problems are much harder to solve than biobjective ones and run times are too prohibitive. Hence, we randomly generated GAI networks with all cliques of size 3 and separators of size 2, and all attributes of domain size 4. For each clique, we generated 5 different utility tables (one for each objective) with numbers drawn randomly between 0 and 20. The run times of the algorithms (in seconds) are displayed in Table 3 ("–" indicate instances where the program failed due to a 2400 s timeout). Column Tcheb shows the times for Tchebycheff optimization (for a fixed set of weights). Note that $\texttt{Pareto}^*_{\mathcal{H}}$ significantly outperforms $\texttt{Pareto}$ (column "Par"). Note also the efficiency of $\texttt{Lorenz}^*_{\mathcal{H}}$, *OWA* and Tcheb compared with $\texttt{Pareto}^*_{\mathcal{H}}$ and $\texttt{Pareto}$. This shows that preference-based optimization can be performed even when the Pareto set cannot be computed.

6. Conclusion

We have shown that GAI networks can be used efficiently to handle preferences in decision problems involving multiple objectives, provided the objectives can be modeled by GAI decomposable utility functions. In particular, it is possible to store m different GAI functions into a single GAI network endowed with local vector-valued utility tables. Then we have proposed a heuristic search procedure exploiting the GAI structure to compute all Pareto-optimal elements. This procedure bears some similarity with labels propagation algorithms such as multiobjective *MOA** [30] used in state space graphs, but it works on a junction tree and must satisfy compatibility constraints induced by separators. The procedure we propose also bears some similarity with multiobjective approaches to constraint satisfaction problems [46–48] with some specificities linked to (i) the use of heuristic information and (ii) the management of labels candidate to expansion (they can belong to different cliques, we do not require to treat all Pareto-optimal labels of a given clique before propagation). These specificities make it possible to modify the initial procedure so as to determine the preferred solutions for any preference model compatible with Pareto dominance. This is not the case of ranking approaches proposed in [21] that only apply to concave utility functions.

We have provided various examples where our approach appears to be useful, first with Lorenz dominance and OWA models for fair multiagent decision making, and then with weighted Tchebycheff distances for multicriteria problems. Note that related approaches have been used successfully to perform fair optimization or compromise search in multiobjective shortest path problems [41,45]. This paper shows that such focused search algorithms specializing Pareto search can also be imported into Graphical Models to solve a wide range of multiobjective combinatorial optimization problems involving sophisticated preferences. Knowing that the size of the Pareto set can be huge in combinatorial domains, a useful complementary study might be to design near admissible algorithms to approximate the Pareto set with performance guarantees. Several recent works on multiobjective combinatorial problems have shown the power of approximations in solving large size instances [5,49,50]. It is likely that such ideas could be imported with benefit in the world of graphical multiobjective utility models. A first step in this direction is proposed in [51].

Appendix A. Proofs

Proof of Proposition 1. We establish the proof for $m = 2$ (biobjective case). The result obviously extends to problems involving more than two objectives. The decision problem P_u associated to PO is clearly in NP. To establish NP-completeness, we reduce the decision version of the Knapsack problem, known as NP-complete [52], to our problem. This problem denoted KP can be stated as follows:

Instance: a utility vector $(v_1, \dots, v_n) \in \mathbb{Z}_+^n$ and a weight vector $(w_1, \dots, w_n) \in \mathbb{Z}_+^n$ and two positive integers V and W.

Question: does there exist $x \in \{0, 1\}^n$ such that $\sum_{j=1}^n v_j x_j \geqslant V$ and $\sum_{j=1}^n w_j x_j \leqslant W$.

Given an instance of KP, we construct in polynomial time an instance of P_u with $u = (V, \sum_{j=1}^n w_j - W)$ as follows: we consider n Boolean attributes: $X_j = \{0, 1\}$, $j = 1, \dots, n$, such that, for all $x_j \in X_j$, $u^1(x_j) = v_j x_j$ and $u^2(x_j) = w_j(1-x_j)$. Thus, to any vector $x \in \{0, 1\}^n$ we associate a utility vector defined by $(u^1(x), u^2(x)) = (\sum_{j=1}^n v_j x_j, \sum_{j=1}^n w_j(1 - x_j))$. We know that the answer to KP is YES if and only if $\sum_{j=1}^n v_j x_j \geqslant V$ and $\sum_{j=1}^n w_j x_j \leqslant W$. By construction of the utility functions, these two inequalities are equivalent to $u^1(x) \geqslant V$ and $u^2(x) \geqslant \sum_{j=1}^n w_j - W$, meaning that the answer to P_u is YES. This shows that P_u is at least as hard as KP. □

