Partage équitable de ressources communes

(2) Éléments de complexité et d'algorithmique

Hélène Fargier* — Jérôme Lang*
Michel Lemaître — Gérard Verfaillie****

* *Institut de Recherches en Informatique de Toulouse,*
118, route de Narbonne, F-31062 Toulouse Cedex

{fargier, lang}@irit.fr

** *Office National d'Études et de Recherches Aérospatiales, Centre de Toulouse*
2, avenue E. Belin, B.P. 4025, F-31055 Toulouse Cedex 4

{Michel.Lemaitre, Gerard.Verfaillie}@cert.fr

RÉSUMÉ. *Le partage de ressources communes est un problème rencontré par de nombreuses organisations industrielles et économiques. Il s'agit, pour un ensemble d'agents ayant financé une ressource commune, d'exploiter conjointement celle-ci de manière équitable et efficace. Dans un article précédent [FAR], nous avons proposé un modèle générique et formel de ce problème combinatoire, et montré son application à un problème de partage de ressources satellitaires. Dans le présent article, nous apportons des éléments concernant la complexité théorique du problème et proposons des algorithmes pour le résoudre.*

ABSTRACT. *Many industrial and economic organizations come up against the problem of sharing common property resources. In this problem, a set of agents, having co-funded a common property resource, must exploit it in a fair and efficient way. In a previous article [FAR], we described a generic and formal model for this problem, and showed its application to a satellite resource sharing problem. In the present article, we characterize the theoretical complexity of this problem under different hypothesis, and discuss some algorithms for solving it.*

MOTS-CLÉS : *Partage équitable, enchères combinatoires, agrégation de préférences, aide multi-critère à la décision*

KEYWORDS: *Equitable resource sharing, combinatorial auctions, preference aggregation, multi-criteria decision aid*

2e soumission à *TSI*, le 29 juillet 2003.

1. Introduction

Le partage de ressources communes est un problème qui se pose à de nombreuses organisations industrielles et économiques. Un exemple typique d'un tel problème se rencontre dans le domaine spatial au niveau européen : les futures constellations de satellites d'observation de la Terre, qui seront financées par plusieurs pays, devront être exploitées en commun. Plus précisément, les pays participants devront se partager les opportunités d'observation offerte chaque jour par la constellation. Le partage doit être **équitable** : chaque pays concerné est en droit d'obtenir un juste retour sur investissement, en rapport avec sa participation financière. En outre, la ressource ne doit pas être sous-exploitée : son exploitation doit être **efficace**. Notons que les besoins et les souhaits des différents pays peuvent être conflictuels, et qu'enfin, le problème est fortement combinatoire, parce qu'une solution d'une instance d'un problème de partage est constituée de nombreuses décisions ponctuelles — concernant des variables discrètes — liées entre elles par des contraintes.

Dans un premier article intitulé « Partage équitable de ressources communes : (1) un modèle général et son application au partage de ressources satellitaires » nous avons proposé un modèle générique et formel du problème de partage de ressources communes et discrètes entre plusieurs agents, et montré son application au problème du partage de ressources satellitaires qui vient d'être évoqué. La modélisation proposée repose sur l'idée de deux niveaux d'utilité : l'utilité individuelle d'un partage pour un agent, puis l'utilité collective du partage pour l'ensemble des agents. L'utilité collective, qu'il s'agit de maximiser, résulte de l'agrégation des utilités individuelles. Nous allons, en section 2, rappeler l'essentiel des notions et notations introduites dans ce premier article, de façon à rendre compréhensible le second sans avoir à se référer au précédent.

Dans le présent article, nous apportons des éléments concernant la complexité théorique du problème (section 3). Puis nous décrivons des algorithmes généraux, complets et incomplets, calculant des partages admissibles maximisant l'utilité collective (section 4). Enfin, nous concluons et indiquons les poursuites possibles de ce travail.

2. Rappels sur le modèle général et hypothèses importantes

Nous rappelons ici l'essentiel du modèle général que nous avons longuement introduit dans le premier article. Par rapport à ce dernier, nous faisons ici plusieurs hypothèses simplificatrices :

– les utilités (individuelles et collectives) sont à valeurs dans $\mathbb{R}^+$;

– les contraintes sont exprimées en logique propositionnelle ;

– les utilités individuelles sont obtenues en maximisant les demandes satisfaites, plutôt qu'en minimisant les demandes insatisfaites (voir la section 5.3 de [FAR]) ;

– notant f (resp. g) la fonction d'agrégation utilisée pour le calcul des utilités individuelles (resp. collectives), nous ne considérerons que $f = +$ et $f = \max$, ainsi que $g = +$ et $g = \min$.

Définition 1 (Problème de partage ; partages) *Un* **problème de partage** *est un quadruplet* $\pi = \langle N, \mathcal{O}, \mathcal{C}, \mathcal{U} \rangle$ *où :*

– $N = \{1, \ldots, n\}$ *est un ensemble fini d'***agents** *;*

– $\mathcal{O}$ *est un ensemble fini d'***objets** *à attribuer aux agents ;*

– $\mathcal{C}$ *est un ensemble fini de* **contraintes** *; chaque contrainte de* $\mathcal{C}$ *est une formule propositionnelle du langage généré par l'ensemble de symboles propositionnels* $\{alloc(o, i) \mid o \in \mathcal{O},\ i \in N\}$.

– $\mathcal{U}$ *est une* **structure d'utilité**.

Un **partage** *pour* π *est un vecteur* $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ *où* $x_i \subseteq \mathcal{O}$ *pour tout* $i \in N$. *La composante* x_i *est la* **part** *de l'agent* i *dans* $\mathbf{x}$*, c'est-à-dire l'ensemble des objets* **attribués** *à l'agent* i *par le partage* $\mathbf{x}$.
Un partage est **admissible** *si et seulement si il satisfait toutes les contraintes de* $\mathcal{C}$. $Adm(\pi)$ *est l'ensemble de tous les partages admissibles de* π.

Dans l'expression des contraintes, $alloc(o, i)$ signifie que l'objet o est alloué à l'agent i.

Une **contrainte de préemption relative à l'objet** o, qui exprime que cet objet ne peut pas être attribué à plus d'un agent, s'écrit :

$$\neg(alloc(o, i) \wedge alloc(o, j)) \text{ pour tout } i, j \in N,\ i \neq j.$$

Les contraintes qui ne sont pas de préemption peuvent être des contraintes de capacité, d'exclusion, de faisabilité physique ou de dépendance entre objets ([FAR], section 3.2.1).

On fait une hypothèse importante : **on demande qu'il soit possible de déterminer en temps polynomial si un partage donné est admissible, et de calculer l'utilité d'un partage admissible également en temps polynomial**.

