Réseaux GAI pour la prise de décision

Christophe Gonzales — Patrice Perny — Sergio Queiroz

LIP6–Université Paris 6
104 Avenue du président Kennedy, F-75016 Paris
prénom.nom@lip6.fr

RÉSUMÉ. *Cet article traite de l'élicitation de préférences et de la recommandation (choix et rangement) dans le contexte de la théorie de l'utilité de multi-attribut. Nous nous concentrons sur le modèle des utilités GAI décomposables (additivité généralisée) qui permet des interactions entre les attributs tout en préservant une certaine décomposabilité du modèle. Nous présentons d'abord une procédure systématique d'élicitation pour de telles fonctions d'utilité. Cette procédure se fonde sur un modèle graphique nommé* réseau GAI *qui est employé pour représenter et gérer des indépendances entre attributs. Nous proposons ensuite un algorithme de choix et de rangement fondé sur les réseaux GAI pour résoudre efficacement aussi bien des problèmes d'optimisation que des problèmes de rangement sur un produit cartésien. Nous montrons que les réseaux GAI peuvent à la fois intégrer des contraintes et des préférences et donc être efficacement utilisés pour calculer le choix optimal et le rangement sous contraintes. Nous fournissons enfin des résultats d'expérimentations numériques qui montrent l'efficacité de notre approche.*

ABSTRACT. *This paper deals with preference elicitation and preference-based recommendation (choice and ranking) in the context of multiattribute utility theory. We focus on the generalized additive decomposable utility model (GAI) which allows interactions between attributes while preserving some decomposability. We first present a systematic elicitation procedure for such utility functions. This procedure relies on a graphical model called a* GAI-network *which is used to represent and manage independences between attributes. Then, we propose a choice procedure as well as a ranking procedure relying on a GAI-network to solve efficiently both optimization and ranking problems over a product set. We show that GAI networks can jointly integrate constraints and preferences and can thus be efficiently exploited in this case to compute constrained choices and ranking. Finally, we provide results of numerical experiments to highlight the practical efficiency of our approach.*

MOTS-CLÉS : *réseaux GAI ; modèles graphiques; élicitation de préférences; rangement.*

KEYWORDS: *GAI networks ; graphical models; preference elicitation; ranking.*

DOI: 10.3166/RIA.21.555-587

RSTI - RIA – 21/2007. Modèles graphiques non probabilistes, pages 555 à 587

1. Introduction

Le développement de systèmes interactifs d'aide à la décision et de systèmes de recommandation pour le web ont mis en évidence la nécessité de disposer de modèles permettant de manipuler les préférences des utilisateurs et de fournir des recommandations fondées sur leurs préférences. Ainsi, des travaux récents en IA concernant la représentation des préférences visent à développer des systèmes de représentation compacte de préférences réalisant un bon compromis entre deux aspects parfois antagonistes : i) le besoin de modèles suffisamment riches et flexibles pour décrire des comportements de décision sophistiqués ; et ii) la nécessité pratique de maintenir l'effort d'élicitation à un niveau admissible et aussi le besoin de procédures efficaces pour résoudre des problèmes d'optimisation fondés sur des préférences. Ces besoins sont particulièrement présents lorsqu'il s'agit de développer un système interactif d'aide à la décision sur le web où la solution préférée doit être trouvée parmi un ensemble combinatoire de possibilités. Ce type d'application justifie l'intérêt actuel pour les modèles qualitatifs de préférences et les représentations compactes comme les CP-nets (Boutilier *et al.*, 2004a; Boutilier *et al.*, 2004b). Ces modèles sont délibérément simples pour être intégrés efficacement dans les systèmes interactifs de recommandation ; les préférences des agents doivent être capturées en posant quelques questions seulement, afin de trouver rapidement parmi les articles disponibles ceux qui sont les meilleurs selon les préférences de l'utilisateur. Dans d'autres applications (par exemple les problèmes de configuration, les problèmes de répartition équitable de ressources, les enchères combinatoires), on peut consacrer plus de temps à l'étape d'élicitation afin d'obtenir une description plus fine des préférences. Dans ce cadre, les utilités peuvent être plus adaptées que les modèles qualitatifs grâce à leur plus grande puissance descriptive (Boutilier *et al.*, 2001). Par exemple, des utilités sont en général mieux adaptées pour représenter des préférences cardinales (quand le décideur exprime différentes intensités de préférences pour les alternatives).

Dans la littérature, différents modèles quantitatifs fondés sur des utilités ont été développés pour la représentation de préférences individuelles, permettant de tenir compte de différentes structures de préférences sur un espace multiattribut. Le modèle le plus utilisé suppose une indépendance entre attributs qui assure la représentabilité des préférences par une utilité additive. Une telle décomposabilité rend le processus d'élicitation très simple et rapide. Cependant, dans la pratique, cette indépendance n'est pas nécessairement vérifiée puisqu'elle élimine toute possibilité d'interaction entre les attributs. Quelques généralisations ont donc été étudiées. Par exemple, *l'indépendance préférentielle mutuelle* mène à une forme plus sophistiquée qu'on appelle *utilité multilinéaire* (Keeney *et al.*, 1993). Celle-ci est plus générale qu'une utilité additive mais elle ne couvre pas tous les types d'interactions entre les attributs. Pour augmenter la puissance descriptive des utilités additives, les décompositions GAI (indépendance additive généralisée) ont été introduites par (Fishburn, 1970). Ces décompositions GAI permettent des interactions plus générales entre les attributs (Bacchus *et al.*, 1995) tout en préservant une certaine décomposabilité du modèle. Elles ont été

utilisées pour enrichir des CP-nets avec des fonctions d'utilité (UCP-nets) aussi bien dans l'incertain (Boutilier *et al.*, 2001) que dans le certain (Brafman *et al.*, 2004).

Dans ce contexte, (Gonzales *et al.*, 2004) proposent une procédure générale pour éliciter des utilités GAI décomposables dans le cadre de la décision dans le risque. La procédure d'élicitation est guidée par la structure d'un nouveau modèle graphique nommé réseau GAI et consiste en une séquence de questions concernant des loteries simples qui capturent efficacement les caractéristiques principales de l'attitude du décideur vis-à-vis du risque. Par exemple, il est bien connu que la concavité d'une utilité (de von Neumann-Morgenstern) révèle une aversion pour le risque de la part du décideur. L'utilité élicitée dans un cadre risqué ne peut donc s'appliquer dans un cadre non risqué (le certain) puisqu'elle est alors biaisée par, entre autres, l'aversion face au risque. Ainsi, la procédure proposée (Gonzales *et al.*, 2004) ne peut-elle être directement utilisée pour éliciter des utilités dans le certain. Cet article étudie les contributions potentielles des réseaux GAI dans le cadre de la prise de décision dans le certain. La section 2 introduit les réseaux GAI et souligne leur apport vis-à-vis des modèles graphiques existants (CP-nets, TCP-nets). Ensuite, nous abordons le problème de l'élicitation d'une utilité GAI et proposons un algorithme pour le résoudre. Dans la section 4, nous exposons un algorithme efficace pour résoudre des questions d'optimisation (problème de choix) et nous l'étendons dans la section 5 au problème de rangement. Enfin, la section 6 fournit des résultats des expérimentations numériques qui montrent l'efficacité pratique de notre procédure de choix/rangement.

2. Réseaux GAI

Avant de décrire les réseaux GAI, nous devons introduire quelques notations. Dans cet article, $\succsim$ représente la relation de préférence d'un décideur (un préordre large total) sur un ensemble $\mathcal{X}$. $x \succsim y$ signifie que x est au moins aussi bon que y. Nous noterons $\succ$ la partie asymétrique de $\succsim$, et $\sim$ sa partie symétrique. Nous supposerons que l'ensemble $\mathcal{X}$ des alternatives est un produit cartésien d'attributs $X_1, \ldots, X_n$, comme c'est le cas dans la plupart des situations pratiques. Nous adoptons les notations suivantes, classiques en théorie de la décision :

– Les lettres majuscules, comme A, B, X_1, représentent aussi bien les attributs que leurs domaines.

– Les minuscules, comme a, b', x_1, représentent des valeurs d'attributs.

– Les attributs indicés X_i sont caractérisés par leur indice. Leurs valeurs sont également identifiées par cet indice. Ainsi, x_i, y_i, z_i^3 représentent des valeurs de X_i.

– Les attributs non indicés, A, B, sont caractérisés par leur lettre. Leurs valeurs sont donc également identifiées par cette lettre. Ainsi a, a', a^3 appartiennent à A.

– Par abus de notation, si $I = \{i_1, \ldots, i_k\}$ est un sous-ensemble de $\{1, \ldots, n\}$, x_I représentera le k-uplet $(x_{i_1}, \ldots, x_{i_k}) \in X_{i_1} \times \cdots \times X_{i_k}$. De même, si x et y sont deux éléments de $\mathcal{X}$, alors $z = (x_{-I}, y_I)$ représentera le n-uplet $(z_1, \ldots, z_n)$ tel que $z_i = y_i$ pour $i \in I$ et $z_i = x_i$ pour $i \notin I$.

2.1. *Motivation*

Sous certaines hypothèses (Debreu, 1964), la relation $\succsim$ peut être représentée par une utilité, c'est-à-dire une fonction $u : \mathcal{X} \mapsto \mathbb{R}$ telle que $x \succsim y \Leftrightarrow u(x) \geq u(y)$ pour tout $x, y \in \mathcal{X}$. Comme les préférences sont spécifiques à chaque agent, les utilités doivent être élicitées pour chaque individu et chaque élément de $\mathcal{X}$, une tâche difficile en raison de la nature combinatoire de $\mathcal{X}$. Par ailleurs, dans un système de recommandation avec de nombreux utilisateurs, stocker explicitement pour chacun d'entre eux l'utilité de tout $x \in \mathcal{X}$ est impossible, ne serait-ce que d'un point de vue utilisation de la mémoire. Heureusement, les préférences des décideurs ont souvent une structure sous-jacente induite par des indépendances entre les attributs. Ceci diminue sensiblement l'effort d'élicitation et la mémoire nécessaire pour stocker les préférences. Le cas le plus simple est obtenu quand les préférences sur $\mathcal{X} = X_1 \times \cdots \times X_n$ sont représentables par une utilité additive $u(x) = \sum_{i=1}^{n} u_i(x_i)$ pour tout $x = (x_1, \ldots, x_n) \in \mathcal{X}$. Ce modèle requiert seulement le stockage des $u_i(\alpha)$ pour chaque $\alpha \in X_i$. Cependant, une telle décomposition n'est pas toujours appropriée car elle élimine toute possibilité d'interaction entre les attributs. Quand les préférences des agents sont plus complexes, un modèle plus sophistiqué est nécessaire comme le montre l'exemple suivant :

Exemple 1 : Considérons un ensemble $\mathcal{X}$ de menus $x = (x_1, x_2, x_3)$, avec pour plat principal $x_1 \in X_1 = \{\text{steak}(s_1), \text{poisson}(p_1)\}$, pour boisson $x_2 \in X_2 = \{\text{vin rouge}\ (r_2), \text{vin blanc}(b_2)\}$ et pour dessert $x_3 \in X_3 = \{\text{cake}(c_3), \text{glace}(g_3)\}$.

Premier cas. Supposons que les préférences d'un individu soient les suivantes :

– Je préfère toujours un menu avec un steak à un menu avec du poisson.

– Pour accompagner un steak je préfère du rouge au blanc. Cela vaut également pour accompagner du poisson.

– Je préfère le cake à la glace (toutes choses égales par ailleurs).

De telles préférences peuvent être représentées par une utilité additive $u(x) = u_1(x_1) + u_2(x_2) + u_3(x_3)$ caractérisée par les utilités marginales suivantes : $u_1(s_1) = 4$; $u_1(p_1) = 0$; $u_2(r_2) = 2$; $u_2(b_2) = 0$; $u_3(c_3) = 1$; $u_3(g_3) = 0$. Alors les utilités des 2^3 menus $x^{(i)}$ possibles sont :

$u(x^{(1)}) = u(s_1, r_2, c_3) = 7$; $u(x^{(2)}) = u(s_1, r_2, g_3) = 6$; $u(x^{(3)}) = u(s_1, b_2, c_3) = 5$;
$u(x^{(4)}) = u(s_1, r_2, g_3) = 4$; $u(x^{(5)}) = u(p_1, r_2, c_3) = 3$; $u(x^{(6)}) = u(p_1, r_2, g_3) = 2$;
$u(x^{(7)}) = u(p_1, b_2, c_3) = 1$; $u(x^{(8)}) = u(p_1, b_2, g_3) = 0$;

lesquelles induisent l'ordre suivant :

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

Deuxième cas. Supposons qu'un autre agent ait pour préférences : $x^{(1)} \succ x^{(2)} \succ x^{(7)} \succ x^{(8)} \succ x^{(3)} \succ x^{(4)} \succ x^{(5)} \succ x^{(6)}$. Ceci peut être expliqué ainsi : i) le plus important est d'accorder le plat principal avec le vin (du vin rouge pour le steak, du vin blanc pour le poisson) ; ii) à un niveau plus bas de priorité, le steak est préféré au poisson ; et iii) le cake est préféré à la glace (toutes choses égales par ailleurs).

Bien que rationnelles, ces préférences ne sont pas représentables par une utilité additive car $x^{(1)} \succ x^{(5)} \Rightarrow u_1(s_1) > u_1(p_1)$ mais $x^{(7)} \succ x^{(3)} \Rightarrow u_1(p_1) > u_1(s_1)$. Toutefois, il est possible d'avoir des formes désagrégées de décompositions additives telles que : $u(x) = u_{1,2}(x_1, x_2) + u_3(x_3)$. Ainsi, $u_{1,2}(s_1, r_2) = 6$, $u_{1,2}(p_1, b_2) = 4$, $u_{1,2}(s_1, b_2) = 2$, $u_{1,2}(p_1, r_2) = 0$, $u_3(c_3) = 1$, $u_3(g_3) = 0$ représentent $\succsim$.

Troisième cas. Supposons que les préférences d'un troisième agent soient : $x^{(2)} \succ x^{(1)} \succ x^{(7)} \succ x^{(8)} \succ x^{(4)} \succ x^{(3)} \succ x^{(5)} \succ x^{(6)}$. Ces préférences sont une sophistication des préférences du deuxième cas. L'individu préfère le cake à la glace quand le plat principal est du poisson et le contraire quand le plat principal est du steak.

Dans ce cas, on peut constater que la décomposition précédente n'est plus appropriée en raison de l'interaction entre les attributs X_1 et X_3. On peut cependant remarquer que ces préférences peuvent être représentées par une utilité décomposable de la forme : $u(x) = u_{1,2}(x_1, x_2) + u_{1,3}(x_1, x_3)$, en posant par exemple :

$$u_{1,2}(s_1, r_2) = 6\,;\, u_{1,2}(p_1, b_2) = 4\,;\, u_{1,2}(s_1, b_2) = 2\,;\, u_{1,2}(p_1, r_2) = 0\,;$$
$$u_{1,3}(s_1, c_3) = 0\,;\, u_{1,3}(s_1, g_3) = 1\,;\, u_{1,3}(p_1, c_3) = 1\,;\, u_{1,3}(p_1, g_3) = 0.$$

On pourrait objecter que cette représentation n'est pas plus compacte que la représentation en extension, ce qui est vrai dans ce cas particulier du fait de sa petite taille. En général, si m représente la taille des domaines X_i le stockage de cette fonction d'utilité requiert $2m^2$ nombres au lieu de m^3 ce qui procure un gain dès que $m > 2$, gain qui devient de plus en plus important au fur et à mesure que m augmente. ♦

Une telle décomposition de l'utilité admettant des facteurs ayant une intersection non vide est appelée décomposition GAI (Bacchus *et al.*, 1995). Ces décompositions incluent les décompositions additives et multilinéaires comme cas particuliers, mais elle sont beaucoup plus flexibles puisqu'elles autorisent des interactions entre attributs et ne font aucune supposition a priori sur le type d'interaction entre ces attributs.

