Décision collective avec des réseaux GAI

Sergio Queiroz[1], Christophe Gonzales[1], et Patrice Perny[1]

LIP6–pôle IA, 8 rue du Capitaine Scott, 75015 Paris, France
prénom.nom@lip6.fr

Résumé Cet article aborde la prise de décision collective dans le contexte de la théorie de l'utilité multiattribut. Nous nous intéressons ici aux cas où l'ensemble d'alternatives a une structure combinatoire (définie comme le produit cartésien des valeurs des domaines d'un ensemble d'attributs) rendant nécessaire l'utilisation de représentations compactes de préférences. Nous supposons que les préférences des agents sont représentées par une fonction d'utilité GAI (indépendance additive généralisée–de l'anglais *Generalized Additive Independence*) sur le produit cartésien. Les fonctions GAI permettent une représentation efficace des interactions entre attributs tout en gardant une certaine décomposabilité du modèle. Nous traitons le problème de l'agrégation de préférences où nous considérons différents critères pour définir la notion de solution de compromis (max-min, min-max regret, normes pondérées de Tchebytcheff). Pour chacun de ces critères d'agrégation, nous proposons une procédure efficace pour la détermination exacte de la solution optimale sur le produit cartésien. Nous présentons ensuite des résultats expérimentaux qui montrent l'efficacité pratique de la procédure proposée.

Mots-clés : réseaux GAI; modèles graphiques; agrégation de préférences.

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 d'avoir des modèles permettant de manipuler les préférences des utilisateurs et de fournir des recommandations fondées sur leurs préférences. Ainsi, des travaux récents en modélisation des préférences en théorie de la décision visent à développer des formalismes 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 préférence et des représentations compactes comme les CP-nets [1,2] ou, s'agissant de décision collective, les mCP-nets [3]. Ces modèles sont délibérément simples pour être intégrés efficacement dans les systèmes interactifs de recommandation; les préférences des agents doivent être capturées en utilisant 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 purement ordinaux grâce à leur plus grande puissance descriptive [4]. L'utilisation d'utilités cardinales permet par exemple de définir un critère unique de synthèse et ainsi d'échapper aux difficultés de l'agrégation ordinale et du théorème d'impossibilité d'Arrow dans le cadre de la décision collective (voir [5]).

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 un type spécial d'indépendance entre attributs nommé *indépendance préférentielle mutuelle* qui assure que les préférences sont représentables par une utilité additive [6]. Une telle décomposabilité rend le processus d'élicitation très simple et rapide. Cependant, dans la pratique, l'indépendance préférentielle 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 en utilité sur chaque attribut* mène à une forme plus sophistiquée appelée *utilité multilinéaire* [7]. 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 [8]. Ces décompositions GAI permettent des interactions plus générales entre les attributs [9] tout en préservant une certaine décomposabilité du modèle.

En exploitant ces caractéristiques, [10,11] proposent une procédure générale pour éliciter des utilités GAI dans le cadre de la décision dans le risque. Cette procédure est dirigée par la structure d'un modèle graphique nommé réseau GAI et consiste en une séquence de questions concernant des loteries simples qui capturent efficacement les principales caractéristiques de l'attitude de l'agent face au risque. Récemment l'élicitation des modèles GAI a été étudiée dans le contexte de la prise de décision dans le certain. Ainsi, [12] propose des procédures d'élicitation fondées sur des comparaisons simples entre alternatives (par exemple, les questions concernent seulement un sous-ensemble d'attributs ou des alternatives qui diffèrent seulement par quelques attributs). Ils suggèrent également des algorithmes efficaces pour trouver l'élément GAI-optimal dans un produit cartésien, en utilisant des techniques classiques de propagation utilisées dans la littérature des réseaux bayésiens.

Dans cet article, nous considérons les problèmes de prise de décision collective dans des contextes semblables et nous nous concentrons sur la détermination d'une bonne solution de compromis entre les agents lorsque l'ensemble des alternatives est de grande taille. Plus précisément, nous supposons que les alternatives à comparer sont définies sur un espace combinatoire (produit cartésien) de grande taille, ce qui empêche toute énumération exhaustive, et nous supposons qu'une utilité GAI a été préalablement élicitée pour chaque agent. Nous étudions alors le problème d'effectuer des choix efficaces pour le groupe d'agents. L'article est organisé comme suit : dans la section 2, nous discutons la représentation individuelle et collective de préférences. Après avoir rappelé quelques éléments sur les modèles GAI et leur représentation graphique, nous considérons divers critères d'agrégation pour définir la notion de compromis entre les agents. Dans la section 3, nous fournissons des algorithmes efficaces pour déterminer la solution de meilleur compromis pour un groupe d'agents. Enfin la section 4 présente quelques résultats numériques obtenus sur des problèmes d'agrégation de préférences sur des espaces de tailles variées.

2 Utilités GAI : de la modélisation des préférences individuelles à la prise de décision collective

Avant de décrire les modèles GAI, nous devons introduire quelques notations. Dans cet article, $\succsim$ représente la relation de préférence d'un agent (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 $\sim$ la partie symétrique. Nous supposons que l'ensemble $\mathcal{X}$ des alternatives est un produit cartésien d'attributs $X_1, \ldots, X_n$. Pour simplifier les notations, dans la suite de l'article, les lettres majuscules (potentiellement avec des indices) comme A, B, X_1 représentent tant les attributs que leurs domaines. De même, les lettres minuscules (potentiellement avec des exposants) représentent des valeurs de l'attribut avec la lettre majuscule correspondante : x, x^1 (resp. x_i, x_i^1) sont ainsi des valeurs de X (resp. X_i). La sous-section 2.1 traite de la représentation des préférences individuelles. Nous considérerons le problème de décision collective dans la sous-section 2.2.

