République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A. Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de Fin de Cycle

En vue de l'obtention du diplôme de Master Recherche en Informatique

Option : Intelligence Artificielle

Thème

Méthodes avancées de génération de structures de coalitions dans les systèmes multi-agents

Réalisé par

M. TAGUELMIMT Redha

Devant le jury composé de

Présidente :	M^{me} ADEL-AISSANOU Karima	MCA	Université de Béjaïa
Examinateur :	M. AISSANI Sofiane	MCA	Université de Béjaïa
Examinateur :	M. AMROUN Kamal	MCA	Université de Béjaïa
Encadrante :	M^{me} BOUKREDERA Djamila	MCA	Université de Béjaïa
Co-encadrant :	M. AKNINE Samir	Prof.	Université de Lyon 1

Promotion 2019 - 2020

Remerciements

C'est avec un immense plaisir que je réserve ces quelques lignes en signe de gratitude et de reconnaissance à toute personne ayant contribué de prés ou de loin à l'accomplissement de ce travail.

Tout d'abord, je tiens à exprimer ma profonde gratitude et adresser mes remerciements les plus sincères à mon encadrante M^{me} BOUKREDERA Djamila pour sa disponibilité, sa patience, ses recommandations judicieuses et son précieux suivi tout au long de la réalisation de ce travail. Sa supervision, pleine de critiques et de phrases encourageantes, m'a vraiment donné la possibilité de m'exprimer.

Je tiens également à remercier mon co-encadrant M. AKNINE Samir qui a partagé avec moi ses brillantes intuitions et propositions. Je le remercie aussi pour son soutien, ses conseils, sa disponibilité, ses remarques pour améliorer ce travail et l'aide précieuse qu'il m'a apporté.

Je souhaiterais remercier aussi les membres de mon jury, M^{me} Adel-Aissanou Karima (présidente du jury), M. AISSANI Sofiane (examinateur) et M. AMROUN Kamal (examinateur) pour avoir consacré une partie de leur temps à la lecture de ce mémoire et pour l'intérêt qu'ils ont porté à ce travail.

Mes remerciements s'étendent à tous mes enseignants du département d'Informatique de l'Université Abderrahmane Mira de Béjaïa. Je resterais reconnaissant à leur participation à ma formation.

Je remercie enfin l'ensemble des personnes qui ont contribué d'une manière ou d'une autre à l'élaboration de ce travail.

Dédicaces

À toute ma famille et à tous mes amis, surtout à mes très chers parents.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

ADP	Average-based Dynamic Programming
CSE	Coalition Structure Enumeration
CSG	Coalition Structure Generation
CS	Coalition Structure
DP	Dynamic Programming
IDP	Improved Dynamic Programming
IP	Integer Partition
ODP	Optimal Dynamic Programming
SMA	Système Multi-Agents

Introduction générale

L'essor que connait l'informatique ainsi que les récents progrès de l'intelligence artificielle ont permis de résoudre des tâches de plus en plus complexes, ce qui a permis d'intégrer de plus en plus d'intelligence et d'autonomie dans les systèmes informatiques. L'une des préoccupations des recherches qui en résultent est de doter les systèmes informatiques de capacités de raisonnement habituellement attribuées à l'intelligence humaine tout en préservant les propriétés de distribution du contrôle de ces systèmes. Ceci est l'un des principaux objectifs des systèmes multi-agents dont le but est de faire coopérer différents systèmes intelligents appelés agents, et qui sont donc inspirés des systèmes distribués [9].

Les systèmes multi-agents recèlent un énorme potentiel pour modéliser les problèmes et les résoudre, et se distinguent par l'ensemble des agents autonomes qui les composent. Une caractéristique clé d'un système multi-agents est que les différents agents qui le composent ont des compétences et des capacités différentes, et que chaque agent est limité en termes de capacités [16]. La coopération et la coordination entre les agents sont donc nécessaires pour leur permettre d'atteindre des objectifs importants et complexes. Le concept de formation de coalitions, où les agents doivent coopérer avec d'autres pour améliorer leurs capacités et leurs avantages, est pertinent pour la recherche sur les systèmes multi-agents. Un objectif clair des systèmes multi-agents est de construire des agents qui peuvent soit prendre des actions communes et coordonnées pour améliorer leurs performances et compenser les faiblesses de chacun des agents, soit atteindre des objectifs qui dépassent les capacités individuelles des agents.

Dans ce contexte, de nombreux travaux de recherche sont proposés et ont obtenu un certain succès pour la résolution du problème de génération de structures de coalitions. Certains chercheurs se sont intéressé à développer et à améliorer des algorithmes exacts qui retournent la structure de coalitions optimale. D'autres se sont tournés vers le développement d'algorithmes capables de produire des solutions à tout moment de l'exécution afin de permettre une interruption prématurée. En raison du nombre exponentiel de structures de coalitions, d'autres travaux se sont basés sur des heuristiques qui se focalisent sur la rapidité et ne garantissent généralement pas d'avoir la solution optimale. De tels algorithmes, présentent la seule option pratique lorsque le nombre d'agents augmente et que le problème devient trop difficile à résoudre par des algorithmes exacts.

Notre contribution consiste à proposer des algorithmes pour la génération de structures de coalitions. Le premier algorithme proposé est basé sur une nouvelle représentation de l'espace de recherche qui permet d'évaluer directement les structures de coalitions et ainsi de produire des solutions à tout moment de l'exécution. Notre heuristique combinée à cette représentation fournit une nouvelle technique efficace de parcours de l'espace de recherche. L'évaluation de performances a montré son efficacité en termes de temps et de qualité des solutions trouvées par rapport à l'algorithme exact le plus rapide à ce jour. Le deuxième algorithme proposé est un algorithme imparfait, c'est-à-dire que pour la plupart des instances du problème, cet algorithme parvient à trouver la solution optimale, mais il existe des instances pour lesquelles il ne parvient pas à trouver une solution optimale. Le principe de cet algorithme est basé sur deux observations sur les valeurs des coalitions. L'analyse de ses performances à montré que l'algorithme n'échoue presque jamais à fournir le résultat exact et que pour certaines distributions de valeurs, le gain de temps dépasse les 300 secondes pour 25 agents. Le troisième algorithme est un algorithme conçu principalement pour des problèmes de grande taille qui nécessitent un espace mémoire énorme pour pouvoir exécuter un des algorithmes existants. Cet algorithme divise le problème qui nécessite un grand espace mémoire en deux sous-problèmes moins gourmants en espace mémoire. Les résultats ont montré que notre algorithme parvient à trouver des solutions quasi-optimales avec un gain de temps avoisinant les 100 %.

Ce mémoire est structuré comme suit : Dans le chapitre 1 nous introduisons le domaine des systèmes multi-agents. Nous présentons également le concept de formation de coalitions d'agents et ses domaines d'application. Dans le chapitre 2 nous abordons le problème de génération de structures de coalitions et exposons quelques notions que nous estimons indispensables à la compréhension des différents travaux du domaine et ceux effectués dans ce mémoire. Nous présentons ensuite un état de l'art sur certains algorithmes qui existent pour la résolution du problème de génération de structure de coalitions. Une comparaison des différentes approches est ensuite présentée pour mettre en évidence leurs points forts et points faibles en termes de temps d'exécution mais aussi de qualité des solutions produites. Dans le chapitre 3, nous présentons notre nouvelle représentation de l'espace de recherche. Les chapitres 4, 5 et 6 sont consacrés à présenter nos trois contributions pour la génération de structures de coalitions. Chacun de ces trois chapitres présente les détails de chaque algorithme ainsi que les résultats obtenus suite à l'évaluation des performances de l'approche en question. Enfin, nous clôturons ce mémoire par une conclusion générale, où nous concluons sur les travaux que nous avons mené et présentons des perspectives d'études.

Chapitre 1

Agents, systèmes multi-agents et formation de coalitions d'agents

1.1 Introduction

Les Systèmes Multi-Agents (SMA) ont suscité un intérêt considérable dans les travaux de recherche récents en tant qu'outil permettant de résoudre des problèmes complexes en les décomposant en tâches plus petites. Dans ce chapitre, nous présenterons quelques définitions et généralités sur les systèmes multi-agents. Nous introduisons ensuite le problème de la formation de coalitions d'agents. Puis, nous nous intéresserons au lien avec la théorie des jeux. Enfin, nous passerons en revue quelques domaines d'application de la formation de coalitions d'agents.

1.2 Définitions

L'entité centrale des systèmes multi-agents et de la formation de coalitions d'agents est l'agent. Le terme "agent" est utilisé dans plusieurs domaines mais n'admet pas de définition acceptée en unanimité. Cependant, nous pouvons le définir dans notre cas comme suit : [10]

Définition 1.2.1 *Un agent est une entité autonome, évoluant dans un environnement, qui est capable d'agir à l'aide d'actionneurs, de percevoir partiellement à l'aide de capteurs et de communiquer avec d'autres agents au sein de cet environnement, et dont le comportement vise à satisfaire ses besoins et ses objectifs à partir de ses observations, de ses connaissances, de son historique d'actions et de ses interactions avec les autres agents.*

La définition ci-dessus comprend trois mots-clés importants :

1. **Entité :** L'entité désigne le type de l'agent. Un agent peut être une entité physique : un robot, une voiture, un avion, ou une entité virtuelle : un composant logiciel, un module informatique.

2. **Environnement :** L'environnement désigne le lieu où se trouve l'agent, et se caractérise par tout ce qui n'est pas lui. Par exemple, pour un agent taxi, les routes, les autres automobiles, les piétons, les clients, etc., peuvent former l'environnement de l'agent. Un agent utilise les informations captées sur l'environnement pour prendre des décisions.

3. **Action :** Un agent peut effectuer des actions pour modifier l'environnement. Par exemple, lorsqu'un agent taxi accélère, la position du taxi change.

Un agent peut être caractérisé par son comportement (ses actions) qui est analysable sans connaître les détails d'implémentations. Les propriétés suivantes caractérisent les agents et leurs permettent de résoudre des tâches complexes :

- **Réactivité :** un agent peut percevoir l'environnement et répondre en temps réel aux changements, en d'autres termes, un agent est capable d'agir, de réagir et de modifier l'environnement.

- **Proactivité :** un agent est dirigé pas ses buts, il est donc doté d'un comportement orienté but lui permettant de prendre l'initiative. Cette capacité implique qu'un même agent peut prendre des actions différentes lorsqu'il est placé dans des environnements différents.

- **Sociabilité :** un agent est capable d'interagir avec d'autres agents ou utilisateurs. Un agent n'est donc pas isolé mais fait plutôt partie d'une société d'agents qui interagissent pour partager leurs connaissances ainsi que pour demander des informations afin d'améliorer leurs performances dans la réalisation de leurs objectifs.

- **Autonomie :** un agent est capable d'agir sans l'intervention directe d'un autre agent ou d'un utilisateur et de contrôler ses propres actions ainsi que son état interne, et de ce fait, un agent peut refuser des requêtes de ces derniers.

En raison de sa vision partielle de son environnement, un agent ne peut exister indépendamment de l'environnement ni des autres agents. Cet ensemble ainsi que les interactions entre les agents forme un système multi-agents [10].

Définition 1.2.2 *Un système multi-agents est un système composé d'un ensemble d'agents autonomes qui interagissent, communiquent et coopèrent pour résoudre collectivement un problème ou élaborer une expertise sans l'intervention d'un externe au système afin d'atteindre leurs objectifs [10].*

Les systèmes multi-agents permettent ainsi d'exploiter le véritable avantage des agents lorsqu'ils travaillent en collaboration et non en agissant seuls sur la base de l'autonomie.

1.3 Formation de coalitions d'agents

La notion de coalition d'individus a été étudiée par la communauté de la théorie des jeux depuis des décennies, et s'est révélée être une stratégie utile dans les systèmes multi-agents. La formation de coalitions d'agents est utilisée comme un moyen de coordination des agents. Elle leur permet de se regrouper temporairement pour atteindre des objectifs qui dépassent les capacités individuelles des agents. Les coalitions ainsi formées sont dissoutes aussitôt que l'objectif est atteint ou que ce dernier devient irréalisable.

Si nous considérons la population des agents A comme un ensemble, alors chaque sous-ensemble de A est une coalition possible. De manière générale, le processus de formation de coalitions peut être divisé en 5 activités [3] :

1. **Le calcul des valeurs des coalitions :** Cette activité correspond au calcul de la valeur correspondant à chaque coalition possible.

2. **La recherche de la structure de coalitions optimale :** Cette activité vise à former des coalitions entre les agents. L'ensemble de coalitions qui en résulte est appelé, structure de coalitions. La recherche consiste ici à trouver la structure de coalitions qui maximise la valeur de chaque coalition.

3. **L'étude des dépendances aux coalitions externes :** Cette étude est faite dans le cas où chaque coalition possède une influence sur les valeurs des autres coalitions.

4. **La formation de coalitions orientée tâches :** Dans cette activité, la formation de coalitions est dictée par le type et l'urgence des tâches à réaliser.

5. **La répartition des gains entre les membres des coalitions :** Dans cette activité, les bénéfices sont répartis entre les membres des coalitions sur la base de certains critères souhaités et préétablis.

Dans ce document, nous nous intéressons à la deuxième activité qui consiste à diviser un ensemble d'agents en sous-ensembles disjoints afin de maximiser le gain, c'est-à-dire, à générer la

structure de coalitions optimale.

La complexité du processus de formation de coalitions dépend des conditions dans lesquelles les coalitions existent et des types de coalitions qui sont autorisés. Par exemple, si la répartition des agents n'est pas disjointe, c'est-à-dire que les agents peuvent appartenir simultanément à plusieurs coalitions, cela complexifie encore plus le problème [11].

1.4 Domaines d'applications de la formation de coalitions d'agents

Un objectif clair des systèmes multi-agents est de construire des agents qui peuvent soit prendre des actions communes et coordonnées pour améliorer leurs performances et compenser les faiblesses de chaque agent, soit atteindre des objectifs qui dépassent les capacités individuelles des agents. C'est pourquoi, la formation de coalitions a suscité une attention considérable dans les travaux de recherche récents, et son utilité s'est révélée être très significative pour un certain nombre de domaines d'application.

Dans les réseaux de capteurs distribués, des coalitions de capteurs sont utilisées pour fournir une meilleure couverture sur une large zone afin de suivre des cibles (d'intérêt) [7].

Dans le commerce électronique, les acheteurs peuvent former des coalitions pour acheter des produits et obtenir des réductions de prix [27].

Dans les systèmes de livraison, des coalitions de sociétés de livraison peuvent être formées pour réduire les coûts de transport en partageant les livraisons [24].

La formation de coalitions peut également être utilisée pour la collecte d'informations, où plusieurs serveurs peuvent former des coalitions pour répondre aux requêtes des utilisateurs [14].

1.5 Conclusion

Dans ce chapitre, nous avons abordé des concepts de base des systèmes multi-agents. Nous avons d'abord proposé une définition des agents et des systèmes multi-agents et décrit leurs caractéristiques clés. Nous avons ensuite décrit le concept de formation de coalitions d'agents tout en abordant certaines de ses applications. Le chapitre suivant sera consacré à un état de l'art sur les algorithmes de génération de structures de coalitions.

Chapitre 2

État de l'art sur les algorithmes de génération de structures de coalitions

2.1 Introduction

Les coalitions sont largement utilisées dans les systèmes multi-agents pour exécuter des tâches collectives. Elles sont généralement formées pour une courte période pour un ou plusieurs objectifs et sont dissoutes lorsque ces objectifs n'existent plus, ou ne peuvent plus être atteints. Ce chapitre est consacré à la présentation et l'étude critique de quelques solutions existantes pour la résolution du problème de génération de structures de coalitions. Pour cela, nous commencerons par présenter les notions clés nécessaires à la compréhension des modèles existants. Ensuite, nous présenterons certaines représentations théoriquement utilisées par les différents modèles, suivis par une classification des solutions étudiées, puis nous présenterons et discuterons chaque solution. Enfin, nous conclurons ce chapitre par une comparaison des travaux analysés.

2.2 Formulation du problème

Cette section définit le problème de génération de structures de coalitions et présente les principales définitions et notations utilisées tout au long du mémoire. Une instance du problème de génération de structures de coalitions est donné par un ensemble fini $A = \{a_1, a_2, ..., a_n\}$ de n agents où l'utilité de tout sous-ensemble non vide de A est déterminée par une fonction caractéristique v qui attribue une valeur réelle à chaque sous-ensemble non vide de A appelé coalition. Une coalition C est un sous-ensemble non vide de A, où $C \in 2^A$, ensemble des coalitions possibles dans A. Une structure de coalitions CS est une partition de l'ensemble des agents A en coalitions disjointes.

Définition 2.2.1 *Une structure de coalitions est un ensemble de coalitions* $CS = \{C_1, C_2, ..., C_k\}$, *tel que* $k = |CS|$ *et* $\forall C_i \in CS$, $C_i \in 2^A$ *et* $C_i \neq \emptyset$, *qui satisfait les contraintes suivantes :* $\bigcup_{j=1}^{k} C_i = A$ *et pour tout* $i, j \in \{1, 2, ..., k\}$ *tel que* $i \neq j$, $C_i \cap C_j = \emptyset$.

Par exemple, étant donné un ensemble de cinq agents, $A = \{a_1, a_2, a_3, a_4, a_5\}$ et une coalition $C = \{a_2, a_3, a_4\}$, une structure de coalitions possible sur A est : $\{\{a_1\}, \{a_2, a_3\}, \{a_4, a_5\}\}$, et une structure de coalitions possible sur C est : $\{\{a_2\}, \{a_3, a_4\}\}$.

Nous notons par $\Pi(A)$, l'ensemble de toutes les structures de coalitions. La valeur d'une structure de coalitions CS est évaluée comme la somme des valeurs des coalitions qui la composent : $V(CS) = \sum_{C \in CS} v(C)$.