2.2. *Définition des réseaux GAI*

Les décompositions GAI peuvent être définies plus formellement comme suit :

Définition 1 (décomposition GAI)**.** *Soit $\mathcal{X} = \times_{i=1}^{n} X_i$. Soit $C_1, \ldots, C_k$ des sous-ensembles de $N = \{1, \ldots, n\}$ tels que $N = \bigcup_{i=1}^{k} C_i$. $\forall i$, soit $X_{C_i} = \times_{j \in C_i} X_j$; autrement dit, X_{C_i} est le produit cartésien des attributs dont les indices appartiennent à C_i. L'utilité $u(\cdot)$ qui représente $\succsim$ est GAI-décomposable par rapport aux X_{C_i} ssi il existe des fonctions $u_i : X_{C_i} \mapsto \mathbb{R}$ telles que :*

$$u(x) = \sum_{i=1}^{k} u_i(x_{C_i}), \;\; \forall x = (x_1, \ldots, x_n) \in \mathcal{X},$$

où x_{C_i} est le n-uplet formé par les x_j, $j \in C_i$.

Les décompositions GAI peuvent être représentées par des structures graphiques que nous appelons *réseaux GAI* (Gonzales *et al.*, 2004) ou *GAI-net*. Ceux-ci sont similaires aux graphes de jonction utilisés pour les réseaux bayésiens (Jensen, 1996) :

Définition 2 (réseau GAI)**.** *Soit $\mathcal{X} = \times_{i=1}^{n} X_i$. Soit $C_1, \ldots, C_k$ des sous-ensembles de $N = \{1, \ldots, n\}$ tels que $N = \bigcup_{i=1}^{k} C_i$. Supposons que $\succsim$ est représentable par une utilité GAI $u(x) = \sum_{i=1}^{k} u_i(x_{C_i})\ \forall x \in \mathcal{X}$. Alors un réseau GAI qui représente $u(\cdot)$ est un graphe non-orienté $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ qui satisfait les propriétés suivantes :*

– Propriété 1 : $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_k}\}$;

– Propriété 2 : $(X_{C_i}, X_{C_j}) \in \mathcal{E} \Rightarrow C_i \cap C_j \neq \emptyset$. $\forall X_{C_i}, X_{C_j}$ tels que $C_i \cap C_j = T_{ij} \neq \emptyset$, il existe un chemin dans $\mathcal{G}$ qui connecte X_{C_i} et X_{C_j} tel que tous ses nœuds contiennent tous les indices de T_{ij} (propriété d'intersection courante).

Les nœuds de $\mathcal{C}$ sont appelés cliques*. Chaque arête $(X_{C_i}, X_{C_j}) \in \mathcal{E}$ est étiquetée par $X_{T_{ij}} = X_{C_i \cap C_j}$ et est appelée un* séparateur.

Les cliques sont représentées par des ellipses et les séparateurs par des rectangles. Ici, nous nous intéressons seulement à des arbres GAI. Comme mentionné dans (Gonzales *et al.*, 2004), ceci n'est pas restrictif puisque des réseaux GAI généraux peuvent toujours être recompilés dans des arbres GAI. Pour toute décomposition GAI, selon la définition 2, les cliques du réseau GAI doivent être les ensembles de variables des facteurs d'utilité. À ce titre, les arêtes reliant les cliques indiquent simplement la présence de certains attributs dans plusieurs facteurs. Autrement dit, elles représentent des intersections entre ensembles d'attributs. Or, l'intersection étant une opération commutative, il convient de représenter le réseau GAI par un graphe non orienté. Notons que cela contraste avec les UCP-nets où les relations de dépendances entre les facteurs sont conditionnelles et justifient l'utilisation de graphes orientés.

Exemple 2 : Si $u(a, b, c, d, e, f, g) = u_1(a, b) + u_2(c, e) + u_3(b, c, d) + u_4(b, d, f) + u_5(b, g)$ alors, comme le montre la figure 1, les cliques sont : AB, CE, BCD, BDF et BG. Par la propriété 2 de la définition 2, l'ensemble d'arêtes d'un réseau GAI peut être déterminé par des algorithmes qui préservent la propriété d'intersection courante (voir la littérature sur réseaux bayésiens (Cowell *et al.*, 1999)). ♦

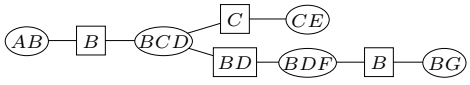

Figure 1. *Un arbre GAI*

Notation : dans la suite, si $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ est un arbre GAI, $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_k}\}$ représente l'ensemble des cliques de $\mathcal{G}$ ordonnées de l'extérieur vers l'intérieur, c'est-à-dire tel que, pour tout i, si $\mathcal{G}_{\mathcal{C}_i}$ représente le sous-graphe de $\mathcal{G}$ induit par $\mathcal{C}_i = \mathcal{C} \backslash \{X_{C_j} : j < i\}$, alors $\mathcal{G}_{\mathcal{C}_i}$ est connexe. De plus, $\forall i \in \{1, \ldots, |\mathcal{C}| - 1\}$, la clique X_{C_i} a exactement une clique voisine dans $\mathcal{G}_{\mathcal{C}_i}$, que l'on désignera par $X_{C_{n(i)}}$. Enfin, nous noterons $S_i = C_i \cap C_{n(i)}$ et $D_i = C_i \backslash S_i$. X_{S_i} correspondra donc au séparateur entre X_{C_i} et $X_{C_{n(i)}}$ et X_{D_i} aux attributs de X_{C_i} n'appartenant pas à ce séparateur.

2.3. *CP-nets, TCP-nets et GAI-nets*

Une des caractéristiques des CP-nets (Boutilier *et al.*, 2004a), est de reposer sur des préférences *Ceteris Paribus*, éventuellement conditionnées par les valeurs de certains attributs. Ainsi, pour utiliser des CP-nets dans l'exemple 1, on doit formuler des assertions ceteris paribus prenant l'une des deux formes suivantes :

– une préférence inconditionnelle, par exemple « *je préfère le steak au poisson* », ce qui s'écrit de manière compacte $s_1 \succ p_1$ et signifie que $\forall x_2 \in X_2, \forall x_3 \in X_3, (s_1, x_2, x_3) \succ (p_1, x_2, x_3)$;

– une préférence conditionnelle, par exemple « *avec un steak, je préfère le vin rouge au vin blanc mais c'est le contraire avec du poisson* », ce qui s'écrit de manière compacte $s_1 : r_2 \succ b_2$ et $p_1 : b_2 \succ r_2$ et signifie que $\forall x_3 \in X_3, (s_1, r_2, x_3) \succ (s_1, b_2, x_3)$ et $(p_1, b_2, x_3) \succ (p_1, r_2, x_3)$.

En faisant l'hypothèse que les préférences sont transitives, on peut, en chaînant des préférences ceteris paribus, déduire d'autres préférences mais cela ne permet pas pour autant de tout exprimer. Ainsi dans le premier cas de l'exemple 1, il ne serait pas possible d'exprimer directement la préférence $(s_1, b_2, g_3) \succ (p_1, r_2, c_3)$ ni de l'obtenir par une chaîne de préférences ceteris paribus dans un CP-net incluant les informations élémentaires connues telles que $s_1 \succ p_1, r_2 \succ b_2$ et $c_3 \succ g_3$. Afin de discuter plus précisément le potentiel descriptif des deux systèmes de représentation compacte, considérons l'exemple suivant :

Exemple 3 : on s'intéresse à un système de recommandation permettant de proposer à l'utilisateur des voyages dont la configuration est optimisée en fonction des préférences de l'utilisateur. Pour simplifier, intéressons-nous à la réservation d'un voyage en train, l'ensemble $\mathcal{X}$ des configurations possibles étant constitué de triplets $x = (x_1, x_2, x_3)$ caractérisés par la classe $x_1 \in X_1 = \{$première classe (c_1^1), deuxième classe $(c_1^2)\}$, le choix d'un wagon fumeur ou non-fumeur $x_2 \in X_2 = \{$fumeur(f_2), non fumeur$(\bar{f}_2)\}$ et la période tarifaire $x_3 \in X_3 = \{$période bleue(b_3), période rouge $(r_3)\}$. Supposons que les préférences d'un individu soient les suivantes :

– je préfère voyager en deuxième classe qu'en première classe[1] ;

– je préfère un wagon non-fumeur à un wagon fumeur[1] ;

– je préfère voyager en wagon 1ère classe non-fumeur qu'en 2ème classe fumeur[1] ;

– je préfère voyager en période bleue qu'en période rouge quelles que soient les conditions du voyage.

De telles préférences peuvent aisément être représentées par une utilité additivement décomposable $u(x) = u_{12}(x_1, x_2) + u_3(x_3)$ caractérisée par les utilités marginales suivantes : $u_{1,2}(c_1^2, \bar{f}_2) = 3$, $u_{1,2}(c_1^1, \bar{f}_2) = 2$, $u_{1,2}(c_1^2, f_2) = 1$, $u_{1,2}(c_1^1, f_2) = 0$, $u_3(b_3) = 4$, $u_3(r_3) = 0$. En effet, on obtient alors les utilités suivantes pour les 2^3 configurations possibles $x^{(i)}$:

1. Ces préférences s'entendent *ceteris paribus*, c'est-à-dire toutes choses égales par ailleurs.

$u(x^{(1)}) = u(c_1^2, \bar{f}_2, b_3) = 7$; $u(x^{(2)}) = u(c_1^2, \bar{f}_2, r_3) = 3$; $u(x^{(3)}) = u(c_1^1, \bar{f}_2, b_3) = 6$;
$u(x^{(4)}) = u(c_1^1, \bar{f}_2, r_3) = 2$; $u(x^{(5)}) = u(c_1^2, f_2, b_3) = 5$; $u(x^{(6)}) = u(c_1^2, f_2, r_3) = 1$;
$u(x^{(7)}) = u(c_1^1, f_2, b_3) = 4$; $u(x^{(8)}) = u(c_1^1, f_2, r_3) = 0$;

lesquelles induisent l'ordre suivant :

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

Le modèle GAI rend ici parfaitement compte des préférences exprimées. En revanche, si les deux premières assertions sont facilement exprimables dans le langage des CP-nets (il suffit d'écrire $c_1^2 \succ c_1^1$ et $\bar{f}_2 \succ f_2$) on ne peut exprimer les deux suivantes. En effet la première d'entre-elles exprime une préférence impliquant le couple de valeurs (x_1, x_2) et révèle une interaction entre ces deux attributs. Elle n'est donc pas décomposable. Ainsi, la seule façon de représenter les préférences du type $(c_1^1, \bar{f}_2, x_3) \succ (c_1^2, f_2, x_3)$ dans un CP-net tel que $\bar{f}_2 \succ f_2$ est d'ajouter la préférence conditionnelle $\bar{f}_2 : c_1^1 \succ c_1^2$ ce qui donnerait $(c_1^1, \bar{f}_2, x_3) \succ (c_1^2, \bar{f}_2, x_3) \succ (c_1^2, f_2, x_3)$. Mais une telle préférence $\bar{f}_2 : c_1^1 \succ c_1^2$ est incompatible avec la préférence inconditionnelle $c_1^2 \succ c_1^1$ exprimée par l'agent. Il suffit de chercher à comparer les solutions $(c_1^1, \bar{f}_2, b_3)$ et $(c_1^2, \bar{f}_2, b_3)$ pour s'en convaincre. De même, la dernière assertion (préférence unilatérale pour la période bleue) n'est pas une conséquence de $b_3 \succ r_3$. En effet, sachant que $\bar{f}_2 \succ f_2$ et $c_1^2 \succ c_1^1$, une préférence telle que $(c_1^1, f_2, b_3) \succ (c_1^2, \bar{f}_2, r_3)$ ne peut s'obtenir en chaînant des préférences ceteris paribus.

Ces limites descriptives des CP-nets ont motivé l'introduction d'un formalisme plus riche permettant de différencier les poids relatifs des attributs dans les comparaisons. Cette idée est à l'origine des TCP nets (Brafman *et al.*, 2002) qui introduisent des « tradeoffs » dans les CP-nets. On peut ainsi stipuler que l'attribut x_1 est plus important que x_2 (noté $1 \triangleright 2$) ce qui permet d'induire des préférences croisées du type $(c_1^1, \bar{f}_2, x_3) \succ (c_1^2, f_2, x_3)$ à partir des préférences ceteris paribus $c_1^2 \succ c_1^1$ et $\bar{f}_2 \succ f_2$ (le conflit entre l'attribut classe et l'attribut fumeur étant résolu par le fait que l'on considère que l'attribut fumeur est le plus important). Cette sophistication des CP-nets ne corrige pourtant que partiellement le problème descriptif évoqué ci-dessus. En effet supposons que le domaine X_1 de la variable x_1 ait 5 éléments $c_1^1, c_1^2, c_1^3, c_1^4, c_1^5$ correspondant à des niveaux de prix décroissants, quelle que soit la période de voyage. Alors il est serait tout à fait naturel qu'un agent ait les préférences suivantes :

$$(c_1^4, \bar{f}_2, x_3) \succ (c_1^5, f_2, x_3) \quad \text{et} \quad (c_1^1, \bar{f}_2, x_3) \prec (c_1^5, f_2, x_3).$$

La première de ces deux préférences s'explique par le fait qu'il est prêt à payer un peu plus (on passe de c_1^5 à c_1^4) pour bénéficier d'un wagon non fumeur. La seconde préférence s'explique par le fait que l'écart de prix entre un billet de classe c_1^1 et un billet de classe c_1^5 est trop important pour accéder à un wagon non fumeur. Dans un TCP-net, la première de ces deux préférences nécessiterait d'introduire l'information $1 \triangleright 2$ mais elle aurait comme effet collatéral d'induire la préférence $(c_1^1, \bar{f}_2, x_3) \succ (c_1^5, f_2, x_3)$ qui n'est pas pertinente. On voit bien ici une limite descriptive intrinsèque des TCP nets qui ne permettent pas de prendre en compte les intensités de préférence entre les valeurs d'attributs.

On peut résumer l'apport des GAI-nets ainsi :

– les GAI-nets ne se limitent pas à l'expression de préférences *Ceteris Paribus* et peuvent décrire n'importe quel préordre total sur un espace produit, ce qui n'est pas possible avec un CP-net ni un TCP-net ;

– les tables d'utilités locales exprimées dans les GAI-nets permettent de quantifier la valeur d'un n-uplet et donc de traduire des intensités de préférences ce qui n'est pas possible dans un TCP net.

En revanche il faut signaler que, sauf à valuer la fonction d'utilité dans un ensemble partiellement ordonné (ce que nous n'avons pas envisagé ici), les GAI-nets ne permettent pas de décrire des structures de préférences partielles alors que les CP-nets permettent de décrire certaines d'entre-elles. De plus, les CP-nets traduisant des structures de préférences très simples, elles sont plus faciles à éliciter. Ces systèmes de représentation sont donc complémentaires. On préférera l'un ou l'autre suivant le niveau d'information dont on dispose sur les préférences de l'agent, le degré de sophistication que l'on souhaite atteindre dans les recommandations, et le temps dont on dispose pour éliciter les préférences. La problématique de l'élicitation des préférences dans les GAI-nets est l'objet de la section suivante.