Définition 2 (Structure d'utilité) *Une* **structure d'utilité** *associée à un problème de partage* π *est un triplet* $\mathcal{U} = \langle \Delta, \mathbf{ua}, uc \rangle$ *où*

– $\Delta = \langle \Delta_1, \ldots, \Delta_n \rangle$*, où chaque* Δ_i *est l'ensemble fini des* **demandes pondérées** *de l'agent* i *;*

– $\mathbf{ua} = \langle ua_1, \ldots, ua_n \rangle$ *où* ua_i*, la* **fonction d'utilité individuelle de l'agent** i*, est une fonction de* $Adm(\pi)$ *dans* $\mathbb{R}^+$ *;*

– uc*, la* **fonction d'utilité collective***, est une fonction de* $Adm(\pi)$ *dans* $\mathbb{R}^+$ *;*

Définition 3 (Demandes ; satisfaction d'une demande) *$L^+_{\mathcal{O}}$ est l'ensemble des formules bien formées que l'on peut construire sur l'ensemble de littéraux $\mathcal{O}$ à partir des connecteurs logiques $\vee$ et $\wedge$. Une demande est une formule de $L^+_{\mathcal{O}}$.*
$x \subseteq \mathcal{O}$ satisfait une demande $\varphi \in L^+_{\mathcal{O}}$, noté $x \models \varphi$, si et seulement si $\langle x, \neg(\mathcal{O} \setminus x)\rangle$ est un modèle de φ, où $\langle x, \neg(\mathcal{O} \setminus x)\rangle$ est l'interprétation dont les littéraux positifs sont les éléments de x et les littéraux négatifs les négations des éléments de $\mathcal{O} \setminus x$.

Définition 4 (demandes pondérées) *L'ensemble des* **demandes pondérées d'un agent** *i est un ensemble de couples $\Delta_i = \{\langle\varphi^i_1, w^i_1\rangle, \ldots, \langle\varphi^i_{|\Delta_i|}, w^i_{|\Delta_i|},\rangle\}$, où $\varphi^i_k \in L^+_{\mathcal{O}}$ et $w^i_k \in \mathbb{R}^+$.*

En ce qui concerne les demandes, différentes hypothèses peuvent être faites (et ces hypothèses auront un impact sur la complexité algorithmique) :

– limitation à l'expression de demandes élémentaires : les formules autorisées sont celles qui ne contiennent qu'un objet ;

– limitation à l'expression de dépendances positives — le poids d'une demande est toujours supérieur ou égal à la combinaison par f des poids des objets qu'elle demande (une demande élémentaire forme évidemment une dépendance positive) ;

– limitation à l'expression de dépendances négative — le poids d'une demande est toujours inférieur ou égal à la combinaison par f des poids des objets qu'elle demande (une demande élémentaire forme évidemment une dépendance négative).

La notion de dépendance entre demandes est introduite, avec des exemples, dans l'article [FAR], section 4.2.

Définition 5 (calcul de l'utilité individuelle à partir des demandes pondérées) *Étant donné un ensemble de demandes pondérées Δ_i et un partage $\mathbf{x}$, l'utilité individuelle du partage $\mathbf{x}$ pour l'agent i est calculé en agrégeant les pondérations des demandes satisfaites par x_i, c'est-à-dire*

$$ua_i(\mathbf{x}) = f(\{w \mid \langle\varphi, w\rangle \in \Delta_i \text{ et } x_i \models \varphi\})$$

où $f \in \{+, \max\}$.
Le vecteur d'utilités individuelles $\langle ua_1(\mathbf{x}), \ldots, ua_n(\mathbf{x})\rangle$ est appelé **profil de satisfaction** *associé au partage $\mathbf{x}$.*

Définition 6 (calcul de l'utilité collective à partir des utilités individuelles) *L'utilité collective du partage $\mathbf{x}$ est calculée en agrégeant les utilités individuelles par*

$$uc(\mathbf{x}) = g(ua_1(\mathbf{x}), \ldots, ua_n(\mathbf{x}))$$

où $g \in \{+, \min\}$.

Le tableau 1 résume les notations introduites dans les définitions qui précèdent.

π : un problème de partage (les données)
$N = \{1, \ldots, n\}$: ensemble des **agents**
$\mathcal{O}$: ensemble fini des **objets** attribuables aux agents
$\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ où $x_i \subseteq \mathcal{O}$: un **partage**
x_i : part de l'agent i dans le partage $\mathbf{x}$
$\mathcal{C}$: ensemble fini des **contraintes** d'admissibilité sur les partages
$Adm(\pi)$: ensemble des **partages admissibles** de π
φ_k^i : $k^{i\grave{e}me}$ **demande** de l'agent i (formule propositionnelle sur $\mathcal{O}$)
w_k^i : **poids** de la $k^{i\grave{e}me}$ demande de l'agent i
$\Delta_i = \{\langle \varphi_1^i, w_1^i \rangle, \ldots, \langle \varphi_{\lvert\Delta_i\rvert}^i, w_{\lvert\Delta_i\rvert}^i \rangle\}$
: ensemble des **demandes pondérées** de l'agent i
$\mathcal{D}_i$: ensemble des objets de $\mathcal{O}$ demandés par l'agent i
$ua_i(\mathbf{x})$: utilité individuelle du partage $\mathbf{x}$ pour l'agent i
$uc(\mathbf{x})$: utilité collective du partage $\mathbf{x}$

Tableau 1. *Principales notations du modèle, introduites dans [FAR , section 3.2].*

3. Complexité

Avant d'explorer des méthodes algorithmiques pertinentes pour la résolution de problèmes de partage, il est utile, pour avoir une idée de la difficulté de la tâche, d'en étudier leur complexité algorithmique.

On suppose que les classes de complexité P , NP et coNP sont connues du lecteur (on pourra se référer par exemple à [GAR 79] ou à [PAP 94]).

Il existe plusieurs **problèmes de décision** associés aux problèmes de partage, et dont il est pertinent d'étudier la complexité :

Étant donné un problème de partage π ...

1) *... et une valuation K, existe-t-il un partage admissible $\mathbf{x}$ tel que $uc(\mathbf{x}) \geq K$?*

2) *... et un partage $\mathbf{x}$, $\mathbf{x}$ est-il optimal pour π ?*

3) *... et une propriété P (exprimable logiquement), est-ce que tous les partages optimaux pour π satisfont P ?*

Les classes de complexité dont font partie les trois problèmes étant étroitement liées, nous nous focalisons sur le premier problème — nous évoquerons la complexité des deux autres problèmes de décision à la fin de la section.

Un premier résultat, immédiat, est que le problème d'existence d'un partage admissible $\mathbf{x}$ tel que $uc(\mathbf{x}) \geq K$ est dans NP quelles que soient les hypothèses faites sur les demandes. Ce résultat provient directement de nos hypothèses de calcul d'admissibilité et d'utilité en temps polynomial. Pour aller plus loin, essayons d'identifier des cas où le problème est NP-complet, et des cas où il est dans P.

Un premier cas est trivial :

Proposition 7 *En l'absence de toute contrainte d'admissibilité ($\mathcal{C} = \emptyset$), avec ou sans dépendances entre objets dans les utilités individuelles, le problème d'existence d'un partage admissible d'utilité collective supérieure ou égale à K est dans* P.

Preuve : Dans ce cas, le partage qui consiste à allouer à chaque agent i tous les objets qu'il demande est optimal. La raison en est que les fonctions ua_i et uc sont non-décroissantes — en d'autres termes, (i) allouer plus d'objets ne diminue jamais la satisfaction d'un agent, et (ii) augmenter la satisfaction d'un agent, celle des autres étant constante, ne fait pas baisser l'utilité collective. Il suffit alors de vérifier que son utilité collective est supérieure ou égale à K. Le problème est donc dans P. □

Le second cas est presque aussi simple :

Proposition 8 *Le problème d'existence d'un partage admissible d'utilité collective supérieure ou égale à K est* NP*-complet.*

Preuve : Par réduction depuis le problème SAT : à toute formule propositionnelle ψ sur un alphabet $\{o_1, \ldots, o_p\}$ on associe le problème de partage défini par $N = 1$, $\mathcal{C} = \psi$ et $\Delta_1 = \{\langle o_{p+1}, 1\rangle\}$ (où l'affectation de o_i à **vrai** signifie que l'on attribue l'objet o_i au seul agent). ψ est satisfaisable si et seulement si il existe $\mathbf{x} \in Adm(\pi)$ tel que $uc(\mathbf{x}) \geq 1$. □

Examinons maintenant les autres cas, qui sont moins triviaux.