Propriété 2.2.1 *Pour un ensemble d'agents A de taille n, le nombre de toutes les coalitions dans 2^A est $2^n - 1$ et le nombre de structures de coalitions est un n-ième nombre de Bell [4].*

Le problème de génération de structures de coalitions prend comme entrée un ensemble d'agents A et vise à trouver la structure de coalitions optimale CS^*. La solution optimale au problème de génération de structures de coalitions est la structure de coalitions qui a la valeur la plus élevée possible $CS^* \in \Pi(A)$, i.e., $CS^* = \arg\max_{C \in \Pi(A)} V(CS)$.

Il est intéressant de noter que chaque structure de coalitions en A représente une solution possible au problème de génération de structures de coalitions. Ainsi, les termes "structure de coalitions" et "solution" seront utilisés de manière interchangeable.

2.3 Représentations de l'espace de recherche

Résoudre le problème de génération de structures de coalitions de manière naïve en énumérant les différentes structures de coalitions est couteux en termes de calcul [23]. De ce fait, un certains nombre d'algorithmes de recherche sont proposés pour raccourcir les délais nécessaires pour la génération des structures de coalitions optimales et exploitent, pour cette raison, des représentations de l'espace de recherche qui le permettent.

2.3.1 Le graphe des structures de coalitions

La première représentation que nous considérons a été proposée par Sandholm et al [23], et est appelée le graphe des structures de coalitions (coalition structure graph). Elle permet de représenter l'espace de recherche sous forme d'un graphe composé de noeuds représentant les structures de coalitions. Étant donné un ensemble d'agents A de taille n, ces noeuds sont répartis en n niveaux, où chaque niveau est composé de noeuds représentant des structures de coalitions qui contiennent exactement i coalitions chacune : $i \in \{1, 2, .., n\}$. Chaque arête de ce graphe relie deux noeuds appartenant à deux niveaux consécutifs, où chaque structure de coalitions de niveau i peut être obtenue en divisant une coalition en deux à partir de la structure de coalitions de niveau $i - 1$. Le graphe des structures de coalitions de quatre agents est présenté dans la Figure 2.1.

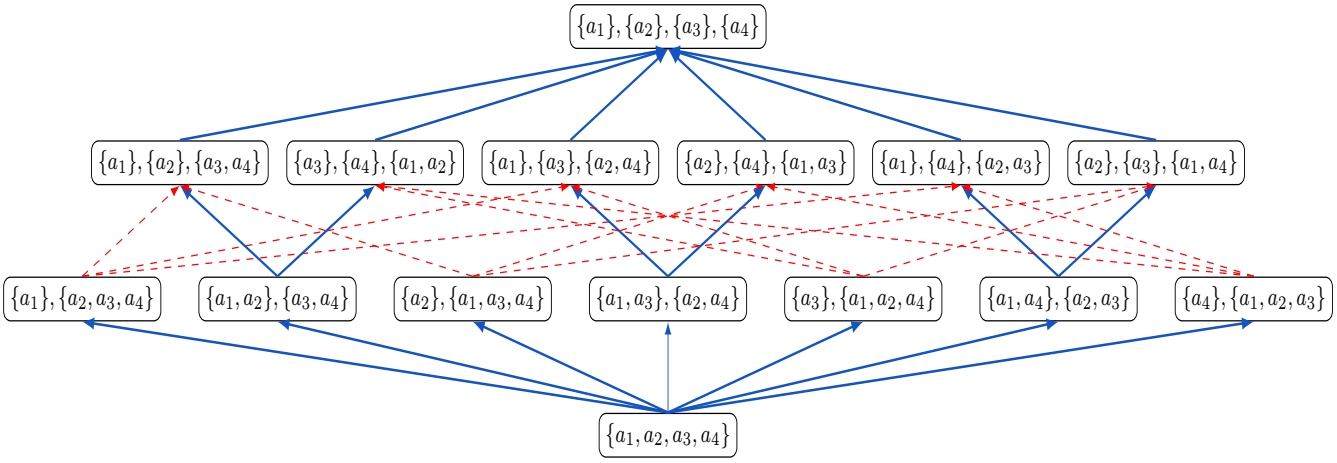


FIGURE 2.1 – Le graphe représentant les structures de coalitions de 4 agents.

2.3.2 Le graphe des partitions

Alors que la représentation des structures de coalitions sous forme d'un graphe catégorise les structures de coalitions en fonction du nombre de coalitions qu'elles contiennent, une représentation alternative a été proposée par Rahwan et al [20] pour les classer en fonction de la taille des coalitions qu'elles contiennent. Étant donné n agents, avec cette représentation, chaque partition entière (Integer-partition) de n est représentée par un noeud, où deux noeuds adjacents sont connectés si et seulement si la partition entière du niveau i peut être obtenue à partir de celle du niveau $i-1$ en divisant un seul entier. La figure 2.2 montre un exemple à quatre agents du graphe des partitions (integer partition graph). Rappelons qu'une partition entière de n est un ensemble de plusieurs entiers positifs, ou parties, dont la somme (avec multiplicités) est égale à n [2]. Par exemple, pour $n = 4$ l'ensemble des partitions est : $\{[4],[1,3],[2,2],[1,1,2],[1,1,1,1]\}$.

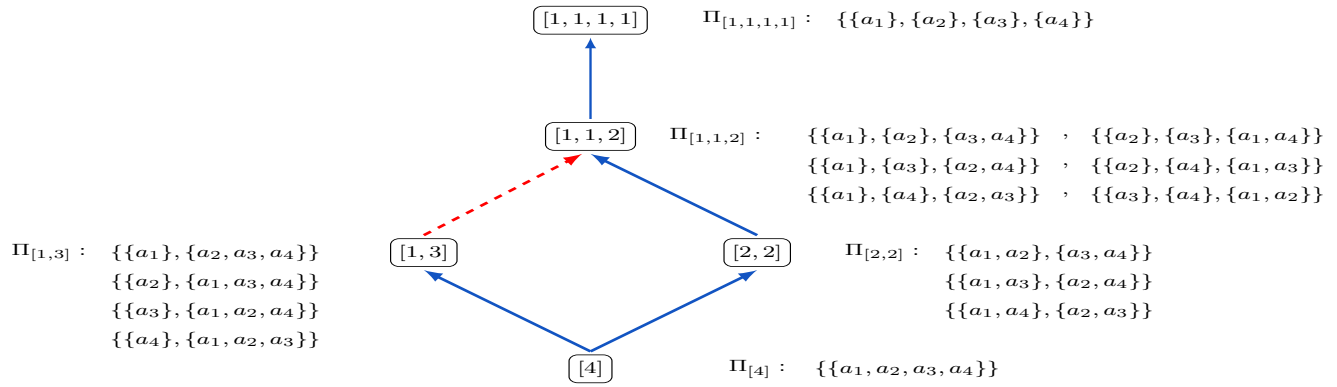


FIGURE 2.2 – Un exemple à quatre agents du graphe des partitions.

Chaque niveau $i \in \{1,2,..,n\}$ dans le graphe des partitions contient des noeuds représentant des partitions de n. Par exemple, le niveau 2 contient des noeuds où les partitions de n ont deux parties (voir la Figure. 2.2). Dans le graphe des partitions, chaque partition P représente un ensemble de

structures de coalitions dans lesquelles les tailles des coalitions correspondent aux parties de P. Par exemple, le noeud [1,1,2] est constitué de toutes les structures de coalitions qui contiennent deux coalitions de taille 1 et une coalition de taille 2. Un exemple à quatre agents est présenté dans la Figure 2.2 où toutes les structures de coalitions correspondant à chaque noeud du graphe des partitions sont représentées.

Chaque mouvement dans le graphe des structures de coalitions correspond à un mouvement dans le graphe des partitions. Par exemple, le mouvement de $\{\{a_1\}, \{a_2, a_3, a_4\}\}$ à $\{\{a_1\}, \{a_2\}, \{a_3, a_4\}\}$ dans la Figure 2.1 correspond au mouvement dans le graphe des partitions du noeud $[1, 3]$ au noeud $[1, 1, 2]$ dans la Figure 2.2.

2.4 Classification des travaux étudiés

Après avoir analysé les travaux collectés, il nous est apparu qu'une classification était nécessaire afin de répertorier les différentes approches suivies. De nombreuses méthodes pour résoudre le problème de génération de structures de coalitions ont été proposées dans la littérature et peuvent être divisées en 5 catégories :

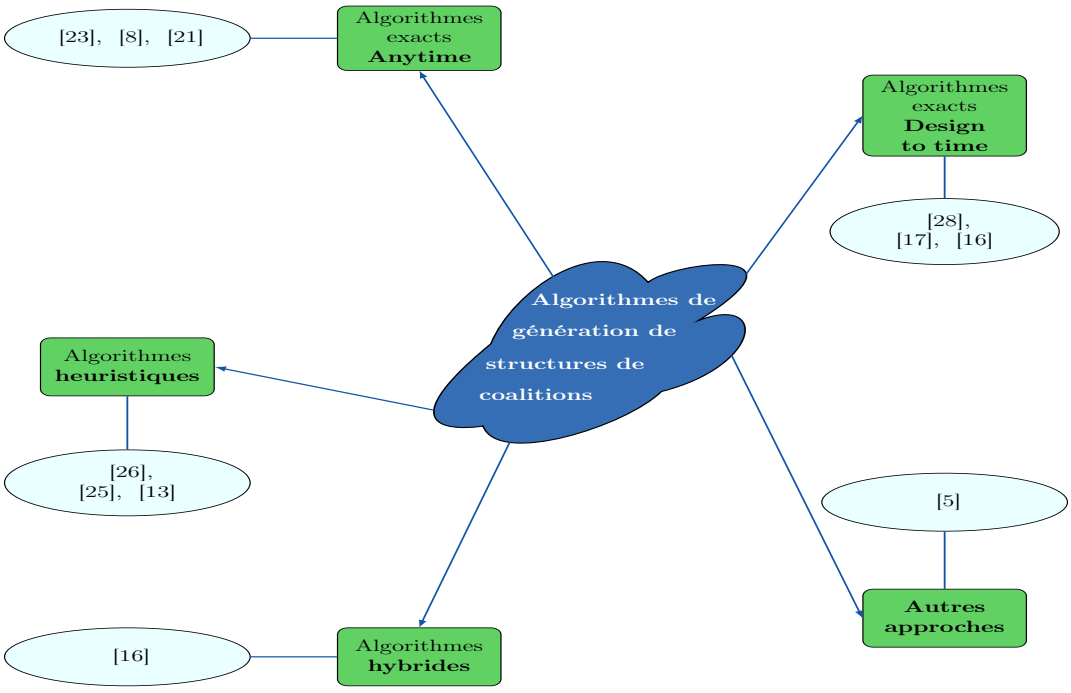


FIGURE 2.3 – Classification des travaux étudiés.

Les algorithmes exacts "**anytime**" ont la propriété anytime, c'est-à-dire, qu'ils sont capables de retourner des solutions à n'importe quel moment de l'exécution, et de ce fait, ils permettent une terminaison prématurée (avant qu'une solution optimale ne soit trouvée) tout en améliorant

la qualité de la solution générée à mesure que le temps d'exécution augmente. Cette propriété est souhaitable pour pouvoir arrêter la recherche une fois qu'une solution de qualité suffisante est trouvée. Ceci est particulièrement intéressant dans les cas où les agents pourraient ne pas être en mesure d'exécuter l'algorithme jusqu'au bout en raison de la taille exponentielle de l'espace de recherche. Cependant, leur inconvénient est qu'ils nécessitent, dans le pire des cas, un temps en $O(n^n)$ pour trouver la solution optimale.

Les algorithmes exacts "**Design to time**" garantissent de trouver une solution optimale et présentent un meilleur temps d'exécution au pire des cas que les algorithmes "anytime" (en $O(3^n)$). Cependant, ils doivent être exécutés jusqu'au bout pour le faire. C'est-à-dire que, contrairement aux algorithmes "anytime", ils sont incapables de produire des solutions intermédiaires.

Les algorithmes **heuristiques** se concentrent sur la rapidité et ne garantissent généralement pas de trouver une solution optimale. Ceci est acceptable si dans la pratique, l'algorithme est capable de retourner rapidement de bonnes solutions. De tels algorithmes, présentent la seule option pratique lorsque le nombre d'agents augmente et que le problème devient trop difficile.

2.5 Étude critique des travaux

Dans ce qui suit, nous allons présenter les travaux les plus significatifs qui ont trait à notre thème de recherche.

2.5.1 Les algorithmes exacts "Design to time"

2.5.1.1 A dynamic programming approach to the complete set partitioning problem [28]

Yeh [28] a proposé le premier algorithme fondé sur la programmation dynamique (DP : Dynamic Programming) pour résoudre le problème de partitionnement qui est proche du problème du calcul de la structure de coalitions optimale. Cet algorithme garantit de trouver une solution optimale mais doit être exécuté jusqu'au bout pour y parvenir.

Étant donné un ensemble de coalitions 2^A, soit P_t un tableau de partitions qui stocke une partition optimale de chaque coalition C dans $P_t(C)$, et V_t un tableau de valeurs qui stocke la valeur optimale de la coalition C dans $V_t(C)$. Il peut y avoir plus d'une partition optimale d'une coalition C. Pour remplir ces tableaux, l'algorithme DP procède en évaluant toutes les manières possibles de diviser en deux chaque coalition C de taille 2 jusqu'à la taille n et vérifie s'il est bénéfique ou non de diviser C. Par exemple, la Figure 2.4 montre un exemple à quatre agents et la façon dont

DP procède.

Une fois que DP évalue toutes les coalitions possibles en remplissant les tableaux P_t et V_t, la structure de coalitions optimale CS^* est calculée de manière récursive. Dans notre exemple (voir la Figure 2.4), DP commence avec $CS^* = \{a_1, a_2, a_3, a_4\}$, la grande coalition, ensuite, DP trouve que la structure de coalitions qui a la plus grande valeur est $\{\{a_1, a_3\}, \{a_2, a_4\}\}$, ensuite, il constate qu'il est bénéfique de diviser la coalition $\{a_1, a_3\}$ en coalitions $\{a_1\}$ et $\{a_3\}$. De même, il juge qu'il est avantageux de conserver la coalition $\{a_2, a_4\}$ dans sa forme initiale. Enfin, DP conclu que $\{\{a_1\}, \{a_3\}, \{a_2, a_4\}\}$ est la solution optimale avec une valeur de 160.

Taille	C	$v(C)$	Division	$P_t(C)$	$V_t(C)$
1	$\{a_1\}$	37	$v[\{a_1\}] = \mathbf{37}$	$\{a_1\}$	**37**
	$\{a_2\}$	25	$v[\{a_2\}] = \mathbf{25}$	$\{a_2\}$	**25**
	$\{a_3\}$	41	$v[\{a_3\}] = \mathbf{41}$	$\{a_3\}$	**41**
	$\{a_4\}$	39	$v[\{a_4\}] = \mathbf{39}$	$\{a_4\}$	**39**
2	$\{a_1, a_2\}$	61	$v[\{a_1, a_2\}] = 61$, $V_t[\{a_1\}] + V_t[\{a_2\}] = \mathbf{62}$	$\{a_1\}, \{a_2\}$	**62**
	$\{a_1, a_3\}$	63	$v[\{a_1, a_3\}] = 63$, $V_t[\{a_1\}] + V_t[\{a_3\}] = \mathbf{78}$	$\{a1\}, \{a_3\}$	**78**
	$\{a_1, a_4\}$	76	$v[\{a_1, a_4\}] = \mathbf{76}$, $V_t[\{a_1\}] + V_t[\{a_4\}] = 76$	$\{a_1, a_4\}$	**76**
	$\{a_2, a_3\}$	54	$v[\{a_2, a_3\}] = 54$, $V_t[\{a_2\}] + V_t[\{a_3\}] = \mathbf{66}$	$\{a_2\}, \{a_3\}$	**66**
	$\{a_2, a_4\}$	82	$v[\{a_2, a_4\}] = \mathbf{82}$, $V_t[\{a_2\}] + V_t[\{a_4\}] = 64$	$\{a_2, a_4\}$	**82**
	$\{a_3, a_4\}$	68	$v[\{a_3, a_4\}] = 68$, $V_t[\{a_3\}] + V_t[\{a_4\}] = \mathbf{80}$	$\{a_3\}, \{a_4\}$	**80**
3	$\{a_1, a_2, a_3\}$	89	$v[\{a_1, a_2, a_3\}] = 89$, $V_t[\{a_1\}] + V_t[\{a_2, a_3\}] = 103$ $V_t[\{2\}] + V_t[\{1, 3\}] = 103$, $V_t[\{3\}] + V_t[\{1, 2\}] = \mathbf{103}$	$\{a_3\}, \{a_1, a_2\}$	**103**
	$\{a_1, a_2, a_4\}$	117	$v[\{a_1, a_2, a_4\}] = 117$, $V_t[\{a_1\}] + V_t[\{a_2, a_4\}] = \mathbf{119}$ $V_t[\{a_2\}] + V_t[\{a_1, a_4\}] = 101$, $V_t[\{a_4\}] + V_t[\{a_1, a_2\}] = 101$	$\{a_1\}, \{a_2, a_4\}$	**119**
	$\{a_1, a_3, a_4\}$	99	$v[\{a_1, a_3, a_4\}] = 99$, $V_t[\{a_1\}] + V_t[\{a_3, a_4\}] = 117$ $V_t[\{a_3\}] + V_t[\{a_1, a_4\}] = \mathbf{117}$, $V_t[\{a_4\}] + V_t[\{a_1, a_3\}] = 117$	$\{a_3\}, \{a_1, a_4\}$	**117**
	$\{a_2, a_3, a_4\}$	114	$v[\{a_2, a_3, a_4\}] = 114$, $V_t[\{a_2\}] + V_t[\{a_3, a_4\}] = 105$ $V_t[\{a_3\}] + V_t[\{a_2, a_4\}] = \mathbf{123}$, $V_t[\{a_4\}] + V_t[\{a_2, a_3\}] = 105$	$\{a_3\}, \{a_2, a_4\}$	**123**
4	$\{a_1, a_2, a_3, a_4\}$	141	$v[\{a_1, a_2, a_3, a_4\}] = 141$, $V_t[\{a_1\}] + V_t[\{a_2, a_3, a_4\}] = 160$ $V_t[\{a_2\}] + V_t[\{a_1, a_3, a_4\}] = 142$, $V_t[\{a_3\}] + V_t[\{a_1, a_2, a_4\}] = 160$ $V_t[\{a_4\}] + V_t[\{a_1, a_2, a_3\}] = 142$, $V_t[\{a_1, a_2\}] + V_t[\{a_3, a_4\}] = 142$ $V_t[\{a_1, a_3\}] + V_t[\{a_2, a_4\}] = \mathbf{160}$, $V_t[\{a_1, a_4\}] + V_t[\{a_2, a_3\}] = 142$	$\{a_1, a_3\}, \{a_2, a_4\}$	**160**

FIGURE 2.4 – Principe de fonctionnement de l'algorithme DP calculant $P_t(C)$, $V_t(C)$ pour chaque coalition $C \subseteq A$.