3. Élicitation

L'élicitation d'utilités GAI décomposables est étroitement liée à celle des utilités additives. À titre d'exemple, considérons l'arbre GAI de la figure 2. Cet arbre représente la décomposition GAI suivante :

$$u(a,b,c) = u_1(a,b) + u_2(b,c)\ \forall (a,b,c) \in A \times B \times C.$$

Fixons maintenant la valeur de l'attribut B, par exemple à $B = \overline{b}$. Alors, $u(a,\overline{b},c) = u_1(a,\overline{b}) + u_2(\overline{b},c)$ devient une utilité additive sur $A \times C$ et les techniques classiques d'élicitation d'utilités additives permettent de construire indépendamment de la valeur accordée à B des collections de fonctions $u_1(A,\overline{b}) + u_2(\overline{b},C)$. Pour achever la construction de $u(a,b,c)$ sur $A \times B \times C$, il ne reste plus qu'à « raccorder » les morceaux obtenus par ces collections de manière à former une fonction d'utilité globale. Au début de cette section, nous allons donc rappeler les techniques classiques d'élicitation des fonctions d'utilité dans le certain. Nous en dériverons dans les sous-sections suivantes une technique permettant l'élicitation dans un cadre GAI.

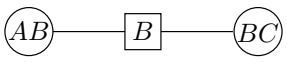

Figure 2. *Liens entre l'élicitation de fonctions d'utilité additives et GAI*

3.1. *Une procédure générique d'élicitation d'utilités additives*

Les procédures d'élicitation d'utilités additives s'appuient principalement sur une propriété particulière de ces utilités, à savoir leur unicité à une transformation affine strictement positive près. Celle-ci dérive naturellement des ensembles d'axiomes garantissant l'existence même de ces utilités (Fishburn, 1970; Krantz *et al.*, 1971; Wakker, 1989; Nakamura, 2002). Ceux-ci sont en outre exploités pour déterminer les questions à poser pour construire les fonctions d'utilité. Cela étant dit, bien que différentes axiomatiques aient été développées (approche algébrique, topologique, etc), les procédures d'élicitation induites sont essentiellement similaires. C'est pourquoi, dans cette sous-section, nous allons nous attacher à présenter une méthode d'élicitation générique qui mettra en valeur les idées forces de ce type de technique. Dans la sous-section suivante, nous l'instancierons en nous fondant sur un ensemble d'axiomes particulier. Pour l'instant, nous allons donc faire l'hypothèse suivante :

Hypothèse 1. *Les utilités additives sont uniques à une transformation affine strictement positive près, c'est-à-dire que si* $u(x_1, \ldots, x_n) = \sum_{i=1}^{n} u_i(x_i)$ *et* $v(x_1, \ldots, x_n) = \sum_{i=1}^{n} v_i(x_i)$ *sont des utilités additives représentant* $\succsim$*, alors* $\exists\ \alpha > 0$ *et* $\exists\ \beta_i \in \mathbb{R}$*,* $\forall i \in \{1, \ldots, n\}$*, tels que* $u_i(\cdot) = \varphi_i(v_i(\cdot)) \Rightarrow \varphi_i(x) = \alpha x + \beta_i$ $\forall x \in v_i(X_i)$.

Les procédures d'élicitation tirent parti des indépendances entre les attributs nécessaires à l'existence des utilités additives, de manière à construire ces dernières en deux étapes bien distinctes : tout d'abord, chaque facteur d'utilité (les $u_i(\cdot)$) est construit indépendamment des autres facteurs. Dans un second temps, ces facteurs sont agrégés de manière à former une utilité globale et c'est l'hypothèse 1 qui garantit que cela est possible. Procéder ainsi en deux temps nous assure que la méthode d'élicitation est à la fois rapide et fiable. Elle est rapide car l'élicitation d'une utilité sur chaque X_i évite l'explosion combinatoire engendrée par le produit cartésien des X_i. Elle est fiable car les questions posées au décideur sont cognitivement simples dans la mesure où elles ne font intervenir que peu d'attributs avec des valeurs différentes (en principe, seuls deux attributs varient). Bien entendu, ces questions peuvent varier d'une procédure d'élicitation à l'autre, c'est pourquoi, afin de présenter une méthode générale, nous allons faire la supposition suivante :

Hypothèse 2. *Pour chaque attribut* X_i*, il existe un questionnaire permettant de construire le facteur d'utilité* $u_i : X_i \to \mathbb{R}$ *indépendamment des autres facteurs.*

Supposons maintenant que $\succsim$ soit représentable par une utilité additive $u(x_1, \ldots, x_n) = \sum_{i=1}^{n} u_i(x_i)$. Alors, d'après l'hypothèse 2, il existe des questionnaires permettant d'éliciter des fonctions d'utilité $v_i : X_i \to \mathbb{R}$. À cette étape, nous ne savons pas encore si les fonctions $u_i(\cdot)$ que nous recherchons sont égales aux $v_i(\cdot)$. Nous savons seulement que, puisque les $u_i(\cdot)$ et les $v_i(\cdot)$ sont des utilités sur la projection de $\succsim$ sur X_i, il existe une fonction strictement croissante $\varphi_i : \mathbb{R} \to \mathbb{R}$ telle que $u_i(\cdot) = \varphi_i(v_i(\cdot))$. En général, éliciter φ_i peut être aussi coûteux que d'éliciter v_i. Heureusement, l'hypothèse 1 nous assure que les fonctions $\varphi_i(\cdot)$ sont des fonctions

affines strictement positives, autrement dit qu'il existe des nombres réels $k_i \in \mathbb{R}_+^*$ et $r_i \in \mathbb{R}$, $i \in \{1, \ldots, n\}$, tels que $u_i(\cdot) = k_i v_i(\cdot) + r_i$. Par conséquent, pour achever la construction de la fonction d'utilité $u(\cdot)$ sur l'ensemble de $\mathcal{X}$, il ne reste plus qu'à déterminer les valeurs de k_i et de r_i. Une procédure générale d'élicitation peut donc être décrite de la manière suivante :

Algorithme 1 (Procédure générale d'élicitation d'utilités additives)
01 **pour tout** i dans $\{1, \ldots, n\}$ **faire**
02 Construire un facteur d'utilité $v_i(\cdot)$ sur X_i
03 Déterminer k_i et r_i de telle sorte que v_i s'accorde avec les $v_j(\cdot)$, $j < i$
04 **fait**

Puisque toute transformée affine strictement positive d'une utilité produit une autre utilité, le processus d'élicitation peut être simplifié en affectant la valeur 0 à tous les r_i. En effet, cela revient juste à rajouter une valeur constante à $u(\cdot)$. Pour la même raison, on peut affecter la valeur 1 à k_1. Voyons maintenant comment l'on peut déterminer aisément les valeurs des k_i pour $i \geq 2$. Supposons que, pour des valeurs a_i, b_i, a_j, b_j, $j < i$, le décideur soit capable d'émettre des jugements du type « je suis indifférent entre (a_i, a_j) et (b_i, b_j), tous les autres attributs étant fixés à une certaine valeur ». Soit x_p la valeur des attributs X_p pour $p \neq i, j$. On peut conclure que :

$$k_i v_i(a_i) + k_j v_j(a_j) + \sum_{p \neq i,j} k_p v_p(x_p) = k_i v_i(b_i) + k_j v_j(b_j) + \sum_{p \neq i,j} k_p v_p(x_p),$$

ou, en d'autres termes,

$$k_i = k_j \frac{v_j(b_j) - v_j(a_j)}{v_i(a_i) - v_i(b_i)} \quad \forall i \geq 2. \qquad [1]$$

Dans l'algorithme 1, les valeurs des k_i sont déterminées après celles de $v_i(\cdot)$, $v_j(\cdot)$ et k_j. L'équation [1] permet donc d'identifier de manière unique les k_i. Par conséquent, l'algorithme 1 permet de construire itérativement une utilité additive représentant $\succsim$. Notons que, dans les jugements mentionnés ci-dessus, les x_p peuvent être choisis arbitrairement puisqu'ils n'apparaissent pas dans l'équation [1]. Voyons maintenant comment l'algorithme 1 peut être instancié sur un ensemble d'axiomes particulier. Dans ce qui suit, nous avons choisi l'axiomatique de (Krantz *et al.*, 1971).

3.2. *Spécialisation de la procédure d'élicitation d'utilités additives*

D'une manière générale, contrairement à la décision dans l'incertain ou dans le risque, les utilités additives dans le certain ne sont pas nécessairement uniques à des transformations affines strictement positives près (Adams, 1965; Gonzales, 2003). Par conséquent, la procédure décrite ci-dessus ne peut être appliquée dans toutes les situations. Fort heureusement, l'ensemble d'axiomes que nous allons présenter maintenant, et qui couvre nombre de situations en pratique, assure que l'hypothèse 1 est satisfaite. Le premier de ces axiomes, que nous allons supposer être vrai dans toute la suite de

cette section, est une hypothèse structurelle sans doute un peu restrictive en pratique, mais elle peut être considérablement allégée (Gonzales, 2003).

Axiome 1 (solvabilité restreinte)**.** *Soit* $\mathcal{X} = \times_{i=1}^{n} X_i$. $\forall i \in \{1, \ldots, n\}$, *si* $(x_1^0, \ldots, x_{i-1}^0, x_i^0, x_{i+1}^0, \ldots, x_n^0) \precsim (x_1, \ldots, x_n) \precsim (x_1^0, \ldots, x_{i-1}^0, x_i^1, x_{i+1}^0, \ldots, x_n^0)$, *alors* $\exists\, x_i^2 \in X_i$ *tel que* $(x_1^0, \ldots, x_{i-1}^0, x_i^2, x_{i+1}^0, \ldots, x_n^0) \sim (x_1, \ldots, x_n)$.

Cette propriété implique normalement que le produit cartésien $\mathcal{X}$ contient de nombreux éléments, sinon un nombre infini d'éléments. Cependant, cela ne limite ni le champ d'application potentiel des utilités additives ni l'applicabilité du processus d'élicitation. En effet, dans le cas où un attribut n'est pas solvable, son domaine de définition peut toujours être étendu de sorte que la solvabilité restreinte soit vérifiée. Il importe juste de se limiter à la projection sur l'espace d'origine lorsque l'on prend les décisions. Quant à l'applicabilité de l'élicitation, la non finitude de $\mathcal{X}$ n'est pas problématique dans le sens où les courbes d'indifférence —les ensembles d'éléments appartenant à la même classe d'indifférence de $\succsim$ — forment des courbes en général très lisses et peuvent donc être approchées en connaissant seulement un nombre limité de points (par exemple en utilisant des splines ou des fonctions affines par morceaux).

En décision dans le certain, l'idée force pour éliciter les $v_i(\cdot)$ réside dans la construction de ce que l'on appelle des séquences standards : soit $\succsim$ une relation de préférence sur $X_1 \times X_2$ représentable par une utilité additive $k_1 v_1(\cdot) + k_2 v_2(\cdot)$. Soit $x_1^0 \in X_1$, $x_2^0, x_2^1 \in X_2$ des éléments quelconques tels que $(x_1^0, x_2^1) \succ (x_1^0, x_2^0)$. Puisque toute transformée affine strictement positive préserve la représentabilité des fonctions d'utilité, on peut supposer sans perte de généralité que $k_1 v_1(x_1^0) = k_2 v_2(x_2^0) = 0$ et $k_2 v_2(x_2^1) = 1$. Soit $x_1^1 \in X_1$ tel que $(x_1^1, x_2^0) \sim (x_1^0, x_2^1)$ (voir la figure 3 qui représente des courbes d'indifférence dans l'espace $X_1 \times X_2$), alors :

$$k_1 v_1(x_1^1) + k_2 v_2(x_2^0) = k_1 v_1(x_1^0) + k_2 v_2(x_2^1).$$

En d'autres termes, $k_1 v_1(x_1^1) = 1$. Plus généralement, pour tout $i \in \{1, \ldots, n\}$, soit x_1^i, x_1^{i+1} tels que $(x_1^{i+1}, x_2^0) \sim (x_1^i, x_2^1)$. Alors, si $\succsim$ est représentable par une utilité additive, $k_1 v_1(x_1^i) = i$. La suite (x_1^i) est appelée une séquence standard :

Définition 3 (séquence standard)**.** *Soit* N *un ensemble d'entiers consécutifs.* $\{x_1^k, k \in N\}$ *est une séquence standard pour* X_1 *ssi* $\exists x^0 = (x_2^0, \ldots, x_n^0)$ *et* $\exists x^1 = (x_2^1, \ldots, x_n^1)$ *tels que* $(x_1^0, x_{-1}^0) \not\sim (x_1^0, x_{-1}^1)$ [2] *et,* $\forall k, k+1 \in N$, $(x_1^{k+1}, x_{-1}^0) \sim (x_1^k, x_{-1}^1)$. $\{x^0, x^1\}$ *est appelé la maille de la séquence standard. Des définitions analogues s'appliquent aux autres* X_i, $i > 1$.

Naturellement, pour pouvoir éliciter une utilité en employant ces séquences, il faut que tout l'espace puisse être parcouru par celles-ci, ou du moins que le maillage de ces séquences soit suffisamment fin pour pouvoir inférer les valeurs de la fonction d'utilité sur tout l'espace (par exemple par densité). Cela nécessite l'axiome suivant.

2. Rappelons que si I est un ensemble d'indices d'attributs, alors $z = (x_{-I}, y_I)$ représente le n-uplet $(z_1, \ldots, z_n)$ tel que $z_i = y_i$ pour $i \in I$ et $z_i = x_i$ pour $i \notin I$. Par abus de notation, (x_{-1}, y_1) représente le n-uplet $(y_1, x_2 \ldots, x_n)$.

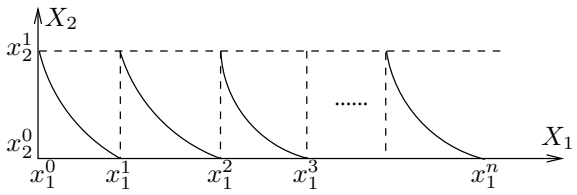

Figure 3. *Une séquence standard pour X_1*

Axiome 2 (axiome archimédien)**.** *Soit $\{x_1^k, k \in N\}$ une séquence standard par rapport à X_1 de maille $\{x^0, x^1\}$. Si $\exists\, y, z \in \mathcal{X}$ tels que $y \succ (x_1^k, x_{-1}^0) \succ z\ \forall x_1^k \in X_1$, alors N est fini. Des définitions analogues s'appliquent aux autres X_i, $i > 1$.*

Le dernier axiome nécessaire à l'existence d'utilités additives est une condition d'indépendance entre les attributs qui capture le fait que l'utilité se décompose comme une somme de fonctions définies sur des espaces complètement distincts (les X_i).

Axiome 3 (indépendance en coordonnées)**.** $\forall i \in \{1, \ldots, n\}$, $\forall z_i, t_i \in X_i$ *et* $\forall x_j, y_j \in X_j$, $j \neq i$, $(z_i, x_{-i}) \succsim (z_i, y_{-i}) \Leftrightarrow (t_i, x_{-i}) \succsim (t_i, y_{-i})$.