Proposition 9 *Si l'expression de $\mathcal{C}$ est limitée aux contraintes de préemption et si l'expression des préférences est limitée aux demandes élémentaires, la complexité du problème d'existence d'un partage admissible d'utilité collective supérieure ou égale à K est reportée sur le tableau suivant :*

$\Downarrow$ *f (phase 1)* | *g (phase 2)* $\Rightarrow$	$+$	$\min$
$+$	P	NP-*complet*
$\max$	P	P

Preuve : Soit $\Delta_i = \{\langle o^i_k, w^i_k\rangle, k = 1 \ldots |\Delta_i|\}$. Puisqu'il y a des contraintes de préemption, posons $a(\mathbf{x}, o)$ l'agent (unique) tel que $o \in x_i$.

– **cas (+,+)** : $\mathbf{x}$ est un partage optimal si et seulement s'il maximise

$$\sum_{i \in N} \sum_{k | o^i_k \in x_i} w^i_k = \sum_{o \in \mathcal{O}} \sum_{i \in N, o \in x_i} w^i_o = \sum_{o \in \mathcal{O}} w^{a(\mathbf{x},o)}_o,$$

donc si et seulement s'il maximise chacune des quantités $w^{a(\mathbf{x},o)}_o$ (on vend chaque objet au plus offrant) ; on détermine donc un partage optimal $\mathbf{x}^*$ en temps de l'ordre de $N \cdot |\mathcal{O}|$, et le résultat suit facilement, puisqu'il existe un partage d'utilité collective supérieure ou égale à K si et seulement si $uc(\mathbf{x}^*)(= \sum_{o \in \mathcal{O}} \max_{i \in N} w^i_k) \geq K$.

– **cas (+,min)** : on montre la NP-complétude dans le cas $N = 2$ (et donc **a fortiori** la NP-complétude dans le cas général). La NP-complétude du problème est prouvée par réduction polynomiale du problème de PARTITION : étant donné un ensemble de nombres $T = \{t_1, \ldots, t_p\}$ tel que $\sum_{i=1}^{p} t_i = 2M$, existe-t-il une partition de T en deux sous-ensembles A, B tels que $\sum\{t_i | t_i \in A\} = \sum\{t_i | t_i \in B\} = M$? À une instance de PARTITION on associe le problème de partage π défini par $\mathcal{O} = \{o_1, \ldots, o_p\}$, $N = 2$, $\mathcal{C}$ = contraintes de préemption, $\Delta_1 = \Delta_2 = \{\langle o_1, t_1\}, \ldots, \langle o_p, t_p\rangle\}$. Soit $\mathbf{x}$ un partage pour π . On a $uc(\mathbf{x}) \geq M$ si et seulement si $ua_1(\mathbf{x}) \geq M$ et $ua_2(\mathbf{x}) \geq M$. Or, les contraintes de préemption et l'identité $\sum_{i=1}^{p} t_i = 2M$ impliquent qu'on a $ua_1(\mathbf{x}) + ua_2(\mathbf{x}) \leq 2M$. On en conclut les équivalences entre (1) il existe $\mathbf{x}$ tel que $uc(\mathbf{x}) \geq M$, (2) il existe un partage $\mathbf{x}$ tel que $uc(\mathbf{x}) = M$ et (3) il existe un partage $\mathbf{x}$ tel que $ua_1(\mathbf{x}) = ua_2(\mathbf{x}) = M$. Enfin, on obtient aisément l'équivalence entre (3) et l'existence d'une partition pour T, d'où le résultat.

– **cas (max,+)** : soit $\mathbf{x}$ un partage. On a $uc(\mathbf{x}) \geq K$ si et seulement si $\sum_{i=1\ldots N} ua_i(\mathbf{x}) \geq K$, avec $K > 0$ et où $ua_i(\mathbf{x})$ est l'utilité de l'objet le plus cher (selon i) qui lui est alloué par $\mathbf{x}$. Puisque seule l'utilité de l'objet alloué d'utilité maximale compte, il suffit de considérer des partages où chaque agent reçoit au plus **un** objet : de tels partages sont dits **injectifs**. Soit $p = |\mathcal{O}|$. L'existence d'un partage injectif d'utilité collective $\geq K$ équivaut à l'existence d'un couplage de poids K dans le graphe biparti $G = (S, A)$ où $S = \mathcal{O} \cup N$ et $A = \{(o, i) \mid i \in N \text{ et } o \in F_i\}$, où F_i est l'ensemble des objets demandés par i (c'est-à-dire tels que Δ_i contienne un couple $\langle o_i, w_j^i\rangle$), et où le poids w_o^i de l'arête (o, i) est le poids de o dans Δ_i. Or, il est bien connu (voir par exemple [GON 79], chapitre 7) que le problème de couplage de poids maximum est polynomial. D'où le résultat.

– **cas (max, min)** : soit $\mathbf{x}$ un partage. On a $uc(\mathbf{x}) \geq K$ si et seulement si pour tout i, $ua_i(\mathbf{x}) \geq K$, c'est-à-dire : pour tout i il existe un objet o tel que $w_o^i \geq K$ et $o \in x_i$. On se ramène cette fois à la recherche d'un couplage de poids N dans le même graphe que ci-dessus, mais avec le poids c_o^i de l'arête (o, i) défini par

$$c_o^i = \begin{cases} 1 & \text{si } w_o^i \geq K \\ 0 & \text{sinon.} \end{cases}$$

Un couplage de poids N donne nécessairement un poids 1 à chaque agent dans ce problème de couplage, et correspond de façon univoque à un partage donnant à chaque agent i un objet o tel que $w_o^i \geq K$.

□

Nous n'avons étudié en détail que les quatre cas précédents. Il resterait — et c'est sans aucun doute une piste de recherche prometteuse — à étudier la complexité du problème de partage en l'absence de contraintes autre que les contraintes de préemption, lorsque d'autres choix sont faits pour les opérateurs d'agrégation, et en particulier lorsque la fonction choisie pour la phase 1 est $f = \text{leximax}$ et/ou la fonction choisie pour la phase 2 est $g = \text{leximin}$. On obtient facilement les deux résultats suivants : étant donné un problème de partage et un vecteur d'entiers $\vec{K} = (K_1, \ldots, K_N)$ tel que $K_1 \leq \ldots \leq K_N$, le problème de l'existence d'un partage $\mathbf{x}$ tel que $\overrightarrow{uc(\mathbf{x})} \geq_{\text{leximin}} \vec{K}$ est :

– NP-complet pour le cas $(f = +, g = \text{leximin})$: il suffit de reprendre la même réduction depuis le problème de PARTITION que dans la preuve pour le cas $(+, \min)$, en « remplaçant » M par le vecteur (M, M).

– dans P pour le cas $(\max, \text{leximin})$: il suffit de reprendre la même transformation en un problème de couplage que dans la preuve pour le cas $(\max, \min)$.

Par contre, on ignore sa complexité dans les cas $(f = \text{leximax}, g = \min)$ et $(f = \text{leximax}, g = \text{leximin})$.