2.1 Modélisation de préférences individuelles

Sous certaines hypothèses [13], la relation de préférence $\succsim$ peut être représentée par une utilité, c.-à-d., 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 individu, des utilités doivent être élicitées pour chaque individu et chaque élément du produit cartésien, 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 élément de $\mathcal{X}$ est impossible, ne serait-ce que d'un point de vue utilisation de la mémoire.

Heureusement, en pratique, les préférences des agents ont naturellement une structure sous-jacente induite par des indépendances entre les attributs. Cela 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 $\alpha \in X_i$, $i = 1, \ldots, n$. Cependant, une telle décomposition n'est pas toujours appropriée parce qu'elle élimine toute possibilité d'interaction entre les attributs. Quand les préférences des agents sont plus complexes, un modèle plus sophistiqué est nécessaire, comme nous le montrons ci-dessous :

Exemple 1 *Considérons un ensemble $\mathcal{X}$ de menus $x = (x_1, x_2, x_3)$, avec pour plat principal $x_1 \in X_1 = \{steak(S), poisson(P)\}$, pour boisson $x_2 \in X_2 = \{vin\ rouge\ (R), vin\ blanc(B)\}$ et pour dessert $x_3 \in X_3 = \{cake(C), glace(G)\}$.*

Premier cas. *Supposons que les préférences d'un individu soient bien représentées par une utilité additive u caractérisée par les utilités marginales suivantes : $u_1(S) = 4$; $u_1(P) = 0$; $u_2(R) = 2$; $u_2(B) = 0$; $u_3(C) = 1$; $u_3(G) = 0$. Alors les utilités pour les 2^3 menus $x^{(i)}$ possibles sont :*

$$\begin{array}{llll} u(x^{(1)}) = u(S,R,C) = 7\,; & & u(x^{(2)}) = u(S,R,G) = 6\,; \\ u(x^{(3)}) = u(S,B,C) = 5\,; & & u(x^{(4)}) = u(S,B,G) = 4\,; \\ u(x^{(5)}) = u(P,R,C) = 3\,; & & u(x^{(6)}) = u(P,R,G) = 2\,; \\ u(x^{(7)}) = u(P,B,C) = 1\,; & & u(x^{(8)}) = u(P,B,G) = 0\,; \end{array}$$

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 que l'ordre de préférence d'un autre agent soit : $x^{(1)} \succ x^{(2)} \succ x^{(7)} \succ x^{(8)} \succ x^{(3)} \succ x^{(4)} \succ x^{(5)} \succ x^{(6)}$. Ceci peut être expliqué de la manière suivante : 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 (ceteris paribus).*

Bien que rationnelles, de telles préférences ne sont pas représentables par une utilité additive parce que $x^{(1)} \succ x^{(5)} \Rightarrow u_1(S) > u_1(P)$ mais $x^{(7)} \succ x^{(3)} \Rightarrow u_1(P) > u_1(S)$. Toutefois, ceci n'élimine pas la possibilité de formes désagrégées de décompositions additives, telles que : $u(x) = u_{1,2}(x_1, x_2) + u_3(x_3)$. Par exemple, $u_{1,2}(S,R) = 6$, $u_{1,2}(P,B) = 4$, $u_{1,2}(S,B) = 2$, $u_{1,2}(P,R) = 0$, $u_3(C) = 1$, $u_3(G) = 0$ représentent $\succsim$.

Troisième cas. *Supposons que l'ordre de préférence d'un troisième agent est : $x^{(2)} \succ x^{(1)} \succ x^{(7)} \succ x^{(8)} \succ x^{(4)} \succ x^{(3)} \succ x^{(5)} \succ x^{(6)}$. 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 :

$$\begin{array}{l} u_{1,2}(S,R)=6\,;\ u_{1,2}(P,B)=4\,;\ u_{1,2}(S,B)=2\,;\ u_{1,2}(P,R)=0\,; \\ u_{1,3}(S,C)=0\,;\ u_{1,3}(S,G)=1\,;\ u_{1,3}(P,C)=1\,;\ u_{1,3}(P,G)=0. \end{array}$$

Une telle décomposition avec des facteurs ayant une intersection non vide est nommée une décomposition GAI [9]. Elle inclut les décompositions additives et multilinéaires comme cas particuliers, mais elle est beaucoup plus flexible puisqu'elle ne fait aucune supposition sur le type d'interaction entre les attributs. Les décompositions GAI peuvent être définies plus formellement comme suit :

Définition 1 (Décomposition GAI) *Soit $\mathcal{X} = \prod_{i=1}^{n} X_i$. Soit $Z_1, \ldots, Z_k$ des sous-ensembles de $N = \{1, \ldots, n\}$ tels que $N = \cup_{i=1}^{k} Z_i$. Pour chaque i, soit $X_{Z_i} = \prod_{j \in Z_i} X_j$. L'utilité $u(\cdot)$ qui représente $\succsim$ est GAI-décomposable par rapport aux X_{Z_i} ssi il existe des fonctions $u_i : X_{Z_i} \mapsto \mathbb{R}$ telles que :*

$$u(x) = \textstyle\sum_{i=1}^{k} u_i(x_{Z_i}),\ \textit{pour tout}\ x = (x_1, \ldots, x_n) \in \mathcal{X},$$

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

Les décompositions GAI peuvent être représentées par des structures graphiques que nous appelons *réseaux GAI* [10]. Ces modèles graphiques sont similaires aux graphes de jonction utilisés pour les réseaux bayésiens [14,15] :