Proof of Proposition 2. Assume that $u_1(x_{\mathbf{D}}) \succ_P u_1(y_{\mathbf{D}})$, then, by definition, for any $x_{\mathbf{E}} \in X_{\mathbf{E}}$, $u_1(x_{\mathbf{D}}) + u_2(x_{\mathbf{E}}) \succ_P u_1(x_{\mathbf{D}}) + u_2(x_{\mathbf{E}})$. Hence, no vector in $ND(\mathbb{U})$ can result from the addition of a vector $u_2(x_{\mathbf{E}})$ to $u_1(y_{\mathbf{D}})$. For the same reason, no vector in $ND(\mathbb{U})$ can result from the addition of a vector $u_1(x_{\mathbf{D}})$ to a dominated vector $u_2(x_{\mathbf{E}})$. As $\mathbb{U} = \{u(x_{\mathbf{D}}),\ x_{\mathbf{D}} \in X_{\mathbf{D}}\} \boxtimes \{u(x_{\mathbf{E}}),\ x_{\mathbf{E}} \in X_{\mathbf{E}}\}$ since $\mathbf{D} \cap \mathbf{E} = \emptyset$ and $\mathbf{D} \cup \mathbf{E} = \mathbf{N}$, the result obtains. □

Proof of Corollary 1. By the running intersection property (Property 3 of Definition 2), $\mathbf{D_1} \cap \mathbf{D_2} = \emptyset$. Hence $\mathbb{U} = \bigcup_{x_{\mathbf{S}_{12}} \in X_{\mathbf{S}_{12}}} (\{u_1(x_{\mathbf{D}_1}, x_{\mathbf{S}_{12}}),\ x_{\mathbf{D}_1} \in X_{\mathbf{D}_1}\} \boxtimes \{u_2(x_{\mathbf{D}_2}, x_{\mathbf{S}_{12}}),\ x_{\mathbf{D}_2} \in X_{\mathbf{D}_2}\})$. Now, when the values of the attributes $X_{\mathbf{S}_{12}}$ are fixed to, say, $x_{\mathbf{S}_{12}}$, $u_1(x_{\mathbf{D}_1}, x_{\mathbf{S}_{12}})$ and $u_2(x_{\mathbf{D}_2}, x_{\mathbf{S}_{12}})$ become subutilities defined over $X_{\mathbf{D}_1}$ and $X_{\mathbf{D}_2}$ respectively. As $\mathbf{D_1} \cap \mathbf{D_2} = \emptyset$, in this case, $u_1 + u_2$ is an additive utility and the application of Proposition 2 completes the proof. □

Proof of Corollary 2. The utility function u defined by the GAI network can be decomposed as $u = u_1 + u_2$, with $u_1 : X_{\mathbf{D}} \times X_{\mathbf{S}_{ij}} \mapsto \mathbb{Z}_+^m$ and $u_2 : X_{\mathbf{E}} \times X_{\mathbf{S}_{ij}} \mapsto \mathbb{Z}_+^m$ defined as $u_1(x_{\mathbf{D}}, x_{\mathbf{S}_{ij}}) = \sum_{t=1}^r u_t(x_{\mathbf{C}_{i_t}})$ and $u_2(x_{\mathbf{E}}, x_{\mathbf{S}_{ij}}) = \sum_{t=r+1}^k u_t(x_{\mathbf{C}_{i_t}})$. Now we are in the conditions of application of Corollary 1 and, thus, the result obtains. □

Proof of Proposition 3. Function `Pareto()` is completed in a finite number of steps else function `Pareto_Collect` would call itself an infinite number of times. But, by induction, it is easily seen that when `Pareto_Collect`$(X_{\mathbf{C}_k}, X_{\mathbf{C}_i})$ is called on line 03, clique $X_{\mathbf{C}_i}$ is "between" $X_{\mathbf{C}_k}$ and `root`, so that, as $\mathcal{G}$ is a tree, there can be only a finite number of calls of `Pareto_Collect`.

Now, let us prove by induction that `Pareto()` returns $ND(\mathbb{U})$. Consider first the leaves $X_{\mathbf{C}_i}$ of the GAI tree: `Pareto_Collect`$(X_{\mathbf{C}_i}, X_{\mathbf{C}_j})$ transforms u_i into label set $\mathcal{M}_{ij}$ and projects it on $X_{\mathbf{S}_{ij}}$, that is, it computes $\mathcal{M}_{ij \Downarrow X_{\mathbf{S}_{ij}}}$. By Corollary 2 applied on separator $X_{\mathbf{S}_{ij}}$, only the subutility vectors of u_i that are undominated for fixed values $x_{\mathbf{S}_{ij}}$ of $X_{\mathbf{S}_{ij}}$ need be taken into account for computing $ND(\mathbb{U})$. This corresponds precisely to label set $\mathcal{M}_{ij \Downarrow X_{\mathbf{S}_{ij}}}$. By induction hypothesis, assume that all the label sets $\mathcal{M}_{ki}$ of line 04 correspond to the undominated vectors described in Corollary 2 for fixed values of $X_{\mathbf{S}_{ki}}$. To apply Corollary 2 on separator $X_{\mathbf{S}_{ij}}$, we should compute, for each value $x_{\mathbf{S}_{ij}} \in X_{\mathbf{S}_{ij}}$, $\mathcal{V}_{x_{\mathbf{S}_{ij}}} = ND(\{\sum_{t=1}^r u_t(x_{\mathbf{C}_{i_t}}), x_{\mathbf{D}} \in X_{\mathbf{D}}\})$, where the $\mathbf{C}_{i_t}$, $t = 1, \dots, r$ are $X_{\mathbf{C}_i}$ and all the cliques having $X_{\mathbf{C}_i}$ on their path toward `root`, and where $\mathbf{D} = \bigcup_{t=1}^r \mathbf{C}_t \backslash \mathbf{S}_{ij}$. The for loop of lines 02–05 does not compute exactly $\{\sum_{t=1}^r u_t(x_{\mathbf{C}_{i_t}}), x_{\mathbf{D}} \in X_{\mathbf{D}}\}$ but rather a subset $\mathcal{W}_{x_{\mathbf{S}_{ij}}}$ where the discarded elements are those that are known to be dominated (by Corollary 2). Hence $ND(\mathcal{V}_{x_{\mathbf{S}_{ij}}}) = ND(\mathcal{W}_{x_{\mathbf{S}_{ij}}})$. So, each call to `Pareto_Collect` returns a set of labels that are undominated for each value of separator $X_{\mathbf{S}_{ij}}$.