Cet axiome entraîne par récurrence l'existence d'une relation de préférence $\succsim_i$ sur chaque X_i définie par : $x_i \succsim_i y_i$ si et seulement si il existe un $(n-1)$-uplet $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \in \times_{j \neq i} X_j$ tel que $(x_i, x_{-i}) \succsim (y_i, x_{-i})$. Il a deux conséquences importantes en ce qui concerne la procédure d'élicitation :

1) il implique que le choix des mailles des séquences standards n'a aucune influence sur la valeur des $v_i(\cdot)$ construits ;

2) on n'a pas besoin d'éliciter la fonction d'utilité « globale » sur l'ensemble de $\mathcal{X}$ mais seulement sur des sous-espaces beaucoup plus petits (les X_i), et ces constructions « locales » s'étendent naturellement pour former une utilité sur l'ensemble de $\mathcal{X}$.

Bien entendu, seuls les attributs impliqués dans les prises de décision devraient être pris en compte lors de la phase d'élicitation et on devrait donc restreindre $\mathcal{X}$ au produit cartésien des seuls attributs non ainsi superflus. C'est ce qu'indique le dernier axiome utilisé dans l'axiomatique de (Krantz *et al.*, 1971) :

Axiome 4 (essentialité)**.** $\forall i \in \{1, \ldots, n\}$, $\exists\, x_i, y_i \in X_i$ *et* $\exists\, z_j \in X_j\ \forall j \neq i$, *tels que* $(z_{-i}, x_i) \succ (z_{-i}, y_i)$.

La combinaison des axiomes ci-dessus ainsi que d'une condition dite de Thomsen (que nous ne détaillerons pas ici car nous ne l'utilisons pas dans la suite) garantit que $\succsim$ est représentable par une utilité additive et qu'en outre l'hypothèse 1 d'unicité est vérifiée (Krantz *et al.*, 1971) :

Théorème 1 (Existence et unicité des utilités additives)**.** *Supposons que la solvabilité restreinte et l'essentialité soient vérifiées pour chaque attribut. Alors les deux assertions suivantes sont équivalentes :*

1) $\succsim$ est un préordre large total vérifiant, pour chaque attribut, l'indépendance en coordonnées, la condition de Thomsen (si $\mathcal{X}$ a 2 attributs) et l'axiome archimédien ;

2) $\succsim$ est représentable par une fonction d'utilité additive et, de plus, celle-ci est unique à une transformation affine strictement positive près.

Maintenant que l'on a la garantie que l'hypothèse 1 est vérifiée, voyons comment l'hypothèse 2 peut l'être. Choisissons arbitrairement deux valeurs x_n^0, x_n^1 de l'attribut X_n telles que $x_n^1 \succ_n x_n^0$, autrement dit telles que $x_n^1 \succsim_n x_n^0$ et $x_n^1 \not\sim_n x_n^0$. Choisissons de plus une valeur x_j^0 arbitraire pour les autres attributs. Si $\succsim$ est représentable par une utilité additive $\sum_{i=1}^n u_i(\cdot)$, alors, il existe $\alpha > 0$ et $\beta_i \in \mathbb{R}$ tels que $\alpha u_1(x_1^0) + \beta_1 = \alpha u_n(x_n^0) + \beta_n = 0$ et $\alpha u_n(x_n^1) + \beta_n = 1$. Les $v_i(\cdot) = \alpha u_i(\cdot) + \beta_i$ forment une utilité additive représentant $\succsim$ puisque $v(\cdot) = \sum_{i=1}^n v_i(\cdot)$ est une transformée affine de $u(\cdot)$. La construction d'une séquence standard $(x_1^k)_{k\in N}$ de maille $\{(x_n^1, x_{-\{1n\}}^0), (x_n^0, x_{-\{1n\}}^0)\}$ sur l'attribut X_1 a pour conséquence immédiate que $v_1(x_1^k) = k$ pour tout $k \in N$. Donc l'utilité $v_1(\cdot)$ peut être élicitée sur une grille $(x_1^k)_{k\in N}$ simplement en construisant une séquence standard. Cette fonction peut alors être étendue sur l'ensemble de l'espace X_1 en affinant itérativement la grille par utilisation de mailles de plus en plus fines, ou bien encore en utilisant des fonctions comme des splines interpolant ou approximant $v_1(x_1^k)$. Bien entendu, le même procédé peut être employé pour définir les autres $v_i(\cdot)$. Cela justifie le questionnaire suivant :

Algorithme 2 (questionnaire pour éliciter v_i)

01 choisir $(x_1^0, \ldots, x_n^0)$ un élément quelconque de $\mathcal{X}$
02 **pour tout** $i \in \{1, \ldots, n\}$ **faire** $v_i(x_i^0) \leftarrow 0$ **fait**
03 choisir $j \neq i$ puis $x_j^1 \in X_j$ un élément quelconque tel que $x_j^1 \succ_j x_j^0$
03 **pour** k = 1,2,3,… **faire**
04 demander au décideur pour quel x_i^k il est indifférent entre $(x_{-\{ij\}}^0, x_i^k, x_j^0)$ et $(x_{-\{ij\}}^0, x_i^{k-1}, x_j^1)$, et faire $v_i(x_i^k) \leftarrow k$
05 **fait**
06 **pour** k = -1,-2,-3,… **faire**
07 demander au décideur pour quel x_i^k il est indifférent entre $(x_{-\{ij\}}^0, x_i^{k+1}, x_j^0)$ et $(x_{-\{ij\}}^0, x_i^k, x_j^1)$, et faire $v_i(x_i^k) \leftarrow k$
08 **fait**
09 étendre la définition de $v_i(\cdot)$ sur X_i par approximation/interpolation

Les étapes 04 et 07 correspondent bien évidemment à la construction de séquences standards par rapport au $i$ème attribut. Notons que la construction de l'utilité $v_i(\cdot)$ est peu coûteuse en ce sens que, si la grille induite par la séquence standard a r points, seulement r questions sont posées au décideur. De plus, ces questions sont cognitivement relativement simples puisque les n-uplets sur lesquels portent ces questions (les indifférences des étapes 04 et 07) ne diffèrent que par deux attributs.

3.3. *Vers une unicité des utilités GAI décomposables*

Comme nous l'avons vu, l'unicité à une transformation affine strictement positive près est d'une importance primordiale pour la validité de la procédure d'éli-

citation des fonctions d'utilité additives. Malheureusement, même sous l'hypothèse de solvabilité restreinte, cette unicité ne peut être garantie lorsqu'il s'agit d'utilités GAI décomposables. En effet, considérons une relation de préférence $\succsim$ représentable sur $X_1 \times X_2 \times X_3$ par $u_1(x_1, x_2) + u_2(x_2, x_3)$. Alors $\succsim$ est tout aussi représentable par $v_1(x_1, x_2) + v_2(x_2, x_3)$, avec $v_1(x_1, x_2) = u_1(x_1, x_2) - f(x_2)$, $v_2(x_2, x_3) = u_2(x_2, x_3) + f(x_2)$ et $f(\cdot)$ une fonction quelconque de X_2 dans $\mathbb{R}$. À moins que $f(\cdot)$ ne soit une constante, il y peu de chances pour que $v(\cdot)$ soit une transformée affine de $u(\cdot)$. Le problème vient du fait que, les facteurs d'utilité ayant des attributs en commun, il est possible d'opérer des transferts d'utilité d'un facteur à l'autre. En conséquence, si l'on veut obtenir une propriété d'unicité à une transformation affine strictement positive près, on doit impérativement se restreindre au sous-ensemble des utilités GAI décomposables interdisant ces transferts. La proposition suivante montre comment un tel sous-ensemble peut être déterminé. Notons que les hypothèses de cette proposition ne sont nullement restrictives en ce sens que si une relation de préférence $\succsim$ est représentable par une utilité GAI, il existera une fonction GAI vérifiant les propriétés de la proposition et représentant elle aussi la relation $\succsim$.

Proposition 1. *Soit $\mathcal{X} = \times_i X_i$ et soit $u(\cdot)$ une utilité décomposable selon un arbre GAI $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, où $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_k}\}$ est l'ensemble des cliques de $\mathcal{G}$ ordonnées de l'extérieur vers l'intérieur*[3]*, autrement dit $u(x) = \sum_{i=1}^{|\mathcal{C}|} u_i(x_{C_i})$. Soit x^0 un élément quelconque de $\mathcal{X}$. Alors :*

– propriété 1 : $\forall i \in \{1, \ldots, |\mathcal{C}|-1\}$, la clique X_{C_i} a exactement une clique voisine dans $\mathcal{G}_{\mathcal{C}_i}$, que l'on notera $X_{C_{n(i)}}$;

– propriété 2 : Soit $S_i = C_i \cap C_{n(i)}$ et $D_i = C_i \backslash S_i$. Il existe une utilité v GAI décomposable selon $\mathcal{G}$ telle que $\forall i < |\mathcal{C}|$ et $\forall x_{S_i}$, $v_i(x^0_{D_i}, x_{S_i}) = 0$, et telle que $v_{|\mathcal{C}|}(x^0_{C_{|\mathcal{C}|}}) = 0$. Une telle utilité est dite non transférable *par rapport à $(x^0, \mathcal{C})$;*

– propriété 3 : Si u et v sont des utilités GAI décomposables vérifiant la propriété 2 ci-dessus, alors il n'existe aucune fonction de transfert non nulle transformant u en v. Autrement dit, il n'existe pas de fonctions $f_i : S_i \mapsto \mathbb{R}$ telles que :

- $v_1(x_{C_1}) = u_1(x_{C_1}) - f_1(x_{S_1})$ *et* $v_k(x_{C_k}) = u_k(x_{C_k}) + \sum_{j \in n^{-1}(k)} f_j(x_{S_j})$,
- $v_i(x_{C_i}) = u_i(x_{C_i}) - f_i(x_{S_i}) + \sum_{j \in n^{-1}(i)} f_j(x_{S_j})$ $\forall i \geq 2$,
- $\exists\, i \in \{2, \ldots, k-1\}$ *et* $\exists\, x_{S_i} \in \times_{j \in C_i} X_j$ *tels que* $f_i(x_{S_i}) \neq 0$.

Démonstration. Propriété 1 : démonstration par récurrence. Si la clique X_{C_1} avait au moins deux voisins, X_{C_a} et X_{C_b}, alors ceux-ci appartiendraient à la même composante connexe de $\mathcal{G}$. Cependant, après la suppression de la clique X_{C_1} du graphe, elles appartiendraient à deux composantes connexes différentes sinon il y aurait eu un cycle dans $\mathcal{G}$, ce qui est impossible puisque, par hypothèse, $\mathcal{G}$ est un arbre. Mais alors si, après suppression de X_{C_1}, les cliques X_{C_a} et X_{C_b} appartiennent à deux composantes connexes différentes, cela implique que $\mathcal{G}_{C_1}$ n'est pas connexe, d'où une contradiction

3. Rappelons que les cliques de $\mathcal{C}$ sont ordonnées de l'extérieur vers l'intérieur si : pour tout i, si $\mathcal{G}_{\mathcal{C}_i}$ représente le sous-graphe de $\mathcal{G}$ induit par $\mathcal{C}_i = \mathcal{C} \backslash \{X_{C_j} : j < i\}$, alors $\mathcal{G}_{\mathcal{C}_i}$ est connexe.

avec la définition de $\mathcal{G}_{\mathcal{C}_i}$. Par conséquent, X_{C_1} a au plus un voisin. En outre, X_{C_1} a au moins un voisin sinon $\mathcal{G}$ ne serait pas connexe (c'est un arbre). Dans $\mathcal{G}_{C_1}$, les mêmes arguments s'appliquent à X_{C_2} et, par récurrence, la propriété 1 est démontrée.

Propriété 2 : soit u une utilité GAI décomposable selon $\mathcal{G}$. S_1 est l'intersection de C_1 et de son voisin $C_{n(1)}$. Donc les attributs dans X_{S_1} appartiennent aux n-uplets passés en argument des facteurs u_1 et $u_{n(1)}$. Ainsi, ajouter n'importe quelle fonction de X_{S_1} à valeurs dans $\mathbb{R}$ à u_1 ou $u_{n(1)}$ ne change pas les arguments passés à ces fonctions. Remarquons que $u_1(x^0_{D_1}, x_{S_1})$ ne dépend que de x_{S_1} puisque $x^0_{D_1}$ est une constante. Donc on peut soustraire cette fonction à u_1 et l'ajouter à $u_{n(1)}$ sans compromettre la représentabilité de la fonction d'utilité GAI décomposable. Mais $v_1(x_{C_1}) = u_1(x_{D_1}, x_{S_1}) - u_1(x^0_{D_1}, x_{S_1})$ est telle que $v_1(x^0_{D_1}, x_{S_1}) = 0$ pour tout x_{S_1}. Le même procédé peut être appliqué par récurrence à tous les u_i, $i < n$. En ce qui concerne u_n, notons que toute transformée affine strictement positive d'une fonction d'utilité est encore une fonction d'utilité. Donc soustraire à u_n la constante $u_{|\mathcal{C}|}(x^0_{C_{|\mathcal{C}|}})$ préserve la représentabilité de la fonction d'utilité et satisfait la condition selon laquelle $u_{|\mathcal{C}|}(x^0_{C_{|\mathcal{C}|}}) = 0$, ce qui achève la démonstration de la propriété 2.

Propriété 3 : supposons qu'il existe des fonctions f_i non nulles. $v_1(x_{D_1}, x_{S_1}) = u_1(x_{D_1}, x_{S_1}) - f_1(x_{S_1})$ pour tout $(x_{D_1}, x_{S_1}) \in X_{C_1}$, et en particulier, pour $x_{D_1} = x^0_{D_1}$, $v_1(x^0_{D_1}, x_{S_1}) = u_1(x^0_{D_1}, x_{S_1}) - f_1(x_{S_1})$ et ce quel que soit $x_{S_1} \in X_{S_1}$. Mais, d'après la propriété 2, $v_1(x^0_{D_1}, x_{S_1}) = u_1(x^0_{D_1}, x_{S_1}) = 0$. Donc $f_1(x_{S_1}) = 0 \; \forall x_{S_1} \in X_{S_1}$. Par récurrence, cette propriété est vraie pour tout $i \in \{1, \ldots, k\}$. □

Illustrons la proposition 1 sur le réseau GAI de la figure 1 : on peut prendre pour $\mathcal{C}$ l'ensemble $\{AB, CE, BCD, BDF, BG\}$. En effet, puisque AB et CE sont des cliques externes, les supprimer conserve la connexité du graphe. De même, après avoir supprimé AB et CE, la clique BCD devient externe puisque la connexité du graphe induit n'est pas remise en cause par sa suppression, et ainsi de suite. La propriété 1 stipule que la clique AB a seulement un voisin, ici BCD ; après avoir supprimé AB et CE, BCD est connectée à seulement une clique, en l'occurrence BDF, etc. La propriété 2 stipule que si $(a^0, b^0, c^0, d^0, e^0, f^0, g^0)$ est un élément quelconque de $\mathcal{X}$, alors il existe une utilité GAI décomposable $v(a, b, c, d, e, f, g) = v_1(a, b) + v_2(c, e) + v_3(b, c, d) + v_4(b, d, f) + v_5(b, g)$ telle que $v_1(a^0, b) = v_2(c, e^0) = v_3(b, c^0, d) = v_4(b, d^0, f^0) = v_5(b^0, g^0) = 0$. L'idée sous-jacente à cette propriété est simple : puisque les X_{S_i} sont des attributs appartenant aux séparateurs, la propriété peut être établie en transférant par l'intermédiaire du séparateur X_{S_i} une certaine quantité qui dépend seulement du X_{S_i} d'une clique à sa clique voisine. Par exemple, supposons que $v_1(a^0, b) \neq 0$ pour un b donné, alors en remplaçant $v_1(a, b)$ par $v_1(a, b) - v_1(a^0, b)$ pour tout $(a, b) \in A \times B$ et $v_3(b, c, d)$ par $v_3(b, c, d) + v_1(a^0, b)$ pour tout $(b, c, d) \in B \times C \times D$, la propriété 2 se trouve trivialement vérifiée pour v_1 et la fonction GAI décomposable ainsi obtenue est encore une utilité représentant $\succsim$. Le même procédé peut être appliqué sur la deuxième clique de $\mathcal{C}$ et, par récurrence, à toutes les cliques de $\mathcal{C}$. Quant à la dernière, en soustrayant la constante $v_5(b^0, g^0)$ de $v_5(b, g)$, on obtient la propriété 2.

