Using and Learning GAI-Decompositions for Representing Ordinal Rankings

Damien Bigot[1], **Hélène Fargier**[1], **Jérôme Mengin**[1], **Bruno Zanuttini** [2 3]

Abstract. We study the use of GAI-decomposable utility functions for representing ordinal rankings on combinatorial sets of objects. Considering only the relative order of objects leaves a lot of freedom for choosing a particular utility function, which allows one to get more compact representations. We focus on the problem of learning such representations, and give a polynomial PAC-learner for the case when a constant bound is known on the degree of the target representation. We also propose linear programming approaches for minimizing such representations.

1 INTRODUCTION

The development of interactive systems for supporting decision-making and of recommender systems highlighted the need for models capable of using a user's preferences to guide her choices. For over fifteen years, the modelling and compact representation of preferences have been active topics in Artificial Intelligence [15, 16, 5, 6, 11].

Existing formalisms are rich and flexible enough to describe the behaviour of complex decision rules. However, for being interesting in practice, these formalisms must also permit fast elicitation of a user's preferences, involving a reasonable amount of interaction only. Configuration of combinatorial products in *business-to-customer* problems [14] and preference-based search [18] are good examples of decision problems in which the user's preferences are not known *a priori*. In such applications, a single interaction with the user must typically last at most 0.25 s, and the whole session must typically last at most 20 minutes, even if the object to be recommended to the user is searched for in a combinatorial set.

When the user's preferences are qualitative and have a "simple" structure (for instance, when they are separable), conditional preference networks (CP-nets) and their variants [5, 4, 6] are popular representation frameworks. In particular, CP-nets come with efficient algorithms for finding most preferred extensions of objects (*outcome optimisation problem*), and with efficient elicitation procedures [13]. Unfortunately, CP-nets suffer a lack of expressivity, since most complete pre-orders cannot be represented by simple (acyclic) CP-nets.

Contrastingly, generalised additively independent decompositions (GAI-decompositions) of utility functions [9, 1, 11] can represent complete pre-orders in a compact way, by exploiting the independencies between sets of variables. In a word, these are representations of utility functions by sums of the form $\sum_{i=1}^{n} u_i(Z_i)$, where the u_i's are sub-utility functions with (hopefully) small scopes Z_i.

Typically, a GAI-decomposition is used to represent a *utility function*, which assigns a value to each possible object and hence, implicitly defines a complete pre-order on them (the greater the value, the more preferred the object). Such values may in some cases represent an amount of money which the user is ready to spend for the object, or may be defined implicitly by preferences on lotteries. However, in many applications, the actual values of the utility function are not important: it is the ranking of the objects that is induced by the utility function, and the properties of this ranking, that are interesting. Furthermore, the representation of the utility function is important too: it should enable fast answers for a variety of queries, not only dominance queries like "Is object o preferred to object o' ?", but also more complex queries like "What are the top-k objects that fulfil some given constraints?", useful for typical recommendation systems.

In this paper, we investigate the use of GAI-decompositions for representing such ordinal rankings. Precisely, we take a GAI-decomposition to represent the ranking defined by the associated utility function. Since in general many different utility functions represent the same ranking, this leaves a lot more freedom for finding compact decompositions than if the utility function is fixed.

In this context, we focus on the problem of learning a compact GAI-decomposition from (ordinal) examples, that is, essentially from statements of the form "I prefer object o to object o'". While some works on this topic have focused on a fixed target *utility function* (rather than a ranking) and assumed the structure (the scopes X_i) to be known in advance, we consider the issue of learning *any* utility function which induces the target *ranking*, and assume nothing about the target structure except for a constant, known bound on its degree. We aim at finding a simple structure, in order to ease optimization queries (among others). This issue has not been addressed in the "learning to rank" literature, where the aim is usually to find a ranking function that can be used to answer dominance queries, as in e.g. [12, 10, 8].

After a review of GAI-decompositions (Section 2), we show in Section 3 that the GAI-decompositions consistent with a set of examples can be defined as the feasible solutions of a linear program. We give an efficient PAC-learner for our problem in Section 4, and extend our approach to the problem of learning *minimal* decompositions in Section 5. Some perspectives and are discusset in Section 6.

2 GAI-DECOMPOSITIONS

In our context, the *preference relation* (or *preferences*) of a user on a set of objects χ is a complete pre-order $\succeq$, that is, a complete and transitive binary relation. Given two objects $o, o' \in \chi$, we take $o \succeq o'$ to mean that o is at least as interesting to the user as o'. We write $\succ$ for the asymmetric part of the relation $\succeq$, and $\sim$ for its symmetric

[1] IRIT, Univ. Toulouse, France; email: prenom.nom@irit.fr
[2] GREYC, Univ. Caen, France; email: prenom.nom@unicaen.fr
[3] Partially funded by the ANR (projet LARDONS, ANR-2010,BLAN-0215)

part. Hence $\succeq$ is a linear order with possible ties on χ, $\succ$ is the strict part of it, and $\sim$ contains the ties.