Définition 2 (Réseau GAI) *Soit* $\mathcal{X} = \prod_{i=1}^{n} X_i$. *Soit* $Z_1, \ldots, Z_k$ *des sous-ensembles de* $N = \{1, \ldots, n\}$ *tels que* $N = \cup_{i=1}^{k} Z_i$. *Supposons que* $\succsim$ *est représentable par une utilité GAI* $u(x) = \sum_{i=1}^{k} u_i(x_{Z_i})$ *pour tout* $x \in \mathcal{X}$. *Alors un réseau GAI qui représente* $u(\cdot)$ *est un graphe non-orienté* $G = (V, E)$ *qui satisfait les propriétés suivantes :*

1. $V = \{X_{Z_1}, \ldots, X_{Z_k}\}$;
2. *Pour tout* $(X_{Z_i}, X_{Z_j}) \in E$, $Z_i \cap Z_j \neq \emptyset$. *Pour tout* X_{Z_i}, X_{Z_j} *tel que* $Z_i \cap Z_j = T_{ij} \neq \emptyset$, *il existe un chemin dans* G *qui connecte* X_{Z_i} *et* X_{Z_j} *tel que tous ses nœuds contiennent tous les indices de* T_{ij} *(propriété d'intersection courante).*

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

Les cliques sont dessinées comme des ellipses et les séparateurs comme des rectangles. Dans cet article, nous nous intéressons seulement à des arbres GAI. Comme mentionné dans [10], 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 sous-utilités. Par exemple, 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 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 [15]).

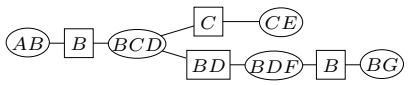

Figure 1. Un arbre GAI

2.2 Prise de décision collective ou multicritère

On considère maintenant un problème de décision multi-agent impliquant un ensemble $\mathcal{A} = \{1, \ldots, m\}$ d'agents. On suppose que les préférences de chaque agent $j \in \mathcal{A}$ sur un ensemble $\mathcal{X}$ sont représentées par une fonction GAI u^j qui a été précédemment élicitée. Une manière classique de trouver la meilleure solution de compromis pour un ensemble d'agents est de définir une fonction d'utilité globale $u(x)$ telle que, pour tout $x \in \mathcal{X}$, $u(x)$ donne la valeur d'utilité de x pour l'ensemble d'agents. On considère $u(x) = h(u^1(x), \ldots, u^m(x))$ où h est une fonction d'agrégation qui établit implicitement le type de compromis désiré pour la solution. La meilleure solution de compromis est celle qui optimise u sur $\mathcal{X}$. Quand la fonction h est non-décroissante en chaque composante, les solutions u-optimales sont des solutions faiblement Pareto-optimales (c.-à-d. qu'il n'y a pas d'autres solutions qui améliorent l'utilité pour tous les agents). De plus, si h est strictement croissante en chaque composante alors les solutions u-optimales sont Pareto-optimales (c.-à-d. qu'il n'y a pas d'autre solution qui améliore l'utilité pour un agent sans diminuer l'utilité pour un autre agent).

Si h est linéaire, alors u est la somme d'utilités GAI et, par conséquent, u est aussi une fonction GAI. Ainsi, le problème se réduit à un problème avec un seul agent avec une fonction d'utilité GAI u. Toutefois, les fonctions d'agrégation linéaires ne sont pas une bonne option, puisqu'elles peuvent choisir une solution avec un profil très déséquilibré concernant la satisfaction individuelle des différents agents. Par exemple, si l'on considère un problème avec 3 agents et tel que $\mathcal{X} = \{x, y, z, w\}$ avec $u^1(x) = 0, u^2(x) = u^3(x) = 100$, $u^2(y) = 0, u^1(y) = u^3(y) = 100$, $u^3(z) = 0, u^1(z) = u^2(z) = 100$, $u^1(w) = u^2(w) = u^3(w) = 65$. Seule la solution w est acceptable par tous les agents. Donc elle constitue la meilleure solution de compromis. Toutefois, w ne peut pas être obtenue par une combinaison linéaire (avec des coefficients positifs) des utilités des agents. Pour trouver des solutions de compromis, les critères non linéaires suivants sont les plus appropriés :

Le critère max-min : On maximise sur $\mathcal{X}$ le critère $u(x) = \min_{j \in \mathcal{A}} u^j(x)$. Ceci correspond à maximiser la satisfaction de l'agent le moins satisfait de l'ensemble d'agents. Cela suppose toutefois que les utilités des agents soient exprimées sur la même échelle.

Le critère min-max regret : On minimise sur $\mathcal{X}$ le critère $u(x) = \max_{j\in\mathcal{A}} r^j(x)$, où $r^j(x) = u^j(x^j) - u^j(x)$, et $x^j = \text{Argmax}_{x\in\mathcal{X}} u^j(x)$ (c.-à-d. x^j est la solution optimale pour l'agent j). $r^j(x)$ représente le regret de l'agent j quand l'alternative x est choisie au lieu de son alternative préférée x^j. Cela que les différences d'utilités représentent la même intensité de préférence pour tous les agents. Lorsque ce n'est pas le cas, il est préférable d'utiliser le critère suivant.