L'algorithme DP a été redécouvert par Rothkopf et al [22] qui l'ont utilisé pour résoudre le problème d'enchères combinatoires.

Discussion Le fonctionnement de DP peut être visualisé sur le graphe des structures de coalitions (voir la Figure 2.1). En particulier, DP procède réellement comme suit : il évalue chaque mouvement vers le haut dans le graphe, il stocke les meilleurs mouvements dans le tableau P_t, et enfin il se déplace vers le haut dans le graphe. Il est intéressant de noter aussi que chaque mouvement vers le haut dans le graphe correspond à la division d'une coalition en deux. Cependant, DP explore toutes les possibilités de division et de ce fait, il explore toutes les arêtes du graphe des structures de coalitions y compris les arêtes rouges de ce graphe (voir la Figure 2.1), ainsi, DP évalue certaines opérations de division inutiles.

2.5.1.2 An improved dynamic programming algorithm for coalition structure generation [17]

Rahwan et al [17] ont proposé une version améliorée de DP, à savoir IDP (Improved Dynamic Programming) en exploitant le fait que DP explore des opérations de division inutiles. L'idée principale de l'algorithme IDP est d'éviter l'évaluation de certaines opérations de fractionnement, sans perdre les garanties de trouver la structure de coalitions optimale. Les auteurs dans [17] ont prouvé que, pour n agents, l'algorithme IDP n'évalue aucune des manières possibles de diviser une coalition de taille t, où $t \in \{\left\lfloor \frac{2n}{3} \right\rfloor + 1, ..., n-1\}$, et ceci sans perdre les garanties de trouver la structure de coalitions optimale.

Dans l'exemple de la Figure 2.4, IDP n'évalue aucune coalition de taille 3 car $3 \nleq \left\lfloor \frac{2*4}{3} \right\rfloor = 2$. De ce fait, IDP évalue 12 fractionnements de moins que DP.

L'idée de IDP consiste à supprimer toutes les arêtes rouges dans le graphe des structures de coalitions (voir la Figure 2.1), sans perdre les garanties d'atteindre n'importe quel noeud du graphe à partir du noeud inférieur. La façon dont IDP recherche tous les noeuds dans le graphe des structures de coalitions peut être visualisée par les mouvements dans le graphe des partitions présenté dans la Figure 2.2. Ainsi, la suppression des arêtes en pointillés de la figure 2.1 correspond à la suppression de l'arête en pointillés dans le graphe des partitions de la Figure 2.2. Éviter les arêtes rouges dans ce graphe correspond à éviter toutes les arêtes rouges dans la Figure 2.1.

2.5.1.3 An optimal dynamic programming algorithm [16]

Michalak et al [16] ont développé une version de DP qui évite l'évaluation de nombreux mouvements, sans perdre la garantie d'avoir la solution optimale. Pour comprendre l'idée de cet algorithme, nous avons besoin d'établir un lien entre le fonctionnement de DP et le graphe des structures de coalitions (voir la Figure. 2.1). L'algorithme DP évalue tous les mouvements suivant les arcs du graphe et enregistre les mouvements les plus bénéfiques dans le tableau P_t. Ensuite, en partant de la structure de coalitions contenant tous les agents (le noeud le plus inférieur), DP

effectue une série de mouvements jusqu'à ce qu'il atteigne un noeud optimal. Par exemple, la façon dont DP atteint le noeud $\{\{a_1\},\{a_3\},\{a_2,a_4\}\}$ dans la Figure 2.1 peut être visualisée sous forme de mouvements dans la Figure. 2.1, où le premier mouvement consiste à partitionner $\{a_1,a_2,a_3,a_4\}$ en $\{a_1,a_3\}$ et $\{a_2,a_4\}$, et le deuxième mouvement consiste à diviser $\{a_1,a_3\}$ en $\{a_1\}$ et $\{a_3\}$. Nous représentons l'ensemble des mouvements possibles évalués par DP dans le graphe par M, où un mouvement possible $m \in M$ qui correspond à la division de la coalition $C = C_1 \cup C_2$ en coalitions disjointes C_1 et C_2 est désigné par m^{C_1,C_2}.

Cette visualisation suggère que certains mouvements ne sont pas nécessaires pour atteindre ce noeud optimal. En effet, pour chaque structure de coalitions CS où $|CS| > 2$, il existe plusieurs chemins qui partent du noeud inférieur et qui mènent au noeud contenant CS. De plus, Michalak et al [16] ont prouvé que tant qu'il existe encore un chemin de mouvements évalués, menant du noeud inférieur à une solution optimale, DP pourra encore atteindre cette solution. Soit M^* un sous ensemble de mouvements défini comme suit :

$$M^* = \{m^{C_1,C_2} \in M, \text{ where } (C_1 \cup C_2 = A) \text{ ou } (C_1 < C_2 < A \setminus (C_1 \cup C_2))\}.$$

L'algorithme proposé par Michalak et al. n'évalue que ce sous ensemble de mouvements M^* qui représente un tiers des mouvements évalués par DP, mais garantie toutefois d'obtenir la solution optimale.

Michalak et al [16] ont prouvé que partant du noeud inférieur du graphe des structures de coalitions, il est possible d'atteindre tous les noeuds du graphe en évaluant uniquement les mouvements dans M^*. Ils ont également prouvé qu'il est impossible d'évaluer moins de mouvements que $|M^*|$ sans perdre la garantie de trouver la solution optimale, c'est-à-dire que M^* est optimal en termes de nombre de mouvements évalués. Ainsi, l'algorithme proposé est appelé ODP pour Optimal DP.

Un autre avantage de ODP est qu'il n'utilise pas le tableau P_t pour stocker les partitions optimales de chaque coalition, celles-ci sont calculées à la fin, il utilise uniquement le tableau V_t pour stocker la fonction caractéristique, ce qui permet de réduire les besoins en mémoire de 50% par rapport à DP.

2.5.2 Les algorithmes exacts "Anytime"

Dans le cas où l'espace de recherche est trop grand pour être entièrement exploré, un certain nombre d'algorithmes, dits "anytime", ne recherchent qu'à travers un sous-ensemble de cet espace et garantissent de trouver une solution qui se situe à une certaine marge β de la solution optimale. Ces algorithmes procèdent généralement comme suit : ils divisent l'espace de recherche en sous-espaces disjoints et exhaustifs, puis, ils identifient une séquence dans laquelle ces sous-ensembles

sont recherchés, de sorte à garantir que la qualité de la solution s'améliore après chaque sous-espace.

2.5.2.1 Coalition structure generation with worst case guarantees [23]

Sandholm et al [23] ont proposé le premier algorithme "anytime" basé sur le graphe des structures de coalitions où ils ont démontré que la recherche des deux niveaux les plus bas du graphe des structures de coalitions garantit une certaine qualité de la solution trouvée. Il est intéressant de noter que la totalité des coalitions apparaissent dans ces deux niveaux 1 et 2. Les deux premiers niveaux nécessitent le calcul de la valeur de $2^n - 1$ structures de coalitions. Ensuite, tant qu'il reste du temps de calcul, l'algorithme poursuit la recherche sur les niveaux restants et commence par le haut du graphe en faisant un parcours en largeur.

2.5.2.2 Generating coalition structures with finite bound from the optimal guarantees [8]

Dang et Jennings [8] ont proposé une amélioration qui utilise le même graphe avec une méthode différente de parcours. Tout comme l'algorithme proposé par Sandholm et al [23], il commence par les deux niveaux inférieurs, ensuite, il ne fait pas des parcours par niveau (comme le fait l'algorithme de Sandholm et al [23]), mais il recherche plutôt des sous-ensembles de structures de coalitions dépendant à la fois du nombre de coalitions ainsi que du nombre d'agents par coalition (il commence par les structures de coalitions qui ont au moins une coalition dont la taille n'est pas inférieure à un certain seuil, qui diminue au cours de l'exécution). Cet algorithme permet ainsi d'améliorer les résultats de manière significative. En effet, cette méthode permet de diminuer rapidement la marge de la meilleure solution trouvée. Cependant, l'algorithme ne réduit pas la complexité de la recherche de la solution optimale qui nécessite toujours un parcours de la totalité des structures de coalitions possibles.

2.5.2.3 An anytime algorithm for optimal coalition structure generation [21]

Rahwan et al [21] ont proposé un algorithme (appelé IP) basé sur une un nouvelle représentation de l'espace de recherche (voir la Figure 2.2) regroupant les structures de coalitions en un ensemble de sous-espaces disjoints en fonction de la taille de toutes les coalitions qu'elles contiennent (voir la Figure 2.2 pour un exemple de 4 agents). Avec cette représentation, il est possible de calculer les bornes supérieures et inférieures de la qualité de la meilleure solution dans chaque sous-espace. En comparant les bornes des différents sous-espaces, il est possible d'identifier, et donc de se concentrer sur les sous-espaces les plus prometteurs. L'algorithme améliore progressivement les solutions en éliminant les sous-espaces qui n'ont pas de borne supérieure meilleure que la dernière meilleure solution trouvée. Pour chaque sous-espace de ce type, l'algorithme construit plusieurs arbres de recherche dont les noeuds représentent des coalitions (la taille de ces coalitions correspond aux

parties du sous espace) et chaque chemin de la racine à une feuille représente une structure de coalitions qui peut ou non être valide. Ensuite pour accélérer la recherche, IP applique une technique de "branch-and bound" pour identifier et éviter les branches qui ne présentent pas le potentiel de contenir une solution optimale. La Figure 2.5 montre une illustration de la manière dont IP procède sur un exemple à 8 agents et un sous-espace à trois parties $[1, 3, 4]$.

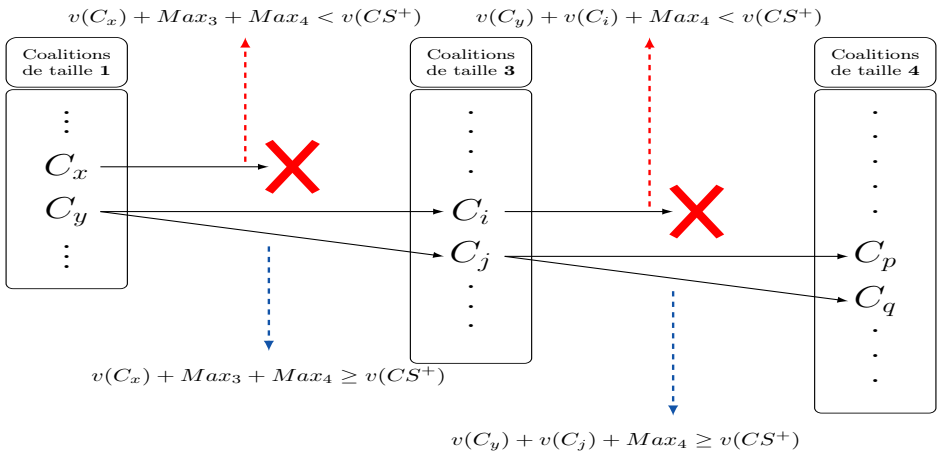


FIGURE 2.5 – Une illustration de la technique de "branch-and bound" lors de la recherche de $\Pi_{[1,3,4]}$. Dans cet exemple, l'algorithme reconnaît que les structures de coalitions contenant la coalition C_x ou les deux coalitions C_y et C_i, ne peuvent pas être optimales. Ainsi IP ne poursuit pas la recherche plus loin dans l'arbre. Ici, Max_i correspond à la valeur maximale qu'une coalition de taille i peut prendre et CS^+ correspond à la dernière meilleure solution trouvée.

L'algorithme IP a une complexité de $O(n^n)$, et dans le pire des cas, il peut finir par construire toutes les structures de coalitions possibles.

2.5.3 Algorithmes hybrides

2.5.3.1 A hybrid exact algorithm for complete set partitioning [16]

Michalak et al [16] ont proposé un algorithme appelé ODP-IP, qui combine la technique de programmation dynamique et l'approche "anytime". Comme son nom l'indique, l'algorithme combine ODP [16] et IP [21]. La relation entre ces deux algorithmes peut être visualisée sur le graphe des partitions (voir la section 2.3). L'algorithme IP est principalement basé sur cette représentation, et les différentes opérations de ODP (qui peuvent être visualisées sur le graphe des structures de coalitions) peuvent être visualisées sur le graphe des partitions (le lien entre les deux représentations est présentée dans la section 2.3). Cependant, comme expliqué précédemment, pour chaque taille de coalitions t, ODP évite d'évaluer seulement quelques coalitions de cette taille t, ceci présente un problème pour sa compatibilité avec IP qui nécessite d'évaluer tous les mouvements possibles. De ce fait, Michalak et al ont plutôt utilisé l'algorithme IDP qui, contrairement à ODP, évalue soit

tous les mouvements dans lesquels une coalition de taille t est divisée en deux coalitions, soit aucun. IDP est alors exécuté en parallèle avec IP, ce qui permet d'utiliser les résultats intermédiaires de IDP pour accélérer la technique de branch and bound de IP et de rechercher simultanément dans plusieurs sous-espaces.

L'algorithme ODP-IP qui en résulte est l'algorithme exact "anytime" le plus rapide à ce jour. De plus, ODP-IP possède les qualités des deux algorithmes qui le constituent et évite leurs faiblesses, c'est-à-dire qu'il est "anytime" et qu'il s'exécute en $O(3^n)$, de ce fait, il est plus rapide que les deux algorithmes.

2.5.4 Algorithmes heuristiques

Dans tous les algorithmes présentés jusqu'à présent, la priorité était de trouver une solution optimale, ou une solution qui se situe à une marge connue de l'optimum. Cependant, à mesure que le nombre d'agents augmente, le problème devient trop difficile, et la seule option pratique est d'utiliser des algorithmes heuristiques. De tels algorithmes ne garantissent pas qu'une solution optimale soit trouvée, et ne donnent aucune garantie sur la qualité de leurs solutions. Cependant, ils peuvent généralement être appliqués avec des problèmes de grande taille. Nous décrivons dans ce qui suit un certain nombre de ces algorithmes.

2.5.4.1 Methods for task allocation via agent coalition formation [26]

Shehotry et Kraus [26] ont présenté l'un des premiers algorithmes heuristiques du problème de génération de structures de coalitions pour la répartition des tâches, l'algorithme limite la taille des coalitions évaluées à un certain nombre d'agents. Il renvoie une structure de coalitions CS qui est construite itérativement à la manière d'un algorithme glouton. À chaque itération, la meilleure des coalitions candidates est ajoutée à CS, où une coalition candidate est une coalition qui ne chevauche aucune des coalitions qui ont été ajoutées à CS lors des itérations précédentes. La recherche de la meilleure coalition candidate se fait de manière distribuée, les agents négocient pour savoir lequel d'entre eux recherche quelles coalitions. Bien que cet algorithme soit assez simple, il a fait le premier pas pour répondre au besoin de développer des algorithmes qui peuvent produire une structure de coalitions réalisable.

2.5.4.2 Searching for optimal coalition structures [25]

Sen et Dutta [25] ont proposé un algorithme génétique qui part d'un ensemble initial de structures de coalitions, appelé population, qui est généré de manière aléatoire. Après cela, l'algorithme répète les trois étapes suivantes : évaluation, sélection et recombinaison.

D'abord, chaque membre (c'est-à-dire une structure de coalitions) de la population actuelle est évalué. Ensuite, lors de la phase de sélection, les structures de coalitions sont choisies en fonction des résultats de l'évaluation. Celles-ci forment la prochaine génération. La dernière étape est la recombinaison, dans laquelle les nouveaux membres sont construits à partir des structures de coalitions qui ont été sélectionnées en échangeant et/ou en modifiant leurs contenus.

2.5.4.3 Simulated annealing for multi-agent coalition formation [13]

Keinänen [13] a proposé un algorithme basé sur le Recuit Simulé (Simulated Annealing, une technique de recherche locale générique et stochastique) ainsi que sur la recherche à voisinage (neighbourhood search). L'algorithme commence par générer une structure de coalitions aléatoire CS et par fixer une température initiale, qui est un paramètre ($temp$) qui permet de spécifier la probabilité d'acceptation d'une solution p.

À chaque itération, l'algorithme passe de la structure de coalitions actuelle à une autre structure de coalitions dans son voisinage. Les voisinages peuvent être définis à l'aide de plusieurs critères. En d'autre termes, l'algorithme génère une structure de coalitions aléatoire CS. Puis, il génère une autre structure de coalitions aléatoire CS' dans le voisinage de CS. Si CS' est meilleure que CS, alors l'algorithme établit $CS \leftarrow CS'$. Sinon, il établit $CS \leftarrow CS'$, mais cette fois avec une probabilité $p = e^{(V(CS')-V(CS))/temp}$.