On s'intéresse maintenant aux cas où il existe des dépendances préférentielles (positives ou négatives) entre objets.

Proposition 10 *Si l'expression de $\mathcal{C}$ est limitée aux contraintes de préemption, le problème d'existence d'un partage admissible d'utilité collective supérieure ou égale à K est* NP*-complet dans le cas $f = +$, et ceci même avec la restriction que toutes les dépendances entre objets sont positives ou avec la restriction qu'elles sont toutes négatives.*

Preuve : On montre la NP-complétude dans le cas où les dépendances sont négatives et où $(f = +, g = +)$ (les preuves dans les cas de dépendances positives et/ou $(f = +, g \in \{\min, \text{leximin}\})$ sont similaires) à l'aide de la réduction polynomiale suivante du problème NP-complet DOMINATING SET [GAR 79] vers le problème d'existence de partage d'utilité collective supérieure ou égale à K. Le problème ENSEMBLE DOMINANT s'exprime ainsi : étant donné un graphe non dirigé $G = (V, E)$ et un entier K, existe-t-il $V' \subseteq V$ tel que $|V'| \leq K$ et tel que pour tout $v \in V$, il existe un $u \in V'$ adjacent à v (c'est-à-dire tel que $\{u, v\} \in E$) ? À une instance $\langle G = (V, E), K \rangle$ de ce problème on associe le problème de partage suivant :

– $N = 2$;
– $\mathcal{O} = V$;
– $\Delta_1 = \{\langle \bigvee \{v \mid \{u, v\} \in E\}, 1 \rangle \mid u \in V\}$;
– $\Delta_2 = \{\langle u, 1 \rangle \mid u \in V\}$;
– $K' = 2 \cdot |V| - K$.

On remarque que les dépendances dans le profil de préférence Δ_1 sont négatives : en effet, si X_1 et X_2 sont deux ensembles d'objets tels que $X_1 \cap X_2 = \emptyset$, l'utilité pour l'agent 1 résultant de l'attribution de $X_1 \cup X_2$ est inférieure ou égale à la somme de l'utilité résultant de l'attribution de X_1 et de l'utilité résultant de l'attribution de X_2. Par ailleurs, il n'y a pas de dépendances dans Δ_2. Supposons qu'il existe un ensemble dominant $V' \subseteq V$ tel que $|V'| \leq K$. Soit alors le partage $\mathbf{x} = \langle x_1, x_2 \rangle$ avec $x_1 = V'$ et $x_2 = V \setminus V'$. Puisque V' est dominant on a $ua_1(\mathbf{x}) = |V|$; par ailleurs on a $ua_2(\mathbf{x}) = |x_2| = |V| - |V'|$; d'où $uc(\mathbf{x}) = 2 \cdot |V| - |V'| \geq 2 \cdot |V| - K$. Réciproquement, supposons qu'il existe un partage $\mathbf{x}$ tel que $uc(\mathbf{x}) \geq K'$. Les contraintes de préemption impliquent que $x_1 \cap x_2 = \emptyset$, et on peut par ailleurs supposer sans encombre que $x_1 \cup x_2 = V$ (puisque les utilités ne décroissent pas avec l'augmentation des ensembles d'objets alloués). Supposons maintenant que x_1 ne soit pas dominant pour G : il existe donc $a \in V$ tel que $\forall u \in V$ avec $\{u, a\} \in E$, on ait $u \notin x_1$. Considérons alors le partage $\mathbf{x}' = (x_1 \cup \{a\}, x_2 \setminus \{a\})$. $\mathbf{x}'$ est admissible et $ua_1(\mathbf{x}') \geq ua_1(\mathbf{x}) + 1$, $ua_2(\mathbf{x}') = ua_2(\mathbf{x}) - 1$, d'où $uc(\mathbf{x}') \geq uc(\mathbf{x})$. En itérant ce procédé jusqu'à ce que la part de 1 soit dominante pour G, on obtient un partage $\mathbf{y}$ tel que y_1 est un ensemble dominant pour G et $uc(\mathbf{y}) \geq K'$. Or, $uc(\mathbf{y}) = 2 \cdot |V| - |y_1|$ et $uc(\mathbf{y}) \geq K'$ impliquent $|y_1| = 2|V| - uc(\mathbf{y}) \leq 2 \cdot |V| - (2 \cdot |V| - K)$, c'est-à-dire $|y_1| \leq K$: il existe donc un sous-ensemble de sommets dominant pour G de cardinalité inférieure ou égale à K. $\square$

Les résultats sont résumés sur la figure 1.

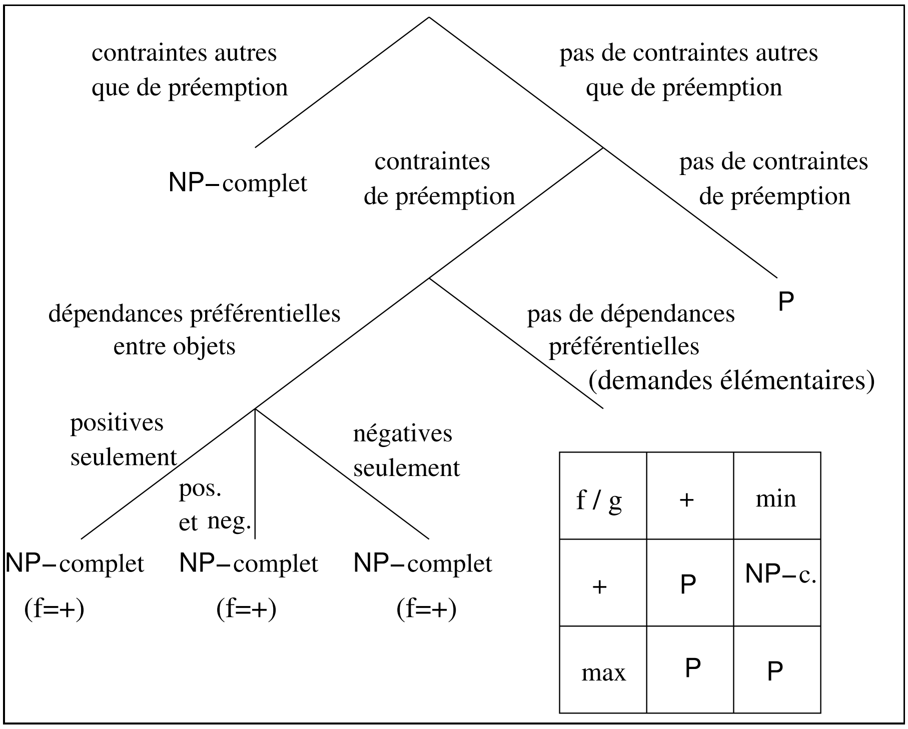

Figure 1. *Résumé des résultats acquis sur la complexité du problème de décision.*

Nous finissons cette section par de brèves considérations sur la complexité des problèmes (2) et (3). En ce qui concerne le problème 2 (déterminer si un partage donné est optimal) : lorsqu'une classe de problèmes de partage est telle que le problème d'existence d'un partage d'utilité supérieure ou égale à K est dans P (respectivement, NP-complet), alors le problème qui consiste à déterminer si un partage donné est optimal est également dans P (respectivement, coNP-complet). En ce qui concerne le problème 3 (déterminer si tous les partages optimaux satisfont une propriété exprimable par une formule logique) : lorsqu'une classe de problèmes de partage est telle que le problème d'existence d'un partage d'utilité supérieure ou égale à K est dans P, on ne peut toujours garantir que le problème reste dans P, puisque la prise en compte de la propriété équivaut techniquement à l'ajout d'une contrainte (autre que de préemption) – on peut seulement dire qu'il est dans coNP (il est cependant dans P si le nombre de partages optimaux reste polynomialement borné) ; enfin, lorsqu'une classe de problèmes de partage est telle que le problème d'existence d'un partage d'utilité supérieure ou égale à K est NP-complet, le problème (3) est dans Δ_2^p, c'est-à-dire que sa résolution nécessite un nombre polynomial de résolutions de problèmes dans NP ou dans coNP[1].