Le critère de Tchebycheff : On minimise sur $\mathcal{X}$ le critère $u(x) = \max_{j\in\mathcal{A}} w_j r^j(x)$ où $r^j(x) = u^j(x^j) - u^j(x)$, $x^j = \text{Argmax}_{x\in\mathcal{X}} u^j(x)$ et les w_j sont des coefficients positifs qui pondèrent l'importance de chaque agent. Ce critère correspond à la distance (par rapport à une norme de Tchebycheff pondérée) entre deux profils d'utilité : $(u^1(x), \ldots, u^n(x))$, obtenu avec la solution x, et le profil de satisfaction idéal $(u^1(x^1), \ldots, u^m(x^m))$ qui correspond à la situation, normalement fictive, où la satisfaction de chaque agents est maximale. Ce point idéal est une borne supérieure de l'ensemble des solutions non-dominées au sens de Pareto. Le critère de Tchebycheff est un outil standard pour générer des solutions de compromis dans le cadre de l'optimisation multi-objectif [16]. On peut le voir comme une sophistication du critère min-max regret, avec l'introduction des poids $w_j = \omega_j/(u^j(x^j) - u^j_*)$ où $u^j_* = \min_{k\in\mathcal{A}} u^j(x^k)$, ω_j est l'importance accordée à l'agent j et $1/(u^j(x^j) - u^j_*)$ est un facteur de normalisation. Ce dernier peut être vu comme un facteur de correction pour l'échelle d'utilité quand les agents attribuent leurs valeurs d'utilité dans des intervalles différents. Il peut être aussi utile quand les utilités des agents sont exprimées sur des échelles distinctes. Le paramètre ω_j permet de contrôler l'importance de chaque agent et aussi le type de compromis recherché. Ainsi, on sait que toute solution Pareto-optimale peut être obtenue avec un choix approprié des ω_j [17], ce qui permet d'engendrer tout type de compromis en utilisant le critère de Tchebycheff.

Dans le même formalisme, nous pourrions considérer des problèmes de décision multicritère au lieu de problèmes multi-agent. Dans ce contexte, nous aurions un ensemble de critères $\mathcal{A} = \{1, \ldots, m\}$. L'utilité de chaque critère serait définie par une fonction GAI-décomposable sur un sous-ensemble d'attributs de $\mathcal{X}$, et nous chercherions une solution de compromis entre les critères. Par exemple, considérons le problème de choix de la configuration d'une voiture, caractérisée par les attributs $\mathcal{X}$ = {couleur, nombre de portes, dir. assistée, freinage ABS, type de carburant, engrenage automatique}, selon 3 critères : *esthétique* (E), *confort* (C), et *sécurité* (S). Les valeurs d'utilité des critères sont représentées par les fonctions GAI-décomposables : u^E(couleur, nombre de portes), u^C(nombre de portes, dir. assistée, type de carburant, engrenage automatique), et u^S(dir. assistée, freinage ABS, type de carburant). Ainsi, le même traitement appliqué au problème de décision multi-agent peut être utilisé dans ce cadre multicritère. Comme la correspondance entre les deux types de problèmes est assez directe, nous nous limitons, sans perte de généralité, à discuter le cadre multi-agent.

Les fonctions d'agrégation envisagées ci-dessus (max-min, min-max regret, Tchebycheff) sont croissantes par rapport à chaque composante, ce qui garantit déjà la Pareto-optimalité faible des solutions maximisant $u(x)$. Cependant, ces fonctions pourraient être légèrement modifiées pour devenir strictement croissantes (afin de générer uniquement des solutions Pareto-optimales). Par exemple, il suffit d'ajouter (resp. soustraire) $\epsilon \sum_{i\in\mathcal{A}} u^i(x)$ pour la maximisation (resp. minimisation), ϵ étant une valeur positive arbitrairement petite. Nous discutons dans la section suivante l'optimisation de chacune des fonctions ci-dessus pour déterminer la meilleure solution de compromis entre les agents.

3 Algorithmes

3.1 Recherche de la solution de compromis

La détermination de la solution optimale de compromis des agents pour une fonction du type $u(x) = h(u^1(x), \ldots, u^m(x))$ n'est pas une tâche facile en raison de deux difficultés principales : la nature combinatoire de $\mathcal{X}$; et la probable non-décomposabilité de la fonction u (quand h n'est pas une fonction linéaire). Par exemple, si $h = \min$ (utilité max-min), la détermination de la meilleure solution de compromis est un problème NP-difficile dès qu'il y a $n \geq 3$ attributs, $m \geq 2$ agents, chacun possédant une fonction d'utilité GAI avec au moins un terme de taille égale ou supérieure à 3. Ceci peut être prouvé en utilisant une réduction de 3-SAT. Considérons en effet une instance de

3-SAT avec n variables et m clauses. À chaque variable nous associons un attribut booléen X_i et à chaque clause C_j sur des variables nous associons un agent avec une fonction d'utilité booléenne u^j. Par exemple $C_j = x \vee y \vee \neg z$ sera représentée par la fonction $u^j(x,y,z) = 1-(1-x)(1-y)z$. Ainsi, la valeur optimale du problème d'optimisation max-min sur $\mathcal{X} = X_1 \times \cdots \times X_n$ et les fonctions $u^1, \ldots, u^m$ est 1 si et seulement si le problème 3-SAT initial est satisfiable. Des réductions similaires peuvent être proposées pour le min-max regret et le critère de Tchebycheff. Ceci montre la difficulté de trouver efficacement la solution de meilleur compromis. Pour faire face à cette difficulté et être capable d'optimiser une fonction non-décomposable f sur $\mathcal{X}$, nous proposons de recourir à une technique de rangement. Ainsi nous proposons une procédure en 3 étapes pour optimiser une fonction $u(x) = h(u^1(x), \ldots, u^m(x))$ lorsque h n'est pas linéaire :

Étape 1 : reformulation. Nous reformulons le problème comme un problème monoagent, en utilisant un critère global $g(x)$ défini comme une combinaison linéaire des utilités individuelles. Une telle fonction est plus facile à optimiser que f puisque la somme de fonctions GAI est aussi une fonction GAI.