Se focaliser sur les fonctions d'utilité non transférables constitue une première étape vers l'unicité à une transformation affine strictement positive près. Cependant, ce n'est pas encore suffisant pour assurer cette unicité, et des conditions structurelles doivent être ajoutées pour parvenir à ce résultat. Dans le cas des utilités additives, c'est la solvabilité restreinte qui joue ce rôle. Intuitivement, la solvabilité force les ensembles d'arrivée des facteurs à être des sous-ensembles *extrêmement réguliers* de $\mathbb{R}$, c'est-à-dire que s'ils ont des « trous », ils les ont tous aux mêmes endroits, et ces trous se retrouvent tous à des intervalles réguliers de $\mathbb{R}$. Par exemple, il serait impossible d'avoir l'ensemble d'arrivée d'un facteur égal à $[1, 3[\cup]3, 6]$ car, $[1, 3[$ étant un continuum, $\{3\}$ serait un trou non régulier : s'il n'existe pas de trou entre 1 et 2,5, il ne peut en exister entre 2,5 et 4. De la même manière, dans l'ensemble $\{1, 2, 3, 5, 6\}$, 4 serait un trou. La régularité des ensembles d'arrivée est précisément ce qui assure l'unicité à une transformation affine strictement positive près des fonctions d'utilité additives. Dans le cas des fonctions d'utilité GAI décomposables en général, la solvabilité restreinte ne suffit plus à garantir cette régularité. En effet, considérons l'exemple suivant : $\succsim$ est représenté sur $\mathbb{R} \times \mathbb{R}^* \times \mathbb{R}$ par la fonction d'utilité :

$$u(x_1, x_2, x_3) = \begin{cases} \pi + \text{arctg}(x_1 + \ln x_2) + \text{arctg}(x_3 + \ln x_2) & \text{si } x_2 > 0, \\ -\pi - \text{arctg}(x_1 + \ln(-x_2)) - \text{arctg}(x_3 + \ln(-x_2)) & \text{si } x_2 < 0. \end{cases}$$

Alors il est aisé de montrer qu'il y a solvabilité restreinte par rapport aux trois attributs X_1, X_2 et X_3. Cependant l'ensemble d'arrivée de la fonction d'utilité est $]-2\pi, 0[\cup]0, 2\pi[$ et possède donc un « trou » en 0. À cause de ce dernier, l'unicité à une transformation affine strictement positive près ne peut être assurée. En effet, la fonction suivante est aussi une fonction d'utilité quel que soit $\alpha, \beta \in \mathbb{R}_+^*$:

$$u(x_1, x_2, x_3) = \begin{cases} \alpha \times [\pi + \text{arctg}(x_1 + \ln x_2) + \text{arctg}(x_3 + \ln x_2)] & \text{si } x_2 > 0, \\ \beta \times [-\pi - \text{arctg}(x_1 + \ln(-x_2)) - \text{arctg}(x_3 + \ln(-x_2))] & \text{si } x_2 < 0. \end{cases}$$

Le problème vient ici du fait que, bien que la solvabilité assure la régularité des codomaines de la projection de la fonction d'utilité pour des valeurs fixées des séparateurs X_{S_i}, elle est incapable d'empêcher l'apparition de trous sur la réunion de ces codomaines. Dans l'exemple ci-dessus, séparément, les fonctions $\pi + \text{arctg}(x_1 + \ln x_2) + \text{arctg}(x_3 + \ln x_2)$ et $-\pi - \text{arctg}(x_1 + \ln(-x_2)) - \text{arctg}(x_3 + \ln(-x_2))$ ont des codomaines réguliers : $]0, 2\pi[$ et $]-2\pi, 0[$. Mais la réunion de ces deux ensembles fait apparaître un trou en 0. L'unicité à une transformation affine près passe donc par l'ajout d'un nouvel axiome empêchant ce genre de situation d'apparaître.

Axiome 5 (connexité des séparateurs)**.** *Soit $\mathcal{X} = \times_{i=1}^{n} X_i$ et soit $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ un arbre GAI, avec $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_k}\}$. Soit x un élément quelconque de $\mathcal{X}$. Soit X_{C_i} et X_{C_j} deux cliques, pas obligatoirement voisines dans l'arbre GAI, et soit $X_{S_{ij}} = X_{C_i \cap C_j}$ leur intersection/séparateur non vide. Notons $D_i = C_i \backslash S_{ij}$, $D_j = C_j \backslash S_{ij}$ et $E = \{1, \ldots, n\} \backslash (C_i \cup C_j)$.*

Alors, $\forall x_{S_{ij}}, y_{S_{ij}} \in X_{S_{ij}}$, il existe une suite finie $(z^k_{S_{ij}})_{k=0}^{p}$ d'éléments de $X_{S_{ij}}$ telle que $z^0_{S_{ij}} = x_{S_{ij}}$, $z^p_{S_{ij}} = y_{S_{ij}}$, et $\forall k \in \{2, \ldots, p\}$ $\exists\, a^k, b^k, c^k, d^k \in \mathcal{X}$ tels que :

1) $(a^k_{D_i}, z^{k-1}_{S_{ij}}, a^k_{D_j}, x_E) \sim (b^k_{D_i}, z^k_{S_{ij}}, b^k_{D_j}, x_E)$;

2) $(c^k_{D_i}, z^{k-1}_{S_{ij}}, c^k_{D_j}, x_E) \sim (d^k_{D_i}, z^k_{S_{ij}}, d^k_{D_j}, x_E)$ *;*
3) $(a^k_{D_i}, z^{k-1}_{S_{ij}}, a^k_{D_j}, x_E) \not\sim (c^k_{D_i}, z^{k-1}_{S_{ij}}, c^k_{D_j}, x_E)$.

La figure 4 illustre cet axiome : les hyperplans correspondant à des valeurs fixes $z^k_{S_{ij}}$ des séparateurs sont représentés par les rectangles. L'axiome 5 stipule que, si deux hyperplans ne sont pas trop « éloignés », on peut trouver deux points A et C dans le premier hyperplan, et deux points B et D dans le second tels que : $A \sim B$, $C \sim D$ et $A \not\sim C$. En termes de préférences, l'axiome de connexité des séparateurs stipule que les séparateurs ne jouent pas un rôle de « dictateur », c'est-à-dire que leur poids n'est pas suffisamment grand pour que, dès qu'un élément $x \in \mathcal{X}$ a x_{S_i} pour valeur d'un séparateur X_{S_i} donné, x est préféré à tout autre élément $y \in \mathcal{X}$ tel que $x_{S_i} \neq y_{S_i}$, et ce quelles que soient les valeurs des autres attributs.

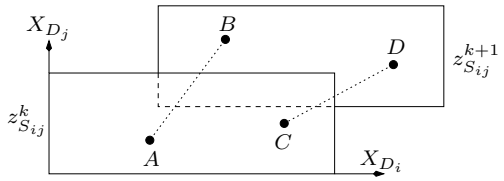

Figure 4. *L'axiome de connexité des séparateurs*

Muni de l'axiome de connexité des séparateurs, (Gonzales *et al.*, 2006) montre que la propriété d'unicité à une transformation affine strictement positive près peut être garantie :

Proposition 2. *Soit* $\succsim$ *une relation de préférence sur* $\mathcal{X} = \times_{i=1}^n X_i$ *représentable par une fonction d'utilité GAI décomposable selon un arbre GAI* $\mathcal{G}$*. Soit* $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_n}\}$ *l'ensemble des cliques de* $\mathcal{G}$ *ordonné des cliques extérieures de* $\mathcal{G}$ *vers les cliques intérieures. Soit* x^0 *un élément quelconque de* $\mathcal{X}$*. Supposons que la solvabilité restreinte soit vérifiée pour tout attribut et que l'axiome de connexité des séparateurs le soit aussi pour tout couple de cliques* X_{C_i}, X_{C_j} *d'intersection non vide. Alors les fonctions d'utilité GAI décomposables selon* $\mathcal{G}$ *et non transférables selon* $(x^0, \mathcal{C})$ *sont uniques à une transformation linéaire strictement positive près.*

3.4. *Une procédure générique d'élicitation d'utilités GAI décomposables*

À l'instar de la sous-section 3.1, nous allons présenter ici une procédure d'élicitation suffisamment générique pour ne pas être liée précisément à un questionnaire donné. Une instanciation de cette procédure sera présentée dans la sous-section suivante. Comme dans la sous-section 3.1, la procédure sera fondée sur un ensemble d'hypothèses, dont la première est l'unicité des utilités GAI décomposables à une transformation linéaire strictement positive près.

Hypothèse 3. *Les utilités GAI décomposables sont uniques à une transformation linéaire strictement positive près.*

Figure 5. *Élicitation d'une fonction d'utilité GAI décomposable*

Considérons l'arbre GAI de la figure 5. Soit $x^0 = (a^0, b^0, c^0, d^0)$ un élément quelconque du produit cartésien $A \times B \times C \times D$. Comme nous l'avions mentionné au début de la section 3, si l'on fixe la valeur du séparateur B à une valeur arbitraire b, on obtient une utilité additive $u_1(A, b) + [u_2(b, C) + u_3(C, D)]$. Par conséquent, en appliquant les résultats des sous-sections 3.1 et 3.2, on est capable d'éliciter le facteur $u_1(A, b)$ sur A. Il ne reste plus alors qu'à « recoller les morceaux » obtenus pour différentes valeurs de B pour construire l'utilité $u_1(\cdot)$ sur $A \times B$. Une fois celle-ci déterminée, le même procédé s'applique pour $u_2(\cdot)$. En effet, pour toute valeur c de C, l'utilité u se sépare additivement en $[u_1(A, B) + u_2(B, c)] + u_3(c, D)$. u_1 étant connu, la seule inconnue du premier terme est $u_2(B, c)$. En posant des questions dans lesquelles la valeur de A est toujours fixée à a^0, on peut même éliminer le terme $u_1(A, B)$ pour peu que l'on fasse en sorte de construire une fonction u non transférable par rapport à $(x^0, \{AB, BC, CD\})$. Cela suggère donc l'hypothèse suivante :

Hypothèse 4. *Soit $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_k}\}$ l'ensemble des cliques de l'arbre GAI $\mathcal{G}$ ordonné des cliques extérieures de $\mathcal{G}$ vers les cliques intérieures. Pour toute clique X_{C_i} de $\mathcal{C}$, appelons $X_{C_{n(i)}}$ sa clique voisine telle que $n(i) > i$ (cf. la proposition 1), $D_i = C_i \backslash C_{n(i)}$ et $S_i = C_i \cap C_{n(i)}$. Pour tout $i < k$ et toute valeur x_{S_i} de X_{S_i}, il existe un questionnaire permettant de construire le facteur d'utilité $u_i(X_{D_i}, x_{S_i})$. Il existe en outre un questionnaire permettant de construire u_k sur X_{C_k}.*

Reprenons l'exemple de la figure 5 et supposons que l'on ait élicité $u_1(A, b)$ et $u_1(A, b')$. Peut-on agréger ces deux morceaux de manière à former une utilité sur $A \times \{b, b'\}$? La réponse nous est fournie par l'observation suivante : dans les hyperplans « B = constante », la fonction d'utilité u est additivement décomposable. Elle est donc unique à une transformation affine près et donc, par non transférabilité, $u_1(A, b)$ et $u_1(A, b')$ sont uniques à une transformation linéaire près. Pour agréger $u_1(A, b)$ et $u_1(A, b')$, il faut et il suffit de multiplier $u_1(A, b')$ par un nombre $k \in \mathbb{R}_+^*$ de telle sorte que $u_1(A, b)$ et $ku_1(A, b')$ forment une utilité sur $A \times \{b, b'\}$. Trouver un tel k est aisé. En effet, si l'on peut trouver des éléments a^i, c^i, d^i tels que :

$$(a^1, b, c^1, d^1) \sim (a^2, b', c^2, d^2) \text{ et } (a^3, b, c^1, d^1) \sim (a^4, b', c^2, d^2), \qquad [2]$$

alors, en termes d'utilité, on obtient :

$$u_1(a^1, b) + u_2(b, c^1) + u_3(c^1, d^1) = ku_1(a^2, b') + u_2(b', c^2) + u_3(c^2, d^2),$$
$$u_1(a^3, b) + u_2(b, c^1) + u_3(c^1, d^1) = ku_1(a^4, b') + u_2(b', c^2) + u_3(c^2, d^2),$$

d'où, en soustrayant ces deux équations :

$$k = \frac{u_1(a^1, b) - u_1(a^3, b)}{u_1(a^2, b') - u_1(a^4, b')}.$$

On peut donc agréger les « morceaux d'utilité » obtenus pour différentes valeurs de B afin d'obtenir une utilité sur $A \times B$. Ceci nous conduit à l'hypothèse suivante :

Hypothèse 5. *À partir d'une collection de fonctions d'utilité u_i définies sur X_{D_i} pour différentes valeurs du séparateur $X_{S_i} = X_{C_i} \cap X_{C_{n(i)}}$, il existe un questionnaire permettant d'agréger cette collection pour former un facteur d'utilité sur X_{C_i}.*

Grâce aux hypothèses précédentes, nous sommes en mesure de définir de manière plus ou moins indépendante les facteurs u_1, u_2 et u_3. Mais rien ne garantit que la somme de ces facteurs soit une utilité sur $\mathcal{X}$. Or, d'après l'hypothèse 3, agréger ces différents facteurs u_i revient à les multiplier par de nouveaux nombres réels k_i strictement positifs. Un procédé similaire au précédent peut être appliqué pour déterminer les k_i. En effet, si l'on peut trouver des éléments a^i, b^i, c, c', d, d' tels que :

$$(a^1, b^1, c, d) \sim (a^2, b^2, c', d') \text{ et } (a^3, b^3, c, d) \sim (a^4, b^4, c', d'),$$

alors, en termes d'utilité, on obtient :

$$k_2 = \frac{u_1(a^2, b^2) + u_1(a^3, b^3) - u_1(a^1, b^1) - u_1(a^4, b^4)}{u_2(b^1, c) - u_2(b^2, c') + u_2(b^3, c) - u_2(b^4, c')}.$$

Ceci nous suggère notre dernière hypothèse :

Hypothèse 6. *À partir d'une collection de fonctions d'utilité u_i définies sur les différentes cliques X_{C_i}, il existe un questionnaire permettant d'agréger cette collection pour former une utilité sur $\mathcal{X}$.*

De ces hypothèses, on peut déduire la procédure générique d'élicitation ci-dessous. Dans celle-ci, $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_k}\}$ représente l'ensemble des cliques de l'arbre GAI ordonnées des cliques extérieures vers les cliques intérieures.