Generalized additive independence provides a representation for preferences on combinatorial domains. Hence we assume that the objects in χ are described over a set of n variables $X = \{X_1, \ldots, X_n\}$. We write D_i for the (finite) domain of X_i, hence the set of all objects is $\chi = D_1 \times \cdots \times D_n$. Though our results can be extended to arbitrary finite domains, for simplicity of exposition we consider *Boolean* domains, and we write $D_i = \{x_i, \bar{x}_i\}$. Slightly abusing notation, we also write objects of χ as sequences of values instead of as vectors. For instance, with $X = \{X_1, X_2, X_3, X_4\}$, the object $(x_1, \bar{x}_2, x_3, x_4) \in \chi$ will be denoted by $x_1\bar{x}_2x_3x_4$. Finally, for any subset of variables $Z \subseteq X$ and object o, $o[Z]$ denotes the projection of o onto the variables in Z, and given a set of objects $O \subseteq \chi$, $O[Z] = \{o[Z] \mid o \in D_1 \times \cdots \times D_n\}$ denotes the set of projections of elements of O onto Z.

It is easy to see that any complete and transitive preference relation $\succeq$ can be represented by a *utility function* $u : \chi \mapsto \mathbb{R}$ satisfying $o \succeq o' \Leftrightarrow u(o) \geq u(o')$ for all $o, o' \in \chi$. Clearly, since the set χ is combinatorial (it contains 2^n objects), it is impractical to directly elicit or explicitly store the relation $\succeq$ or a representation u. However, in some cases, the utility function u satisfies strong independency properties between attributes [7], so that it can be represented by a set of *local* utility functions $\{u_i : D_i \mapsto \mathbb{R} \mid i = 1, \ldots, m\}$ each of arity 1, satisfying $u(o) = \sum_{i=1}^{m} u_i(o[\{X_i\}])$ for all objects o. Such representations are clearly very compact, easy to elicit, and allow for efficiently computing optimal objects. Unfortunately, preference relations seldom satisfy this property of *additive independence*.

Example 1. *Consider the set of variables $X = \{X_1, X_2, X_3\}$ with $D_1 = \{\text{beef}(b), \text{fish}(f)\}$, $D_2 = \{\text{redWine}(r), \text{whiteWine}(w)\}$, $D_3 = \{\text{lemon}(l), \text{mustard}(m)\}$: χ contains 8 possible combinations. Consider the following ordering over χ:*

$$brm \succ brl \succ frm \succ frl \sim bwm \succ bwl \succ fwm \succ fwl$$

It can be represented with the additive utility function u defined by the following tables:

u_1 :

b	5
f	2

u_2 :

r	5
w	1

u_3 :

l	2
m	3

Consider now the following ordering:

$$brm \succ bwm \sim fwl \succ brl \succ fwm \sim frl \succ bwl \succ frm$$

To represent this ordering with an additive utility, we should have $u_1(b) > u_1(f)$ since brm is preferred to frm, and $u_1(b) < u_1(f)$ since fwl is preferred to bwl. However, this ordering can be represented using local utilities over several variables: define for any object o, $u(o) = u_{\{X_1,X_2\}}(o[\{X_1, X_2\}]) + u_{\{X_1,X_3\}}(o[\{X_1, X_3\}])$ where $u_{\{X_1,X_2\}}$ and $u_{\{X_1,X_3\}}$ are defined by:

$u_{\{X_1,X_2\}}$:

b, r	5
f, r	1
b, w	2
f, w	4

$u_{\{X_1,X_3\}}$:

b, l	3
b, m	7
f, l	5
f, m	2

Example 1 shows that some variables may depend on one another, and that in this case the utility function must be decomposed onto *sets* of variables rather than onto singletons.

Definition 1 (GAI-decomposition)**.** *Let $X = \{X_1, ..., X_n\}$ be a set of variables, $\chi = D_1 \times \cdots \times D_n$ be a set of objects, and $u : \chi \mapsto \mathbb{R}$ be a utility function on χ. A* GAI-decomposition *of u is a finite set $G = \{u_{Z_1}, \ldots, u_{Z_m}\}$ of utility functions on subsets Z_i of X (*i.e.*, $u_{Z_i} : \chi[Z_i] \mapsto \mathbb{R}$) such that $u(o) = \Sigma_{i=1}^{m} u_{Z_i}(o[Z_i])$ holds for all $o \in \chi$. The* degree *of G is defined to be $deg(G) = \max_{i=1,\ldots,m} |Z_i|$, where $|Z_i|$ denotes the cardinality of Z_i.*

Definition 2. *Let u be a utility function, and let G be a GAI-decomposition of u. Then u (resp. G) is said to* represent *a preference relation $\succeq$ iff $o \succeq o' \Leftrightarrow u(o) \geq u(o')$ for all $o, o' \in \chi$.*
Considering u (resp. G) as given, we also say that it induces *this relation and denote it by $\succeq_u$ (resp. $\succeq_G$).*

The local utility functions $u_{Z_1}, \ldots, u_{Z_m}$ are also called *GAI-tables*, because they are typically implemented in tabular form.

Clearly, for any utility function u, $\{u\}$ is a GAI-decomposition of u of degree $|X|$. Also, writing u_c for the constant function with value c, for any GAI-decomposition G of u and any set of variables $Z \subseteq X$, $G \cup \{u_0(Z)\}$ is also a GAI-decomposition of u. Similarly, $G \cup \{u_c(Z)\}$ is a GAI-decomposition of $u + c$, and hence induces the same preferences as G. However, the most interesting decompositions are those which properly refine u.