Le paramètre $temp$ diminue après chaque itération selon un programme de recuit $temp \leftarrow \alpha temp$, avec $0 < \alpha < 1$ (lorsque $temp$ diminue, $\frac{1}{temp}$ augmente).

L'algorithme boucle alors continuellement jusqu'à ce qu'une certaine condition de terminaison soit satisfaite. Cette condition est satisfaite lorsqu'une valeur de température minimale qui est utilisée comme paramètre de sortie est atteinte ou que le nombre maximal d'itérations a été atteint et qu'une solution de qualité suffisamment bonne a été trouvée.

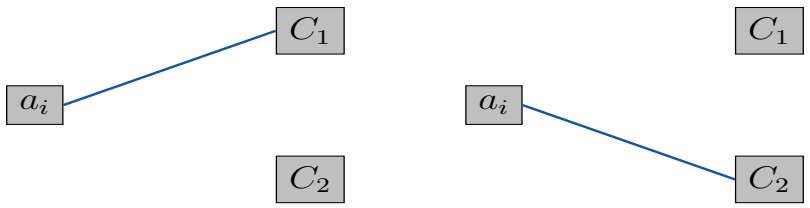
2.5.5 Autres approches pour la résolution du problème de formation de coalitions

2.5.5.1 An Imperfect Algorithm for Coalition Structure Generation [5]

Changder et al [5] ont proposé un algorithme imparfait ImDP (Imperfect Dynamic Programming), c'est-à-dire que pour certaines distributions de données, l'algorithme imparfait ne parvient pas à fournir la solution optimale. L'algorithme proposé est basé sur la programmation dynamique et utilise pour l'évaluation les deux tableaux P_t et V_t précédemment présentés pour l'algorithme DP. Cependant, contrairement à DP, ImDP ne teste pas s'il est bénéfique ou non de diviser les

coalitions de taille 2 jusqu'à la taille n, mais teste plutôt les coalitions dont la taille ne dépasse pas $\left\lceil \frac{n}{2} \right\rceil$. De plus, ImDP n'évalue pas tous les mouvements possibles pour chaque coalition, mais utilise plutôt une énumération partielle des mouvements. Nous détaillons dans ce qui suit les deux fonctions, $Merge_1$ et $Merge_2$ utilisées par ImDP pour l'énumération partielle.

La fonction $Merge_1$ est utilisée pour énumérer de façon partielle les mouvements possibles de chaque coalition de taille 2 jusqu'à la taille $\left\lceil \frac{n}{2} \right\rceil$. Pour chaque coalition $C = \{a_i \cup C \setminus a_i\}$, $Merge_1$ est appliquée pour chaque agent $a_i \in C$ pour créer 2 mouvements possibles : la coalition $\{C \setminus a_i\}$ est stockée dans le tableau des partitions P_t sous la forme de deux coalition C_1 et C_2 (i.e., dans P_t, $\{C \setminus a_i\} = C_1 \cup C_2$). Ainsi, les deux mouvements évalués pour l'agent a_i consistent à diviser la coalition $C = \{a_i \cup C_1 \cup C_2\}$ en deux coalitions $\{a_i \cup C_1\}$ et C_2 pour le premier mouvement, et, en $\{a_i \cup C_2\}$ et C_1 pour le second mouvement. La Figure 2.6 montre les détails de la fonction $Merge_1$.



(a) $\{a_i \cup C_1\}, \{C_2\}$ (b) $\{a_i \cup C_2\}, \{C_1\}$

FIGURE 2.6 – Principe de la fonction $Merge_1$. $Merge_1$ est appliquée sur la coalition $\{a_i \cup C \setminus a_i\}$, où la coalition $\{C \setminus a_i\}$ est stockée dans le tableau des partitions en $\{C_1\}, \{C_2\}$.

La fonction $Merge_2$ est utilisée entre deux coalitions de tailles $\left\lceil \frac{n}{2} \right\rceil$ et $n - \left\lceil \frac{n}{2} \right\rceil$ pour générer 4 nouvelles structures de coalitions entre chaque paire de coalitions X et Y stockées respectivement en $\{\{x_1\}, \{x_2\}\}$ et $\{\{y_1\}, \{y_2\}\}$ dans le tableau des partitions. Le détail de cette fonction est donné par la Figure 2.7.

La structure de coalitions ayant la plus grande valeur est ensuite calculée de façon récursive. À la fin du programme, ImDP ne recherche pas tous les noeuds du graphe des partitions (qui représente l'espace de recherche), ainsi, il ne parcours pas toutes les structures de coalitions et de ce fait, l'algorithme ne garantie pas de trouver la solution optimale.

La différence avec les heuristiques, réside dans le fait que pour certaines distributions de valeurs, l'algorithme ne fournit pas la solution optimale, mais pour d'autres il arrive à atteindre cette solution optimale. De plus, la probabilité de fournir une solution optimale est beaucoup plus élevée avec un algorithme imparfait qu'avec une heuristique. Les résultats obtenus montrent que le taux d'erreur sur certaines distributions n'atteint pas 5%, et dans le cas où ImDP ne trouve pas la solution optimale, la qualité de la solution produite est très proche de l'optimale.

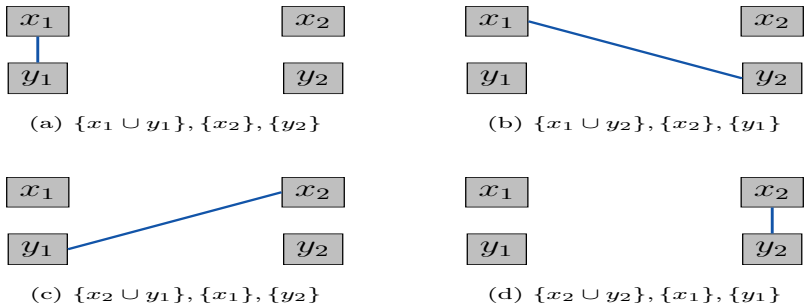


FIGURE 2.7 – Principe de la fonction $Merge_2$. $Merge_2$ opère entre deux coalitions X et Y de taille $\left\lceil \frac{n}{2} \right\rceil$ et $n - \left\lceil \frac{n}{2} \right\rceil$ enregistrées chacune en deux coalitions dans le tableau des partitions. $Merge_2$ génère 4 structures de coalitions en fusionnant 2 coalitions de X et de Y. Par exemple, dans la Figure (a), le composant $\{x_1\}$ de X est fusionné avec le composant $\{y_1\}$ de Y pour former la structure de coalitions $\{\{x_1 \cup y_1\}, \{x_2\}, \{y_2\}\}$

La complexité de l'algorithme ImDP est de $O(n2^n)$ et pour un certain nombre de distributions de valeurs, ImDP est plus rapide que ODP-IP (l'algorithme le plus rapide à ce jour).

2.6 Étude comparative

Le tableau 2.1 illustre une étude comparative que nous avons mené sur les différents algorithmes de génération de structures de coalitions.

Algorithme	Exact	Anytime	Complexité	Mémoire requise
DP [28]	Oui	Non	$O(3^n)$	Un tableau de partitions P_t de taille 2^n Un tableau de valeurs optimales V_t de taille 2^n
IDP [17]	Oui	Non	$O(3^n)$	Un tableau de partitions P_t de taille 2^n Un tableau de valeurs optimales V_t de taille 2^n
ODP [16]	Oui	Non	$O(3^n)$	Un tableau de valeurs optimales V_t de taille 2^n
IP [21]	Oui	Oui	$O(n^n)$	Un tableau de valeurs V_t de taille 2^n
ODP-IP [16]	Oui	Oui	$O(3^n)$	Un tableau de partitions P_t de taille 2^n Un tableau de valeurs optimales V_t de taille 2^n
ImDP [5]	Non	Non	$O(n2^n)$	Un tableau de partitions P_t de taille 2^n Un tableau de valeurs optimales V_t de taille 2^n

TABLE 2.1 – Comparaison des solutions analysées par rapport à : Optimalité (Exact/Approchée), Anytime (Renvoie des solutions à tout moment de l'exécution/Doit s'exécuter jusqu'au bout), Complexité, Mémoire nécessaire.

2.7 Conclusion

Ce chapitre a fourni une synthèse de l'état actuel de la recherche sur le problème de génération de structures de coalitions. Tout d'abord, nous avons présenté différentes représentations de l'espace de recherche. Puis, nous avons proposé une classification des solutions en fonction de l'approche suivie. Nous avons ensuite décrit les solutions étudiées pour chaque approche. Enfin, nous avons comparé les différents travaux analysés selon certains critères choisies.

Chapitre 3

Nouvelle représentation de l'espace de recherche

3.1 Introduction

Les deux représentations de l'espace de recherche utilisées par la plupart des algorithmes "anytime" existants sont pour la première représentation, un graphe non orienté (voir la Figure 2.1) dont les sommets représentent des structures de coalitions. Cette première représentation oblige cependant à explorer toutes les solutions possibles afin de garantir que la solution optimale soit trouvée. Pour la seconde représentation dont les sommets représentent des sous-espaces de structures de coalitions (voir la Figure 2.2), elle permet le calcul des solutions de manière anytime mais aussi de diviser l'espace de recherche en sous-espaces plus petits et indépendants. Cette représentation permet ainsi de regrouper des structures de coalitions dans le but d'éviter la recherche d'un grands nombre de sous-espaces. Cependant, jusqu'à ce jour, aucune représentation ne permet une représentation distincte de chaque structure de coalitions. Dans ce contexte, nous pensons qu'une représentation idéale de l'espace de recherche devrait permettre de générer une structure de coalitions sans passer par le parcours des coalitions qui la compose en fonction de leurs tailles. C'est avec cet objectif en tête que nous décrivons dans ce chapitre une telle représentation de l'espace de recherche (c'est-à-dire l'espace des structures de coalitions possibles) basée sur le graphe des partitions. Cette représentation permettra l'évaluation de toutes les structures de coalitions de n'importe quel noeud du graphe des partitions.

3.2 Nouvelle représentation pour la génération de structures de coalitions

Dans le chapitre précédent, nous avons montré que l'évaluation des structures de coalitions de chaque noeud du graphe des partitions garantit de trouver la solution optimale. Cependant, les algorithmes qui existent ne procèdent pas vraiment à une évaluation directe de chaque structure de coalitions sans passer par l'énumération et le parcours de chaque coalition qui la compose.

Dans ce qui suit, nous montrons qu'il est possible d'exploiter le graphe des partitions d'une nouvelle manière.

Pour mieux expliquer le principe de notre approche, considérons l'exemple suivant. Soit A un ensemble d'agents constitué de $n = 10$ agents, nous considérons le noeud $[1, 3, 6]$, et $CS = \{\{a_1\}, \{a_2, a_3, a_4\}, \{a_5, a_6, a_7, a_8, a_9, a_{10}\}\}$ une structure de coalitions de ce noeud contenant trois coalitions : $C_0 = \{a_1\}$ avec $|C_0| = 1$, $C_1 = \{a_2, a_3, a_4\}$ avec $|C_1| = 3$, et $C_2 = \{a_5, a_6, a_7, a_8, a_9, a_{10}\}$ avec $|C_2| = 6$.

A présent, nous codifions chaque coalition avec un numéro d'ordre comme suit :
$C_0 \rightarrow 0$, $C_1 \rightarrow 1$,$C_2 \rightarrow 2$. Ainsi, chaque agent d'une coalition C_i donnée aura une valeur i, qui correspond à la coalition à laquelle il appartient. De ce fait, une structure de coalitions sera codifiée par un ensemble de valeurs x_i. Par exemple, $CS = \{\{a_1\}, \{a_2, a_3, a_4\}, \{a_5, a_6, a_7, a_8, a_9, a_{10}\}\} = [x_1\ x_2\ ...\ x_{10}]$, où $x_i = 0 \Leftrightarrow a_i \in C_0$, $x_i = 1 \Leftrightarrow a_i \in C_1$ et, $x_i = 2 \Leftrightarrow a_i \in C_2$. La Figure 3.1 montre la représentation de la structure de coalitions CS en utilisant la nouvelle approche.

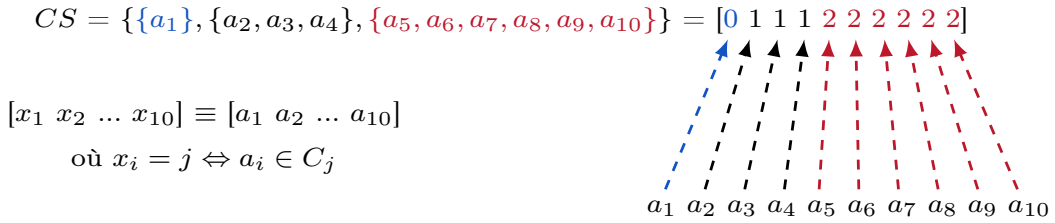


FIGURE 3.1 – Un exemple de notre représentation de l'espace de recherche étant donné le noeud [1,3,6] et la structure de coalitions $CS = \{\{a_1\}, \{a_2, a_3, a_4\}, \{a_5, a_6, a_7, a_8, a_9, a_{10}\}\}$.

Chaque nombre de la Figure 3.1 représente la coalition de chaque agent. Dans cette représentation, tous les nombres de valeur 0 représentent l'appartenance des agents correspondants à la coalition C_0, tous les nombres de valeur 1 représentent l'appartenance des agents correspondants à la coalition C_1 et tous les nombres de valeur 2 représentent l'appartenance des agents correspondants à la coalition C_2. Par exemple, l'agent a_1 appartient à la coalition C_0 et l'agent a_2 à la coalition C_1. Toute permutation de ces chiffres produit une autre structure de coalitions, comme illustré à la Figure 3.2, où un autre positionnement des chiffres conduit à une nouvelle structure de coalitions.

Avec cette deuxième combinaison, nous obtenons la valeur 0 qui représente la coalition C_0 pour l'agent a_3 (troisième chiffre), donc $a_3 \in C_0$. De même, nous obtenons la valeur 1 qui représente la coalition C_1 pour les agents a_2, a_4 et a_6, donc $a_2, a_4, a_6 \in C_1$, et la valeur 2 qui représente la

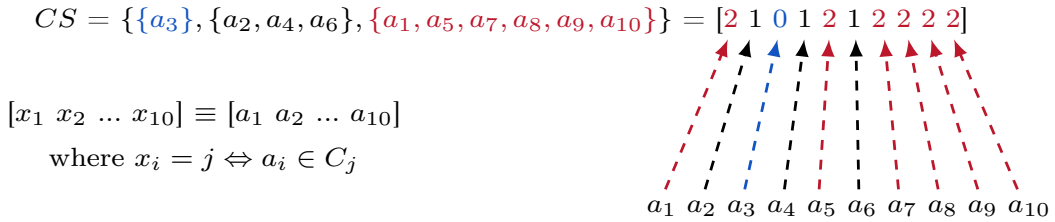


FIGURE 3.2 – Un autre exemple de notre représentation de l'espace de recherche étant donné le noeud $[1, 3, 6]$ et la structure de coalitions $CS = \{\{a_3\}, \{a_2, a_4, a_6\}, \{a_1, a_5, a_7, a_8, a_9, a_{10}\}\}$.

coalition C_2 pour les agents a_1, a_5, a_7, a_8, a_9 et a_{10}, ainsi, $a_1, a_5, a_7, a_8, a_9, a_{10} \in C_2$.

Avoir toutes les combinaisons de ces nombres garantit la génération de toutes les structures de coalitions du noeud $[1, 3, 6]$. Généralisons à présent cette représentation pour l'ensemble du graphe des partitions. Chaque noeud de ce graphe fournit le nombre de coalitions qui le composent et la taille de chacune d'entre elles. Le principe décrit pour le noeud $[1, 3, 6]$ sera appliqué à chaque noeud du graphe des partitions. Pour chaque noeud du niveau i, nous avons i coalitions, nous codifions alors chaque coalition avec un nombre, i.e., $C_j \rightarrow j$, $j \in N = \{0, 1, .., i-1\}$, ensuite, pour générer le premier positionnement des nombres, nous positionnons chaque nombre k (en partant de 0 à $i-1$) avec sa multiplicité qui est la taille de la coalition qu'il représente comme le montre la Figure 3.2. Enfin, pour générer toutes les structures de coalitions du noeud, il suffit de générer toutes les combinaisons en utilisant l'ensemble des nombres apparaissant dans le premier positionnement (voir la Figure. 3.3 pour un exemple de 4 agents).

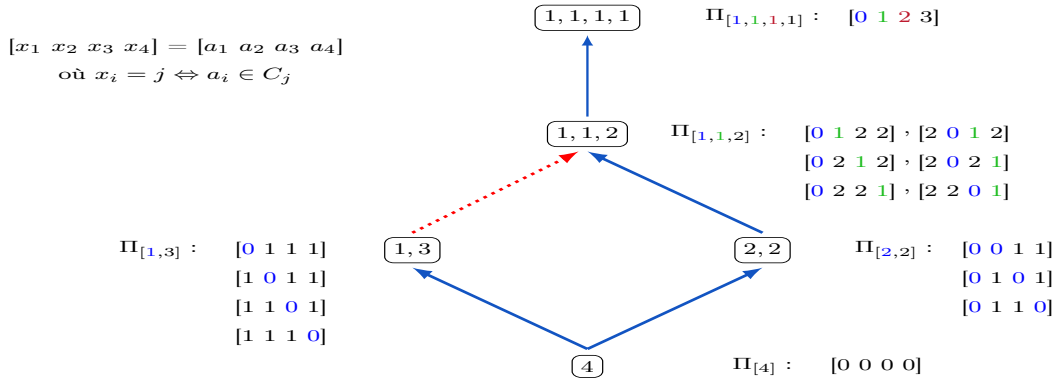


FIGURE 3.3 – Un exemple de notre représentation de l'espace de recherche pour quatre agents.