Étape 2 : rangement. Nous énumérons les solutions de $\mathcal{X}$ dans l'ordre décroissant de l'utilité selon $g(x)$. Cela nécessite un algorithme efficace qui exploite la structure GAI de g pour permettre une énumération rapide. Ce point sera discuté dans la sous-section 3.2.

Étape 3 : critère d'arrêt. Nous devons arrêter l'énumération le plus vite possible en raison de la taille exponentielle de $\mathcal{X}$. Pour cela nous utilisons la proposition suivante :

Proposition 1 *Considérons deux fonctions : f et g à minimiser, telles que $f, g : \mathcal{X} \mapsto \mathbb{R}$, f étant une fonction quelconque et g une fonction GAI-décomposable telle que $f(x) \geq g(x)$ pour tout $x \in \mathcal{X}$. Soit $x^{(1)}, ..., x^{(k)}$ les k-meilleures solutions selon la fonction g (c.-à-d. les solutions ayant les k plus petites valeurs possibles pour g). Si $g(x^{(k)}) \geq f(x^*)$ avec $x^* = Argmin_{i=1,\ldots,k} f(x^{(k)})$ alors x^* est optimale pour f, c.-à-d. $f(x^*) = \min_{x \in \mathcal{X}} f(x)$.*

Démonstration. Pour tout $i > k$, nous avons, par construction, $f(x^{(i)}) \geq g(x^{(i)}) \geq g(x^{(k)})$. Comme $g(x^{(k)}) \geq f(x^*)$ par hypothèse, nous avons $f(x^{(i)}) \geq f(x^*)$, ce qui montre qu'aucune solution trouvée après $x^{(k)}$ dans le rangement ne peut améliorer la meilleure solution actuelle x^*. □

Ce résultat simple peut être utilisé pour optimiser les critères d'agrégation introduits. Le tableau suivant définit g, l'approximation GAI à utiliser, pour chacun des critères f considérés :

Critère	$f(x)$	$g(x)$
max-min	$-\min_j(u^j(x))$	$-\frac{1}{m}\sum_j u^j(x)$
min-max regret	$\max_j(r^j(x))$	$\frac{1}{m}\sum_j r^j(x)$
Tchebycheff	$\max_j(w_j r^j(x))$	$\frac{1}{m}\sum_j w_j r^j(x)$

Comme $1/m \sum_j r^j(x)$ diffère de $-1/m \sum_j u^j(x)$ seulement par un terme constant, la même méthode de rangement utilisée pour les critères max-min vaut aussi pour min-max regret. Tchebycheff est simplement une extension de min-max regret avec des poids et ainsi peut être traité de manière analogue. Voyons maintenant comment procéder pour ranger efficacement les éléments de $\mathcal{X}$ avec une fonction GAI g (étape 2).

3.2 Rangement avec une fonction GAI

La procédure de rangement que nous présentons dans cette sous-section est fondée sur une autre procédure conçue pour répondre à des questions de choix, c'est-à-dire pour trouver le n-uplet préféré dans $\mathcal{X}$. Pour éviter des comparaisons exhaustives entre les paires d'alternatives – qui seraient infaisables en pratique du fait de la nature combinatoire de $\mathcal{X}$ – les deux procédures

exploitent la structure du réseau GAI pour décomposer le problème en une séquence d'optimisations locales, permettant de maintenir le coût du calcul global à un niveau admissible. D'abord nous présentons brièvement l'algorithme permettant d'obtenir la solution optimale (problème de choix) et, ensuite, nous présentons la procédure générale de rangement. Le problème de choix est équivalent à résoudre :

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

L'optimum peut être trouvé efficacement en exploitant les propriétés suivantes :

1. le max sur $X_1,\ldots,X_n$ de $u(X_1,\ldots,X_n)$ peut être décomposé comme :

$$\max_{X_1}\ldots\max_{X_n} u(X_1,\ldots,X_n),$$

   et l'ordre dans lequel les max sont calculés n'est pas important ;
2. si $u(X_1,\ldots,X_n)$ peut être décomposé comme $f() + g()$ où $f()$ ne dépend pas de 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 qui appartient à une clique externe C (c.-à-d. une clique avec au plus un voisin) mais pas à sa clique voisine n'apparaît pas ailleurs dans le réseau GAI.

Les propriétés 2 et 3 suggèrent une stratégie consistant à calculer l'utilité maximale en utilisant d'abord les variables qui sont seulement dans les cliques externes (de telles cliques seront les seuls facteurs incluant ces variables) et en additionnant ensuite les résultats à la clique voisine après l'élimination des cliques externes. On itère la procédure jusqu'à ce que toutes les cliques aient été éliminées. Ceci nous conduit à l'algorithme suivant :

Function `Collect`(clique C_i, F)
01 pour tout C_j dans {cliques adjacentes à C_i}$\backslash F$ dans le réseau GAI faire
02 soit $S_{ij} = C_i \cap C_j$ le séparateur entre C_i et C_j
03 soit u_j^* définie sur S_{ij} par `Collect`$(C_j, \{C_i\})$
04 substituer $u_i(x_{C_i})$ par $u_i(x_{C_i}) + u_j^*(x_{S_{ij}})$ pour tout x_{C_i}
05 fait
06 si $F \neq \emptyset$ alors
07 soit C_j la seule clique $\in F$ et soit $S_{ij} = C_i \cap C_j$
08 soit $M_i^*(x_{S_{ij}}) = \text{Argmax}\{u_i(y_{C_i}) : y_{S_{ij}} = x_{S_{ij}}\}$ et soit $u_i^*(x_{S_{ij}}) = u_i(M_i^*(x_{S_{ij}}))$ pour tout $x_{S_{ij}}$ dans $\prod_{X_k\in S_{ij}} X_k$
09 stocker la matrice M_i^* dans le séparateur S_{ij} et renvoyer u_i^*
10 finsi