Definition 3 (utility-preserving refinement)**.** *Let G, G' be two GAI-decompositions of the same utility function u. Then $G = (u_{Z_1}, \ldots, u_{Z_m})$ is said to* u-refine *$G' = (u'_{Z'_1}, \ldots, u'_{Z'_{m'}})$ if for $i = 1, \ldots, m'$, there is $j \in \{1, \ldots, m\}$ with $Z'_i \subseteq Z_j$.*

Definition 4 (preference-preserving refinement)**.** *Let G, G' be two GAI-decompositions of utility functions u, u', respectively. Then $G = (u_{Z_1}, \ldots, u_{Z_m})$ is said to* refine *$G' = (u'_{Z'_1}, \ldots, u'_{Z'_{m'}})$ if $\succeq_G$ and $\succeq_{G'}$ are the same relation and for $i = 1, \ldots, m'$, there is $j \in \{1, \ldots, m\}$ with $Z'_i \subseteq Z_j$.*

For both definitions, the refinement is said to be *proper* if moreover, for one relation $Z'_i \subseteq Z_j$ as in the definition it holds $Z'_i \neq Z_j$, or for one Z_j there is no $Z'_i \subseteq Z_j$.

Refinement differs from u-refinement because the same preference relation can be represented by several utility functions. We will pay a particular attention to the maximally refined GAI-decompositions which represent a given preference relation.

Example 2. *Consider the set of boolean variables $X = \{X_1, X_2, X_3, X_4\}$. Let u be the GAI utility defined as the sum of $u_{X_1X_2}$, $u_{X_1X_3}$ and $u_{X_1X_4}$, where these sub-utilities are defined by the following tables:*

x_1x_2	9
$x_1\bar{x}_2$	5
$\bar{x}_1x_2$	5
$\bar{x}_1\bar{x}_2$	2

x_1x_3	8
$x_1\bar{x}_3$	9
$\bar{x}_1x_3$	6
$\bar{x}_1\bar{x}_3$	9

x_1x_4	5
$x_1\bar{x}_4$	2
$\bar{x}_1x_4$	4
$\bar{x}_1\bar{x}_4$	1

It can easily be checked that the order over χ induced by u is also induced by the utility u' defined as the sum of $u'_{X_1X_2}$ and $u'_{X_2X_3X_4}$ with the following tables:

x_1x_2	6
$x_1\bar{x}_2$	2
$\bar{x}_1x_2$	1
$\bar{x}_1\bar{x}_2$	0

$x_2x_3x_4$	3
$x_2x_3\bar{x}_4$	0
$x_2\bar{x}_3x_4$	7
$x_2\bar{x}_3\bar{x}_4$	1

$\bar{x}_2x_3x_4$	2
$\bar{x}_2x_3\bar{x}_4$	0
$\bar{x}_2\bar{x}_3x_4$	4
$\bar{x}_2\bar{x}_3\bar{x}_4$	1

Using a small program based on the ideas developed in the next section, we have checked that none of these two GAI-decompositions of the same pre-order can be refined.

This example shows that there is not always a unique maximally refined GAI-decomposition inducing a given pre-order.

3 REPRESENTATION OF EXAMPLES

Our aim in this paper is to learn GAI-decompositions which induce a hidden target preference relation. Hence in the following we assume that there is a set of Boolean variables $X = \{X_1, \ldots, X_n\}$, which defines a set of objects χ, and a target preference relation $\succeq$ on χ, hidden to the learner. The learner has access to information on $\succeq$ through examples.

Definition 5 (example)**.** *An example e of $\succeq$ is a triple of the form (o, R, o'), where $o, o' \in \chi$ and R is one of $\succ, \succeq, \sim, \preceq, \prec$.*
For a set of examples E, we write O_E for the set of all objects involved in at least one example of E.

Examples formalize the information received by the learner, especially by observing the user. For instance, if the learner observes that the user always chooses o over o', it may represent this as the example $(o, \succ, o')$. Similarly, if the user sometimes chooses o over o', and sometimes o' over o, this may be represented as the example $(o, \sim, o')$, etc.

Definition 6 (consistency)**.** *A GAI-decomposition G of a utility function u is said to be* consistent *with a set of examples E if for every example $(o, R, o') \in E$, $u(o) > u(o')$ (respectively $u(o) \geq u(o'), u(o) = u(o'), \ldots$) holds if R is the relation $\succ$ (respectively $\succeq, \sim, \ldots$).*

Clearly, given a constant k, there is not always a GAI-decomposition of degree k (or less) which is consistent with a given set of examples E. To formalize this, we define a set of examples E to be *k-sound* if there is at least one utility function u and a decomposition G of u, of degree at most k, that is consistent with E.

We now define a system of linear inequalities, whose solutions essentially correspond to the GAI-decompositions of degree k consistent with E.

Definition 7 (linear representation of an example)**.** *Let $e = (o, R, o')$ be an example of the target preference relation $\succeq$, and let $k \in \mathbb{N}$. Moreover, let $\sigma > 0$ be a real constant, positive but arbitrary. Finally, for all subsets of variables $Z \subseteq X$ with $0 < |Z| \leq k$ and assignments z to Z, let $U_{Z,z}$ be a formal variable.*

The linear inequality *for $e = (o, R, o'), k, \sigma$, written $ineq_k^\sigma(E)$ (or simply $ineq_k(E)$) is defined to be*

$$\sum_{Z \subseteq X, 0 < |Z| \leq k} U_{Z,o[Z]} \geq \sigma + \sum_{Z \subseteq X, 0 < |Z| \leq k} U_{Z,o'[Z]}$$

if R is the relation $\succ$, to be

$$\sum_{Z \subseteq X, 0 < |Z| \leq k} U_{Z,o[Z]} \geq \sum_{Z \subseteq X, 0 < |Z| \leq k} U_{Z,o'[Z]}$$

if R is the relation $\succeq$, and similarly for the relations $\sim$ (using $=$ in $ineq_k(E)$), $\preceq$ (using $\leq$), and $\prec$ (using $\leq$ and σ).