Finally, for each clique $X_{\mathbf{C}_i}$, the loop of lines 02–05 parses all the neighbors of $X_{\mathbf{C}_i}$ except that which leads to `root`, hence the whole of the GAI net has been parsed when function `Pareto()` is completed. As a consequence, the labels in $\mathcal{M}_{pp}$

computed by `Pareto()` correspond to utility values u of complete instantiations. Moreover, by the recursive applications of Corollary 2, we know that $ND(\mathbb{U}) \subseteq \mathcal{M}_{pp}$. As the final step returns $ND(\mathcal{M}_{pp})$, function `Pareto()` returns u's Pareto set. □

Proof of Proposition 4. See the proof of Theorem 3 of [28]. □

Proof of Proposition 5. Consider a call to `initial_labels`$(X_{\mathbf{C}_i}, X_{\mathbf{C}_j})$ where $X_{\mathbf{C}_i}$ is a leaf of the GAI tree. Then on line 06, $\mathcal{V} = (Labels(X_{\mathbf{C}_i}))_{\Downarrow X_{\mathbf{S}_{ij}}}$. By applying Corollary 2 with separator $X_{\mathbf{S}_{ij}}$, we know that it cannot be the case that a label of $Labels(X_{\mathbf{C}_i}) \backslash \mathcal{V}$ be part of a Pareto element of $ND(\mathbb{U})$. On lines 07 and 08, $\mathcal{V}$ is partitioned into message $\mathcal{M}_{ij}$ and $\mathcal{W} = \mathcal{V} \backslash \mathcal{M}_{ij}$, the latter being added to $\mathcal{L}^{open}$. As a consequence, for any utility vector $u(x) \in ND(\mathbb{U})$, either $\langle u_i(x_{\mathbf{C}_i}), x_{\mathbf{C}_i}, X_{\mathbf{S}_{ij}} \rangle$ belongs to $\mathcal{W} \subseteq \mathcal{L}^{open}$ and Property 2 of the proposition holds (with $r = 1$ and $j_1 = i$), or $\langle u_i(x_{\mathbf{C}_i}), x_{\mathbf{C}_i}, X_{\mathbf{S}_{ij}} \rangle \in \mathcal{M}_{ij}$.

Now let $X_{\mathbf{C}_i}$ be a clique that is not a leaf. Let $X_{\mathbf{C}_{j_2}}, \ldots, X_{\mathbf{C}_{j_r}}$ denote the set of all the cliques that have $X_{\mathbf{C}_i}$ on their path toward clique $X_{\mathbf{C}_p}$. Clearly, `initial_labels`$(X_{\mathbf{C}_i}, X_{\mathbf{C}_j})$ recursively calls on line 03 `initial_labels`$(X_{\mathbf{C}_{j_t}}, X_{\mathbf{C}_{h_t}})$ for $t = 2, \ldots, r$, where $X_{\mathbf{C}_{h_t}}$ denotes the clique adjacent to $X_{\mathbf{C}_{j_t}}$ which is on the path between $X_{\mathbf{C}_{j_t}}$ and `root`. Assume by induction hypothesis that, for any vector $u(x) \in ND(\mathbb{U})$, one of the two following cases obtains:

(i) there exists a clique $X_{\mathbf{C}_{j_t}}$, $t \in \{2, \ldots, r\}$, such that `initial_labels`$(X_{\mathbf{C}_{j_t}}, X_{\mathbf{C}_{h_t}})$ created a new label in $\mathcal{L}^{open}$ satisfying Property 2 of Proposition 5,

(ii) there exists a label $\langle w, y_{\mathbf{D}}, X_{\mathbf{S}_{j_t h_t}} \rangle \in \mathcal{M}_{j_t h_t}$ such that $y_{\mathbf{D}} = x_{\mathbf{D}}$, i.e., a label that, when combined appropriately, will produce $\langle u(x), x, X_{\mathbf{C}_p} \rangle$.