Algorithme 3 (procédure générique d'élicitation de fonctions d'utilité GAI)

01 choisir x^0 un élément quelconque de $\mathcal{X}$
02 **pour** i variant de 1 à k **faire**
03 éliciter les $u_i(X_{D_i}, x_{S_i}) : X_{D_i} \mapsto \mathbb{R}$ pour différentes valeurs du séparateur $X_{S_i} = X_{C_i} \cap X_{C_{n(i)}}$
04 agréger ces u_i pour former un facteur d'utilité u_i sur X_{C_i}
05 trouver $k_i \in \mathbb{R}_+^*$ tel que $\sum_{j<i} u_j + k_i u_i$ forme une utilité
06 **fait**

3.5. *Spécialisation de la procédure d'élicitation d'utilités GAI décomposables*

Nous nous intéressons à présent à proposer une solution pour les étapes 03, 04, 05 de l'algorithme ci-dessus, c'est-à-dire une méthode instanciant les hypothèses 5 et 6. Considérons donc un arbre GAI $\mathcal{G}$ et $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_k}\}$ l'ensemble de ses cliques ordonnées de l'extérieur vers l'intérieur, et choisissons arbitrairement $x^0 \in \mathcal{X}$, qui va nous servir de point de référence pour la non transférabilité. À l'instar de la proposition 1, appelons $X_{C_{n(i)}}$ la clique voisine de X_{C_i} dans $\mathcal{G}_{\mathcal{C}_i}$ et $X_{S_i} = X_{C_i} \cap X_{C_{n(i)}}$ le séparateur entre ces deux cliques. Si l'on fixe la valeur des attributs de ce séparateur,

disons à x_{S_i}, nous avons vu que nous obtenons une utilité additive puisque l'arête entre X_{C_i} et $X_{C_{n(i)}}$ disparaît. On peut donc appliquer les algorithmes des sous-sections 3.2 et 3.1 afin d'éliciter celle-ci. Évidemment, seul le facteur de cette utilité contenant les attributs de X_{C_i} nous intéresse. Nous utiliserons donc pour l'éliciter une séquence standard dont la maille correspond aux attributs du 2ème facteur. De plus, il est inutile de faire varier tous les attributs du 1er facteur puisque seul $u_i(X_{C_i})$ nous intéresse. Nous fixerons donc tous les autres attributs à leur valeur dans x^0. Ceci présente en outre l'avantage de supprimer tous les termes $u_j(X_{C_j})$, $j < i$, des équations. Les séquences standards nous donnent donc directement la valeur de $u_i(X_{D_i}, x_{S_i})$.

L'étape 04 de l'algorithme consiste à généraliser les indifférences de l'équation [2]. Or, celles-ci ont le bon goût de correspondre à celles de l'axiome 5 de connexité des séparateurs. En effet, si l'on note $D_i = C_i \backslash S_i$, et si l'on suppose que l'on a élicité $u_i(x_{D_i}, z_{S_i}^{k-1}) : X_{D_i} \mapsto \mathbb{R}$ et $u_i(x_{D_i}, z_{S_i}^{k}) : X_{D_i} \mapsto \mathbb{R}$ pour deux valeurs $z_{S_i}^{k-1}$ et $z_{S_i}^{k}$ du séparateur X_{S_i}, si l'on fixe les valeurs des attributs sur D_j et E dans l'axiome de connexité des séparateurs à x^0, l'élément de référence de la non transférabilité, les trois équations de l'axiome de connexité des séparateurs deviennent :

1) $(a_{D_i}^k, z_{S_i}^{k-1}, x_{-C_i}^0) \sim (b_{D_i}^k, z_{S_i}^k, x_{-C_i}^0)$;
2) $(c_{D_i}^k, z_{S_i}^{k-1}, x_{-C_i}^0) \sim (d_{D_i}^k, z_{S_i}^k, x_{-C_i}^0)$;
3) $(a_{D_i}^k, z_{S_i}^{k-1}, x_{-C_i}^0) \not\sim (c_{D_i}^k, z_{S_i}^{k-1}, x_{-C_i}^0)$.

En traduisant ces équations sous forme d'utilités et en soustrayant les deux premières, il est aisé de voir que tous les termes $u_p(\cdot)$, pour $p < i$, sont égaux à 0 (par non transférabilité) et que les termes $u_p(\cdot)$, pour $p > i$, s'annulent tous. On obtient donc que k, le nombre par lequel on doit multiplier la fonction $u_i(x_{D_i}, z_{S_i}^k)$, est égal à :

$$k = \frac{u_i(a_{D_i}^k, z_{S_i}^{k-1}) - u_i(c_{D_i}^k, z_{S_i}^{k-1})}{u_i(b_{D_i}^k, z_{S_i}^k) - u_i(d_{D_i}^k, z_{S_i}^k)},$$

et l'on peut ainsi construire une utilité sur X_{C_i} et donc réaliser l'étape 04.

L'étape 05 est assez similaire. En effet, supposons que tous les facteurs $u_p(\cdot)$, $p < i$ aient déjà été agrégés et supposons que l'on ait élicité un nouveau facteur $u_i(\cdot)$. Soit $S_1 = C_1 \cap C_{n(1)}$, autrement dit X_{S_1} est le séparateur entre X_{C_1} et son voisin dans l'arbre GAI. Soit $D_1 = C_1 \backslash S_1$, $E_1 = C_1 \backslash C_i$, $F_1 = C_1 \backslash E_1$ et $E = \{1, \ldots, n\} \backslash (D_1 \cup D_i)$. Autrement dit, X_{E_1} sont les attributs appartenant à X_{C_1} mais pas à X_{C_i}, X_{F_1} sont les attributs communs à X_{C_1} et X_{C_i}, et X_{D_1} sont les attributs n'appartenant qu'à la clique X_{C_1}. Alors, comme dans le paragraphe précédent, les trois préférences suivantes :

1) $(a_{D_1}, a_{D_i}, x_E^0) \sim (b_{D_1}, b_{D_i}, x_E^0)$,
2) $(c_{D_1}, c_{D_i}, x_E^0) \sim (d_{D_1}, d_{D_i}, x_E^0)$,
3) $(a_{D_1}, a_{D_i}, x_E^0) \not\sim (c_{D_1}, c_{D_i}, x_E^0)$,

impliquent par soustraction des utilités relatives aux deux premières préférences que :

$$k_i = \frac{u_1(b_{F_1}, x_{E_1}^0) - u_1(a_{F_1}, x_{E_1}^0) + u_1(c_{F_1}, x_{E_1}^0) - u_1(d_{F_1}, x_{E_1}^0)}{u_i(a_{D_i}, x_{S_i}^0) - u_i(b_{D_i}, x_{S_i}^0) - u_i(c_{D_i}, x_{S_i}^0) + u_i(d_{D_i}, x_{S_i}^0)}.$$

Enfin, pour clore cette section, mentionnons que ce processus peut être facilement adapté pour traiter les utilités décomposables selon des forêts GAI, c'est-à-dire les ensembles d'arbres GAI (comme sur la figure 6). En effet, la somme des utilités sur chaque composante connexe consiste en une utilité additive sur $\mathcal{X}$. Ainsi, pour éliciter une utilité dans cet espace il suffit d'appliquer le processus précédemment décrit sur chaque composante connexe et, ensuite, mettre sur la même échelle toutes ces sous-utilités comme nous le ferions pour une utilité additive ordinaire.

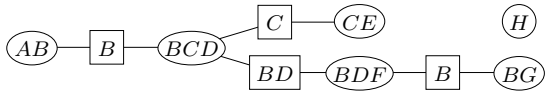

Figure 6. *Une forêt GAI*

4. Choix optimal

Une fois la fonction d'utilité du décideur élicitée, elle peut être employée pour des tâches de recommandation. Plusieurs questions intéressantes peuvent être distinguées :

– *questions globales de choix :* trouver le n-uplet x^* préféré dans l'ensemble $\mathcal{X}$;

– *questions de choix contraints :* trouver le n-uplet x^* préféré dans l'ensemble $\mathcal{X}$ sous la contrainte que certains attributs prennent des valeurs spécifiques ;

– *questions de comparaison :* trouver quelle est l'alternative préférée par le décideur parmi deux alternatives données $(x, y) \in \mathcal{X} \times \mathcal{X}$;

– *questions de rangement :* déterminer et ordonner les k n-uplets préférés de $\mathcal{X}$.

Remarquons que la deuxième question n'est qu'un cas simplifié de la première. Le troisième type n'est pas critique d'un point de vue algorithmique puisque, pour un couple (x, y) donné, cette question peut être résolue en calculant et comparant $u(x)$ et $u(y)$. En revanche, dès lors qu'il faut déterminer un meilleur élément ou classer les éléments du meilleur au pire, l'évaluation de la fonction u en chaque point $x \in \mathcal{X}$ est trop coûteuse et pose un problème computationnel. L'objet de cette section est de montrer comment la structure des réseaux GAI permet d'organiser les calculs efficacement en se ramenant à des séquences d'optimisations locales. Dans un premier temps, nous nous concentrons sur des questions globales de choix et nous présenterons ensuite une procédure de rangement. Pour la clarté de la présentation, la procédure est d'abord illustrée sur un exemple jouet puis un algorithme général est présenté.

Exemple 4 : Considérons le problème consistant à déterminer la configuration maximale pour l'ensemble $\mathcal{X} = A \times B \times C \times D \times E \times F \times G$ où $A = \{a^0, a^1, a^2\}$, $B = \{b^0, b^1\}$, $C = \{c^0, c^1\}$, $D = \{d^0, d^1\}$, $E = \{e^0, e^1, e^2\}$, $F = \{f^0, f^1\}$, $G = \{g^0, g^1\}$, et les préférences du décideur sont représentées par la fonction d'utilité $u(a, b, c, d, e, f, g) = u_1(a, b) + u_2(c, e) + u_3(b, c, d) + u_4(b, d, f) + u_5(b, g)$ avec les valeurs d'utilité de la figure 7. Remarquons que l'utilité u est complètement

caractérisée par seulement 32 valeurs tandis que le stockage de u en extension requiert $|\mathcal{X}| = 288$ entiers. Si les attributs étaient continus, les tables de la figure 7 seraient remplacées par des fonctions fournissant une représentation analytique de u. La figure 1 montre un réseau GAI représentant la décomposition u.

$u_1(a,b)$	b^0	b^1
a^0	8	2
a^1	4	3
a^2	1	7

$u_2(c,e)$	e^0	e^1	e^2
c^0	6	3	5
c^1	3	4	0

$u_3(b,c,d)$	b^0		b^1	
	d^0	d^1	d^0	d^1
c^0	0	2	7	1
c^1	5	1	2	4

$u_4(b,d,f)$	b^0		b^1	
	f^0	f^1	f^0	f^1
d^0	4	2	5	8
d^1	3	8	9	0

$u_5(b,g)$	g^0	g^1
b^0	0	9
b^1	6	4

Figure 7. *Valeurs d'utilité pour* $u(\cdot)$

Trouver la configuration optimale correspond à résoudre le problème suivant : $\max_{a,b,c,d,e,f,g} u_1(a,b)+u_2(c,e)+u_3(b,c,d)+u_4(b,d,f)+u_5(b,g)$. Les propriétés ci-dessous peuvent être exploitées pour résoudre efficacement ce problème :

1) le max sur un ensemble de variables $X_1,\ldots,X_n$ de $u(X_1,\ldots,X_n)$, peut être décomposé comme $\max_{X_1}\max_{X_2}\ldots\max_{X_n} u(X_1,\ldots,X_n)$ où l'ordre des max n'a aucune importance ;

2) si $u(X_1,\ldots,X_n)$ peut être décomposé comme $f()+g()$ où $f()$ ne dépend pas de la variable X_i, alors $\max_{X_i}[f()+g()] = f()+\max_{X_i} g()$;

3) dans un réseau GAI, la propriété d'intersection courante garantit qu'une variable appartenant à une clique externe X_C et qui n'appartient pas à la clique voisine de X_C n'apparaît dans aucune autre clique du réseau GAI.

Les propriétés 2 et 3 suggèrent une stratégie dans laquelle, pour calculer l'utilité maximale, on maximise en jouant d'abord sur les variables figurant uniquement dans les cliques externes, on transmet ensuite les résultats à la clique voisine en éliminant les cliques externes. On itère ainsi ce processus de l'extérieur vers l'intérieur du réseau, jusqu'à ce que toutes les cliques soient éliminées. Dans l'exemple, on résout le problème d'optimisation :

$$\begin{array}{c}\max_{b,c,d}[\, u_3(b,c,d)+\max_f[u_4(b,d,f)+\max_g u_5(b,g)] \\ +[\max_e u_2(c,e)]+[\max_a u_1(a,b)]]\end{array} \qquad [3]$$

à travers des opérations suivantes :

1) dans la clique AB, calculer $u_1^*(b)=\max_{a\in A} u_1(a,b)$ pour tout $b\in B$;

2) dans la clique CE, calculer $u_2^*(c)=\max_{e\in E} u_2(c,e)$ pour tout $c\in C$;

3) dans la clique BG, calculer $u_5^*(b)=\max_{g\in G} u_5(b,g)$ pour tout $b\in B$;

4) dans la clique BDF, substituer $u_4(b,d,f)$ par $u_4(b,d,f) + u_5^*(b)$ pour tout n-uplet $(b,d,f) \in B \times D \times F$. Ensuite, calculer $u_4^*(b,d) = \max_{f \in F} u_4(b,d,f)$ pour tout n-uplet $(b,d) \in B \times D$;

5) dans la clique BCD, substituer $u_3(b,c,d)$ par $u_3(b,c,d) + u_1^*(b) + u_2^*(c) + u_4^*(b,d)$ pour tout n-uplet $(b,c,d) \in B \times C \times D$. Ensuite, calculer $\max_{b,c,d} u_3(b,c,d)$, l'utilité maximale du réseau GAI (34, dans l'exemple).