4. Éléments d'algorithmique

Dans cette section, nous proposons dans un premier temps un algorithme simple et très général, de type séparation-évaluation (*Branch and Bound*), qui calcule un partage d'utilité collective maximale pour une instance de problème de partage donné. Nous discutons ensuite de son adaptation à des cas spécifiques, notamment aux choix effectués pour l'application de partage de ressources satellitaires décrite dans notre précédent article [FAR], section 7. Enfin, nous évoquons des pistes d'algorithmes non complets (c'est-à-dire sans garantie d'optimalité).

4.1. *Préliminaires*

Un partage est totalement déterminé lorsqu'on connait, pour chaque objet $o \in \mathcal{O}$, l'ensemble des agents auxquels il est attribué (rappelons que dans le cas général dans lequel nous nous plaçons, un objet « non consommable » peut être attribué à plusieurs agents). Autrement dit, un partage est complètement déterminé si pour chaque couple $(o, i) \in \mathcal{O} \times N$, on sait si l'objet o est attribué à l'agent i ou non. Il est donc intéressant de considérer comme **variables de décision** d'un problème de partage les éléments $(o, i) \in \mathcal{O} \times N$. Ces variables prendront deux valeurs possibles : 1 lorsque o est attribué à i, et 0 dans le cas contraire. Nous nommerons **attribution**, toute fonction $a : \mathcal{O} \times N \longmapsto \{0, 1\}$. Une attribution a détermine trivialement et de manière unique un partage $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ défini par $x_i = \{o \in \mathcal{O} \mid a(o, i) = 1\}$.

On dira qu'une attribution est admissible lorsque l'unique partage correspondant est admissible. Nous étendrons naturellement les fonctions d'utilité individuelle et collective aux attributions : l'utilité d'une attribution est l'utilité du partage correspondant.

On remarque qu'il est suffisant d'examiner les partages $\mathbf{x}$ tels que $x_i \in \mathcal{D}_i$ (on rappelle que $\mathcal{D}_i$ est l'ensemble des objets demandés par l'agent i), ceci parce que attribuer à un agent un objet qu'il ne demande pas ne modifie pas l'utilité individuelle de cet agent. L'utilité collective n'est pas modifiée non plus, car elle dépend uniquement des utilités individuelles. Il est donc inutile d'examiner les partages attribuant des objets non demandés puisque ces partages ne peuvent pas augmenter l'utilité collective. Les attributions intéressantes étant limitées à celles concernant les objets demandés, il suffit de considérer l'ensemble des couples $\mathcal{A} = \{(o, i) \in \mathcal{O} \times N \mid o \in \mathcal{D}_i\}$, au lieu

1. Dans ce dernier cas, on peut réutiliser des résultats de complexité de [KON 02].

de $\mathcal{O} \times N$. Nous nommerons $\{v_1, v_2, \ldots, v_m\}$ les $m = \sum_{i \in N} |\mathcal{D}_i|$ éléments de $\mathcal{A}$, que nous prenons donc comme variables de décision.

La notion d'attribution est intéressante non seulement pour l'expression de l'algorithme qui suit, mais aussi pour l'expression des contraintes d'admissibilité $\mathcal{C}$. Ainsi, une contrainte de préemption étendue telle que « *l'objet o ne peut être attribué qu'à p agents au plus* » s'exprime par

$$\sum_{i \in N_o} a(o,i) \leq p, \text{ avec } N_o = \{i \mid o \in \mathcal{D}_i\}$$

(N_o est l'ensemble des agents qui demandent o) et peut s'exprimer quasi-directement dans un langage de programmation par contraintes tel que OPL [Van 99] ou celui du système CHOCO [LAB 00]. Pour une contrainte d'exclusion entre objets telle que « *il est impossible de sélectionner en même temps tous les éléments du sous-ensemble d'objets $\mathcal{Q}$* » on écrira

$$\sum_{o \in \mathcal{Q}} s_o < |\mathcal{Q}|, \text{ avec } s_o = \max_{i \in N_o} a(o,i)$$

($s_o = 1$ si l'objet o a été sélectionné, c'est-à-dire attribué à un agent au moins, et 0 sinon).

Nous avons, dans la section 3, étudié la complexité du problème de décision suivant : étant donné un problème de partage π et une valuation K, existe-t-il un partage admissible $\mathbf{x}$ tel que $uc(\mathbf{x}) \geq K$? Il a été montré que dans la plupart des cas intéressants ce problème de décision est NP-complet. Autrement dit dans ces cas — et en particulier dans le cas de l'application de partage de ressources satellitaires décrite dans [FAR , section 7] — le problème d'optimisation associé est NP-difficile. Il est donc probable que seuls des algorithmes énumératifs du type de celui proposé dans le paragraphe suivant sont susceptibles de fournir des partages optimaux.

4.2. *Un algorithme énumératif de calcul de partages optimaux*

L'algorithme utilise les structures de données suivantes. On représente une attribution a par le tableau $A[\ell], \ell = 1, \ldots, m$, avec $A[\ell] = a(v_\ell)$ et $m = \sum_{i \in N} |\mathcal{D}_i|$. Autrement dit, soit $v_\ell = (o,i)$, alors $A[\ell]$ représente la décision (affirmative, négative, ou non encore prise) d'attribution de o à i. On notera $A[1, \ldots, k]$ l'attribution partielle courante, représentant des décisions d'attribution pour les k premiers couples de $\mathcal{A}$. Le tableau $\widehat{A}$ mémorise la meilleure attribution (complète) trouvée jusqu'ici par l'algorithme.

L'algorithme, de type séparation-évaluation, est décrit figure 2. Il s'agit d'un parcours arborescent de l'ensemble des attributions possibles, en profondeur d'abord, exprimé récursivement. On commence par initialiser $\widehat{A}$ par une attribution admissible, obtenue rapidement, si possible de bonne qualité. Dans la plupart des problèmes de

```
PARTAGEOPTIMAL() :
// Entrée : une instance d'un problème de partage
// Sortie : un partage maximisant la fonction d'utilité collective
  Â ← une attribution admissible quelconque
  PARCOURS(0)
  retourner le partage correspondant à Â

PARCOURS(k) :
  si k = m alors Â ← A // on a trouvé une meilleure attribution
  sinon
    pour d ← 1, 0
      A[k] ← d
      si SNONADM(A, k) ou uc(Â) ≥ uc̄(A, k)
      alors retour
      sinon PARCOURS(k + 1)
```

Figure 2. *Algorithme de type séparation-évaluation — Branch and Bound — calculant un partage optimal.*

```
uc̄(A, k) :
  pour ℓ ← 1, ..., m
    Ā[ℓ] ← A[ℓ] si ℓ ≤ k, 1 sinon
  retourner uc(Ā)
```

Figure 3. *Fonction calculant un majorant de l'utilité collective.*

partage, l'attribution vide (qui n'attribue rien) est admissible et peut servir de $\widehat{A}$ initial. L'argument k de la procédure PARCOURS est le nombre de décisions d'attribution dans l'attribution partielle courante (il y a m décisions d'attribution à prendre).