Let us prove that, then, this will also hold for the set of cliques $X_{\mathbf{C}_i}, X_{\mathbf{C}_{j_2}}, \ldots, X_{\mathbf{C}_{j_r}}$. Let $u(x)$ be any element of $ND(\mathbb{U})$. If there exists a clique $X_{\mathbf{C}_{j_t}}$, $t \in \{2, \ldots, r\}$, such that (i) holds, then the result is obvious. Hence assume that, for all $t \in \{2, \ldots, r\}$, (ii) holds. In particular, it holds for the neighbors of clique $X_{\mathbf{C}_i}$, which means that, on line 04, each label set $\mathcal{M}_{ki}$ contains a label corresponding to a partial instantiation of x. So, at the end of the for loop of lines 02–05, $\mathcal{V}$ necessarily contains a label $\langle w, x_{\mathbf{D}}, X_{\mathbf{C}_i} \rangle$ such that $w = u_i(x_{\mathbf{C}_i}) + \sum_{t=2}^{r} u_{j_t}(x_{\mathbf{C}_{j_t}})$ and $\mathbf{D} = \mathbf{C}_i \bigcup_{t=2}^{r} \mathbf{C}_{j_t}$ because the labels corresponding to partial instantiations of x can be combined together. If label $\langle w, x_{\mathbf{D}}, X_{\mathbf{C}_i} \rangle$ were discarded on line 06, this would mean that it is dominated by another label for fixed value $x_{\mathbf{S}_{ij}}$ of separator $X_{\mathbf{S}_{ij}}$. But this is impossible because, by Corollary 2, this would imply that $u(x)$ is necessarily dominated and, thus, that $u(x) \notin ND(\mathbb{U})$. Consequently, either label $\langle w, x_{\mathbf{D}}, X_{\mathbf{C}_i} \rangle$ is inserted into message $\mathcal{M}_{ij}$ on line 07 or it is inserted into $\mathcal{L}^{open}$ on line 08.

The application of this induction up to clique $X_{\mathbf{C}_p}$ completes the proof. □

Proof of Proposition 6. First, `Pareto*()` executes a finite number of steps: clearly the call to `initial_labels`$(X_{\mathbf{C}_p}, X_{\mathbf{C}_p})$ ends in a finite number of steps since $\mathcal{G}$ is a tree. In addition, the number of elements it inserts into $\mathcal{L}^{open}$ is finite since the size of each message $\mathcal{M}_{ij}$ is the domain size of $X_{\mathbf{S}_{ij}}$. Each time we go through the while loop of lines 04–09, an element is removed from $\mathcal{L}^{open}$ on line 06, hence, if function `Pareto*()` did run infinitely, this would mean that an infinite number of new elements would be added to $\mathcal{L}^{open}$ on line 08. Now this is impossible because these elements, i.e., set $\mathcal{V}$, are those which result from a move of a given label. But then, by function `move_label`, these new labels are the only ones yet that combine $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ with other labels that are located on neighbors of clique $X_{\mathbf{C}_j}$. In other words, these new labels correspond to new partial instantiations of the attributes. As the number of possible partial instantiations is finite, `Pareto*()` terminates in a finite number of steps.

Note that a given label can never belong both to $\mathcal{L}^{open}$ and to a message $\mathcal{M}_{tl}$. This property clearly holds before the while loop because function `initial_labels` never inserts twice the same label on its lines 07 and 08. In the while loop of `Pareto*`, label $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ is removed from $\mathcal{L}^{open}$ before being added to $\mathcal{M}_{ij}$ by function `move_label`. Finally, label set $\mathcal{V}$ created on line 07 of `Pareto*` contains only new labels, as mentioned in the preceding paragraph. So we can add them to $\mathcal{L}^{open}$: they do not belong to any message $\mathcal{M}_{tl}$ yet.

Let us now prove that, at each step, one and only one of the assertions of Proposition 5 holds (where $\mathcal{M}_{pp}$ is substituted by $\mathcal{L}^{Pareto}$). Clearly, before the while loop, this holds. Let $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ be the label chosen on line 05. For all vectors $u(x) \in ND(\mathbb{U})$ such that there exists an agreeing label $\langle v, y_{\mathbf{E}}, X_{\mathbf{S}_{j_1 l}} \rangle$ as defined in assertion 2 of Proposition 5, with $j_1 \neq i$, then after executing lines 06–08, this label will still exist in $\mathcal{L}^{open}$ since the only line that removes labels from $\mathcal{L}^{open}$ is line 06 and the label removed cannot be $\langle v, y_{\mathbf{E}}, X_{\mathbf{S}_{j_1 l}} \rangle$ since $j_1 \neq i$. But, then, there cannot exist a label in $\mathcal{L}^{Pareto}$ corresponding to $u(x)$ because this label would correspond to a complete instantiation x and, thus, $\langle v, y_{\mathbf{E}}, X_{\mathbf{S}_{j_1 l}} \rangle$ would have been combined with other labels (which is not the case since it belongs to $\mathcal{L}^{open}$). Let now $u(x) \in ND(\mathbb{U})$ be a vector such that there exists only one agreeing label in $\mathcal{L}^{open}$ and this label is precisely that which is chosen on line 05, that is, $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$. For such label, using the notations of Fig. 7, `move_label`$(\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle)$ first computes on lines 01–06 the label set $\mathcal{V} = \{Labels(X_{\mathbf{C}_j}) \otimes \langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle \otimes \mathcal{M}_{k_1 j} \otimes \cdots \otimes \mathcal{M}_{k_r j}\}$ and, then, projects this set on line 07. It is easy to prove by induction that each message $\mathcal{M}_{k_t j}$, $t = 1, \ldots, r$, contains a label agreeing with $u(x)$ else $\langle w, x_{\mathbf{D}}, X_{\mathbf{S}_{ij}} \rangle$ would not be the only label agreeing with $u(x)$ in $\mathcal{L}^{open}$. Hence $\mathcal{V}$ necessarily contains a label agreeing with $u(x)$. As $u(x) \in ND(\mathbb{U})$, this label cannot be discarded on line 07 of `move_label`. So the set $\mathcal{V}$ on line 07 of `Pareto*` contains a label agreeing with $u(x)$ and, on the next line, it is either inserted into $\mathcal{L}^{Pareto}$ or $\mathcal{L}^{open}$, so that one and only one of the assertions of Proposition 5 holds again. Finally, consider a vector $u(x) \in ND(\mathbb{U})$ the label L of which belongs to $\mathcal{L}^{Pareto}$. Then label set $\mathcal{V}$ mentioned above cannot contain another label L' that agrees with $u(x)$ because this one would correspond to the same partial instantiation as L. But then, label L' would already have been combined with other labels to produce L, which is impossible since L' is a new label. Hence the property also holds in this case.