Propriété 3.2.1 *La génération de toutes les combinaisons de l'ensemble des nombres représentant un noeud garantit la génération de toutes les structures de coalitions dans le noeud spécifié.*

Théorème 3.2.1 *La génération de toutes les combinaisons de l'ensemble des nombres représentant chaque noeud du graphe des partitions garantit la génération de la structure de coalitions*

optimale.

Preuve 1 *Dans le chapitre précédent, nous avons vu que chaque noeud du graphe des partitions, représente un ensemble de structures de coalitions répondant aux critères de ce noeud. Par exemple, le noeud* $[1, 1, 2]$ *dans le graphe des partitions (voir la Figure 3.3) représente toutes les structures de coalitions contenant trois coalitions disjointes* C_1, C_2 *et* C_3 *avec* $|C_1| = 1$, $|C_2| = 1$ *et* $|C_3| = 2$. *Ainsi, toute structure de coalitions contenant trois coalitions disjointes* C_1, C_2 *et* C_3 *avec* $|C_1| = 1$, $|C_2| = 1$ *et* $|C_3| = 2$ *est associée au noeud* $[1, 1, 2]$ *dans le graphe des partitions.*

Soit à présent, un noeud P *quelconque contenant la structure de coalitions optimale* CS^*. *Il suffit de prouver que pour ce noeud, la structure de coalitions* CS^* *peut être représentée avec notre approche. Or, sur la base de la propriété précédente, il est clair que chaque structure de coalitions de* P *est représentable avec notre approche. Ainsi, la structure de coalitions optimale* CS^* *peut être représentée avec notre représentation.*

3.3 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle représentation de l'espace de recherche pour le problème de génération de structures de coalitions. Cette représentation est basée sur le graphe des partitions, où chaque coalition d'une structure de coalitions donnée se voit attribuer un numéro d'ordre. Elle permet ainsi d'évaluer directement chaque structure de coalitions sans passer par le parcours des coalitions qui la composent. De plus, en travaillant avec des structures de coalitions, un algorithme qui adopterait cette représentation produirait des solutions à n'importe quel moment de son exécution et permet ainsi une terminaison prématurée.

Chapitre 4

Un algorithme heuristique anytime basé sur une nouvelle représentation de l'espace de recherche pour la génération de structures de coalitions

4.1 Introduction

Jusqu'à ce jour, l'algorithme ODP-IP est l'algorithme exact le plus rapide pour résoudre le problème de génération de structures de coalitions, et est très efficace pour résoudre de nombreux problèmes réels lorsque le temps nécessaire pour produire la solution optimale est plus court que le temps disponible. Cependant, pour un grand nombre d'agents avec un délai strict et un temps d'exécution court, le problème devient trop difficile à résoudre par des algorithmes exacts. Dans ce chapitre, nous proposons d'abord un algorithme heuristique basé sur la représentation présentée dans le chapitre précédent. Nous évaluons ensuite les performances de notre approche, nous présentons dans un premier temps les métriques de performances ainsi que les différentes distributions de valeurs considérées pour l'évaluation des performances de notre solution, puis, dans un second temps, nous interprétons les résultats obtenus à l'issue de cette évaluation et comparons ces résultats avec plusieurs algorithmes étudiés dans le chapitre de l'état de l'art.

4.2 L'algorithme CSE

Dans cette section, nous présentons notre algorithme pour la génération de structures de coalitions appelé Coalition Structure Enumeration (CSE). Notre solution (CSE) est basée sur la nouvelle représentation présentée dans le chapitre précédent. Une fois les différentes partitions possibles de l'entier n sont générées, l'algorithme CSE est basée sur deux phases. La première phase consiste à générer pour chaque partition possible les chiffres représentatifs de la partition sur lesquels se porte la phase 2. La seconde phase consiste à générer les différentes permutations des chiffres ob-

tenues à la fin de la phase 1, ce qui permettra de générer directement les structures de coalitions correspondantes.

4.2.1 Phase de génération des chiffres

Cette phase a pour objectif de générer les chiffres représentatifs de chaque structure de coalitions d'une certaine partition. Par exemple, pour $n = 4$ agents, les partitions possibles sont : $[4], [1, 3], [2, 2], [1, 1, 2], [1, 1, 1, 1]$.

Comme le montre la figure 3.1, l'ensemble des nombres avec lesquels nous calculons les permutations contient des nombres répétés. Ainsi, l'utilisation d'une méthode traditionnelle de calcul des combinaisons n'est pas efficace en raison de la charge de calcul élevée, en particulier pour un grand nombre d'agents. Par conséquent, nous avons besoin d'un autre algorithme pour générer partiellement les permutations. Considérons l'exemple précédent du noeud $[1, 3, 6]$, pour représenter ce noeud avec la nouvelle représentation, nous utilisons les nombres 0, 1 et 2 comme indiqué dans la figure 3.1. Plutôt que de calculer toutes les combinaisons des 10 nombres, CSE calcule les combinaisons des nombres représentatifs, soit $N = \{0, 1, 2\}$. On obtient alors les combinaisons suivantes :

$C = \{\{0, 1, 2\}, \{0, 2, 1\}, \{1, 0, 2\}, \{1, 2, 0\}, \{2, 0, 1\}, \{2, 1, 0\}\}$.

Pour chaque combinaison de C, CSE calcule la première structure de coalitions de la combinaison en question appelée initialisation. Chaque chiffre de la combinaison sera alors reproduit autant de fois que la taille de la coalition qu'il représente. Par exemple, l'initialisation de $\{0, 1, 2\}$ est [0 1 1 1 2 2 2 2 2 2] (voir la Figure 3.1), l'initialisation de $\{0, 2, 1\}$ est [0 2 2 2 2 2 2 1 1 1], et, l'initialisation de $\{2, 0, 1\}$ est [2 2 2 2 2 2 0 1 1 1].

Les différentes initialisations ainsi obtenues seront utilisées dans la seconde phase pour générer les différentes permutations.

4.2.2 Phase de génération des structures de coalitions

Cette seconde phase a pour objectif de générer les permutations entre les chiffres des différentes initialisations obtenues à la fin de la phase 1. À chaque permutation correspondra une structure de coalitions.

Pour chaque initialisation, CSE calcule les permutations comme suit. Dans un premier temps, CSE commence par le premier chiffre (premier agent) et permute sa valeur avec tous les autres agents, puis, CSE passe à un autre agent jusqu'à ce qu'il atteigne le dernier. Enfin, CSE reprend le même processus avec les autres initialisations. Un exemple à quatre agents est présenté à la Figure

4.1, qui considère le noeud [1,3] du niveau 2 (voir la Figure 3.3) qui représente les structures de coalitions composées d'une coalition de taille 1 et d'une coalition de taille 3. Pour représenter ce noeud, CSE utilise $N = \{0, 1\}$ où l'initialisation de $\{0, 1\}$ est [0 1 1 1].

$$[0\ 1\ 1\ 1] \Leftrightarrow \{\{a_1\}, \{a_2, a_3, a_4\}\}$$
$$[0\ 1\ 1\ 1] : [1\ 0\ 1\ 1] \Leftrightarrow \{\{a_2\}, \{a_1, a_3, a_4\}\}$$
$$[0\ 1\ 1\ 1] : [1\ 1\ 0\ 1] \Leftrightarrow \{\{a_3\}, \{a_1, a_2, a_4\}\}$$
$$[0\ 1\ 1\ 1] : [1\ 1\ 1\ 0] \Leftrightarrow \{\{a_4\}, \{a_1, a_2, a_3\}\}$$

FIGURE 4.1 – Une illustration de la technique de l'algorithme CSE lors de la recherche du noeud [1,3] du graphe des partitions.

Le calcul de toutes les permutations pour chaque combinaison n'est toujours pas suffisamment efficace car il génère encore certaines structures de coalitions de manière répétée, en particulier pour les noeuds qui sont loin de la racine. Ainsi, pour traiter ce problème, CSE ne calcule pas toutes les permutations pour chaque initialisation, mais seulement pour certaines initialisations qui sont les plus susceptibles de générer un grand nombre de structures de coalitions différentes. La Figure 4.2 montre un exemple de génération répétée de structures de coalitions.

Dans cette figure, nous montrons que le calcul des permutations pour chaque initialisation conduit à la génération des mêmes structures de coalitions. Nous considérons le noeud [2,2] du niveau 2, qui représente les structures de coalitions avec 2 coalitions de taille 2. Pour représenter les structures de coalitions de ce noeud, CSE utilise deux initialisations, à savoir [0 0 1 1] (initialisation de $\{0, 1\}$) et [1 1 0 0] (initialisation de $\{1, 0\}$). De manière générale, dans le cas où deux coalitions ou plus ont la même taille, au moins deux combinaisons attribueraient un nombre différent à chaque coalition, mais la taille des coalitions correspondantes reste la même et de ce fait le traitement des deux combinaisons générera les mêmes structures de coalitions.

En calculant les permutations sur différentes initialisations au lieu de calculer les combinaisons d'une seule initialisation (Figure 3.1), CSE devient plus rapide mais perd la capacité de trouver la solution optimale car certaines structures de coalitions ne peuvent pas être générées avec cette approche. Par exemple, la structure de coalitions présentée à la figure 3.2 ne peut pas être générée par CSE car il n'y a pas de permutation appliquée à toute initialisation pouvant aboutir à cette structure de coalitions.

De plus, pour les noeuds situés sur les cinq premiers niveaux, CSE calcule les permutations pour toutes les combinaisons pertinentes. Cependant, pour les noeuds qui se trouvent au niveau

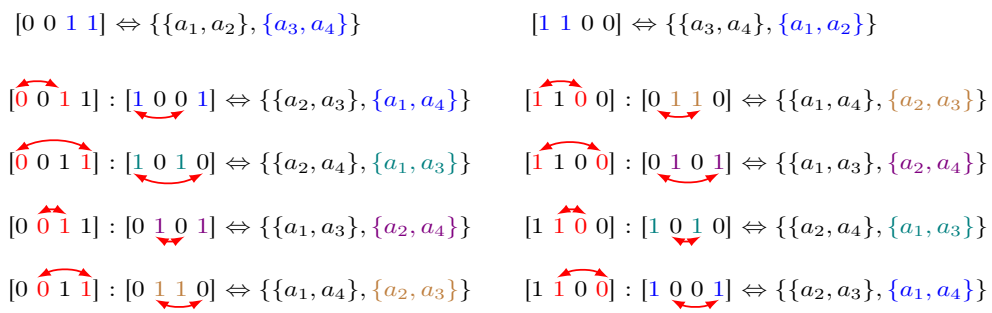


FIGURE 4.2 – Un exemple de génération répétée de structures de coalitions lors de la recherche du noeud [2,2] du graphe des partitions. CSE trouve que les deux combinaisons calculent les mêmes structures de coalitions (chaque combinaison attribue un nombre différent pour chaque coalition mais la taille des coalitions représentées reste la même). Ainsi, CSE ne calcule pas les permutations pour les deux initialisations.

i où $i \in \{6, ..., n\}$, CSE calcule les permutations pour seulement deux initialisations pertinentes. Nous avons constaté que calculer plus de permutations coutera beaucoup plus de temps, sans pour autant améliorer considérablement la qualité des solutions. Pour chaque partition (sous-espace de solutions), l'algorithme garantit que pour toute taille du problème, chaque agent appartiendra à chaque coalition de taille k au moins $2 * k$ fois.

L'algorithme 1 montre le comportement de notre heuristique, qui itère sur chaque noeud du graphe des partitions. Dans l'algorithme 1, les lignes 1 à 5, montrent une boucle qui va de 1 à n pour stocker les valeurs de chaque coalition. La structure de coalitions optimale est d'abord initialisée pour être l'ensemble des agents A lui-même, puis, la valeur de la structure de coalitions optimale s'améliore progressivement. Les lignes 9-26 décrivent la manière dont CSE calcule la structure de coalitions ayant la valeur maximale parmi celles évaluées.

4.3 Evaluation de performances

Cette section présente les différents résultats obtenus en appliquant notre technique qui a été implémentée en utilisant le langage de programmation Java et l'environnement de développement Eclipse. Toutes nos évaluations ont été effectuées sur un ordinateur personnel, le satellite c50-a559 qui a la configuration d'un processeur Intel Core i3-3110M à 2.40GHz et 4 GO de mémoire.

4.3.1 Métriques de performances

Dans cette section, nous présentons les métriques de performances que nous avons utilisés pour évaluer notre heuristique pour la génération se structures de coalitions.

Algorithm 1 L'algorithme CSE

Input Ensemble de toutes les coalitions possibles et la valeur $v(C)$ de chaque coalition $C \in 2^A$. A désigne un ensemble de n agents.
Output Une structure de coalition quasi-optimale CS^* et sa valeur.

```
for i = 1 à n do
   for C ⊆ A, où | C |= i do
      V(C) ← v(C)
   end for
end for
Générer toutes les partitions entières de n et les stocker dans IP
CS* ← (A)
V* ← v(A)
for i = 1 à n do
   Générer les nombres représentatifs N = {0, 1, .., i − 1}
   for p ∈ IP, où | p |= i do
      Sélectionner les combinaisons pertinentes de N.
      for chaque combinaison sélectionnée do
         Générer l'initialisation
         Générer les permutations
         for chaque permutation CS do
            V ← Σ_{j=0}^{i−1} V(C_j), où C_j ∈ CS
            if V > V* then
               V* ← V
               CS* ← CS
            end if
         end for
      end for
   end for
end for
Return CS*, V*
```

Les métriques de performances utilisées pour l'évaluation de notre technique sont le temps d'exécution et la qualité de solutions.

- **Temps d'exécution**
  Le temps nécessaire pour trouver une solution. En d'autres termes, le temps qu'un algorithme requiert pour trouver une solution optimale dans le cas des algorithmes exacts ou une solution approchée dans le cas des heuristiques.

- **Qualité de solutions**
  À quel point une solution heuristique est proche de l'optimum exact. Elle est calculée comme le rapport entre la valeur de la meilleure structure de coalitions CS^+ trouvée par notre

algorithme et la valeur de la solution optimale CS^*. Formellement, la qualité de solutions représente :

$$\frac{v(CS^+) * 100}{v(CS^*)}\%.$$

La qualité de solutions est utilisée par de nombreux chercheurs pour calculer la valeur de la solution trouvée par leurs algorithmes à tout moment de l'exécution. Cependant, dans certains cas, cette métrique de performances ne fournit pas réellement de bonnes informations sur la qualité d'une solution, car la qualité de la plus mauvaise solution peut également être très élevée (autour de 90% dans certains cas). C'est pourquoi, nous introduisons une nouvelle métrique pour évaluer la qualité d'une solution appelée qualité réelle de solutions, où la qualité réelle de solutions d'une structure de coalitions produite est formellement :

$$\frac{(v(CS^+) - v(CS^-)) * 100}{(v(CS^*) - v(CS^-))}\%$$

.

Avec CS^-, la plus mauvaise structure de coalitions. Ainsi, la qualité de la plus mauvaise solution est de 0 %.

4.3.2 Distributions de valeurs

Nous avons considéré les distributions suivantes [6] et pour chaque distribution, nous avons effectué une moyenne de 60 tests.

- **Agent-based Uniform :** Chaque agent a_i se voit attribuer une puissance aléatoire $p_i \sim U(0, 10)$ qui reflète sa performance moyenne sur l'ensemble des coalitions. Ensuite, pour toutes les coalitions C dans lesquelles l'agent a_i apparaît, la puissance réelle de l'agent a_i dans C est déterminée comme suit : $p_i^C \sim U(0.2 \times p_i)$, et la valeur de la coalition est calculée comme la somme des puissances de tous les membres de cette coalition, c'est-à-dire $\forall C$, $v(C) = \sum_{a_i \in C} P_i^C$ [18].

- **Agent-based Normal :** Chaque agent a_i se voit attribuer une puissance aléatoire $p_i \sim U(10, 0.01)$, ensuite, pour toutes les coalitions C dans lesquelles l'agent a_i apparaît, la puissance réelle de l'agent a_i dans C est déterminée comme suit : $p_i^C \sim U(p_i, 0.01)$, et la valeur de la coalition est calculée comme la somme des puissances de tous les membres de cette coalition, c'est-à-dire $\forall C$, $v(C) = \sum_{a_i \in C} P_i^C$ [16].

- **Normal :** Chaque valeur d'une coalition est obtenue à partir de : $v(C) \sim N(\mu, \sigma^2)$, où

$\mu = 10 \times |C|$ et $\sigma = 0.1$ [19].

- **Uniform :** Pour toutes les coalitions $C \in 2^A - 1$, $v(C) \sim U(a, b)$, où $a = 0$ et $b = |C|$ [15].

- **Modified Uniform :** La valeur de chaque coalition C est d'abord calculée de manière uniforme comme suit : $v(C) \sim U(a, b)$, où $a = 0$ et $b = 10 \times |C|$, ensuite, un nombre aléatoire r est généré $r \sim U(0, 50)$ et est ajouté à la valeur de la coalition C avec la probabilité 0.2 [1].

- **Modified Normal :** La valeur de chaque coalition C est d'abord calculée comme suit : $v(C) \sim N(a, b)$, où $a = 10 \times |C|$ et $b = 0.01$, ensuite, un nombre aléatoire r est généré $r \sim U(0, 50)$ et est ajouté à la valeur de la coalition C avec la probabilité 0.2 [18].

- **Beta :** La valeur de chaque coalition C est calculée comme suit : $v(C) \sim |C| \times Beta(\alpha, \beta)$, où $\alpha = \beta = 0.5$ [16].

- **Exponential :** La valeur de chaque coalition C est calculée comme suit : $v(C) \sim |C| \times Exp(\lambda)$, où $\lambda = 1$ [16].

- **Normally Distributed Coalition Structures (NDCS) :** La valeur de chaque coalition C est calculée comme suit : $v(C) \sim N(\mu, \sigma^2)$, où $\mu = |C|$ et $\sigma = \sqrt{|C|}$ [21].