Definition 8 (linear system)**.** *Let E be a set of examples of the target preference relation $\succeq$, and let $k \in \mathbb{N}$, $\sigma > 0$. The* linear system *for E, k, σ is defined to be the conjunction of linear inequalities $\Sigma_k^\sigma(E) = \bigwedge_{e \in E} ineq_k^\sigma(e)$ (also written simply $\Sigma_k(E)$).*

Intuitively, variables $U_{Z,z}$ encode the components of the GAI-tables in a decomposition G of the target relation. We use a constant σ for strict preference with the aim of using linear programming, for which we need a closed topological space. Proposition 1 below shows that this is without loss of generality.

Importantly, note that the system $\Sigma_k(E)$ has at most $\sum_{i=0}^{k} 2^i \binom{n}{i}$ variables (as many as possible assignments to subsets of at most k variables). However, another bound is obtained by observing that the variable $U_{Z,z}$ appears only if there is an object $o \in O_E$ with $o[Z] = z$. Hence the number of variables occurring in $\Sigma_k(E)$ is at most $\sum_{i=0}^{k} \binom{n}{i} . |O_E|$. Whatever formula we use, provided k is bounded by a constant, the size of $\Sigma_k(E)$ is polynomial in the number of variables n and the number of examples E (using $|O_E| \leq 2|E|$).

Example 3. *Let $o = x_1 x_2 \bar{x}_3$ and $o' = \bar{x}_1 x_2 x_3$. The linear inequality associated with the exemple $e = (o, \succ, o')$ for $k = 2$ and $\sigma = 0.1$ is (writing, for exemple, $U_{x_1\bar{x}_2}$ for $U_{\{X_1,X_2\},x_1\bar{x}_2}$) :*

$$\begin{aligned} & U_{x_1} + U_{x_2} + U_{\bar{x}_3} + U_{x_1x_2} + U_{x_1\bar{x}_3} + U_{x_2\bar{x}_3} \\ \geq \quad & U_{\bar{x}_1} + U_{x_2} + U_{x_3} + U_{\bar{x}_1x_2} + U_{\bar{x}_1x_3} + U_{x_2x_3} + 0.1 \end{aligned}$$

We now show that the linear system $\Sigma_k(E)$ characterizes the GAI-decompositions of degree at most k and consistent with E. For technical reasons, we restricted ourselves to utility functions u with *span at least σ*, that is, satisfying $|u(o) - u(o')| \geq \sigma$ for all o, o' with $u(o) \neq u(o')$. This is without loss of generality however, since if u has span $\sigma_u < \sigma$, then u', defined by $u'(o) = \frac{\sigma}{\sigma_u} u(o)$ for all $o \in \chi$, is consistent with E as well and has span σ.

Proposition 1. *Let $\succeq$ be a preference relation on χ, let E be a set of examples for $\succeq$, and let $k \in \mathbb{N}$, $\sigma > 0$. Then the GAI-decompositions of degree at most k, span of at least σ, and consistent with E are exactly the solutions of $\Sigma_k(E)$.*

4 LEARNING

In this section we give an algorithm which, given a constant $k \in \mathbb{N}$ and a k-sound, hidden target preference relation $\succeq$, learns a GAI-decomposition G of $\succeq$ from examples only, in the *Probably Approximately Correct* (PAC) framework [17] (see Section 4.2). Our algorithm essentially maintains the *version space* of all GAI-decompositions of degree k (and span at least σ) consistent with the examples received so far, using a compact representation by $\Sigma_k(E)$.

4.1 VC-Dimension

So as to study the number of examples needed to learn $\succeq$, we first study the Vapnik-Chervonenkis dimension (VC-dimension for short) of the class G_k of all relations $\succeq$ which can be represented by a GAI-decomposition of degree at most k.

The VC-dimension concerns classes of binary concepts, that is, concepts c which assign one of two values to any object x (values $c(x)$ and $\neg c(x)$). Hence we view $\succeq$ as the two binary concepts $\succ$ and $\prec$ over objects $(o, o') \in \chi \times \chi$ (and $G_k^\succ, G_k^\prec$ denote the corresponding classes of concepts). This gives an equivalent view since, for instance, $o \succeq o'$ is equivalent to $o \not\prec o'$, $o \sim o'$ is equivalent to $o \not\prec o' \wedge o \not\succ o'$, etc. Intuitively, the VC-dimension of $\succ$ is the largest number of "independent" couples (o, R, o'), in the sense that the relation R of none depends on the relation of the others.

Definition 9 (VC-dimension)**.** *Let C be a set of binary concepts over $\chi \times \chi$. A set of couples $O \subseteq \chi \times \chi$ is said to be* shattered *by C if for any partition $\{O^+, O^-\}$ of O, there is a concept $c \in C$ satisfying $\forall (o, o') \in O^+, c(o, o')$ and $\forall (o, o') \in O^-, \neg c(o, o')$. The* VC-dimension *of C is the size of the largest set O that is shattered by C.*

We now give the VC-dimension of classes $G_k^{\succ}, G_k^{\prec}$. The fact that it is polynomial could not be taken for granted even for constant k, since *a priori* arbitrary values can occur in each entry of the GAI-tables.

Proposition 2. *The VC-dimension of $G_k^{\succ}$ (resp. $G_k^{\prec}$) is in $O(2^k n^{k+1})$, where n denotes the number of variables over which the objects are defined.*

Proof Let $K = \sum_{i=0}^{k} 2^i \binom{n}{i}$ (hence $K \in O(2^k n^{k+1})$). We show that no set $O \subseteq \chi \times \chi$ containing more than K couples (o, o') is shattered, from what the claim will follow. By duality, we give the proof for $G_k^{\succ}$.