Cette fonction réduit récursivement l'équation (1) en enlevant une par une toutes les sous-utilités (par extraction de leur max). Ainsi, en appelant la fonction `Collect` à partir de n'importe quelle clique on obtient la valeur d'utilité de l'élément préféré. Par exemple, sur l'exemple de la figure 1, en appelant la fonction `Collect`$(BCD, \emptyset)$ on obtient les propagations de message montrées dans la figure 2, où :

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

À la fin de la collecte, $\max_{BCD} u_3^*(B,C,D) = u_3(B,C,D) + u_1^*(B) + u_2^*(C) + u_4(B,D)$ correspond à la valeur maximale de la fonction d'utilité. Soit $(\hat{b},\hat{c},\hat{d})$ une solution de $\max_{BCD} u_3^*(B,C,D)$. Alors $(\hat{b},\hat{c},\hat{d})$ est évidemment une projection sur $B\times C\times D$ d'un des éléments les plus préférés de $\mathcal{X}$. Mais l'utilité correspondante est $u_3(\hat{b},\hat{c},\hat{d}) + u_1^*(\hat{b}) + u_2^*(\hat{c}) + u_4(\hat{b},\hat{d})$, laquelle est obtenue par $M_1(\hat{b})$, $M_2(\hat{c})$ et $M_4(\hat{b},\hat{d})$. Enfin, $M_4(\hat{b},\hat{d})$ correspond à la valeur d'utilité $u_4(\hat{b},\hat{c},\hat{d}) + u_5^*(\hat{b})$, obtenue par $M_5(\hat{b})$. Par conséquent, le n-uplet optimal peut être obtenu en propageant de manière récursive les affectations des attributs (les M_i) de la clique BCD aux cliques plus externes, comme montre la figure 3. Ceci nous conduit à formuler l'algorithme suivant :

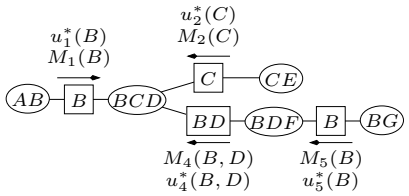

FIGURE 2. La phase de collecte

Function `Instantiate`(C_i, F, x_F)
01 si $F = \emptyset$ alors
02 soit $x^*_{C_i} = \text{Argmax}\{u_i(x_{C_i}) : x_{C_i} \in \prod_{X_k \in C_i} X_k\}$
03 sinon
04 soit C_j la seule clique $\in F$ et soit $x^*_{C_j} = x_F$
05 soit $S_{ij} = C_i \cap C_j$ et $D_{ij} = C_i \backslash C_j$
06 soit $x^*_{C_i} = \text{Argmax}\{u_i(x^*_{S_{ij}}, y_{D_{ij}})\}$
07 finsi
08 soit $\{C_{i_1}, \ldots, C_{i_k}\} = \{\text{cliques adjacentes à } C_i\} \backslash F$
09 pour chaque j de 1 à k faire
10 soit x^{i_j} = `Instantiate` $(C_{i_j}, \{C_i\}, x^*_{C_i})$
11 soit y^{i_j} le n-uplet x^{i_j} sans les valeurs des attributs dans S_{ij}
12 renvoyer le n-uplet $(x^*_{C_i}, y^{i_1}, \ldots, y^{i_k})$

Function `Optimal_choice`(GAI-net)
01 Soit C_0 une clique quelconque dans le réseau GAI
02 appeler `Collect`$(C_0, \emptyset)$ et soit x^* = `Instantiate`$(C_0, \emptyset, \emptyset, \emptyset)$
03 renvoyer le choix optimal x^*

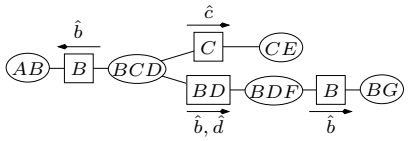

FIGURE 3. La phase d'affectation

Pour le rangement, considérons l'exemple de la figure 2. Supposons que `Optimal_choice` a renvoyé x^*=$(\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{e}, \hat{f}, \hat{g})$. Alors, le prochain meilleur n-uplet x^2 diffère de x^* par au moins un attribut, autrement dit il y a une clique C_i telle que la projection de x^2 sur C_i diffère de celle de x^*. Comme nous ne connaissons pas la clique C_i dans laquelle se produit cette différence, nous considérons toutes les possibilités en divisant l'espace d'alternatives restantes selon la partition suivante :

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

La construction des ensembles ci-dessus suit la décomposition proposée par [18] : les cliques dans lesquelles les attributs doivent être différents de ceux de x^* sont énumérées dans l'ordre dans lequel elles sont appelées par la fonction `Collect` dans l'exécution de `Optimal_choice`. Les ensembles 1 à 5 ci-dessus correspondent ainsi à une phase de collecte qui a parcouru successivement les cliques (B, C, D), (B, D, F), (B, G), (C, E) et (A, B). Trouver le meilleur élément d'un ensemble donné est essentiellement similaire à trouver la solution optimale sauf que les lignes 02 et 06 dans la fonction `Instantiate` doivent être modifiées pour éviter certaines affectations (comme $(\hat{b}, \hat{c}, \hat{d})$).