Les contraintes d'admissibilité sont vérifiées par la procédure SNONADM(A, k) — pour « sûrement non admissible ». Cette procédure rend **vrai** lorsqu'il est certain que toute extension complète de $A[1, \ldots, k]$ est non admissible, et **faux** sinon. Remarquone que la procédure peut rendre **faux** même lorsque l'attribution partielle courante est non admissible, dans le cas où la preuve de l'infaisabilité n'est pas acquise. Ceci vient du fait que l'algorithme vérifiant la faisabilité d'une attribution partielle peut être incomplet, par exemple, ne vérifiant que les contraintes « instanciées ». La recherche est donc abandonnée dans cette branche — on parle de « coupe » — si la procédure SNONADM rend **vrai**.

Examinons maintenant l'autre possibilité de coupe. L'algorithme utilise la fonction $\overline{uc}(A, k)$, qui donne par définition un majorant de l'utilité collective de la meilleure attribution admissible étendant l'attribution partielle courante. Lorsque $uc(\widehat{A}) \geq \overline{uc}(A, k)$, il est certain qu'on ne pourra jamais dans cette branche trouver strictement mieux que

$\widehat{A}$. La figure 3 décrit une façon simple d'obtenir un tel majorant : soit $\overline{A}$ l'attribution obtenue en attribuant à chaque agent tous les objets qu'il a demandés, parmi ceux dont l'attribution n'est pas encore décidée dans $A[1,\ldots,k]$. Les propriétés de monotonie des différentes fonctions d'utilité font que $uc(\overline{A})$ est bien un majorant. Montrons cela formellement.

Proposition 11 *La propriété $uc(\overline{A}) \geq uc(A)$ est vraie pour toute extension admissible A de $A[1,\ldots,k]$.*

Preuve : Soit $\overline{\mathbf{x}} = \langle \overline{x}_1, \ldots, \overline{x}_n \rangle$ et $\mathbf{x}$ les partages correspondant aux attributions $\overline{A}$ et A respectivement, A étant une extension admissible quelconque de $A[1,\ldots,k]$. Il faut montrer

$$uc(\overline{\mathbf{x}}) \geq uc(\mathbf{x}) \tag{1}$$

Nous allons d'abord montrer la propriété

$$o \in x_i \implies o \in \overline{x}_i, \text{ pour tout } o \in \mathcal{D}_i \tag{2}$$

Soit o un objet quelconque de $\mathcal{D}_i$, et soit $v_\ell = (o, i)$ la variable correspondante dans $\mathcal{A}$. Deux cas sont possibles :

1) $\ell \leq k$. Nous sommes dans la partie commune des attributions $\overline{A}$ et A, donc $\overline{A}[\ell] = A[\ell]$, autrement dit $o \in \overline{x}_i$ si et seulement si $o \in x_i$. La propriété (2) est donc vérifiée.

2) $\ell > k$. Nous sommes dans la partie extension de $A[1,\ldots,\ell]$, et donc $\overline{A}[\ell] = 1$ par construction de $\overline{A}$. Autrement dit $o \in \overline{x}_i$, et la propriété (2) est encore vérifiée.

Nous venons donc de démontrer la propriété (2), qui est équivalente à

$$\overline{x}_i \supseteq x_i, \text{ pour tout } i \in N \tag{3}$$

Nous allons utiliser maintenant les hypothèses de monotonie des fonctions d'utilité individuelle et collective, discutées dans [FAR , sections 5 et 6]. Nous pouvons supposer que ces fonctions sont monotones au sens suivant :

$$y_i \supseteq z_i \implies ua_i(\mathbf{y}) \geq ua_i(\mathbf{z}) \tag{4}$$

$$[ua_i(\mathbf{y}) \geq ua_i(\mathbf{z}) \text{ pour tout } i \in N] \implies uc(\mathbf{y}) \geq uc(\mathbf{z}) \tag{5}$$

où $\mathbf{y}$ et $\mathbf{z}$ sont des partages quelconques. La preuve de (1) s'obtient en combinant (3), (4) et (5). □

4.3. *Considérations sur l'efficacité de l'algorithme*

Le nombre de partages possibles est $2^m = 2^{\sum_i |\mathcal{D}_i|}$, ce qui donne une idée de la nature fortement combinatoire du problème. Il est donc clair que cet algorithme énumératif — même si, grâce aux coupes, il n'énumère pas toutes les attributions possibles — trouve rapidement ses limites lorsque la taille des instances augmente. En pratique, ses performances vont fortement dépendre de quelques facteurs dont nous discutons maintenant.

Un facteur très important, qui vient d'être évoqué, concerne l'efficacité des coupes :

– pour que la procédure SNONADM produise de bonnes coupes, il faut qu'elle puisse décider de l'inadmissibilité d'une attribution partielle[2] si possible sans attendre que toutes les variables $A[\ell]$ soient affectées. Cette possibilité dépend bien sûr de la structure des contraintes d'admissibilité. Pour propager efficacement les contraintes, on se tournera avec profit vers les techniques de filtrage connues dans le domaine des CSP (voir par exemple [BES 97, DEB 98]).

– dans tout algorithme de type séparation-évaluation, un compromis est à trouver entre la qualité du majorant — il doit être le plus petit possible pour assurer de bonnes coupes — et son temps de calcul, qui doit rester court. La solution proposée ci-dessus figure 3 est simpliste. Il conviendrait là encore de l'adapter à la structure des contraintes d'admissibilité et aux structures d'utilité propres à chaque application. On cherchera par exemple à obtenir un meilleur compromis qualité/temps en relaxant partiellement (et non totalement) les contraintes d'admissibilité.

D'autre part, dans la perspective d'un calcul en temps limité — ce qui est toujours le cas en pratique — il est important d'obtenir rapidement de bonnes attributions $\widehat{A}$. Cela favorise en outre les bonnes coupes et renforce donc l'efficacité de l'algorithme. Il faut donc diriger l'algorithme vers les régions de l'espace de recherche de meilleure qualité. Les propriétés de monotonie des fonctions d'utilité nous indiquent qu'on doit pour cela essayer d'attribuer le plus d'objets au plus grand nombre d'agents. C'est la raison pour laquelle, dans la procédure PARCOURS de la figure 2, la valeur $d = 1$ est tentée en premier (heuristique sur les valeurs). On pourra d'autre part jouer sur l'ordre d'affectation des variables $A[\ell]$, en commençant par celles dont l'affectation à 1 favorise le plus la qualité du partage, c'est-à-dire celles qui correspondent aux demandes les plus importantes de chaque agent. Dans la version présentée figure 2, l'ordre d'affectation des variables est fixe, mais dans ce type d'algorithme, un ordre calculé dynamiquement est souvent beaucoup plus efficace.

4.4. *Normalisation*

Les opérations de normalisation des utilités individuelles [FAR , section 5.4] nécessitent le calcul préalable des utilités individuelles maximales ua_i^* pour chaque agent. L'algorithme précédent est aisément adaptable à de calcul : il suffit de le restreindre au cas d'un partage réduit à un seul agent. Nous avons dans ce cas identité entre utilité collective et utilité individuelle.

2. Nous dirons qu'une attribution partielle est non admissible lorsque toute extension complète de cette attribution partielle est non admissible.