Now, to complete the proof, we know by the preceding paragraph that for each $u(x) \in ND(\mathbb{U})$, either there exists a label $\langle u(x), x, X_{\mathbf{C_p}}\rangle \in \mathcal{L}^{Pareto}$ or there exists a label in $\mathcal{L}^{open}$ that agrees with $u(x)$. When function Pareto* returns, $\mathcal{L}^{open}$ is empty, so $\mathcal{L}^{Pareto}$ contains the Pareto set and, by line 08, it is precisely equal to $ND(\mathbb{U})$. □

Proof of Proposition 7. Vectors on separators and within $\mathcal{L}^{open}$ are precisely those sent on the separators by function Pareto. Hence the space complexity of Pareto* is identical to that of Pareto. As for the time complexity, the combinations of sets of labels and their projections differ from Pareto only in the order in which they are done. In addition, labels selected on line 05 can be determined in $O(1)$. Hence Pareto* time complexity of is equal to that of Pareto. □

Proof of Proposition 8. Proof by induction. On the leaves, by construction, labels $\mathcal{H}_{ir}[x_{\mathbf{S}_{ir}}]$ obviously Pareto dominate the $u_r(x_{\mathbf{S}_{ir}})$. Now, the for loop of lines 02–05 computes $\mathcal{V} = Labels(X_{\mathbf{C}_i}) \otimes \mathcal{H}_{ir_1} \otimes \cdots \otimes \mathcal{H}_{ir_p}$, where $\{X_{\mathbf{C}_1}, \ldots, X_{\mathbf{C}_p}\} = \mathrm{Adj}(X_{\mathbf{C}_i}) \backslash X_{\mathbf{C}_j}$. Assume as induction hypothesis that each $\mathcal{H}_{ir_t}$ weakly Pareto dominates the sum of the subutilities of the cliques $X_{\mathbf{C}_s}$ such that $X_{\mathbf{C}_{r_t}}$ is on their path toward $X_{\mathbf{C}_i}$. Then, clearly, $\mathcal{V}$ weakly Pareto dominates the sum of the subutilities of $X_{\mathbf{C}_i}$ and of the cliques $X_{\mathbf{C}_s}$ such that $X_{\mathbf{C}_i}$ is on their path toward $X_{\mathbf{C}_j}$. Now, by definition of $\mathcal{M}\mathrm{ax}_{\downarrow X_{\mathbf{S}_{ij}}}$, the same property holds for $\mathcal{H}_{ji}$ as defined on line 06. □

Proof of Proposition 9. In a “usual” scalar collect algorithm, the space and time complexities are known to be $O(k \times d^{w^*})$ and time $O(k \times d^{w^*+1})$ respectively. Here, the only difference is that we do not manipulate scalars but vectors of size m. Hence the overall complexities. □

Proof of Proposition 10. The only difference between $\mathtt{Pareto}^*_{\mathcal{H}}$ and Pareto* is that the former prunes $\mathcal{L}^{open}$ using operator ND_H on lines 04, 09 and 10. But, by Proposition 8, the only labels that can be pruned are those that can only produce at the root Pareto dominated labels. Hence, discarding such labels cannot remove any element from $ND(\mathbb{U})$. As Pareto* was proved to return the Pareto set, $\mathtt{Pareto}^*_{\mathcal{H}}$ must return it as well. □

Proof of Proposition 11. There are fewer undominated vectors than $K = \prod_{i=1}^{m} K_i$. As a consequence, there are fewer possible labels on $\mathcal{L}^{open}$ and on separators than kKd^{w^*} because there are at most k separators and each separator's size is lower than d^{w^*+1} and, for each separator's value, there are fewer than K undominated vectors. Hence the space complexity.