Supposons maintenant que le deuxième meilleur n-uplet, disons $x^2 = (a^2, b^2, \hat{c}, d^2, \hat{e}, \hat{f}, g^2)$, soit le choix optimal de l'ensemble 1. Alors le prochain n-uplet, x^3, est le meilleur n-uplet qui est différent de x^* et de x^2. Il peut être trouvé en utilisant le même processus. Comme x^2 appartient à l'ensemble 1, nous devons remplacer l'ensemble 1 par les ensembles ci-dessous pour exclure x^2 et ensuite réitérer le même processus :

Ensemble 1.1 : $(B, C, D) \notin \{(\hat{b}, \hat{c}, \hat{d}), (b^2, \hat{c}, d^2)\}$
Ensemble 1.2 : $(B, C, D) = (b^2, \hat{c}, d^2)$ et $(B, D, F) \neq (b^2, d^2, \hat{f})$
Ensemble 1.3 : $(B, C, D, F) = (b^2, \hat{c}, d^2, \hat{f})$ et $(B, G) \neq (b^2, g^2)$
Ensemble 1.4 : $(B, C, D, F, G) = (b^2, \hat{c}, d^2, \hat{f}, g^2)$ et $(C, E) \neq (\hat{c}, \hat{e})$
Ensemble 1.5 : $(B, C, D, E, F, G) = (b^2, \hat{c}, d^2, \hat{e}, \hat{f}, g^2)$ et $(A, B) \neq (a^2, b^2)$

Ceci justifie l'algorithme suivant :

Function `k-best`(GAI-net, k)
01 soit x^* le n-uplet renvoyé par `Optimal_choice`
02 soit $\mathcal{S} = \{$Ensembles $i\}$ selon décrit ci-dessus et soit $kbest = \emptyset$
03 pour chaque Ensemble i, soit opt(Ensemble i) le choix optimal dans l'Ensemble i
04 pour $i = 2$ à k faire
05 soit Ensemble j un élément quelconque de
$\{$Ensemble $j \in \mathcal{S} : opt$(Ensemble $j) \geq opt$(Ensemble $p)$, Ensemble $p \in \mathcal{S}\}$
06 soit x^i, le $i$ème meilleur élément, soit opt(Ensemble j)
07 ajouter x^i à $kbest$ et enlever l'Ensemble j de $\mathcal{S}$
08 substituer l'Ensemble j dans $\mathcal{S}$ par les ensembles $\not\supseteq \{x^i\}$ selon décrit ci-dessus
09 fait
10 renvoyer $kbest$

4 Expérimentations

Pour évaluer l'efficacité pratique de notre méthode, nous avons fait des expérimentations avec différents jeux d'essais. Nous avons observé les temps de calcul ainsi que le nombre de solutions énumérées pour renvoyer la solution de meilleur compromis selon les critères considérés. Les expérimentations ont été réalisées à l'aide d'un PC 3.2GHz PC en utilisant une implantation en Java.

4.1 Jeux d'essais

Pour les expérimentations, nous avons généré aléatoirement des données pour des préférences GAI-décomposables. Toutes les décompositions GAI concernaient 20 variables, avec 10 termes d'utilité $u_i(x_{Z_i})$ de taille $|x_{Z_i}|$ choisie aléatoirement entre 2 et 4 (il semble irréaliste de considérer des interactions encore plus complexes, car l'élicitation serait trop difficile à réaliser). Le domaine de chaque terme u_i a été choisi aléatoirement. Pour les variables qui n'ont été sélectionnées dans aucun domaine, nous avons créé des termes d'utilité unaires. Ensuite, nous avons créé 5 fonctions d'utilité différentes avec la structure précédemment générée. Ces fonctions représentent les préférences de 5 agents. Pour chaque terme d'utilité u_i^j d'un agent j, nous générons d'abord sa valeur maximale d'utilité $\max(u_i^j)$ dans l'intervalle $[0, 1]$. Ensuite nous générons uniformément les valeurs d'utilité pour toutes les configurations de u_i^j dans l'intervalle $[0, \max(u_i^j)]$. Ceci nous donne 5 fonctions d'utilité GAI différents avec la même structure. Nous avons généré des données pour les tailles de domaine 2, 5 et 10, ce qui induit respectivement 2^{20}, 5^{20} et 10^{20} configurations possibles.

Nous présentons ici les résultats des essais avec 5 agents. D'autres essais que nous avons réalisés montrent une croissance progressive du nombre de solutions énumérées (et par conséquent du temps d'exécution) avec l'augmentation du nombre d'agents. Notons cependant que notre approche est destinée essentiellement à traiter les problèmes d'agrégation avec un nombre restreint d'agents. En effet, au delà d'une dizaine d'individus l'élicitation précise d'utilités GAI sur un ensemble combinatoire semble délicate à mettre en œuvre.

4.2 Résultats

Les résultats moyens (t : temps en ms et $\#gen$: nombre de solutions générées) sur 100 exécutions sont résumés ci-dessous :

Taille du	max-min		min-max Regret		Tchebycheff	
domaine	t(ms)	$\#gen$	t(ms)	$\#gen$	t(ms)	$\#gen$
2	371	9145	118	2437	96	1916
5	2784	60020	1100	22364	1052	21640
10	4703	99709	3934	78518	3104	60479