La figure 8 montre le contenu des u_i^* et u_i après substitution. À la fin de l'étape 5 nous avons calculé la valeur maximale de l'utilité, ici 34, définie par l'équation [3].

	b^0	b^1
$u_1^*(b)$	8	7

	c^0	c^1
$u_2^*(c)$	6	4

	b^0	b^1
$u_5^*(b)$	9	6

$u_4(b,d,f)$	b^0		b^1	
	f^0	f^1	f^0	f^1
d^0	13	11	11	14
d^1	12	17	15	6

$u_4^*(b,d)$	b^0	b^1
d^0	13	14
d^1	17	15

$u_3(b,c,d)$	b^0		b^1	
	d^0	d^1	d^0	d^1
c^0	27	33	34	29
c^1	30	30	27	30

Figure 8. *Contenu des u_i^* et u_i après les substitutions*

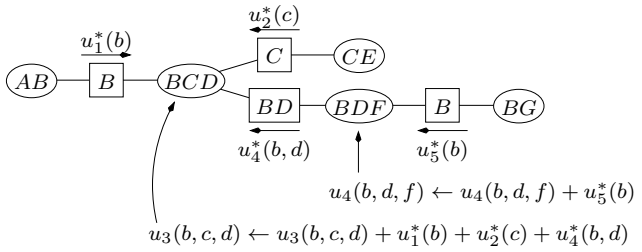

Figure 9. *Étapes 1 à 5 pour calculer l'utilité à l'optimum*

Au terme de cette phase de collecte de valeurs, on dispose donc de la valeur optimale de la fonction u sur $\mathcal{X}$. Pour déterminer à quelle configuration des attributs correspond cette valeur, il suffit de réaliser une phase d'instanciation des attributs qui consiste à propager en sens inverse du sens de la collecte les arguments des calculs opérés. Ainsi, à la dernière étape de notre phase de collecte, on voit que l'utilité 34 correspond en u_3 au n-uplet (b^1, c^0, d^0), ce qui permet de déduire que, dans la configuration optimale, on a $B = b^1, C = c^0, D = d^0$. À l'étape 4, $u_4^*(b^1, d^0)$ correspond à $u_4(b^1, d^0, f^1) = 14$ ce qui implique que $F = f^1$. Ensuite, à l'étape 3, on constate que $u_5^*(b^1) = 6$ correspond à $u_5(b^1, g^0)$ et par conséquent $G = g^0$, ce qui achève de caractériser le n-uplet optimal. Pour procéder de manière efficace il faut, lors du calcul de chaque u_i^*, sauvegarder une fonction $M_i^* : X_{S_i} \mapsto X_{D_i}$, où X_{S_i} et X_{D_i} sont définis comme à la fin de la sous-section 2.2, M_i^* mémorisant les arguments pour lesquels u_i^* est atteint. Dans l'exemple :

1) $M_1^*(b) = \text{Argmax}_{a \in A} u_1(a,b)$ pour tout $b \in B$;

2) $M_2^*(c) = \text{Argmax}_{e \in E} u_2(c, e)$ pour tout $c \in C$;

3) $M_5^*(b) = \text{Argmax}_{g \in G} u_5(b, g)$ pour tout $b \in B$;

4) $M_4^*(b, d) = \text{Argmax}_{f \in F} u_4(b, d, f)$ pour tout $(b, d) \in B \times D$;

5) $\text{Argmax}_{b,c,d} u_3(b, c, d)$.

Ainsi, pour trouver les valeurs des attributs pour la configuration optimale, il suffit de consulter les fonctions M_i^*. La phase d'instanciation reprend en sens inverse les étapes de la phase de collecte. Ainsi, dans l'exemple :

– *Étape 5 :* $\text{Argmax}_{b,c,d} u_3(b, c, d) = (b^1, c^0, d^0)$;

– *Étape 4 :* $M_4^*(b^1, d^0) = \text{Argmax}_{f \in F} u_4(b^1, d^0, f) = f^1$;

– *Étape 3 :* $M_5^*(b^1) = \text{Argmax}_{g \in G} u_5(b^1, g) = g^0$;

– *Étape 2 :* $M_2^*(c^0) = \text{Argmax}_{e \in E} u_2(c^0, e) = e^0$;

– *Étape 1 :* $M_1^*(b^1) = \text{Argmax}_{a \in A} u_1(a, b^1) = a^2$.

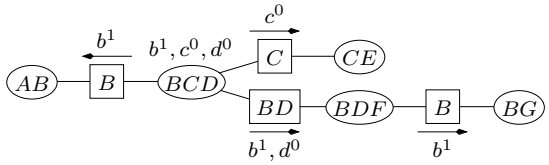

Figure 10. *Étapes 5 à 1 pour calculer l'instanciation à l'optimum*

La configuration optimale est donc $(a^2, b^1, c^0, d^0, e^0, f^1, g^0)$. ♦

La fonction `Optimal_choice` ci-dessous implante la procédure utilisée pour trouver la configuration maximale. Son principe est similaire à celui utilisé pour trouver l'explication la plus probable dans un réseau bayésien (Jensen, 1996; Cowell *et al.*, 1999). L'algorithme possède deux phases : d'abord la fonction `Collect` qui calcule les utilités maximales et ensuite la fonction `Instantiate` qui calcule la configuration optimale correspondante. Dans les fonctions ci-dessous, nous supposerons que l'ensemble des cliques $\mathcal{C} = \{X_{C_1}, \ldots, X_{C_k}\}$ est ordonné dans l'ordre inverse des appels de la fonction `Collect` (ce qui garantit que les cliques sont effectivement ordonnées de l'extérieur vers l'intérieur). X_{C_k} est donc la la dernière clique de $\mathcal{C}$. Enfin, dans la fonction `Optimal_choice`, la valeur de x_{C_k} passée en paramètre lors de l'appel à `Instantiate` est une instanciation quelconque des attributs de X_{C_k}.

Fonction `Optimal_choice`($\mathcal{G}$)

01 appeler `Collect`(X_{C_k}, X_{C_k})

02 appeler `Instantiate`(X_{C_k},X_{C_k},x_{C_k},$\emptyset$)

03 retourner les $x^*_{C_i}$ calculés par `Instantiate` qui composent la configuration optimale

`Collect` effectue la collecte en X_{C_i} tout en évitant la clique X_{C_r}. Cela permet de ne pas boucler indéfiniment sur la boucle 01-06.

Fonction `Collect`(X_{C_i}, X_{C_r})
01 **pour toutes** les cliques $X_{C_j} \in \{$cliques adjacentes à $X_{C_i}\} \backslash \{X_{C_r}\}$ **faire**
02 appeler `Collect` (X_{C_j}, X_{C_i})
03 **pour tout** $x_{C_i} \in X_{C_i}$ **faire**
04 $u_i(x_{C_i}) \leftarrow u_i(x_{C_i}) + u_j^*(x_{S_j})$ où x_{S_j} est la projection de x_{C_i} sur $X_{C_i \cap C_j}$
05 **fait**
06 **fait**
07 **si** $X_{C_i} \neq X_{C_r}$ **alors**
08 **pour tout** $x_{S_i} \in X_{S_i}$ **faire** $u_i^*(x_{S_i}) \leftarrow \max_{x_{D_i}} u_i(x_{C_i})$ **fait**
09 **fin si**

La fonction `Instantiate` réalise l'instanciation des attributs. À l'instar de la fonction `Collect`, son second argument, X_{C_r}, évite que la boucle 07-09 soit infinie. L'argument $x^*_{C_r}$ contient les valeurs optimales des attributs de X_{C_r}. Enfin, *Forbidden* désigne un ensemble de configurations interdites de la clique X_{C_i}. Ce champ ne nous servira que pour le rangement, c'est-à-dire dans la section suivante.

Fonction `Instantiate`(X_{C_i}, X_{C_r}, $x^*_{C_r}$, *Forbidden*)
01 **si** $X_{C_i} = X_{C_r}$ **alors**
02 $x^*_{C_i} \leftarrow$ Argmax$\{u_i(x_{C_i}) : x_{C_i} \in X_{C_i} \setminus$ *Forbidden*$\}$
03 **sinon**
04 $x^*_{S_i} \leftarrow$ la projection de $x^*_{C_r}$ sur $X_{C_i \cap C_r}$
05 $x^*_{C_i} \leftarrow$ Argmax$\{u_i(x^*_{S_i}, x_{D_i}) : x_{D_i} \in X_{D_i}$ et $(x^*_{S_i}, x_{D_i}) \notin$ *Forbidden*$\}$
06 **finsi**
07 **pour toutes** les cliques X_{C_j} dans $\{$cliques adjacentes à $X_{C_i}\} \backslash \{X_{C_r}\}$ **faire**
08 appeler `Instantiate` (X_{C_j}, X_{C_i}, $x^*_{C_i}$, $\emptyset$)
09 **fait**
10 retourner les valeurs des attributs trouvées au point maximal.

5. Rangement

Pour réaliser un rangement des meilleures solutions il faut, après avoir déterminé le n-uplet optimal comme indiqué dans la section précédente, être capable de trouver le deuxième meilleur élément, puis le suivant, etc. Le deuxième meilleur élément, x^1, se distingue du premier x^* par au moins un attribut, autrement dit il y a une clique X_{C_i} telle que la projection de x^1 sur X_{C_i} diffère de celle de x^*. Comme nous ne savons pas dans quelle clique X_{C_i} se produit cette différence, on considère toutes les possibilités en divisant l'espace d'alternatives restantes selon la partition proposée par (Nilsson, 1998) :

E1 : $(B, C, D) \neq (b^1, c^0, d^0)$
E2 : $(B, C, D) = (b^1, c^0, d^0)$ et $(B, D, F) \neq (b^1, d^0, f^1)$
E3 : $(B, C, D, F) = (b^1, c^0, d^0, f^1)$ et $(B, G) \neq (b^1, g^0)$
E4 : $(B, C, D, F, G) = (b^1, c^0, d^0, f^1, g^0)$ et $(C, E) \neq (c^0, e^0)$
E5 : $(B, C, D, E, F, G) = (b^1, c^0, d^0, e^0, f^1, g^0)$ et $(A, B) \neq (a^2, b^1)$

Trouver le choix optimal dans l'ensemble E1 est équivalent à résoudre :

$$\max_{(b,c,d)\neq(b^1,c^0,d^0)} [\, u_3(b,c,d) + \max_f(u_4(b,d,f) + \max_g u_5(b,g)) + (\max_e u_2(c,e)) + (\max_a u_1(a,b))\,].$$

Ceci suggère d'utiliser la fonction `Collect` comme dans la section précédente et ensuite d'appeler `Instantiate` en interdisant le choix de (b^1, c^0, d^0) dans l'étape 5 : c'est-à-dire qu'on appelle `Instantiate`$(BCD, BCD, (b^1, c^0, d^0), \{(b^1, c^0, d^0)\})$. De la même manière, pour trouver le choix optimal dans l'ensemble E2 on doit résoudre :

$$u_3(b^1, c^0, d^0) + \max_{f\neq f^1}[u_4(b^1, d^0, f) + \max_g u_5(b^0, g)] + [\max_e u_2(c^0, e)] + [\max_a u_1(a, b^0)].$$

Cela consiste à appeler la fonction `Collect` comme précédemment, puis `Instantiate`$(BDF, BCD, (b^1, c^0, d^0), \{(b^1, d^0, f^1)\})$. Il est clair que cet appel n'instanciera que les cliques BDF et BG, les autres cliques n'étant pas appelées par la fonction puisque, BCD étant passé en 2ème argument, la ligne 07 de `Instantiate` interdira tout appel récursif passant par la clique BCD. Pour former le n-uplet recherché, on affectera aux attributs de ces cliques les valeurs qu'ils avaient à l'optimum. Ainsi `Instantiate`$(BDF, BCD, (b^1, c^0, d^0), \{(b^1, d^0, f^1)\})$ est précisément ce qui est nécessaire pour déterminer le choix optimal dans E2. Naturellement, cela se généralise aux autres ensembles et, dans notre exemple, nous obtenons :

E1 : `Instantiate`$(BCD, BCD, (b^1, c^0, d^0), \{(b^1, c^0, d^0)\})$, $u(x) = 33$,
$x = (a^0, b^0, c^0, d^1, e^0, f^1, g^1)$,
E2 : `Instantiate`$(BDF, BCD, (b^1, c^0, d^0), \{(b^1, d^0, f^1)\})$, $u(x) = 31$,
$x = (a^2, b^1, c^0, d^0, e^0, f^0, g^0)$,
E3 : `Instantiate`$(BG, BDF, (b^1, d^0, f^1), \{(b^1, g^0)\})$, $u(x) = 32$,
$x = (a^2, b^1, c^0, d^0, e^0, f^1, g^1)$,
E4 : `Instantiate`$(CE, BCD, (b^1, c^0, d^0), \{(c^0, e^0)\})$, $u(x) = 33$,
$x = (a^2, b^1, c^0, d^0, e^2, f^1, g^0)$,
E5 : `Instantiate`$(AB, BCD, (b^1, c^0, d^0), \{(a^2, b^1)\})$, $u(x) = 30$
$x = (a^1, b^1, c^0, d^0, e^0, f^1, g^0)$.

Le deuxième n-uplet préféré est ainsi le choix optimal dans E1 ou E4. Arbitrairement, nous choisissons le n-uplet dans E1. Le prochain n-uplet, x^2, est le n-uplet préféré qui est donc différent de $(a^2, b^1, c^0, d^0, e^0, f^1, g^0)$ et de $(a^0, b^0, c^0, d^1, e^0, f^1, g^1)$. Il peut être recherché en utilisant le même processus. Comme x^1 est dans E1, nous devons partitionner E1 de la manière suivante pour exclure x^1 :

E1.1 : $(B, C, D) \notin \{(b^1, c^0, d^0), (b^0, c^0, d^1)\}$
E1.2 : $(B, C, D) = (b^0, c^0, d^1)$ et $(B, D, F) \neq (b^0, d^1, f^1)$
E1.3 : $(B, C, D, F) = (b^0, c^0, d^1, f^1)$ et $(B, G) \neq (b^0, g^1)$
E1.4 : $(B, C, D, F, G) = (b^0, c^0, d^1, f^1, g^1)$ et $(C, E) \neq (c^0, e^0)$
E1.5 : $(B, C, D, E, F, G) = (b^0, c^0, d^1, e^0, f^1, g^1)$ et $(A, B) \neq (a^0, b^0)$

et ensuite réitérer le même processus. Ce découpage peut être implanté grâce à la fonction `Split` ci-dessous : chaque ensemble de la partition courante est décrit par

un quadruplet de la forme $(X_{C_i}, \textit{Forbidden}_{C_i}, z^*, u^*)$, où X_{C_i} correspond à la clique sur laquelle on impose les contraintes d'exclusion, $\textit{Forbidden}_{C_i}$ désigne l'ensemble des $|C_i|$-uplets interdits pour la clique X_{C_i}, z^* est un n-uplet optimal dans l'ensemble considéré et u^* est égal à la valeur de l'utilité en z^*. Par exemple, les ensembles E1 à E5 ci-dessus sont représentés par :

$$\begin{array}{ll}
\text{E1} : (BCD, \{(b^1, c^0, d^0)\}, & (a^0, b^0, c^0, d^1, e^0, f^1, g^1), 33),\\
\text{E2} : (BDF, \{(b^1, d^0, f^1)\}, & (a^2, b^1, c^0, d^0, e^0, f^0, g^0), 31),\\
\text{E3} : (BG, \quad \{(b^1, g^0)\}, & (a^2, b^1, c^0, d^0, e^0, f^1, g^1), 32),\\
\text{E4} : (CE, \quad \{(c^0, e^0)\}, & (a^2, b^1, c^0, d^0, e^2, f^1, g^0), 33),\\
\text{E5} : (AB, \quad \{(a^2, b^1)\}, & (a^1, b^1, c^0, d^0, e^0, f^1, g^0), 30).
\end{array}$$

Lorsque l'on veut découper un ensemble, comme par exemple E1, la fonction `Split` doit être appelée en passant en paramètre la clique X_{C_i} de l'ensemble, les $|C_i|$-uplets interdits, le n-uplet optimal dans cet ensemble, ainsi qu'un booléen fixé à `vrai`. Ce booléen permet de différencier le premier appel à `Split` des appels récursifs vers les cliques extérieures.

Fonction `Split`(X_{C_i}, $\textit{Forbidden}_{C_i}$, y^*, *is_root*)
01 **si** *is_root* = vrai **alors**
02 $\quad \textit{Forbidden}_{C_i} \leftarrow \textit{Forbidden}_{C_i} \cup \{y^*_{C_i}\}$
03 **sinon**
04 $\quad \textit{Forbidden}_{C_i} \leftarrow \{y^*_{C_i}\}$
05 **finsi**
06 $z^* \leftarrow$ `Instantiate`$(X_{C_i}, X_{C_{n(i)}}, y^*_{C_{n(i)}}, \textit{Forbidden}_{C_i})$
07 compléter les attributs manquants dans z^* avec leurs valeurs correspondantes dans y^*
08 $u^* \leftarrow u(z^*)$
09 $\mathcal{S} \leftarrow \{(X_{C_i}, \textit{Forbidden}_{C_i}, z^*, u^*)\}$
10 **pour toutes** les cliques X_{C_j} dans {cliques adjacentes à X_{C_i}}$\backslash\{X_{C_{n(i)}}\}$ **faire**
11 $\quad \mathcal{S} \leftarrow \mathcal{S} \cup$ `Split`$(X_{C_j}, \emptyset, y^*,$ faux$)$
12 **fait**
13 retourner $\mathcal{S}$

En utilisant la fonction `Split`, on peut définir la fonction de rangement `k-best`.