4.3.3 Résultats

Dans cette section, nous nous somme intéressés à comparer les performances de notre approche à l'algorithme étudié dans le chapitre de l'état de l'art, ODP-IP, qui est à ce jour, l'algorithme le plus rapide pour la résolution du problème de génération de structures de coalitions. Nous considérons différentes distributions de valeurs pour évaluer pleinement notre technique et pour chaque distribution, nous générons aléatoirement plusieurs instances de problèmes en faisant varier le nombre d'agents impliqués.

Tester sur plusieurs distributions de valeurs est effectué pour obtenir des résultats différents pour l'algorithme ODP-IP. Le temps d'exécution de notre algorithme n'est pas affecté par la distribution de valeurs.

4.3.3.1 Performances en termes de temps d'exécution

Dans ce qui suit, nous discutons les performances de notre algorithme en termes de temps d'exécution et comparons les résultats par rapport aux performances de l'algorithme ODP-IP.

Pour chacune des distributions, l'évolution du temps d'exécution en fonction du nombre d'agents est illustrée sur la Figure 4.3.

En étudiant le graphique de la Figure 4.3, nous constatons que pour toutes les distributions mentionnées, notre algorithme est nettement plus rapide que ODP-IP. Par exemple, avec 25 agents, pour la distribution normale (normal distribution), le rapport entre le temps de ODP-IP et le temps de notre algorithme est de 71,9.

Le temps exact de notre algorithme est indiqué à la figure 4.3 pour différents nombres d'agents. Cela montre que l'utilisation de la nouvelle représentation de l'espace de recherche avec l'algorithme heuristique fournit des résultats très intéressants pour le problème de génération de structures de coalitions. De plus, pour 25 agents, le gain en temps sur les distibutions Normal, Uniform, Agent-based Uniform, Agent-based Normal, Modified Normal et Beta est respectivement, 76, 27, 458, 477, 3,5 et 28 secondes (voir le tableau 4.1).

Distribution	**ODP-IP** (t_1)	**CSE** (t_2)	**Différence** $t_1 - t_2$
Agent-based normal	478.492	1.017	477.475
Agent-based uniform	459.302	1.017	458.285
Uniform	28.465	1.017	27.448
Normal	77.238	1.017	76.221
Beta	29.113	1.017	28.096
NDCS	87.066	1.017	86.49
Modified normal	4.608	1.017	3.591
Modified uniform	69.704	1.017	68.687
Exponential	79.365	1.017	78.348

TABLE 4.1 – Différence de temps entre ODP-IP et CSE en secondes pour 25 agents.

La Figure 4.4 montre le temps d'exécution des algorithmes étudiés dans le chapitre de l'état de l'art sur la distribution Agent-based Normal. Cela montre bien que ODP-IP est l'algorithme le plus rapide parmi les différents algorithmes étudiés, et notre solution arrive à faire beaucoup mieux.

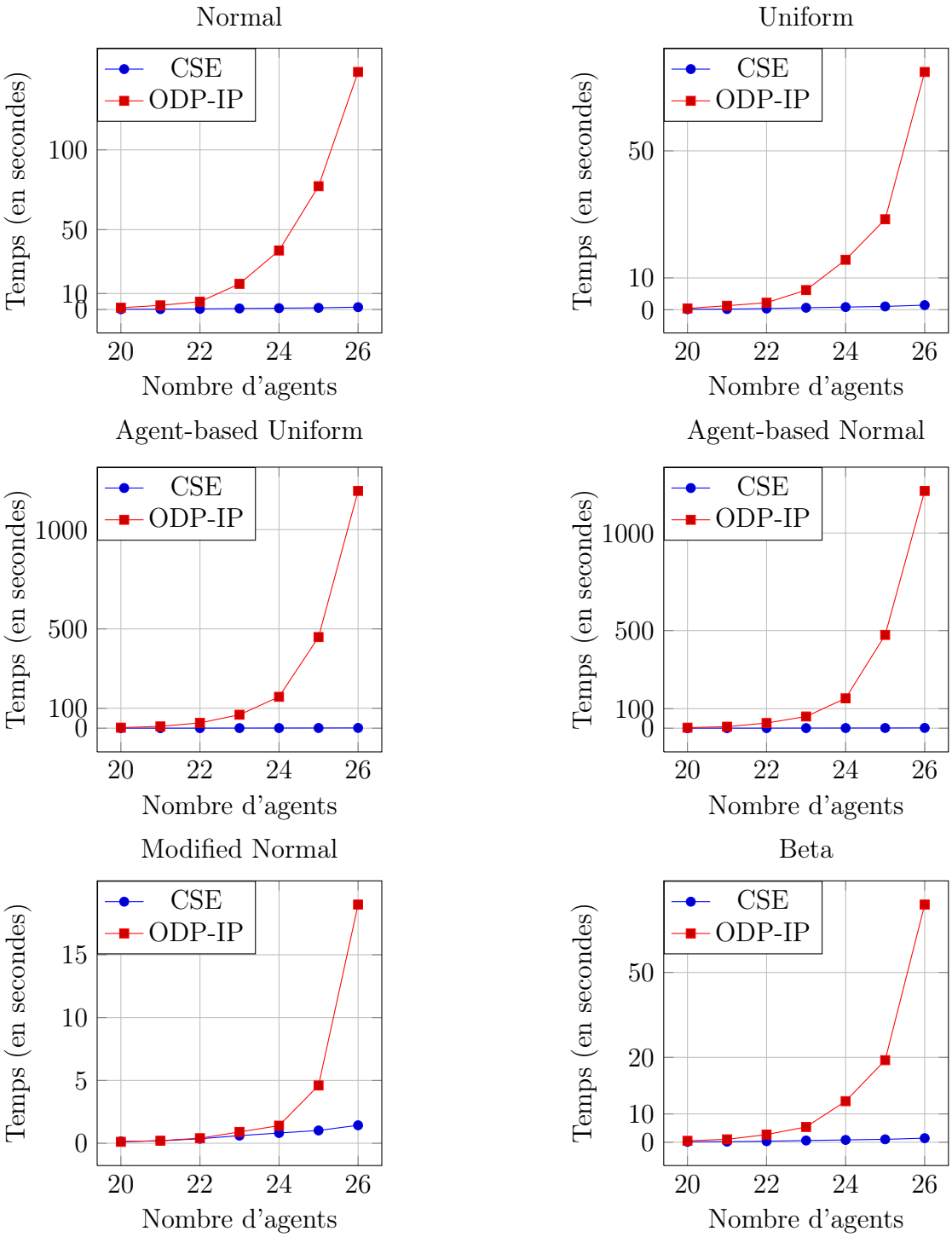


FIGURE 4.3 – Performances en temps d'exécution en seconds de CSE et ODP-IP.

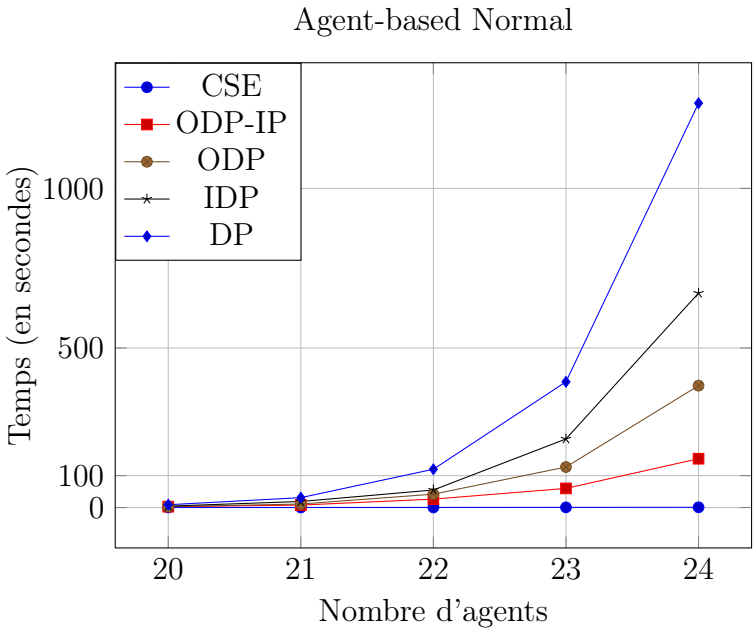


FIGURE 4.4 – Performance en temps d'exécution en seconds de quelque algorithmes étudiés dans le chapitre de l'état de l'art CSE et ODP-IP.

4.3.3.2 Performances en termes de qualité de solutions

Dans cette section, nous évaluons la qualité des solutions trouvées par notre algorithme. Nous les évaluons dans un premier temps par rapport à la qualité de la structure de coalitions optimale. Puis, dans un second temps, par rapport à la qualité de la solution trouvé par ODP-IP au bout du temps d'exécution de CSE.

En utilisant la qualité de solutions comme métrique, notre algorithme fournit généralement une qualité de solution supérieure à 99 %. Cependant, pour avoir plus d'informations sur la qualité des solutions trouvées, nous allons plutôt utiliser la métrique qualité réelle de solutions.

Le tableau 4.2 montre la qualité réelle de la solution générée par CSE en fonction de différentes distributions de valeurs.

Les résultats montrent que notre algorithme génère des solutions très proches de l'optimum, où la qualité réelle de solution est généralement supérieure à 93 %. Par ailleurs, en énumérant les structures de coalitions plutôt que les coalitions, CSE fournit une solution intermédiaire à tout moment de l'exécution, ce qui le rend anytime, et les résultats montrent que CSE produit des solutions de haute qualité s'il est interrompu avant d'avoir terminé.

Distribution	Qualité réelle de solution
Agent-based normal	99%
Agent-based uniform	89%
Uniform	99%
Normal	93%
Beta	99%
NDCS	85%
Modified normal	83%
Modified uniform	78%
Exponential	76%

TABLE 4.2 – Qualité Réelle de solution, de CSE.

4.4 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle technique pour le problème de génération de structures de coalitions. Cette technique se base sur la représentation présentée dans le chapitre précédant où chaque coalition d'une structure de coalitions se voit attribuer un numéro d'ordre. Notre heuristique appelé CSE permet ainsi d'énumérer des structures de coalitions. Notre analyse et notre évaluation ont montré que CSE fournit des solutions de haute qualité qui se rapprochent de la solution optimale. De plus, CSE est plus rapide que ODP-IP pour différents nombres d'agents et sur différentes distributions de valeur.

Chapitre 5

ADP : Un algorithme basé sur la moyenne pour la génération de structures de coalitions

5.1 Introduction

En pratique, l'algorithme ODP-IP est plus efficace pour un certain nombre d'instances de problème, mais pour d'autres, il est incapable de produire un résultat exact dans un temps raisonnable. Ceci est plus particulièrement le cas pour les distributions Agent-based Normal et Agent-based Uniform. Pour ce type de problèmes, si un algorithme efficace peut être trouvé pour résoudre la plupart des instances, à l'exception de quelques cas, alors il pourrait s'agir d'une méthode pratique. Dans ce but, nous avons défini un nouvel algorithme imparfait appelé ADP pour Averge-based Dynamic Programming, cela signifie que pour un large ensemble de problèmes, ADP fournit la solution optimale, mais pour certains cas, il fournit une solution quasi optimale en fonction du sous-espace de solutions qui contient la structure de coalitions optimale. Dans ce chapitre, nous détaillons en premier lieu le principe de l'algoithme ADP et les conditions sous lesquelles il échoue à produire le résultat exact. Nous évaluons par la suite les performances de notre algorithme. Après avoir présenté les métriques de performances ainsi que les différentes distributions de valeurs considérés, nous concluons par une analyse et une interprétation des résultats.

5.2 L'algorithme ADP

L'algorithme ADP que nous proposons est basée sur l'approche programmation dynamique et utilise deux tableaux : un tableau de partitions P_t pour stocker la partition optimale de chaque coalition C dans $P_t(C)$ et un tableau de valeurs V_t pour stocker la valeur optimale de cette coalition C dans $V_t(C)$. Il peut y avoir plus d'une partition optimale d'une coalition C, $P_t(C)$ stocke n'importe quelle partition d'entre elles.

Pour évaluer les coalitions, ADP commence par évaluer toutes les divisions possibles de chaque coalition de taille 2, ensuite, ADP augmente progressivement la taille par une unité de 1 jusqu'à ce que la taille devienne $\left\lceil \frac{n}{2} \right\rceil$ et complète les tableaux P_t et V_t pour chaque coalition évaluée C. De plus, pour une certaine taille t, ADP n'évalue pas toutes les coalitions de taille t. Il teste plutôt un sous-ensemble de ces coalitions. Nous détaillons dans ce qui suit l'idée qui a inspiré cet algorithme.

Partons de ces deux observations :

- Plus la valeur d'une coalition est grande, moins elle est susceptible d'être divisée.
- Plus la valeur d'une coalition est petite, moins sa division ou non n'a d'importance.

Supposons maintenant que nous disposions d'une fonction ACV (Average Coalition Value) qui calcule pour chaque taille t la valeur moyenne de toutes les coalitions de taille t. Ensuite, grâce à cette fonction les coalitions de taille t peuvent être séparées en 2 ensembles :

- **Ensemble 1 :** Les coalitions C de taille t, où $v(C) < ACV(t)$.
- **Ensemble 2 :** Les coalitions C de taille t, où $v(C) > ACV(t)$.

Pour ces deux ensembles, il est possible de calculer la valeur moyenne des coalitions dans chaque ensemble. C'est-à-dire, $ACV_1(t)$ pour l'ensemble 1 et $ACV_2(t)$ pour l'ensemble 2. Notre objectif principal avec cet algorithme est de réduire la charge de calcul. De ce fait, l'algorithme ADP évaluerait pour chaque taille t uniquement les coalitions C dont la valeur est comprise entre $ACV_1(t)$ et $ACV_2(t)$, c'est-à-dire les coalitions C où $ACV_1(t) < v(C) < ACV_2(t)$ (voir la Figure 5.1).

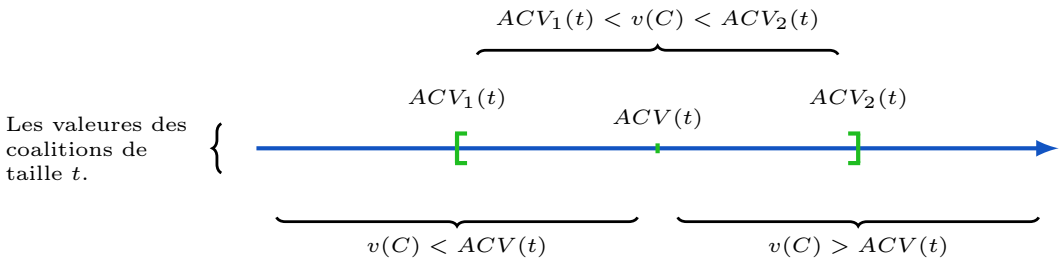


FIGURE 5.1 – Une Illustration de l'intervalle des valeurs des coalitions évaluées par ADP.

Dans notre exemple (voir la Figure 5.2), nous avons $ACV(2) = avg(61, 63, 76, 54, 82, 68) = 67.33$, donc $ACV_1(2) = avg(61, 63, 54) = 59.33$ et $ACV_2(2) = avg(76, 82, 68) = 75.33$, et $ACV(3) = avg(89, 119, 99, 114) = 105.25$, donc $ACV(3) = avg(89, 99) = 94$ et $ACV(3) = avg(119, 114) = 116.5$. Toutes les coalitions représentées en bleu ont des valeurs qui ne sont pas comprises entre

$ACV_1(2)$ et $ACV_2(2)$ pour les coalitions de taille 2 et des valeurs qui ne sont pas comprises entre $ACV_1(3)$ et $ACV_2(3)$ pour les coalitions de taille 3. Les opérations de division correspondantes à ces coalitions représentées en rouge ne sont donc pas évaluées. Cependant, ADP n'évalue aucune coalition de taille 3 car 3 est supérieur à $\left\lceil \frac{4}{2} \right\rceil = 2$.