Nous avons obtenu des temps d'exécution entre 0.1 et 4.7 secondes, selon le critère de compromis utilisé et la taille du domaine des variables. Trouver la solution avec le critère max-min a demandé l'énumération d'un nombre plus grand de solutions qu'avec les critères min-max regret et Tchebycheff pour la même taille du domaine. Toutefois, le facteur dominant est la taille du domaine des attributs. On constate heureusement que le nombre de solutions énumérées avant de trouver la solution optimale augmente plus lentement que la taille du problème. Par exemple, passer d'un problème à 20 attributs avec un domaine de taille 5 à un problème à 20 attributs avec un domaine de taille 10 multiplie le nombre de configurations par plus de 10^6 alors le nombre de solutions énumérées augmente seulement d'un facteur inférieur à 3. Nous avons aussi réalisé des expérimentations où chaque agent avait une fonction GAI de structure différente. Dans ce cas, pour générer la fonction GAI agrégée on doit d'abord trianguler le graphe induit par les fonctions d'utilité des agents. Quand les structures des fonctions GAI des agents sont très différentes, le nouvel arbre GAI agrégé peut avoir des cliques très grandes, ce qui empêche l'utilisation de l'algorithme. Toutefois, en pratique, on peut s'attendre à ce que les différences entre les fonctions d'utilité des agents s'expliquent moins par des différences de cliques que par des différences de tables d'utilité. Dans de tels cas, la somme des utilités GAI des agents ne devrait pas faire apparaître beaucoup de grosses cliques, ce qui devrait contribuer à garantir une certaine efficacité de la procédure proposée.

5 Conclusion

Dans cet article, nous avons montré que les réseaux GAI peuvent être employés non seulement pour fournir efficacement des recommandations (choix et rangement) sur des ensembles combinatoires pour un individu seul, mais également pour des problèmes de recommandation collective dans le cadre de la décision multi-agent ou multi-critère. La procédure proposée permet la détermination de divers types de solutions de compromis et reste très efficace dès lors que le nombre d'agents n'est pas trop élevé. Elle pourrait être employée dans beaucoup de situations réelles comme le choix d'un voyage de vacances fondé sur les préférences d'un groupe, la configuration de la voiture idéale pour une famille, ou pour la recommandation d'un film pour un groupe d'amis.

Dans le cadre de la résolution multi-agent avec différentes structures GAI, nous avons observé que lorsque les structures des modèles GAI des agents différent considérablement, l'arbre de jonction obtenu après re-triangulation peut avoir des cliques trop grandes, ce qui réduit l'efficacité des calculs. Pour faire face à ce problème, on envisage d'approximer inférieurement la fonction GAI $g(\cdot)$ par une fonction GAI $g' : \mathcal{X} \mapsto \mathbb{R}$ encore plus décomposable, c'est-à-dire dont les cliques sont plus petites que celles de g. Une telle fonction g' peut être utilisée à la place de g dans la procédure en 3 phases présentée dans la section 3.1.

D'autres sophistications de notre approche sont possibles, par exemple en utilisant des arbres ET/OU [19,20] dans la structure GAI au lieu de calculer le terme u_i^* entier pendant les phases de collecte. L'algorithme de rangement pourrait ensuite être amélioré en utilisant un choix dynamique de la clique passée comme argument dans les phases de collecte/affectation (selon les n-uplets rangés jusqu'à présent). Enfin nous précisons que le formalisme introduit ici devrait permettre de traiter des problèmes multicritères combinatoires en utilisant les fonctions GAI pour représenter de manière compacte les critères.

Références

1. Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., Poole, D. : CP-nets : A tool for representing and reasoning with conditional ceteris paribus preference statements. J. A.I. Research **21** (2004) 135–191
2. Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., Poole, D. : Preference-based constraint optimization with CP-nets. Computational Intelligence **20** (2004)
3. Rossi, F., Venable, K.B., Walsh, T. : mCP nets : Representing and reasoning with preferences of multiple agents. In : AAAI. (2004) 729–734
4. Boutilier, C., Bacchus, F., Brafman, R. : UCP-networks; a directed graphical representation of conditional utilities. In : UAI'01. (2001)
5. Pini, Rossi, Venable, Walsh : Aggregating partially ordered preferences : Impossibility and possibility results. TARK **10** (2005)
6. Krantz, D., Luce, R.D., Suppes, P., Tversky, A. : Foundations of Measurement (Additive and Polynomial Representations). Volume 1. Academic Press (1971)
7. Keeney, R.L., Raiffa, H. : Decisions with Multiple Objectives : Preferences and Value Tradeoffs. Cambridge University Press (1993)
8. Fishburn, P.C. : Utility Theory for Decision Making. Wiley (1970)
9. Bacchus, F., Grove, A. : Graphical models for preference and utility. In : UAI'95. (1995)
10. Gonzales, C., Perny, P. : GAI networks for utility elicitation. In : KR'04. (2004)
11. Braziunas, D., Boutilier, C. : Local utility elicitation in GAI models. In : UAI'05. (2005)
12. Gonzales, C., Perny, P. : GAI networks for decision making under certainty. In : IJCAI'05 – Workshop on Advances in Preference Handling. (2005)
13. Debreu, G. : Continuity properties of paretian utility. Int. Econ. Review **5** (1964) 285–293
14. Jensen, F. : An introduction to Bayesian Networks. Taylor and Francis (1996)
15. Cowell, R., Dawid, A., Lauritzen, S., Spiegelhalter, D. : Probabilistic Networks and Expert Systems. Stat. for Eng. and Information Science. Springer-Verlag (1999)
16. Steuer, R., Choo, E.U. : An interactive weighted Tchebycheff procedure for multiple objective programming. Math. Prog. **26** (1983) 326–344
17. Wierzbicki, A. : On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spektrum **8** (1986) 73–87
18. Nilsson, D. : An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. Statistics and Computing **8** (1998) 159–173
19. Dechter, R. : AND/OR search spaces for graphical models. Technical report, ICS (2004)
20. Marinescu, R., Dechter, R. : AND/OR tree search for constraint optimization. In : CP'04 – Workshop on Pref. and Soft Constraints. (2004)