As for the time complexity, if we do not take into account the prunings, we perform the same operations as Pareto*() except that we actually do them on fewer labels. So, the only difference lies in the additional domination tests. When the label moved on line 08 reaches the root, $\mathcal{L}^{Pareto}$ is updated on line 09, which means that we compare all pairs (x, y) where $x \in \mathcal{L}^{Pareto}$ and $y \in \mathcal{V}$. Thus, along the whole execution of the function, we cannot perform more than K^2 tests for each value of the root clique, hence a complexity of $O(mK^2 d^{w^*+1})$. On line 09, $\mathcal{L}^{open}$ is updated as well. Note that the labels in $\mathcal{L}^{open}$ are those that will be stored later on separators, hence the size of $\mathcal{L}^{open}$ never exceeds kKd^{w^*}. In addition, labels of $\mathcal{L}^{open}$ are always compared on line 09 with new labels, that is, labels that were not yet part of $\mathcal{L}^{Pareto}$. As a consequence, on overall, the time complexity of all these tests is $kK^2 d^{w^*+1}$. Finally, when $\mathcal{L}^{open}$ is updated on line 10, note that the elements in $\mathcal{V}$ are some of those sent by Pareto* on separator $X_{\mathbf{S}_{ij}}$. As a consequence, during the whole execution of the algorithm, fewer than kKd^{w^*} elements are stored successively in $\mathcal{V}$. As there are fewer elements than K in $\mathcal{L}^{Pareto}$, the time complexity obtains. □

Proof of Proposition 12. We consider instances of LO with two objectives ($m = 2$) on a set $\mathcal{X} = \prod_{j=1}^{n} X_j$, where $X_j = \{0, 1\}$, $j = 1, \ldots, n$. Assume that the objectives are additive utility functions defined, for any Boolean vector $x = (x_1, \ldots, x_n) \in \mathcal{X}$, by $u^i(x) = \sum_{j=1}^{n} u^i_j(x_j)$, $i = 1, 2$, where u^i_j is a marginal utility function defined on X_j by:

$$u^1_j(x_j) = 2^{j-1} x_j \quad \text{and} \quad u^2_j(x_j) = 2^j(1 - x_j) + \big(2^n - 1\big)/n, \quad j = 1, \ldots, n.$$

Then for all $x \in \{0, 1\}^n$, $u^1(x) = \sum_{j=1}^{n} 2^{j-1} x_j$ and $u^2(x) = 2\sum_{j=1}^{n} 2^{j-1}(1 - x_j) + 2^n - 1$. Let $z = \sum_{j=1}^{n} 2^{j-1} x_j$ we get: $u^1(x) = z$ and $u^2(x) = 2(2^n - 1) - 2z + 2^n - 1 = 3(2^n - 1) - 2z$. Hence there exist 2^n different Boolean vectors in $\mathcal{X}$, with distinct images in the utility space of the form $\{(z, 3(2^n - 1) - 2z),\ z \in \{0, \ldots, 2^n - 1\}\}$. Note that the second component is always greater than or equal to the first one for $z \in \{0, \ldots, 2^n - 1\}$. Consequently, the corresponding set of Lorenz vectors can be written as $\{(z, 3(2^n - 1) - z),\ z \in \{0, \ldots, 2^n - 1\}\}$. All these Lorenz vectors have their two components adding to $3(2^n - 1)$. Consequently, they are all located on a same line orthogonal to vector $(1, 1)$ which proves that all these vectors are Pareto-optimal. Hence all initial utility vectors are Lorenz-optimal which proves that $ND^L(\mathbb{U}) = \mathbb{U}$. Clearly, in such instances, the size of the set of L-non-dominated elements grows exponentially with the number of attributes, even if the number of criteria is fixed to 2. □

Proof of Proposition 13. The problem is clearly in NP. To establish NP-completeness, we reduce the partition problem, known as NP-complete [52], to our problem. The partition problem is stated as follows:

Instance: finite set $A = \{a_1, \ldots, a_m\}$ of items and a weight $s(a_i) \in \mathbb{N}$ for each $a_i \in A$.

Question: is it possible to partition A into two sets of objects of equal weights?

Let $u = (\beta, \beta)$ with $\beta = \sum_{a_i \in A} s(a_i)/2$. We construct in polynomial time an instance of L_u with $m = 2$ criteria and n Boolean attributes: $X_j = \{0, 1\}$, $j = 1, \ldots, n$, such that, for all $x_j \in X_j$, $u^1(x_j) = s(a_j)x_j$ and $u^2(x_j) = s(a_j)(1 - x_j)$. Thus, to any partition of A of type $(B, A \setminus B)$, $B \subseteq A$, we associate a Boolean vector $x^B \in \mathcal{X}$ with $n = |A|$ components ($x_i^B = 1$ if and only if $i \in B$). By construction, the image of x^B in the utility space is vector $(\sum_{a \in B} s(a), \sum_{a \in A \setminus B} s(a))$. Hence, the answer to L_u is YES if and only if the answer to the partition problem is YES. Indeed, if there is a solution to the partition problem, then there exists a partition with utility (β, β) and the corresponding Boolean vector in $\mathcal{X}$ is a solution of L_u. Moreover, if the answer to the partition problem is NO, then any partition of A into two subsets is unfair and the corresponding Boolean vector has a utility of type $(\beta - j, \beta + j)$, where j is a positive integer no greater than β. Since $L(\beta - j, \beta + j) = (\beta - j, 2\beta)$ and $L(u) = (\beta, 2\beta)$ we have $u \succ_L (\beta - j, \beta + j)$ and the answer to L_u is NO. □