Size	C	$v(C)$	Splitting	$P_t(C)$	$V_t(C)$
1	$\{a_1\}$	37	$v[\{a_1\}] = \mathbf{37}$	$\{a_1\}$	**37**
	$\{a_2\}$	25	$v[\{a_2\}] = \mathbf{25}$	$\{a_2\}$	**25**
	$\{a_3\}$	41	$v[\{a_3\}] = \mathbf{41}$	$\{a_3\}$	**41**
	$\{a_4\}$	39	$v[\{a_4\}] = \mathbf{39}$	$\{a_4\}$	**39**
2	$\{a_1, a_2\}$	61	$v[\{a_1, a_2\}] = 61$, $V_t[\{a_1\}] + V_t[\{a_2\}] = \mathbf{62}$	$\{a_1\}, \{a_2\}$	**62**
	$\{a_1, a_3\}$	63	$v[\{a_1, a_3\}] = 63$, $V_t[\{a_1\}] + V_t[\{a_3\}] = \mathbf{78}$	$\{a1\}, \{a_3\}$	**78**
	$\{a_1, a_4\}$	76	$v[\{a_1, a_4\}] = \mathbf{76}$, $V_t[\{a_1\}] + V_t[\{a_4\}] = 76$	$\{a_1, a_4\}$	**76**
	$\{a_2, a_3\}$	54	$v[\{a_2, a_3\}] = 54$, $V_t[\{a_2\}] + V_t[\{a_3\}] = \mathbf{66}$	$\{a_2\}, \{a_3\}$	**66**
	$\{a_2, a_4\}$	82	$v[\{a_2, a_4\}] = \mathbf{82}$, $V_t[\{a_2\}] + V_t[\{a_4\}] = 64$	$\{a_2, a_4\}$	**82**
	$\{a_3, a_4\}$	68	$v[\{a_3, a_4\}] = 68$, $V_t[\{a_3\}] + V_t[\{a_4\}] = \mathbf{80}$	$\{a_3\}, \{a_4\}$	**80**
3	$\{a_1, a_2, a_3\}$	89	$v[\{a_1, a_2, a_3\}] = 89$, $V_t[\{a_1\}] + V_t[\{a_2, a_3\}] = 103$ $V_t[\{2\}] + V_t[\{1, 3\}] = 103$, $V_t[\{3\}] + V_t[\{1, 2\}] = \mathbf{103}$	$\{a_3\}, \{a_1, a_2\}$	**103**
	$\{a_1, a_2, a_4\}$	119	$v[\{a_1, a_2, a_4\}] = 117$, $V_t[\{a_1\}] + V_t[\{a_2, a_4\}] = \mathbf{119}$ $V_t[\{a_2\}] + V_t[\{a_1, a_4\}] = 101$, $V_t[\{a_4\}] + V_t[\{a_1, a_2\}] = 101$	$\{a_1\}, \{a_2, a_4\}$	**119**
	$\{a_1, a_3, a_4\}$	99	$v[\{a_1, a_3, a_4\}] = 99$, $V_t[\{a_1\}] + V_t[\{a_3, a_4\}] = 117$ $V_t[\{a_3\}] + V_t[\{a_1, a_4\}] = \mathbf{117}$, $V_t[\{a_4\}] + V_t[\{a_1, a_3\}] = 117$	$\{a_3\}, \{a_1, a_4\}$	**117**
	$\{a_2, a_3, a_4\}$	114	$v[\{a_2, a_3, a_4\}] = 114$, $V_t[\{a_2\}] + V_t[\{a_3, a_4\}] = 105$ $V_t[\{a_3\}] + V_t[\{a_2, a_4\}] = \mathbf{123}$, $V_t[\{a_4\}] + V_t[\{a_2, a_3\}] = 105$	$\{a_3\}, \{a_2, a_4\}$	**123**
4	$\{a_1, a_2, a_3, a_4\}$	141	$v[\{a_1, a_2, a_3, a_4\}] = 141$, $V_t[\{a_1\}] + V_t[\{a_2, a_3, a_4\}] = 160$ $V_t[\{a_2\}] + V_t[\{a_1, a_3, a_4\}] = 142$, $V_t[\{a_3\}] + V_t[\{a_1, a_2, a_4\}] = 160$ $V_t[\{a_4\}] + V_t[\{a_1, a_2, a_3\}] = 142$, $V_t[\{a_1, a_2\}] + V_t[\{a_3, a_4\}] = 142$ $V_t[\{a_1, a_3\}] + V_t[\{a_2, a_4\}] = \mathbf{160}$, $V_t[\{a_1, a_4\}] + V_t[\{a_2, a_3\}] = 142$	$\{a_1, a_3\}, \{a_2, a_4\}$	**160**

FIGURE 5.2 – Principe de fonctionnement des algorithmes DP et ADP calculant $P_t(C)$, $V_t(C)$ pour chaque coalition $C \subseteq A$. Les valeurs surlignées en vert pour les opérations de division possibles de la grande coalition A peuvent avoir des valeurs plus petites en raison de la non-évaluation des coalitions surlignées en bleu qui pourraient également avoir des valeurs plus petites.

Une fois que ADP a calculé P_t et V_t pour chaque coalition évaluée, la structure de coalitions optimale est ensuite calculée de manière récursive en utilisant le tableau de partitions P_t. Dans notre exemple (voir la Figure 5.2), notre algorithme commence par $CS^* = \{a_1, a_2, a_3, a_4\}$, la grande coalition, trouve ensuite que la structure de coalitions qui a la plus grande valeur est $\{\{a_1, a_3\}, \{a_2, a_4\}\}$. Ensuite, il trouve qu'il est bénéfique de diviser la coalition $\{a_1, a_3\}$ en coalitions $\{a_1\}$ et $\{a_3\}$. De même, il conclut qu'il est avantageux de maintenir la coalition $\{a_2, a_4\}$ dans sa forme initiale. Au final, la structure de coalitions optimale est $\{\{a_1\}, \{a_3\}, \{a_2, a_4\}\}$ avec une valeur de 160.

Pour visualiser le résultat de ces opérations, nous introduisons le graphe des partitions (voir la Figure 5.3) qui contient des noeuds représentant des partitions entières de n répartis en n niveaux. Chaque partition entière P représente un ensemble de structures de coalitions (c'est-à-dire un sous-espace de solutions) dans lequel les tailles des coalitions correspondent aux parties de P. Par exemple, le noeud [1,1,8] est composé de toutes les structures de coalitions qui contiennent deux coalitions de taille 1 et une coalition de taille 8.

Dans le graphe de la figure 5.3, nous constatons que ADP peut atteindre chaque noeud à partir du noeud inférieur, à l'exception des noeuds rouges qu'il n'évalue pas, sans évaluer aucune des opérations de division des coalitions de taille $\left\lceil \frac{n}{2} \right\rceil + 1$ à $n-1$ représentées dans le graphe par les pointillés rouges (voir la Figure 5.3). Éviter l'évaluation de ces coalitions correspond à supprimer tous les pointillés rouges dans le graphe des partitions.

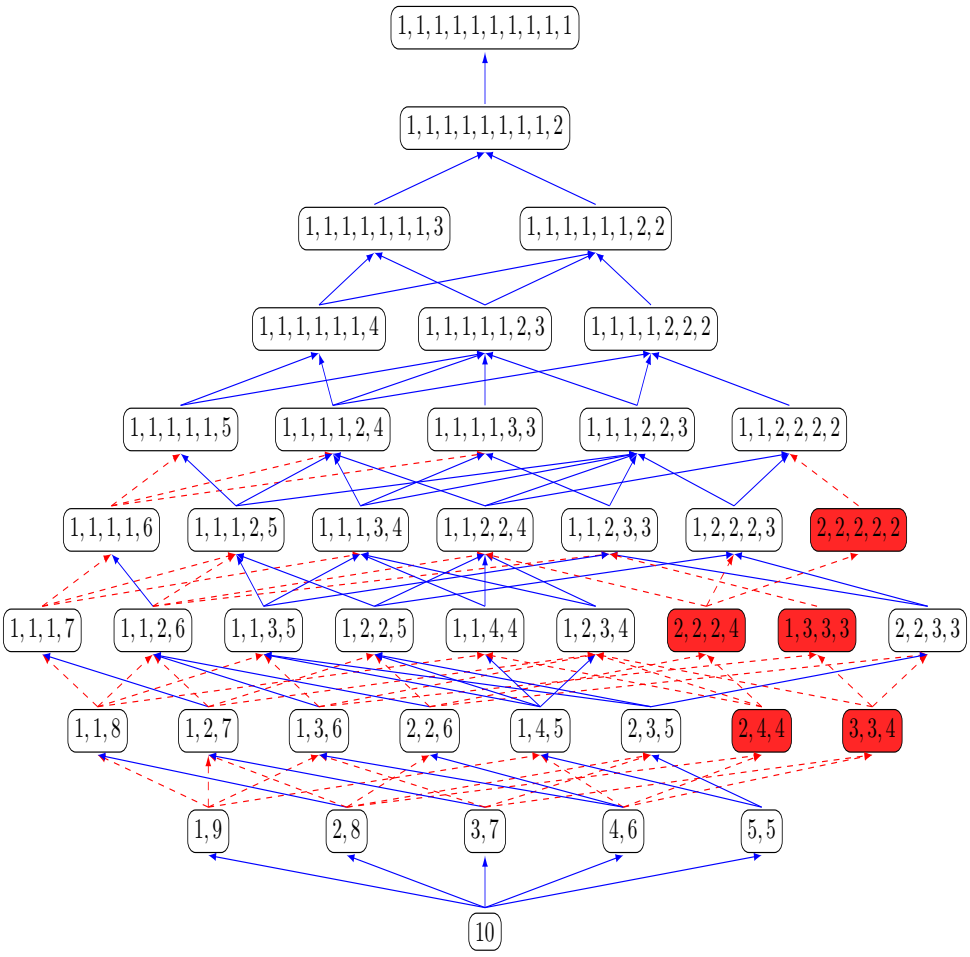


FIGURE 5.3 – Un exemple à dix agents du graphe des partitions. Les noeuds blancs sont recherchés par ADP tandis que les noeuds rouges ne sont pas recherchés par ADP.

Lors de la recherche de la solution optimale, les deux cas suivants se présentent à ADP :

1. **La structure de coalitions optimale appartient aux noeuds blancs :** Supposons que l'un des sous-espaces recherchés par ADP contienne une solution optimale. Dans ce cas, ADP trouve la structure de coalitions optimale pour un large ensemble de problèmes, mais ne parvient pas à le faire pour certains problèmes. Cela est dû au fait que pour une taille donnée de coalitions, ADP évite d'évaluer certaines opérations de division, ce qui lui fait perdre les garanties de trouver la structure de coalitions optimale dans les sous-espaces évalués. Cependant, si ADP ne produit pas le résultat exact, il peut toujours fournir une solution proche de ce résultat exact.

2. **La structure de coalitions optimale appartient aux noeuds rouges :** Si l'un des sous-espaces qui ne sont pas recherchés par ADP contient la solution optimale, ADP ne parvient pas à trouver la solution optimale. Cependant, ADP parvient à produire une solution qui est proche de l'optimum exact. Cette solution correspond à la meilleure structure de coalitions trouvée entre celles qui ont été évaluées, c'est-à-dire la meilleure solution trouvée dans les sous-espaces recherchés.

La différence avec les heuristiques réside dans le fait que pour certains cas, l'algorithme ADP ne fournit pas la solution optimale, mais pour la majorité des cas, il parvient à atteindre cette solution optimale. Cela dépend du sous-espace qui contient la solution optimale. De plus, la probabilité de fournir une solution optimale est beaucoup plus élevée avec l'algorithme ADP qu'avec une heuristique. Par exemple, nous avons trouvé que pour 20 agents, ADP recherche 92% des sous-espaces (c'est-à-dire 577 sous-espaces recherchés sur 627), et pour 25 agents, ADP recherche 97% des sous-espaces (c'est-à-dire 1903 sous-espaces recherchés sur 1958).

Le pseudocode de ADP est donné dans l'algorithme 2, qui itére sur les noeuds parcourues du graphe des partitions. Dans l'algorithme 2, les lignes 1 à 7, montrent une boucle qui va de 1 à n pour stocker les valeurs de chaque coalition et calculer la valeur moyenne ($ACV(t)$) pour chaque taille de coalitions t. Les lignes 8 à 19 calculent la partition optimale de chaque coalition évaluée. Les lignes 20 à 25 évaluent la grande coalition A (c'est-à-dire que le noeud le plus inférieur du graphe des partitions est toujours évalué). Enfin, les lignes 26 à 32 trouvent la solution optimale.

5.3 Evaluation de performances

Cette section est consacrée à l'évaluation des performances de notre algorithme. L'algorithme ADP a été implémenté en Java et les simulations ont été réalisées sur un ordinateur personnel, le satellite c50-a559 qui a la configuration d'un processeur Intel Core i3-3110M à 2.40GHz et 4 GO

Algorithm 2 L'algorithme ADP

Input Ensemble de toutes les coalitions possibles et la valeur $v(C)$ de chaque coalition $C \in 2^A$. A désigne un ensemble de n agents.

Output Une structure de coalition optimale ou quasi-optimale CS^*.

1: **for** $t = 1$ à n **do**
2: **for** $C \subseteq A$, où $\mid C \mid = t$ **do**
3: $V_t(C) \leftarrow v(C)$
4: $P_t(C) \leftarrow C$
5: **end for**
6: $ACV(t) \leftarrow avg_i(v(C_i))$
7: $ACV_1(t) \leftarrow avg^i_{v(C_i)<ACV(t)}(v(C_i))$
8: $ACV_2(t) \leftarrow avg^i_{v(C_i)>ACV(t)}(v(C_i))$
9: **end for**
10: **for** $t = 2$ à $\left\lceil \frac{n}{2} \right\rceil$ **do**
11: **for** $C \subseteq A$, où $\mid C \mid = t$ **do**
12: **if** $ACV_1(t) < v(C) < ACV_2(t)$ **then**
13: **for** $C_1, C_2 \subseteq C$, où $C_1 \cup C_2 = C$ et $C_1 \cap C_2 = \emptyset$ **do**
14: **if** $V_t(C_1) + V_t(C_2) > V_t(C)$ **then**
15: $V_t(C) \leftarrow V_t(C_1) + V_t(C_2)$
16: $P_t(C) \leftarrow \{C_1, C_2\}$
17: **end if**
18: **end for**
19: **end if**
20: **end for**
21: **end for**
22: **for** $C_1, C_2 \subseteq A$, où $C_1 \cup C_2 = A$ et $C_1 \cap C_2 = \emptyset$ **do**
23: **if** $V_t(C_1) + V_t(C_2) > V_t(A)$ **then**
24: $V_t(A) \leftarrow V_t(C_1) + V_t(C_2)$
25: $P_t(A) \leftarrow \{C_1, C_2\}$
26: **end if**
27: **end for**
28: $CS^* \leftarrow A$
29: **for** $C \subseteq CS^*$ **do**
30: **if** $P_t(C) \neq C$ **then**
31: $CS^* \leftarrow CS^* \setminus C \cup P_t(C)$
32: **end if**
33: Aller à la ligne 29 et commencer avec la nouvelle CS^*
34: **end for**
35: Return CS*

de mémoire.

5.3.1 Métriques de performances

Dans cette section, nous présentons les métriques de performances que nous avons utilisés pour l'évaluation de performances de notre algorithme.

Les métriques de performances utilisées pour évaluer notre heuristique CSE, à savoir le temps d'exécution et la qualité de solutions sont utilisées aussi pour évaluer ADP. De plus, une autre métrique est utilisée pour évaluer notre algorithme, à savoir le taux d'échec.

- **Temps d'exécution**
  Le temps nécessaire pour trouver une solution.

- **Qualité de solutions**
  Dans le cas où notre algorithme ne parvient pas à fournir la solution optimale, il est intéressant de calculer à quel point la solution trouvée est proche de l'optimum exact. Elle est calculée comme le rapport entre la valeur de la meilleure structure de coalition CS^+ trouvée par notre algorithme et la valeur de la solution optimale CS^*. Formellement, la qualité de la solution représente :
  $$\frac{v(CS^+) * 100}{v(CS^*)}\%.$$

- **Taux d'échec**
  Le nombre de fois que l'algorithme échoue à fournir le résultat exact sur un ensemble donné d'instances du problème.

5.3.2 Distributions de valeurs

Un algorithme est dit imparfait si, pour certaines entrées, l'algorithme ne parvient pas à fournir le résultat correct [12].

Une façon de valider ou de comparer un algorithme imparfait est de l'exécuter sur des instances du problème et de voir combien de fois il échoue. Dans ce contexte, nous avons considéré les distributions suivantes et effectué des tests pour des ensembles de 8 à 25 agents : agent-based uniform, agent-based normal, beta, exponential, modified normal, modified uniform, Normally Distributed Coalition Structures (NDCS), uniform, normal.

5.3.3 Résultats

Dans cette section, nous présentons les différents résultats obtenues en comparant les performances de notre algorithme à celles de l'algorithme ODP-IP. Nous avons généré de façon aléatoire

plusieurs instances du problème en faisant varier le nombre d'agents impliqués.

5.3.3.1 Performances en termes de temps d'exécution

Dans cette section, nous évaluons les performances de notre algorithme en termes de temps d'exécution et comparons les résultats par rapport aux performances de l'algorithme ODP-IP.

Les Figures 5.4, 5.5 et 5.6 illustrent l'évolution du temps d'exécution en fonction du nombre d'agents pour chaque distribution de valeurs.

Ces tests ont montré que pour les distributions de valeurs Agent-based normal et Agent-based uniform, le gain moyen en temps d'exécution avec ADP est de 69.56% et 57.55% respectivement. De manière générale, nous avons trouvé des distributions de valeurs pour lesquels ADP fonctionne mieux que ODP-IP et d'autres pour lesquels ODP-IP fonctionne mieux.

Nous avons remarqué que ADP et ODP-IP se comportent d'une certaine manière en fonction du nombre d'agents. C'est pourquoi nous avons divisé les résultats des performances en trois ensembles. Pour l'ensemble 1, comme le montre la Figure 5.4, nous constatons que ADP est plus rapide que ODP-IP pour toutes les distributions où le nombre d'agents est inférieur à 16.

Pour l'ensemble 2, le nombre d'agents est entre 16 et 20 comme le montre la Figure 5.5. Nous avons observé que ADP est toujours plus rapide que ODP-IP pour les distributions Agent-based uniform et Agent-based normal. Pour les autres distributions, ADP est parfois plus rapide et parfois non.

Pour l'ensemble 3, le nombre d'agents est compris entre 21 et 25 comme le montre la Figure 5.6. Nous avons constaté que ADP est plus rapide pour Agent-based uniform et Agent-based normal. Pour les autres distributions, ODP-IP est meilleur.

ADP n'est pas un algorithme exact, mais il peut générer des structures de coalitions optimales (ou quasi optimales) en moins de temps que ODP-IP pour certaines distributions. Par exemple, pour Agent-based Normal, ADP est plus rapide que ODP-IP de 335 secondes (i.e. 6 minutes) pour 25 agents. L'efficacité des techniques de branch and bound de IP joue un rôle essentiel dans ODP-IP, qui dépend à son tour des fonctions caractéristiques en question. Pour certaines distributions, ODP-IP est beaucoup plus rapide en raison de l'efficacité de IP sur ces distributions qui supprime ainsi une grande partie de l'espace de recherche. Cependant, l'application de branch-and-bound dans certaines distributions est très difficile, ce qui explique pourquoi ODP-IP est plus rapide dans certains cas et plus lent dans d'autres.