4.5. *Adaptations*

L'algorithme général présenté ci-dessus ne dit rien sur la représentation et le calcul des différentes utilités individuelles et collective. Nous présentons dans ce paragraphe des adaptations de l'algorithme à des situations particulières, et notamment aux choix spécifiques effectués pour l'application de partage de ressources satellitaires décrite dans notre précédent article [FAR], section 7.

4.5.1. *Fonctions d'utilité*

Rappelons les traits importants de l'algorithme énumératif décrit section 4.2 :

– le critère à optimiser est l'utilité collective $uc(\mathbf{x})$,

– on cherche à évaluer des attributions **partielles** plutôt que complètes, de manière à pouvoir éventuellement couper au plus tôt de mauvaises branches de l'arbre de recherche,

– les attributions partielles sont construites par extensions successives, par ajout de décisions concernant à chaque fois l'attribution d'un seul objet à un seul agent (attributions « atomiques »).

Supposons une attribution partielle $A[1,\ldots,k-1]$, et soit à calculer l'utilité collective $uc(A[1,\ldots,k])$ de l'attribution qui en découle par l'allocation atomique $A[k] \leftarrow d$, $d \in \{0,1\}$. Nous savons que cette utilité collective est définie par une certaine agrégation g des utilités individuelles ua_i correspondantes, elles-mêmes définies par agrégation des poids des demandes satisfaites :

$$\begin{aligned} ua_i(\mathbf{x}) &= f(\{w \mid \langle\varphi, w\rangle \in \Delta_i \text{ et } x_i \models \varphi\}) \qquad (6)\\ ua_i(\mathbf{x}) &= 0 \text{ si } Sat_i(\mathbf{x}) = \emptyset \end{aligned}$$

avec f associatif et commutatif (voir justification [FAR , section 5.1]).

Pour calculer $uc(A[1,\ldots,k])$, il serait dommage de recalculer toutes les ua_i à partir de la définition précédente, alors que nous avons déjà calculé $uc(A[1,\ldots,k-1])$ ainsi que toutes les utilités individuelles de $A[1,\ldots,k-1]$, car peu de modifications ont été apportées par la dernière attribution atomique :

– les demandes déjà satisfaites vont le rester, donc leurs contributions dans les ua_i seront inchangées,

– si l'objet alloué ne concerne pas l'agent i, son ua_i ne va pas changer, il est inutile de la recalculer.

Il est donc particulièrement avisé de s'intéresser uniquement aux incidences de la dernière décision d'attribution atomique sur les utilités individuelles des agents. Autrement dit, il faut calculer les utilités individuelles de façon incrémentale, sinon on s'expose à répéter de nombreuses fois les mêmes calculs dans des branches de l'arbre de recherche ayant un tronc commun. Cela est rendu possible grâce aux propriétés

```
LEXIMINSUPEG(u, v) :
// Entrée : deux vecteurs d'utilité individuelle u et v
// Sortie : vrai si u ≥leximin v, faux sinon
  u* ← u ordonné par valeurs non décroissantes
  v* ← v ordonné par valeurs non décroissantes
  pour i ← 1, ..., n
      si u*_i > v*_i retourner vrai
      si v*_i > u*_i retourner faux
  retourner vrai
```

Figure 4. *Algorithme décidant l'ordre leximin.*

d'associativité et de commutativité : on maintiendra de façon incrémentale les quantités $ua_i(A[1,\dots,k])$, définies comme suit, pour $k \in [1,\dots,m]$:

$$\begin{array}{rcl} ua_i(A[]) & = & 0 \\ ua_i(A[1,\dots,k]) & = & f(ua_i(A[1,\dots,k-1]), f(\{w \mid \langle w,\varphi\rangle \in \Delta_i^k\})) \end{array} \tag{7}$$

où Δ_i^k est l'ensemble des demandes de Δ_i non instanciées[3] par $A[1,\dots,k-1]$ mais instanciées et satisfaites par $A[1,\dots,k]$. L'utilité individuelle partielle ainsi définie est un minorant de l'utilité individuelle de toutes les extensions complètes de l'attribution courante. Elle correspond bien à l'utilité individuelle d'une attribution lorsque celle-ci est complète.

Dans le cas simple où toutes les demandes sont élémentaires, c'est-à-dire réduites à un seul objet, les utilités individuelles deviennent

$$\begin{array}{rcl} ua_i(\mathbf{x}) & = & f(\{w \mid \langle o,w\rangle \in \Delta_i \text{ et } o \in x_i\}) \\ ua_i(\mathbf{x}) & = & 0 \text{ si } x_i = \emptyset \end{array} \tag{8}$$

et la maintenance incrémentale des utilités individuelles partielles s'effectue alors simplement, pour $k \in [1,\dots,m]$:

$$\begin{array}{rcl} ua_i(A[]) & = & 0 \\ ua_i(A[1,\dots,k]) & = & f(ua_i(A[1,\dots,k-1]), w_o^i) \text{ si } A[k] = 1 \text{ et } v_k = (o,i) \\ & = & ua_i(A[1,\dots,k-1]) \text{ sinon} \end{array} \tag{9}$$

où w_o^i dénote le poids de o dans Δ_i.

3. On dira qu'une demande φ est instanciée par $A[1,\dots,k]$ si pour tout littéral o de φ, il existe une variable $v_\ell = (o,i)$, avec $\ell \in [1,\dots,k]$.

GLOUTONSTOCHASTIQUE() :
// *Entrée : une instance d'un problème de partage*
// *Sortie : un bon partage pour l'ordre de préférence collective*
$\widehat{A} \leftarrow$ une attribution admissible quelconque
effectuer DESCENTE(0) autant de fois que possible dans le temps imparti
retourner le partage correspondant à $\widehat{A}$

DESCENTE(k) :
si $k = m$ **alors** $\widehat{A} \leftarrow A$
sinon
$A[k] \leftarrow 1$ avec la probabilité p, 0 avec la probabilité $1 - p$
si SNONADM(A, k) **ou** $u(\widehat{A}) \geq \overline{uc}(A, k)$
alors retour
sinon DESCENTE$(k + 1)$

Figure 5. *Un algorithme incomplet calculant un bon partage, basé sur des gloutons successifs suivant une heuristique biaisée.*

4.5.2. *Ordre collectif leximin*

Examinons maintenant le cas où la fonction d'utilité collective code l'ordre leximin (c'est ce que nous avons choisi pour les deux modes hiérarchique et commercial de notre application de partage de ressources satellitaires). Il est alors plus avantageux de ne pas agréger les utilités individuelles en utilité collective, mais de travailler directement sur les vecteurs d'utilité individuelle (profils de satisfaction). Soit $\mathbf{ua}(A) = \langle ua_1(A), \ldots, ua_n(A)\rangle$. En pratique, on remplacera dans l'algorithme figure 2, la comparaison $uc(\widehat{A}) \geq \overline{uc}(A, k)$ par l'appel à la procédure LEXIMINSUPEG$(\mathbf{ua}(\widehat{A}), \mathbf{ua}(\overline{A}))$, décrite figure 4, qui implémente la comparaison selon l'ordre leximin. On calculera $\overline{A}$ comme décrit figure 3. Toutefois deux optimisations sont possibles. La première, évidente, consiste à trier une seule fois le vecteur $\mathbf{ua}(\widehat{A})$, à chaque fois qu'un nouveau $\widehat{A}$ est trouvé. Pour la seconde optimisation, au fur et à mesure de la construction de $\overline{A}$, on met à jour les utilités individuelles partielles $ua_i(\overline{A}[1, \ldots, \ell])$ et dès que l'une d'elle dépasse strictement la quantité $\min_i ua_i(\widehat{A})$, on est certain que le test LEXIMINSUPEG$(\mathbf{u}(\widehat{A}), \mathbf{u}(\overline{A}))$ rendra **faux**, sans attendre la construction complète de $\overline{A}$. D'autre part, dans la procédure PARCOURS figure 2, il est judicieux d'utiliser un ordre dynamique d'affectation des variables dans lequel on choisit comme prochaine attribution élémentaire à considérer, celle qui fait le plus augmenter l'utilité individuelle de l'agent le moins satisfait dans l'attribution partielle courante.