Proof of Proposition 14. The only differences between $\texttt{Lorenz}^*_{\mathcal{H}}$ and $\texttt{Pareto}^*_{\mathcal{H}}$ lie on lines 04, 09 and 10 where Pareto dominance is substituted by Lorenz dominance. Clearly, replacing instructions $\mathcal{L}^{Pareto} \leftarrow ND(\mathcal{M}_{pp})$ and $\mathcal{L}^{Pareto} \leftarrow ND(\mathcal{L}^{Pareto} \cup \mathcal{V})$ by $\mathcal{L}^{Pareto} \leftarrow ND^L(\mathcal{M}_{pp})$ and $\mathcal{L}^{Pareto} \leftarrow ND^L(\mathcal{L}^{Pareto} \cup \mathcal{V})$ respectively cannot discard any element that is not Lorenz dominated. Had we only modified these two instructions, since Lorenz dominance is a refinement of Pareto dominance, function $\texttt{Lorenz}^*_{\mathcal{H}}$ would thus return the set of Lorenz-optimal elements. As for the heuristic, ND^L_H can be used anywhere in the algorithm as it prunes labels that we know for sure cannot be part of the solution. Hence $\texttt{Lorenz}^*_{\mathcal{H}}$ as described above returns the set of Lorenz-optimal labels. □

Proof of Proposition 15. The space complexity is the same as $\texttt{Pareto}^*_{\mathcal{H}}()$ as we store the same elements in $\mathcal{L}^{open}$ and in the separators. The time complexity is that of $\texttt{Pareto}^*_{\mathcal{H}}() \times \log m$ because, when we perform dominance tests, in addition to parsing vectors of size m, we also need to sort them, hence a complexity of $O(m \log m)$ instead of $O(m)$. □

Proof of Proposition 16. The proof is similar to the one of Proposition 13. The problem is clearly in NP. To establish NP-completeness, we reduce the partition problem, known as NP-complete [52], to our problem. Let $\alpha = (w_1 + w_2)\beta$ with $\beta = \sum_{a_i \in A} s(a_i)/2$. From any instance of partition (as introduced in the proof of Proposition 13) we construct in polynomial time an instance of P_α with $m = 2$ criteria and n Boolean attributes: $X_k = \{0, 1\}$, $k = 1, \ldots, n$, such that, for all $x_k \in X_k$, $u^1(x_k) = s(a_k)x_k$ and $u^2(x_k) = s(a_k)(1 - x_k)$. Thus, to any partition of A of type $(B, A \setminus B)$, $B \subseteq A$, we associate a Boolean vector $x^B \in \mathcal{X}$ with $n = |A|$ components ($x_i^B = 1$ if and only if $i \in B$). By construction, the image of x^B in the utility space is vector $(\sum_{a_i \in B} s(a_i), \sum_{a_i \in A \setminus B} s(a_i))$. Hence, the answer to P_α is YES if and only if the answer to the partition problem is YES. Indeed, if there is a solution to the partition problem, then there exists a partition with utility (β, β) and the corresponding Boolean vector in $\mathcal{X}$ gets the value $OWA(\beta, \beta) = w_1\beta + w_2\beta = \alpha$. Moreover, if the answer to the partition problem is NO, then any partition of A into two subsets is unfair and the corresponding Boolean vector has a utility of type $(\beta - k, \beta + k)$, where k is a positive integer not greater than β. Then we have $OWA(\beta - k, \beta + k) = w_1(\beta - k) + w_2(\beta + k) = \alpha - k(w_1 - w_2) < \alpha$ since $w_1 > w_2$. Hence the answer to P_α is NO. □

References

[1] R.E. Steuer, Multiple Criteria Optimization: Theory, Computation and Application, John Wiley, 1986.

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

[3] B. Roy, Multicriteria Methodology for Decision Analysis, Kluwer Academic Publishers, 1996.

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

[5] C.H. Papadimitriou, Y. Yannakakis, On the approximability of trade-offs and optimal access of web sources, in: Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS), 2000, p. 86.

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

[7] F. Bacchus, A. Grove, Graphical models for preference and utility, in: Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI), 1995, pp. 3–10.

[8] D. Braziunas, C. Boutilier, Local utility elicitation in GAI models, in: Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI), 2005, pp. 42–49.

[9] C. Gonzales, P. Perny, GAI networks for utility elicitation, in: Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR), 2004, pp. 224–234.

[10] C. Boutilier, F. Bacchus, R. Brafman, UCP-networks: A directed graphical representation of conditional utilities, in: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI), 2001, pp. 56–64.

[11] R. Brafman, C. Domshlak, T. Kogan, Compact value-function representations for qualitative preferences, in: Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence (UAI), 2004, pp. 51–59.

[12] U. Bertele, F. Brioschi, Nonserial Dynamic Programming, Academic Press, 1972.

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

[14] R. Dechter, Bucket elimination: A unifying framework for reasoning, Artificial Intelligence 113 (1999) 41–85.

[15] Th. Schiex, H. Fargier, G. Verfaillie, Valued constraint satisfaction problems: hard and easy problems, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1995, pp. 631–637.

[16] P. Hansen, Bicriterion path problems, in: G. Fandel, T. Gal (Eds.), Multicriteria Decision Making: Theory and Applications, in: Lecture Notes in Economics and Mathematical Systems, vol. 177, 1980, pp. 109–127.