So let $O \subseteq \chi \times \chi$ with $|O| \geq K + 1$. For all couples $(o, o') \in O$ we define the following formal sum, with variables $U_{Z,z}$ as in Definition 7:

$$V_{o,o'}^k = \sum_{Z \subseteq X, 0 < |Z| \leq k} U_{Z,o[Z]} - U_{Z,o'[Z]}$$

which corresponds to combining the rhs and lhs of any linear inequality associated to o and o'.

All sums $V_{o,o'}^k$ (for $(o, o') \in O$) use variables among the same K variables $U_{Z,z}$ ($0 < |Z| \leq k$). Hence if O contains at least $K + 1$ couples, there is at least one of them, which we write (ω, ω'), such that the sum $V_{\omega,\omega'}^k$ is a linear combination of the others, that is, there are values $\lambda_{o,o'}$ (for $(o, o') \in O \setminus \{(\omega, \omega')\}$) which satisfy

$$V_{\omega,\omega'}^k = \sum_{(o,o') \in O \setminus \{(\omega,\omega')\}} \lambda_{o,o'} V_{o,o'}^k$$

We write $O^{\leq}$ (resp. $O^{>}$) for the set of all couples $(o, o') \in O \setminus \{(\omega, \omega')\}$ with $\lambda_{o,o'} \leq 0$ (resp. $\lambda_{o,o'} > 0$).

First assume $O^{>} \neq \emptyset$. We show that no concept $\succ$ in $G_k^{\succ}$ is consistent with the partition defined by $O^+ = O^{>}$ and $O^- = O^{\leq} \cup \{(\omega, \omega')\}$, that is, no concept $\succ$ satisfies $o \succ o'$ for all $(o, o') \in O^{>}$, $o \not\succ o'$ (*i.e.*, $o \preceq o'$) for all $(o, o') \in O^{\leq}$, and $\omega \not\succ \omega'$. Indeed, given those labels and using Proposition 1, we get that the following linear system must be satisfied (for an arbitrary constant $\sigma > 0$):

$$\begin{aligned} V_{o,o'}^k &\geq \sigma \quad (\forall (o, o') \in O^{>}) \\ V_{o,o'}^k &\leq 0 \quad (\forall (o, o') \in O^{\leq}) \end{aligned}$$

Because of the signs of $\lambda_{o,o'}$'s, it follows that the following inequalities must be satisfied:

$$\begin{aligned} \lambda_{o,o'} V_{o,o'}^k &\geq \lambda_{o,o'} \sigma \quad (\forall (o, o') \in O^{>}) \\ \lambda_{o,o'} V_{o,o'}^k &\geq 0 \quad (\forall (o, o') \in O^{\leq}) \end{aligned}$$

Hence all solutions of this system must satisfy

$$\sum_{(o,o') \in O^{>} \cup O^{\leq}} \lambda_{o,o'} V_{o,o'}^k \geq \sigma \sum_{(o,o') \in O^{>}} \lambda_{o,o'}$$

that is (using $O^{>} \cup O^{\leq} = O \setminus \{(\omega, \omega')\}$),

$$V_{\omega,\omega'} \geq \sigma \sum_{(o,o') \in O^{>}} \lambda_{o,o'}$$

Because of $\sigma > 0$, $O^{>} \neq \emptyset$ and $\lambda_{o,o'} > 0$ for $(o, o') \in O^{>}$, it follows that $\omega \not\succ \omega'$ is impossible, so O is not shattered by $G_k^{\succ}$.

Now assume $O^{>} = \emptyset$. We show that no concept $\succ$ in $G_k^{\succ}$ is consistent with the partition defined by $O^+ = O^{\leq} \cup \{(\omega, \omega')\}$ and $O^- = \emptyset$. Indeed, reasoning as above we get:

$$\begin{aligned} V_{o,o'}^k &\geq 0 \quad (\forall (o, o') \in O^{\leq}) \\ \lambda_{o,o'} V_{o,o'}^k &\leq 0 \quad (\forall (o, o') \in O^{\leq}) \\ \sum_{(o,o') \in O^{\leq}} \lambda_{o,o'} V_{o,o'}^k &\leq 0 \\ V_{\omega,\omega'} &\leq 0 \end{aligned}$$

and hence, $\omega \succ \omega'$ is impossible, showing again that O is not shattered by $G_k^{\succ}$. Since O was arbitrary of size at least K, we conclude that the VC-dimension of $G_k^{\succ}$ is at most K. □

4.2 Algorithm

We now give an algorithm for learning a GAI decomposition of a hidden preference relation $\succeq^*$ accessed through examples. We use the *probably approximately correct learning (PAC learning)* framework proposed by Valiant [17]: the learner asks for a number m of examples (o, R, o') of the target relation $\succeq^*$, and computes a preference relation $\succeq$. The number m of examples in the sample is chosen by the learner as a function of the number of variables n and two real parameters $\varepsilon, \delta \in]0, 1[$. Each example is drawn at random according to a probability distribution D on $\chi \times \chi$; D is fixed but unknown to the learner. For any couple (o, o') drawn from $\chi \times \chi$, the learner receives the example (o, R, o'), where R is determined by $\succeq^*$ (without noise). In this context, an algorithm is a *PAC*-learner if

- it outputs a concept $\succeq$ which with probability at least $1 - \delta$ has error less than ε on couples drawn from $\chi \times \chi$ according to D. Formally, $\sum\{D(o, o') \mid oRo' \text{ but } \neg(oR^*o')\} < \varepsilon$ holds with probability at least $1 - \delta$, where R is any relation in $\{\succ, \sim, \prec\}$,
- the number m of examples asked by the learner is polynomial in $n, 1/\varepsilon, 1/\delta$,
- the algorithm runs in time polynomial in $n, 1/\varepsilon, 1/\delta$ (counting unit time for asking and receiving an example).