4.6. *Algorithmes incomplets*

Dans beaucoup de cas, l'algorithme précédent, qualifié de « complet » parce qu'il fournit un solution garantie optimale, utilise un temps de calcul prohibitif et n'est donc pas utilisable en pratique. Il faut alors se résigner à des algorithmes incomplets, c'est-à-dire calculant des partages de bonne qualité mais sans garantie d'optimalité.

Toute la panoplie des recherches locales à base de parcours dans des voisinages sont à notre disposition (voir par exemple [AAR 97]). L'efficacité de ces méthodes repose là aussi en bonne partie sur la capacité de mise-à-jour incrémentale et rapide des différentes structures de préférence.

Dans une veine différente, citons deux méthodes qui sont des adaptations simples et non complètes de l'algorithme complet décrit plus haut. La première est une recherche à divergence limitée (*Limited Discrepancy Search*) [HAR 95] qui tente d'exploiter au mieux l'heuristique sur le choix de valeurs. On conserve l'idée d'exploration arborescente systématique, mais on la limite à une région de l'espace dans lequel on s'écarte peu de l'heuristique du choix de valeurs. Pour cela, on compte le nombre de fois b où les choix de valeurs dans l'affectation partielle courante s'écartent de l'heuristique (dans notre cas, on compte le nombre de fois où on fait le choix $A[k] \leftarrow 0$ dans une branche, c'est-à-dire $b = \sum_{\ell=1}^{k}(1 - A[\ell])$). Dès que b est supérieur à un seuil fixé δ, la branche est coupée. On peut soit fixer δ à une petite valeur, soit effectuer m recherches successives avec $\delta = 0, \ldots, m - 1$, auquel cas l'algorithme redevient complet.

La seconde adaptation simple de l'algorithme complet est décrite figure 5. L'algorithme consiste en une suite de choix gloutons (sans retour arrière) guidés par des choix heuristiques aléatoirement biaisés (*Heuristic-Biased Stochastic Searching*) [BRE 96]. On remarquera la proximité de la structure de cet algorithme avec celle de l'algorithme complet figure 2. Le comportement de l'algorithme dépend de la valeur du paramètre p : plus p est proche de 1, plus il suit l'heuristique de valeur. Pour de bons résultats, la valeur de p doit être adaptée à la structure des contraintes d'admissibilité et à la structure des préférences, si possible dynamiquement. De multiples variantes sont possibles, notamment celle qui choisit dynamiquement l'ordre d'affectation des variables, là aussi selon une heuristique biaisée aléatoirement.

5. Conclusion

Le problème du partage équitable de ressources discrètes est un problème pratique et théorique important. C'est un cadre général pour les problèmes de répartition d'objets entre agents, tenant compte de leurs préférences, et des contraintes d'admissibilité sur les partages. Ce cadre englobe le cadre déjà connu des enchères combinatoires [ROT 98].

Nous avons dans cet article analysé la complexité théorique du problème, sous diverses hypothèses réalistes. Nous avons montré que dans tous les cas intéressants, le problème de décision associé est NP-complet. Puis nous avons décrit et discuté des algorithmes très généraux de résolution du problème du partage d'utilité collective maximale, à la fois dans des versions complètes et incomplètes.

Beaucoup reste à faire dans le domaine de la résolution des problèmes de partage. En particulier, les algorithmes proposés ici constituent seulement une première approche de leur résolution. Aller plus loin nécessite soit de poser des hypothèses supplémentaires concernant les différents opérateurs d'agrégation et les propriétés des demandes, soit, dans une démarche opposée, de plonger le problème dans un cadre de résolution général tel que la programmation par contraintes. À cet égard, explorer la connexion entre problèmes de partage et CSP pondérés [BIS 99] — pour lesquels d'intéressants travaux algorithmiques sont en cours [LAR 99, SCH 00] — pourrait s'avérer très fructueux.

6. Bibliographie

[AAR 97] AARTS E., LENSTRA J. K., *Local Search in Combinatorial Optimization*, John Wiley & Sons, 1997.

[BES 97] BESSIÈRE C., RÉGIN J., « Arc Consistency for General Constraint Networks : Preliminary Results », *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, 1997, p. 398-404.

[BIS 99] BISTARELLI S., MONTANARI U., ROSSI F., SCHIEX T., VERFAILLIE G., FARGIER H., « Semiring-Based CSPs and Valued CSPs : Frameworks, Properties and Comparison », *Constraints*, vol. 4, n° 3, 1999, p. 199-240.

[BRE 96] BRESINA J. L., « Heuristic-Biased Stochastic Sampling », *Proc. of the 13th National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, USA, 1996, p. 271–278.

[DEB 98] DEBRUYNE R., « Consistances locales pour les problèmes de grande taille », mémoire de thèse, Université Montpellier II, Décembre 1998.

[FAR] FARGIER H., LANG J., LEMAÎTRE M., VERFAILLIE G., « Partage équitable de ressources communes : (1) un modèle général et son application au partage de ressources satellitaires », *Technique et Science Informatiques*, dans ce numéro.

[GAR 79] GAREY M., JOHNSON D., *Computers and intractability : a guide to the theory of NP-completeness*, Freeman, 1979.

[GON 79] GONDRAN M., MINOUX M., *Graphes et algorithmes*, Eyrolles, 1979.

[HAR 95] HARVEY W. D., GINSBERG M. L., « Limited Discrepancy Search », *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995, p. 607–613.

[KON 02] KONIECZNY S., LANG J., MARQUIS P., « Distance-based merging : a general framework and some complexity results », *Proceedings of KR2002*, 2002, p. 97-108.

[LAB 00] LABURTHE F., « CHOCO : Implémentation du noyau d'un système de contraintes », *Actes des 6ièmes Journées Nationales sur la Résolution Pratique de Problèmes NP-Complets (JNPC-00)*, Marseille, France, 2000, `http://www.choco-constraints.net`.

[LAR 99] LARROSA J., MESEGUER P., SCHIEX T., « Maintaining reversible DAC for Max-CSP », *Artificial Intelligence*, vol. 107, 1999, p. 149–163.

[PAP 94] PAPADIMITRIOU C. H., *Computational complexity*, Addison–Wesley, 1994.

[ROT 98] ROTHKOPF M. H., PEKEČ A., HARSTAD R. M., « Computationally manageable combinatorial auctions », *Management Science*, vol. 44, n° 8, 1998, p. 1131-1147.

[SCH 00] SCHIEX T., « Arc Consistency for Soft Constraints », *Proc. of the 6th International Conference on Principles and Practice of Constraint Programming (CP-00)*, Singapore, 2000.

[Van 99] VAN HENTENRYCK P., *The OPL Optimization Programming Language*, The MIT Press, Cambridge, Massachusetts, 1999.