Fonction `k-best`($\mathcal{G}$, K)
01 $x^* \leftarrow$ `Optimal_choice`$(\mathcal{G})$
02 $\mathcal{S} \leftarrow$ `Split`$(X_{C_k}, \emptyset, x^*,$ vrai$)$
03 $i \leftarrow 1$
04 **tant que** $i < K$ et $\mathcal{S} \neq \emptyset$ **faire**
05 $\quad S \leftarrow$ l'élément $(X_{C_j}, \textit{Forbidden}_{C_j}, z^*, u^*)$ de $\mathcal{S}$ ayant la plus grande valeur de u^*
06 $\quad x^i \leftarrow z^*$
07 $\quad \mathcal{S} \leftarrow \mathcal{S} \cup$ `Split`$(X_{C_j}, \textit{Forbidden}_{C_j}, z^*,$ vrai$) \setminus \{S\}$
08 $\quad i \leftarrow i + 1$
08 **fait**
09 retourner $(x^*, x^1, \ldots, x^{i-1})$

La complexité de cet algorithme se déduit directement de celle donnée par (Nilsson, 1998), soit :

$2k\bar{c} + 2kK\bar{d} + kK \log kK$, où :

– k = le nombre de cliques de l'arbre GAI ;

– K = la taille de la tête de classement recherchée (top-K éléments) ;

– $\bar{c} = \frac{\sum_{j=1}^{k} |X_{C_j}|}{k}$ et $|X_{C_j}| = |\{x_{C_j} \in X_{C_j}\}|$ représente le nombre de configurations d'une clique ;

– $\bar{d} = \frac{|X_{C_k}| + \sum_{j=1}^{k-1} |X_{D_j}|}{k}$ et $|X_{D_j}| = |\{x_{D_j} \in X_{D_j}\}|$.

En pratique, on peut toutefois accélérer `k-best` en implantant la fonction `Instantiate` à l'aide de tableaux d'utilités triées. En effet, la majorité des opérations réalisées dans `Instantiate` consiste à calculer les Argmax des lignes 02 et 05. Or, les expérimentations montrent que l'on calcule souvent des Argmax avec les mêmes valeurs de séparateurs $x^*_{S_i}$ sur la ligne 05 et/ou des Argmax « inconditionnels » sur la ligne 02, tous ceux-ci différant uniquement par la liste des $|C_i|$-uplets interdits. Cela suggère donc de préparer ces calculs en triant les portions d'utilités sur lesquelles portent ces Argmax. Autrement dit :

1) lorsque l'on calcule $x^*_{C_k}$ en ligne 02, on vérifie si l'on a déjà trié la table d'utilité de la clique X_{C_k} par ordre décroissant d'utilité. Si ce n'est pas le cas, on le fait et le $x^*_{C_k}$ correspond alors au premier élément de cette table. On place alors un curseur sur cet élément. Sinon, il suffit d'incrémenter la valeur du dernier curseur placé et de retourner l'élément $x^*_{C_k}$ correspondant.

2) lorsque l'on calcule $x^*_{C_i}$ en ligne 05, on vérifie de la même manière si, pour $x^*_{S_i}$ fixé, la sous-table d'utilité de $u_i(x^*_{S_i}, x_{D_i})$ est triée sur X_{D_i} et on effectue les mêmes opérations que ci-dessus.

Ainsi les Argmax des lignes 02 et 05 peuvent être calculés en O(1) simplement par incrémentation de curseurs dans des tables. En outre, la ligne 05 de la fonction `k-best` peut être réalisée en O(1) pourvu que l'on stocke l'ensemble $\mathcal{S}$ sous forme d'un tas trié par ordre décroissant de u^*.

5.1. *Intégration de contraintes*

Jusqu'à présent, nous avons considéré que toutes les configurations du produit cartésien $\mathcal{X}$ étaient réalisables. Toutefois, dans de nombreuses situations, certaines de ces configurations ne sont pas possibles ou pas disponibles. Ces contraintes peuvent être intégrées de façon directe dans le modèle GAI en ajoutant des nouveaux termes d'utilité dans lesquels la valeur est 0 pour les configurations possibles et $-\infty$ dans le cas contraire. La fonction d'utilité joue alors le rôle des fonctions d'appartenance dans les CSP flexibles (Larrosa *et al.*, 2005; Schiex *et al.*, 1997), les préférences étant vues comme des contraintes flexibles et les contraintes de faisabilité comme des contraintes

dures. Supposons que les paires (a^0, e^1), (a^1, e^2), (a^2, e^1) et (a^2, e^2) soient impossibles, on doit alors ajouter le terme d'utilité $u_6(a, e)$ avec les valeurs :

$$\begin{array}{lll} u_6(a^0, e^0) = -\infty; & u_6(a^0, e^1) = 0; & u_6(a^0, e^2) = 0; \\ u_6(a^1, e^0) = 0; & u_6(a^1, e^1) = 0; & u_6(a^1, e^2) = -\infty; \\ u_6(a^2, e^0) = 0; & u_6(a^2, e^1) = -\infty; & u_6(a^2, e^2) = -\infty; \end{array}$$

De même, si les paires (e^0, f^0) et (e^1, f^1) sont aussi interdites, on ajoute un terme d'utilité $u_7(e, f)$ avec les valeurs :

$$\begin{array}{lll} u_7(e^0, f^0) = -\infty; & u_7(e^1, f^0) = 0; & u_7(e^2, f^0) = 0; \\ u_7(e^0, f^1) = 0; & u_7(e^1, f^1) = -\infty; & u_7(e^2, f^1) = 0; \end{array}$$

Après l'addition des termes d'utilité permettant d'exprimer les contraintes, on a de nouvelles cliques à gérer et il est nécessaire de construire un nouvel arbre de jonction. Pour cela, on construit d'abord le réseau de Markov de la fonction d'utilité incluant les facteurs dus aux contraintes. Ce réseau de Markov est un graphe dans lequel les nœuds correspondent aux attributs et les arêtes lient chaque paire d'attributs apparaissant dans un même facteur d'utilité. Ce graphe est ensuite triangulé et le graphe résultant sert pour construire le nouvel arbre de jonction qui sera utilisé pour les calculs (cette procédure est classiquement utilisée pour les réseaux bayésiens, voir par ex. (Jensen, 1996)). La figure 11 montre l'arbre GAI de l'exemple après l'introduction des termes u_6 et u_7.

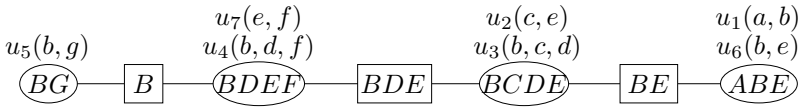

Figure 11. *L'arbre GAI après l'introduction des contraintes*

On remarque que les nouvelles dépendances entre variables introduites par u_6 et u_7 ont créé un arbre avec des cliques plus grandes qu'avant. Ceci demande plus d'espace pour stocker les fonctions relatives aux cliques. Toutefois il est possible d'utiliser une implantation « *lazy* », dans laquelle les utilités intervenant dans chaque clique ne sont pas fusionnées mais gardées dans leur forme originelle et stockées dans des listes chaînées (dans notre exemple, les termes d'utilité composant chaque clique sont indiqués sur la figure 11).

La section suivante présente quelques résultats expérimentaux obtenus avec des réseaux GAI.

6. Expérimentations

Pour tester l'algorithme d'énumération des k-meilleures solutions, nous avons exécuté ce dernier sur différents jeux de données générés automatiquement. Les temps de

calculs pour trouver le meilleur élément ainsi que les 50 et 100 meilleurs ont été enregistrés. On a créé 5 classes de problèmes $K_1, \ldots, K_5$, caractérisées par le nombre d'attributs, la taille du domaine de chaque attribut, le nombre de contraintes, la taille de la décomposition (c'est-à-dire le nombre de termes d'utilité), et l'arité des termes d'utilité (ainsi que des contraintes) :

	Nb. Attr.	Taille Dom.	Nb. Contr.	Taille decomp.	Arité des termes
K_1	40	2	5	5	4
K_2	35	5	30	30	2
K_3	15	10	20	20	2
K_4	30	5	5	5	4
K_5	10	10	5	5	3

Pour chaque classe on a généré aléatoirement 50 instances avec différents nombres de n-uplets interdits par contrainte. Les temps d'exécution sont d'abord donnés sur des produits cartésiens complets (sans contraintes). Ensuite, des contraintes de faisabilité ont été introduites dans le réseau par adjonction de nouveaux termes d'utilité, nous avons triangulé le réseau pour produire un nouvel arbre de jonction et réitéré l'algorithme de rangement. Les expérimentations ont été faites avec un programme en Java en utilisant un PC 2.4GHz. Les temps moyens sur les 50 instances sont résumés dans le tableau 1. Dans ce tableau, l'implantation utilise une stratégie *lazy*. À titre de comparaison, nous indiquons également les temps de réponse de Toolbar[4] pour trouver la première solution sur les mêmes problèmes (colonne TBBest). Toolbar est un logiciel de référence (librement disponible, implanté en langage C (de Givry *et al.*, 2005)) dans le domaine des contraintes flexibles. En revanche, Toolbar ne permet pas d'énumérer les k-meilleures solutions, ce qui ne nous a pas permis de nous comparer en termes de rangement (notons que l'ajout de contraintes interdisant la meilleure solution déjà générée n'est pas possible car cela revient à ajouter une contrainte sur tous les attributs, ce qui a pour conséquence une explosion de l'utilisation mémoire).

Time	sans contraintes				avec contraintes			
(in ms)	TBBest	Best	Top 50	Top 100	TBBest	Best	Top 50	Top 100
K_1	6153	8	44	82	2047	28	64	102
K_2	< 10	2	8	17	< 10	65	100	133
K_3	< 10	4	9	14	< 10	1272	1334	1365
K_4	35510	40	65	82	11257	2553	2647	2702
K_5	60	625	719	780	72	7858	7971	8062

Tableau 1. *Temps d'exécution (en ms) pour les différentes classes de problèmes*

Dans le cas de produits cartésiens complets, on peut remarquer que les algorithmes fondés sur les réseaux GAI sont très efficaces pour résoudre les problèmes de choix et de rangement. Quand des contraintes sont ajoutées, les temps de calcul demeurent

4. `http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro`

satisfaisants en dépit de la complexité additionnelle. En particulier, on peut remarquer que le rangement des éléments de la deuxième à la centième position ne prend pas plus de temps que l'obtention du premier élément. L'augmentation du temps de calcul observée après avoir ajouté des contraintes est causée par les dépendances inter-attributs additionnelles induites par ces contraintes. Ces nouvelles dépendances provoquent la réunion de cliques initialement disjointes, augmentant ainsi la taille moyenne des cliques, et dégradant du même coup les performances de l'algorithme de choix ou de rangement. Cependant, dans des applications pratiques, l'arité des facteurs de sous-utilité est généralement petite (≤ 3 dans la plupart des cas) ; le temps de calcul dépend donc essentiellement de l'arité des contraintes de faisabilité.

7. Conclusion

Dans cet article les résultats obtenus sont doubles : 1) sous une hypothèse de solvabilité, une approche générale pour l'élicitation d'utilités GAI décomposables sur un produit cartésien a été proposée ; 2) nous avons montré comment les réseaux GAI peuvent être exploités pour résoudre efficacement des problèmes de choix et de rangement sur un espace de nature combinatoire. On imagine facilement de nombreuses applications dans des systèmes de recommandation ou d'aide à la décision.

Bien entendu, diverses sophistications de notre approche sont encore possibles. Il serait par exemple intéressant de chercher à utiliser des arbres ET/OU (Dechter, 2004; Marinescu *et al.*, 2004) dans la structure GAI au lieu de faire transiter dans le graphe des messages de la taille des cliques/séparateurs.

Enfin, il est clair que dans les problèmes admettant des contraintes dures d'arité importante (par exemple, sac à dos), la solution proposée ici qui consiste à inclure les contraintes dans des facteurs d'utilité n'est probablement pas idéale. Dans de tels cas, il serait intéressant de séparer les contraintes dures des préférences de manière à obtenir une fonction d'utilité avec des cliques de petite taille et de traiter les contraintes dures séparément avec les outils développés dans la communauté CSP et en recherche opérationnelle.

8. Bibliographie

Adams E., « Elements of a Theory of Inexact Measurement », *Philosophy of Science*, vol. 32, p. 205-228, 1965.

Bacchus F., Grove A., « Graphical Models for Preference and Utility », *UAI'95*, 1995.

Boutilier C., Bacchus F., Brafman R. I., « UCP-networks ; A Directed Graphical Representation of Conditional Utilities », *UAI'01*, p. 56-64, 2001.

Boutilier C., Brafman R., Domshlak C., Hoos H., Poole D., « CP-nets : A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements », *Journal of Artificial Intelligence Research*, vol. 21, p. 135-191, 2004a.

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

Brafman R., Domshlak C., « Introducing variable importance tradeoffs into CP-nets », *UAI'02*, 2002.

Brafman R., Domshlak C., Kogan T., « On Generalized additive Value-Function Decomposition », *UAI'04*, 2004.

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

de Givry S., Zytnicki M., Heras F., Larrosa J., « Existential arc consistency : Getting closer to full arc consistency in weighted CSPs », *IJCAI'05*, 2005.

Debreu G., « Continuity Properties of Paretian Utility », *International Economic Review*, vol. 5, p. 285-293, 1964.

Dechter R., AND/OR Search Spaces for Graphical Models, Technical report, ICS, 2004.

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

Gonzales C., « Additive Utility Without Restricted Solvability on Every Component », *Journal of Mathematical Psychology*, vol. 47, p. 47-65, 2003.

Gonzales C., Perny P., « GAI Networks for Utility Elicitation », *KR'04*, p. 224-234, 2004.

Gonzales C., Perny P., « GAI networks : from axiomatization to applications », *37th European Mathematical Psychology Group meeting*, 2006.

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

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

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

Larrosa J., Meseguer P., Schiex T., « Soft Constraint Processing », *IJCAI'05 Tutorial*, 2005.

Marinescu R., Dechter R., « AND/OR Tree Search for Constraint Optimization », *CP'2004 – Workshop on Preferences and Soft Constraints*, 2004.

Nakamura Y., « Additive Utilities on Densely Ordered Sets », *Journal of Mathematical Psychology*, vol. 46, n° 5, p. 515-530, 2002.

Nilsson D., « An efficient algorithm for finding the M most probable configurations in probabilistic expert systems », *Statistics and Computing*, vol. 8, n° 2, p. 159-173, 1998.

Schiex T., Fargier H., Verfaillie G., « Problèmes de satisfaction de contraintes valués », *Revue d'Intelligence Artificielle*, vol. 11, n° 3, p. 339-373, 1997.

Wakker P., *Additive Representations of Preferences, A New Foundation of Decision Analysis*, Kluwer, 1989.