A concept class is said to be *efficiently PAC-learnable* if such a PAC-learner exists for the class.

The framework of PAC-learning captures situations where the learner observes some objects in its environment (those that come to it — it cannot choose which), and is given by a "teacher" the correct labels for these objects. Some objects occurring possibly more seldom than others (as formalized by the distribution D), the learner has less chances to learn with them, but it is less penalized by errors on them.

In order to show that for fixed, constant k, the class G_k of GAI-decompositions having degree at most k are PAC-learnable, we follow the classical *consistent learning* approach. The learner maintains a concept (in fact, the version space of all concepts) consistent with each of the examples received so far, namely, it maintains the linear system $\Sigma_k(E)$ (for a fixed but arbitrary span $\sigma > 0$).

Figure 1 depicts the algorithm. Since $\succeq^*$ is assumed to be k-sound and our setting is noise-free, the algorithm always returns a solution, *i.e.*, $\Sigma_k(E)$ is necessarily a consistent system. The number m of examples which the learner needs is given by the following proposition.

Proposition 3. *For any constant $k > 0$, the class G_k of GAI-decompositions of degree at most k is efficiently PAC-learnable. The number m of examples required by Algorithm* GAI-Learning *is in* $O(\max(\frac{1}{\varepsilon} \log \frac{1}{\delta}, \frac{2^k n^{k+1}}{\varepsilon} \log \frac{1}{\varepsilon})$.

Algorithm 1: The GAI-Learning Algorithm

begin
  $\Sigma_k(E) \leftarrow \emptyset$;
  for $i = 1, \ldots, m$ **do**
    ask for an example e of the form (o, R, o');
    add $ineq_k(e)$ to $\Sigma_k(E)$;
  compute a solution Sol of $\Sigma_k(E)$;
  return the GAI-decomposition encoded by Sol;
end

Proof. Proposition 1 shows that $\Sigma_k(E)$ is solvable, and that the concept returned by the algorithm is consistent with all the examples received.

We determine m using the VC-dimension of $G_k^{\succ}$ and $G_k^{\prec}$. Following [2], because the binary relations $\succ$ and $\prec$ uniquely determine the concept $\succeq$ (see the discussion before Definition 9), we define the "dimension" d of the class of non-binary concepts G_k to be the maximum of the VC-dimension of $G_k^{\succ}$ and $G_k^{\prec}$, hence $d \in O(2^k n^{k+1})$; intuitively, a learner learns $\succ$ and $\prec$ in parallel using the same examples for learning both, and deduces $\succeq$ (for more details we refer the reader to [2]).

Then we can apply the well-known generic result of Blumer et al. [3, Theorem 2.1 (ii)] to get that a number of examples $m \in O(\max(\frac{1}{\varepsilon}\log\frac{1}{\delta}, \frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$ is enough for any concept $\succeq$ consistent with the examples received to be probably approximately correct.

Finally, since m is polynomial in $n, 1/\varepsilon, 1/\delta$ (k is bounded), the size of $\Sigma_k(E)$ is polynomial. Since linear programming is polynomial, the proof is complete. □

5 MINIMIZING GAI-DECOMPOSITIONS

So far we have shown that for any constant k (small in practice), the class G_k of GAI-decompositions with degree at most k is PAC-learnable. However, our solution does not distinguish between a decomposition with degree k and one with degree $k' \ll k$, nor does it distinguish between one with t clusters of variables Z_i and one with $t' \ll t$ clusters, etc.

We now briefly discuss how such parameters can be optimized. The first natural objective is to learn a GAI-decomposition which is maximally u-refined, that is, which cannot be decomposed further while preserving the function.

Lemma 1. *Let u_Z be a utility function with a nontrivial GAI-decomposition $(u_{Z_1}, \ldots, u_{Z_m})$. Then $\sum_{z\in D(Z)} u_Z(z) > \sum_i \left(\sum_{z_i\in D(Z_i)} u_{Z_i}(z_i)\right)$ holds.*

Proof. We have by definition of a decomposition

$$\sum_{z\in D(Z)} u_Z(z) = \sum_{z\in D(Z)} \sum_i u_{Z_i}(z[Z_i]) = \sum_i \sum_{z\in D(Z)} u_{Z_i}(z[Z_i])$$

Now because there are $2^{|Z|-|Z_i|}$ assignments to Z which match z on Z_i, it follows:

$$\sum_{z\in D(Z)} u_Z(z) = \sum_i \sum_{z_i\in D(Z_i)} 2^{|Z|-|Z_i|} u_{Z_i}(z_i)$$

Finally, because of $Z_i \subseteq Z$ we have $2^{|Z|-|Z_i|} \geq 1$ for all i, and because the GAI-decomposition is not trivial, we have $2^{|Z|-|Z_i|} > 1$ for at least one i, and hence

$$\sum_{z\in D(Z)} u_Z(z) > \sum_i \sum_{z_i\in D(Z_i)} u_{Z_i}(z_i)$$

as desired. □

Corollary 1. *Let E be a k-sound set of examples. Then minimizing the objective function $\sum_{0<|Z|\leq k}\left(\sum_{z\in D(Z)} u_Z(z)\right)$, under the constraints in $\Sigma_k(E)$ plus the constraint $U_{Z,z} \geq 0$ (for all Z, z), yields a GAI-decomposition $(u_{Z_1}, \ldots, u_{Z_m})$ which is consistent with E and in which no u_{Z_i} can be u-refined.*