[17] V. Emelichev, V. Perepelitsa, Multiobjective problems on the spanning trees of a graph, Soviet Mathematics Doklady 37 (1) (1988) 114–117.

[18] H. Hamacher, G. Ruhe, On spanning tree problems with multiple objectives, Annals of Operations Research 52 (1994) 209–230.

[19] G. Debreu, Continuity properties of Paretian utility, International Economic Review 5 (1964) 285–293.

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

[21] C. Gonzales, P. Perny, S. Queiroz, GAI networks: Optimization, ranking and collective choice in combinatorial domains, Foundations of Computing and Decision Sciences 32 (4) (2008) 3–24.

[22] F. Jensen, An Introduction to Bayesian Networks, Taylor and Francis, 1996.

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

[24] U. Kjærulff, Triangulation of graphs – algorithms giving small total state space, Tech. Rep. R-90-09, Department of Mathematics and Computer Science, Aalborg University, 1990.

[25] A. Darwiche, M. Hopkins, Using recursive decomposition to construct elimination orders, jointrees, and Dtrees, in: Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), 2001, pp. 180–191.

[26] F. van den Eijkhof, H.L. Bodlaender, Safe reduction rules for weighted treewidth, in: Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science, in: Lecture Notes in Computer Science, vol. 2573, Springer, 2002, pp. 176–185.

[27] D. Rose, Triangulated graphs and the elimination process, Journal of Mathematical Analysis and Applications 32 (1970) 597–609.

[28] E. Rollon, J. Larrosa, Bucket elimination for multiobjective optimization problems, Journal of Heuristics 12 (4–5) (2006) 307–328.

[29] L. Mandow, J.P. de la Cruz, A new approach to multiobjective A* search, in: Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI), 2005, pp. 218–223.

[30] B.S. Stewart, C.C. White III, Multiobjective A*, Journal of the Association for Computing Machinery 38 (4) (1991) 775–814.

[31] R. Carraway, T. Morin, H. Moskowitz, Generalized dynamic programming for multicriteria optimization, European Journal of Operational Research 44 (1) (1990) 95–104.

[32] C.C. White III, B.S. Stewart, R.L. Carraway, Multiobjective, preference-based search in acyclic OR-graphs, European Journal of Operational Research 56 (3) (1992) 357–363.

[33] R. Korff, W. Zhang, H. Hohwald, Frontier search, Journal of the Association for Computing Machinery 52 (5) (2005) 715–748.

[34] G. Shafer, Probabilistic Expert Systems, Society for Industrial and Applied Mathematics, 1996.

[35] W. Marshall, I. Olkin, Inequalities: Theory of Majorization and Its Applications, Academic Press, London, 1979.

[36] H. Moulin, Axioms of Cooperative Decision Making, Monograph of the Econometric Society, Cambridge University Press, 1988.

[37] A. Sen, On Economic Inequality, expanded edition, Clarendon Press, 1997.

[38] R. Steuer, E.-U. Choo, An interactive weighted Tchebycheff procedure for multiple objective programming, Mathematical Programming 26 (1983) 326–344.

[39] K.M. Chong, An induction theorem for rearrangements, Canadian Journal of Mathematics 28 (1976) 154–160.

[40] R. Yager, On ordered weighted averaging aggregation operators in multicriteria decision making, IEEE Transactions on Systems, Man and Cybernetics 18 (1998) 183–190.

[41] P. Perny, O. Spanjaard, L.-X. Storme, A decision-theoretic approach to robust optimization in multivalued graphs, Annals of Operations Research 147 (1) (2006) 317–341.

[42] W. Ogryczak, Inequality measures and equitable approaches to location problems, European Journal of Operational Research 122 (2000) 374–391.

[43] A. Wierzbicki, On the completeness and constructiveness of parametric characterizations to vector optimization problems, OR Spektrum 8 (1986) 73–87.

[44] C. Gonzales, P. Perny, S. Queiroz, Preference aggregation with graphical utility models, in: Proceedings of the 23rd AAAI Conference on Artificial Intelligence, 2008, pp. 1037–1042.

[45] L. Galand, P. Perny, Search for compromise solutions in multiobjective state space graphs, in: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI), 2006, pp. 93–97.

[46] E. Rollon, J. Larrosa, Multi-objective Russian doll search, in: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, 2007, pp. 249–254.

[47] S. Bistarelli, F. Gadducci, J. Larrosa, E. Rollon, A soft approach to multi-objective optimization, in: Proceedings of the 24th International Conference on Logic Programming (ICLP), in: Lecture Notes in Computer Science, vol. 5366, Springer, 2008, pp. 764–768.

[48] H. Fargier, N. Wilson, Local computation schemes with partially ordered preferences, in: Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), in: Lecture Notes in Computer Science, vol. 5590, Springer, 2009, pp. 34–45.

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

[50] P. Perny, O. Spanjaard, Near admissible algorithms for multiobjective search, in: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI), 2008, pp. 490–494.

[51] J.-P. Dubus, C. Gonzales, P. Perny, Multiobjective optimization using GAI models, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), 2009, pp. 1902–1907.

[52] M. Garey, D. Johnson, Computers and Intractability, W.H. Freeman and Company, 1979.