Proof. Observe that due to the nonnegativity constraint on $U_{Z,z}$'s, the minimum is well-defined. Now towards a contradiction, let $G^* = (u^*_{Z_1}, \ldots, u^*_{Z_m})$ be an optimal solution, and let (wlog) $G_1 = (u_{Z_{11}}, \ldots, u_{Z_{1p}})$ ($Z_{1i} \subseteq Z_1$) be a nontrivial u-refinement of u_{Z_1}. Define G to be obtained from G^* by replacing $u^*_{Z_1}$ with G_1. Then clearly G is a utility-preserving refinement of G^*, hence both represent the same utility function. It follows that G is also consistent with E and also a feasible solution of the program (in particular, it has the same span $\geq \sigma$). Now by Lemma 1, G has a better value than G^*, which contradicts the optimality of G^*. □

Let $D(Z, E)$ denotes the set of the $z \in D(Z)$ such that there si some $o \in O_E$ with $z = o[Z]$: obviously, for every $z \in D(Z)$, if $z \notin D(Z, E)$ then the minimization will yield $U_{Z,z} = 0$. Therefore, the linear program of Corollary 1 can be expressed as follows:

$$\text{(P1)} \begin{cases} \text{minimize} \displaystyle\sum_{Z\subseteq X, 0<|Z|\leq k, z\in D(Z,E)} U_{Z,z} \\ \text{under constraints} \\ \bullet\ ineq_k(e) \text{ for every } e \in E \\ \bullet\ U_{Z,o[Z]} \geq 0 \text{ for every } Z, o \end{cases}$$

However, though (P1) achieves some kind of minimality, it does not tend to minimize the number of nonempty entries (nonnull values) in the tables, as the following example shows.

Example 4. *Consider two boolean variables X_1, X_2, and let $E = \{x_1x_2 \succ x_1\bar{x}_2\ ,\ x_1\bar{x}_2 \succ \bar{x}_1\bar{x}_2\ ,\ \bar{x}_1\bar{x}_2 \succ \bar{x}_1x_2\}$. Fix $k = 2$, $\sigma = 1$, and consider the following decompositions (u_1, u_{12}) and (u'_{12}), which are both consistent with E:*

u_1 :

x_1	1
$\bar{x}_1$	0

u_{12} :

x_1x_2	2
$x_1\bar{x}_2$	1
$\bar{x}_1\bar{x}_2$	1
$\bar{x}_1x_2$	0

u'_{12} :

x_1x_2	3
$x_1\bar{x}_2$	2
$\bar{x}_1\bar{x}_2$	1
$\bar{x}_1x_2$	0

The decomposition (u'_{12}) has fewer non-zero entries than (u_1, u_{12}), however it can be seen that the latter has a better value for (P1) than the former.

Despite this, we can apply some efficient post-processing to the solution computed for (P1), by reporting the values in the table of Z_i to the table of $Z_j \supset Z_i$, if any.

Clearly, minimizing the number of nonempty entries allows for more efficient storage of the decomposition learnt. In order to minimize this number over all possible GAI-decompositions consistent with the example, one has to resort to mixed-integer programming, using an additional set of 0/1 variables of the form $V_{Z,z}$ (recording whether the entry $u_Z(z)$ is nonempty). This yields the following program (using standard constructs):

$$\text{(P2)} \begin{cases} \text{minimize} \sum_{Z \subseteq X, 0<|Z| \leq k, z \in D(Z,E)} V_{Z,z} \\ \text{under constraints} \\ \bullet\ ineq_k(e) \text{ for every } e \in E \\ \bullet\ U_{Z,o[Z]} \geq 0 \text{ for every } Z, o \\ \bullet\ V_{Z,o[Z]} \geq U_{Z,o[Z]} \text{ for every } Z \subseteq X, 0 < |Z| \leq k, o \in O_E \\ \bullet\ V_{Z,o[Z]} \in \{0,1\} \text{ for every } Z \subseteq X, 0 < |Z| \leq k, o \in O_E \end{cases}$$

Note that the constant σ must be small enough so that constraining the $U_{Z,o[Z]}$s to be in the $[0,1]$ interval does not artificially eliminate some solutions.

Finally, a natural objective is to minimize the degree of the GAI-decomposition learnt (given that it will be at most k anyway).

Example 5. *Consider three boolean variables X_1, X_2, X_3, let $k = 3$, $\sigma = 1$, and $E = \{x_1x_2x_3 \succ \bar{x}_1x_2x_3\ ,\ \bar{x}_1x_2\bar{x}_3 \succ \bar{x}_1\bar{x}_2\bar{x}_3\}$. Consider the decompositions (u_{123}) and (u'_1, u'_2) defined as follows:*

u_{123} :

$x_1x_2x_3$	1
$\bar{x}_1x_2\bar{x}_3$	1
else	0

u'_1 :

x_1	1
$\bar{x}_1$	0

u'_2 :

x_2	1
$\bar{x}_2$	0

Both decompositions are consistent with E. Moreover, (u_{123}) is clearly an optimum of (P1) and of (P2), but it does not have minimal degree, since (u'_1, u'_2) has degree 1.

Again, one could resort to mixed-integer programming to minimize the degree over all decompositions consistent with the examples. Nevertheless, it is clearly a more efficient approach to proceed by exhaustive search (or by dichotomy): if there is a decomposition of degree k, then look for one with degree $k-1$, etc.

6 CONCLUSION

In this paper, we have shown that any complete preorder on a combinatorial domain, which can be represented by a GAI-decomposition of degree k, can also be seen as the solution of a system of linear equations. For a given k, a set of examples of such a hidden preorder (encoding a preference relation) leads to a linear system whose variables encode the components of the GAI-tables. A judicious choice of an objective function allows to get a minimal decomposition (with various notions of minimality).

With this in hand, we designed an algorithm which learns a GAI-decomposition of a hidden preference relation, provided a constant bound on the degree of such a decomposition is known *a priori*, in the framework of PAC-learning. To our knowledge, this is the first algorithm able to learn GAI-decompositions without being given the structure of the tables (the sets of variables Z_i). Even if we require a constant bound on the degree to be given, the result could not be taken for granted, since the presence of arbitrary values in the GAI-tables makes *a priori* GAI-decompositions of degree k a very expressive class (hence difficult to learn). On the practical side, requiring a small, constant bound typically fits in the applicative context, where (human) users usually have very local preferences.

When the training set is noisy or the chosen bound k is too low, the linear system has no solution. It is simple to relax the system by introducing a *slack variable* δ_e in the left part of each inequality $ineq_k(e)$. Then the sum of the δ_i's is obviously to be minimized (the variant of the algorithm given in the paper corresponds to null δ_i's).

The next development of this work is the design of a good strategy for the choice of the degree of the GAI to be learnt. A simple approach is to follow an increasing strategy, first assuming that variables are independent, then setting $k = 2$ and so on, until a good coverage of the examples is reached. A finer strategy would be to use a tolerance parameter and to analyze the (imperfect) graph learnt for $k = 2$ in order to have a better idea of the dependencies: the knowledge of a clique on a set Y of variables leading to the necessity of the local utility function u_Y. More generally, such further development would imply interleaving two procedures, one being devoted to learning k (or the structure of the GAI), and the other to learning the entries in the tables using linear programming.

Last, but not least, we shall go back to the development of (active) elicitation methods close to the ones used in CP-net learning. The idea is to ask the user a series of question, whose answer allows the learner to infer (in)dependencies between variables [13]. In this context, the equations and variables of the linear system would show up only when necessary, leading to a much more efficient procedure in terms of memory.

Acknowledgments We thank all anonymous reviewers of ECAI 2012 for helpful comments.

REFERENCES

[1] Fahiem Bacchus and Adam Grove, 'Graphical models for preference and utility', in *Proceedings of UAI'95*, pp. 3–10, (1995).
[2] Shai Ben-David, Nicolò Cesa-Bianchi, David Haussler, and Philip M. Long, 'Characterizations of learnability for classes of {0,...,n}-valued functions', *J. Of Computer and System Sciences*, **50**, 74–86, (1995).
[3] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth, 'Learnability and the vapnik-chervonenkis dimension', *J. ACM*, **36**(4), 929–965, (1989).
[4] Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman, 'Ucp-networks: A directed graphical representation of conditional utilities', in *Proceedings of UAI'01*, (2001).
[5] Craig Boutilier, Ronen Brafman, Holger Hoos, and David Poole, 'Reasoning with conditional ceteris paribus preference statem', in *Proceedings of UAI-99*, pp. 71–80, San Francisco, CA, (1999).
[6] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole, 'Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements', *JAIR*, **21**, 135–191, (2004).
[7] G. Debreu, 'Topological methods in cardinal utility theory', in *Mathematical Methods in the Social Sciences*, eds., K.J. Arrow, S. Karlin, and P. Suppes, 16–26, Stanford University Press, Stanford, (1960).
[8] Carmel Domshlak and Thorsten Joachims, 'Unstructuring user preferences: Efficient non-parametric utility revelation', in *Proc. of UAI '05*, pp. 169–177, (2005).
[9] P.C. Fishburn, *Utility Theory for Decision Making*, Wiley, New York, 1970.
[10] Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer, 'An efficient boosting algorithm for combining preferences', *Journal of Machine Learning Research*, **4**, 933–969, (2003).
[11] Christophe Gonzales and Patrice Perny, 'Gai networks for utility elicitation', in *Proceedings of KR'04*, pp. 224–234, (2004).
[12] Thorsten Joachims, 'Optimizing search engines using clickthrough data', in *Proc. of ACM KDD'02*, pp. 133–142.
[13] Frédéric Koriche and Bruno Zanuttini, 'Learning conditional preference networks with queries', in *Proc. IJCAI'09*, pp. 1930–1935, (2009).
[14] Daniel Mailharro, 'A classification and constraint-based framework for configuration', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 383–397, (September 1998).
[15] Anand S. Rao and Michael P. Georgeff, 'Modeling rational agents within a bdi-architecture', in *Proc. KR'92*, pp. 473–484, (1991).
[16] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie, 'Valued constraint satisfaction problems: Hard and easy problems ', in *Proceedings of IJCAI'95)*, pp. 631–637. Morgan Kaufmann, (aot 1995).
[17] Leslie G. Valiant, 'A theory of the learnable', *Communications of the ACM*, **27**(11), 1134–1142, (1984).
[18] Paolo Viappiani, Boi Faltings, and Pearl Pu, 'Preference-based search using example-critiquing with suggestions', *JAIR*, **27**, 465–503, (2006).