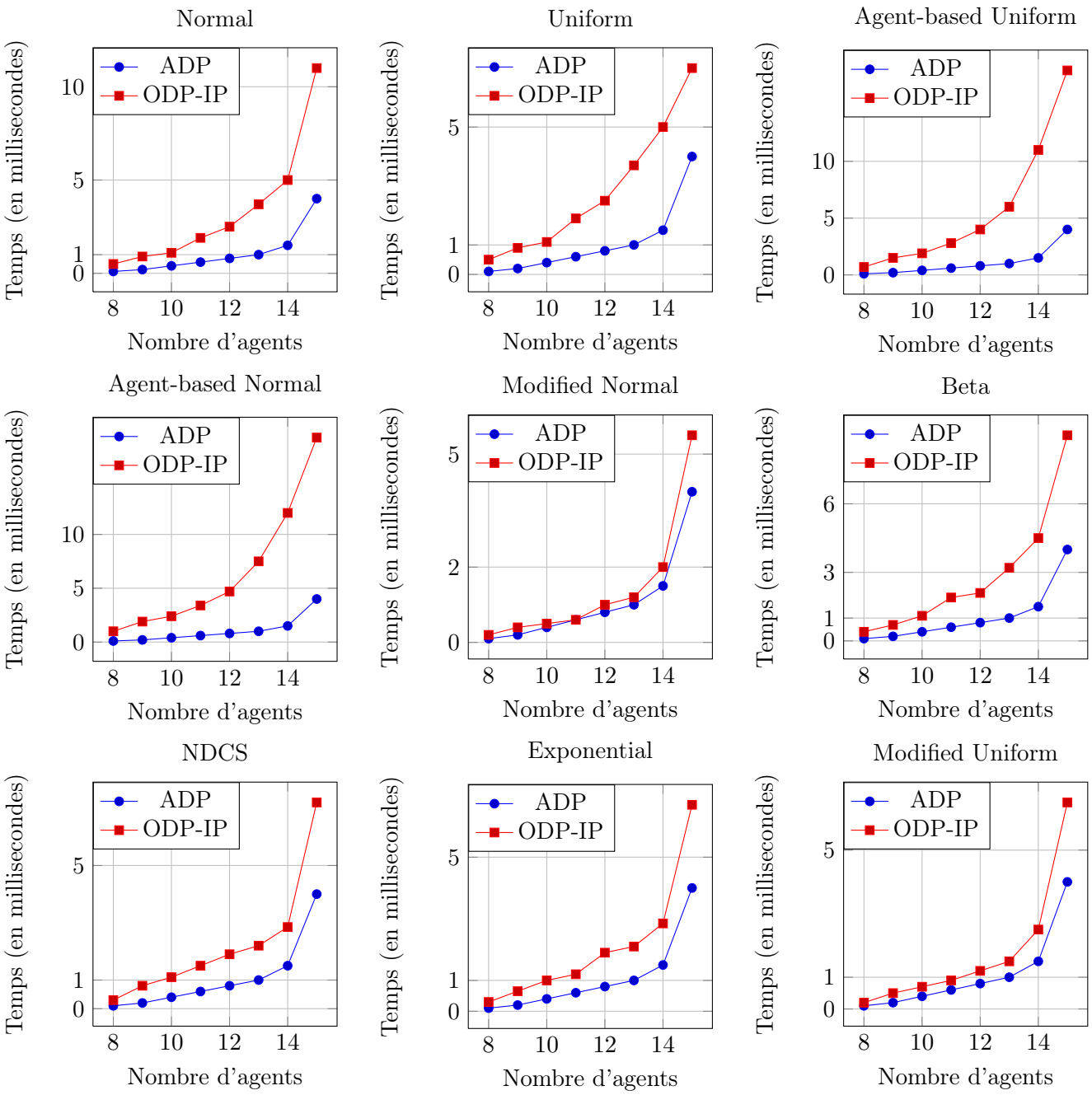


FIGURE 5.4 – Performance en temps d'exécution en milli-seconds de ADP et ODP-IP.

5.3.3.2 Performances en termes de taux d'échec

Dans cette section, nous évaluons la fréquence à laquelle notre algorithme échoue à produire le résultat exact et déterminons les conditions sous lesquelles il échoue.

Comme le montre la Figure 5.3 du graphe des partitions, si la structure de coalitions appartient aux sous espaces non évalués (c'est-à-dire, les noeuds rouges) alors ADP échoue à produire le résultat exact.

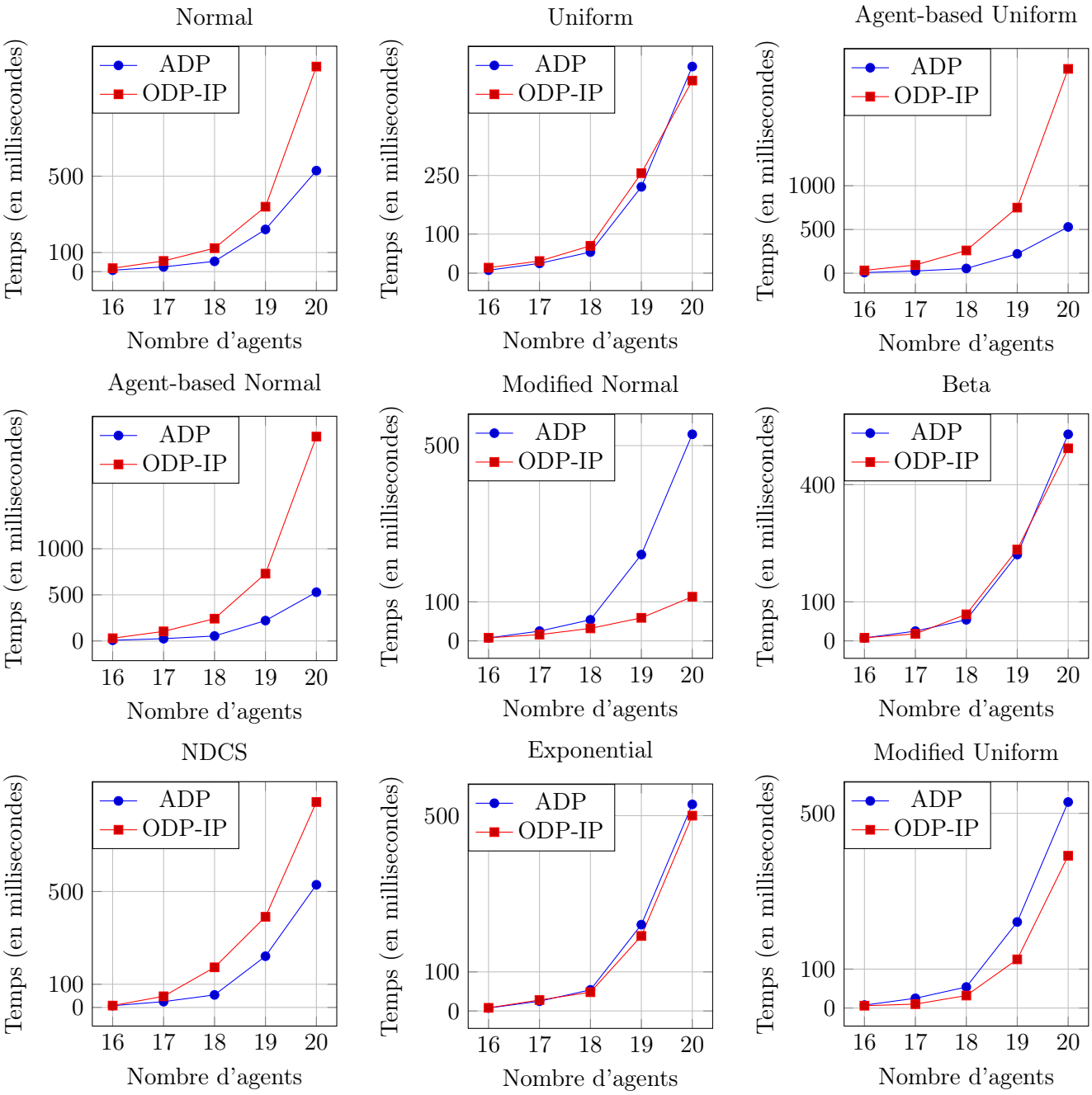


FIGURE 5.5 – Performance en temps d'exécution en milli-seconds de ADP et ODP-IP.

Le tableau 5.1 montre que ADP recherche généralement plus de 90% des sous-espaces de solutions et dans certains cas, plus de 97%. Cela montre que les chances de trouver la solution optimale sont très élevées. Dans la section précédente, nous avons décrit dans quelles conditions ADP ne parvient pas à produire la structure de coalitions optimale. Le nombre exact de sous-espaces de solutions recherchés est indiqué dans le tableau 5.1.

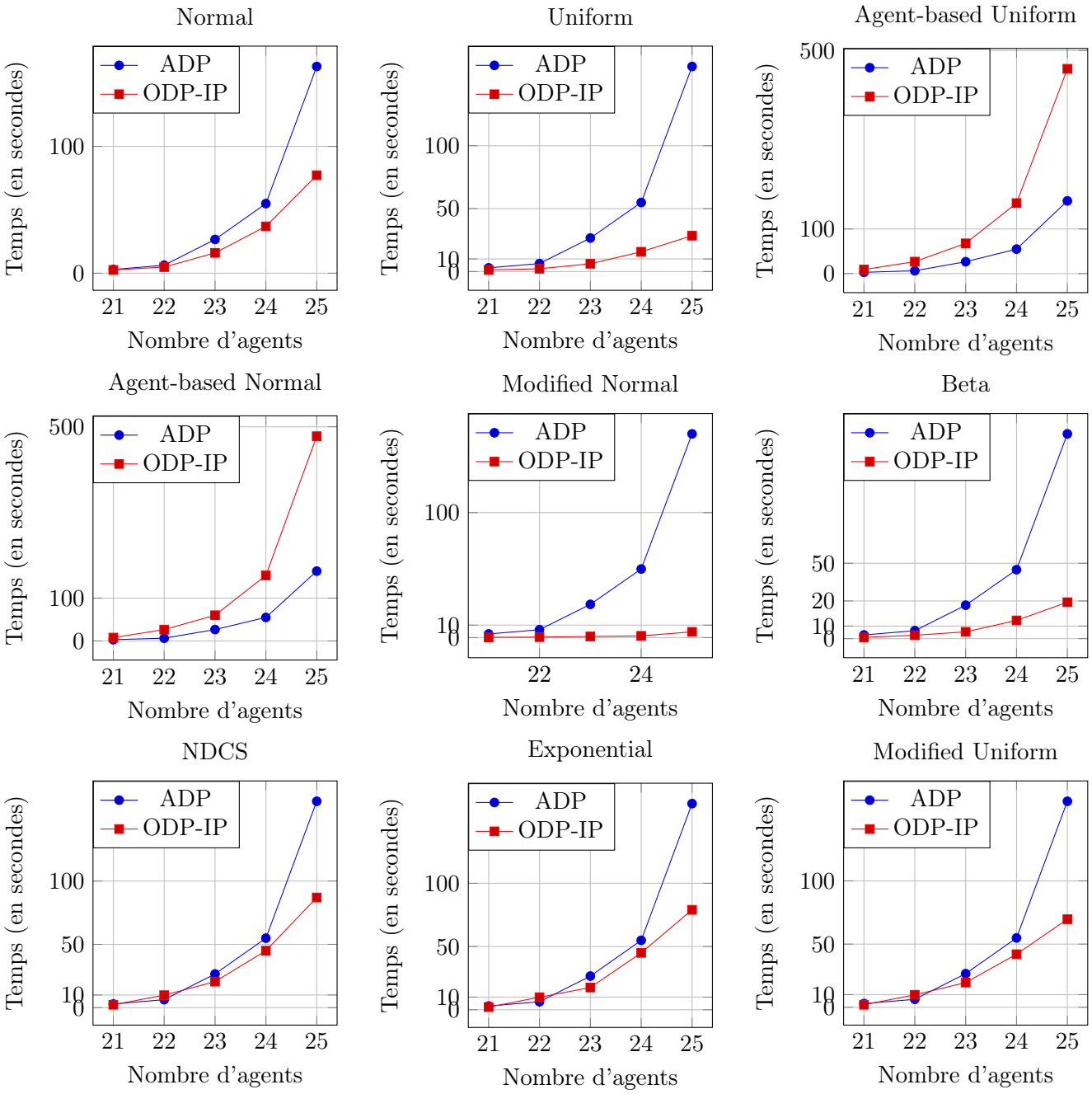


FIGURE 5.6 – Performance en temps d'exécution en seconds de ADP et ODP-IP.

L'algorithme ADP n'échoue que pour quelques instances du problème, et le taux d'échec est très faible. Par exemple, nous avons trouvé que les taux d'échec pour les distributions Agent-based Normal ($A - b\ N$) et Normal (N) sont respectivement 3% et 1% (voir le tableau 5.2).

5.3.3.3 Performances en termes de qualité de solutions

ADP est un algorithme imparfait, pour évaluer la qualité des solutions trouvées par notre algorithme lorsqu'il ne parvient pas à touver la solution optimale, nous avons comparé les valeurs

Nombre d'agents	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Nombre de sous-espaces (n_1)	627	792	1002	1255	1575	1958	2436	3010	3718	4565	5604
Nombre de sous-espaces recherchés (n_2)	577	763	899	1221	1468	1903	2220	2921	3462	4446	5190
Ratio (%) $\frac{n_2*100}{n_1}$ %	92.02	96.33	89.72	97.29	93.21	97.19	91.13	97.04	93.11	97.39	92.61

TABLE 5.1 – Nombre de sous-espaces de solutions partiellement recherchés.

Distribution	**A-b N**	**A-b U**	**Beta**	**NDCS**	**N**	**U**	**M-N**	**M-U**	**Exp**
Nombre d'échecs	3	3	2	9	1	10	1	2	4

TABLE 5.2 – Nombre d'échecs de l'algorithme ADP sur 100 exécutions par distribution.

des structures de coalitions générées pas ADP aux valeurs des structures de coalitions optimales. Nous avons constaté que la qualité de la solution produite par ADP est toujours supérieure à 99%.

5.4 Conclusion

La génération de structures de coalitions dans les systèmes multi-agents est une tâche difficile et les chercheurs ont proposé plusieurs techniques de solutions adaptées pour résoudre ce problème important. Cependant, très peu d'algorithmes imparfaits ont été proposés pour résoudre ce problème. Dans ce chapitre, nous avons présenté un algorithme imparfait pour la génération de structures de coalitions. Tout d'abord, nous avons décrit les observations sur lesquelles nous nous sommes basés pour concevoir notre algorithme. Puis, nous avons présenté notre algorithme qui ne traite qu'un sous-ensemble de coalitions. ADP produit la structure de coalitions optimale pour la majorité des cas. Dans le cas où ADP ne fournit pas la solution optimale, il peut néanmoins produire une solution proche du résultat exact. Ensuite, nous avons présenté les conditions sous lesquelles ADP ne parvient pas à produire la solution optimale et nous avons comparé les performances de ADP à celles de ODP-IP. Notre analyse et notre évaluation ont montré que ADP est plus performant que ODP-IP sur plusieurs distributions de valeur sous les 20 agents et pour les distributions Agent-based Normal et Agent-based Uniform, ADP est toujours plus rapide que ODP-IP. De plus la qualité des solutions générées par ADP est toujours supérieure à 99%.

Chapitre 6

Un algorithme à faible mémoire pour la génération de structures de coalitions

6.1 Introduction

Le temps d'exécution est le problème principal dans la génération de structures de coalitions. En effet, pour différencier deux algorithmes, on compare généralement leurs temps d'exécution et la qualité des solutions générées. Cependant, un autre problème qui est tout aussi important que le temps d'exécution, mais qui est relativement négligé, est celui de l'espace mémoire requis par les algorithmes existants qui nécessitent au moins de stocker la fonction caractéristique. Pour les problèmes de grande taille, ce problème rend ainsi les algorithmes existants totalement impraticables. Dans ce chapitre, nous aborderons en premier lieu quelques problèmes liés aux algorithmes existants. Nous présenterons, par la suite, notre algorithme de génération de structures de coalitions qui permet de réduire les besoins en mémoire d'un problème de taille n en le divisant en deux sous-problèmes de taille $\frac{n}{2}$. Ensuite, nous décrirons la technique que nous utilisons pour répartir les agents en deux groupes afin de former les deux sous-problèmes. Enfin, nous concluons par une analyse des performances et une interprétation des résultats.

6.2 Problèmes des algorithmes existants

Un problème majeur des algorithmes exacts existants est leur complexité temporelle dans le pire des cas. S'il y a suffisamment de temps pour effectuer les calculs nécessaires pour trouver le résultat exact, les algorithmes exacts restent pratiques. Toutefois, dans certains cas, lorsque les agents n'ont pas suffisamment de temps pour effectuer les calculs requis par un algorithme exact, une approche qui donne une bonne approximation, dans un délai raisonnable, est plus utile. C'est pourquoi des heuristiques et des algorithmes imparfaits sont proposés pour générer la structure de coalitions optimale ou quasi-optimale en moins de temps que les algorithmes exacts.

Cependant, en concentrant toute l'attention sur le problème ci-dessus, nous en négligeons un

autre, qui est tout aussi important que le temps d'exécution. Ce problème est celui des besoins élevés en mémoire des algorithmes existants qui nécessitent au moins de stocker la fonction caractéristique à disposition. De ce fait, dans le cas des problèmes de grande taille, la mémoire disponible peut ne pas être suffisante.

Par exemple, pour 26 agents, pour stocker la fonction caractéristique, les algorithmes existants ont besoin de plus de 2 Go d'espace mémoire disponible. Cependant, pour 50 agents, les algorithmes existants nécessiteraient plus de 16 millions de fois l'espace requis pour 26 agents, ce qui rend les algorithmes existants totalement impraticables.

Dans ce contexte, nous décrivons dans ce qui suit une nouvelle technique facile à mettre en oeuvre pour générer une structure de coalitions quasi optimale, en particulier pour les problèmes de grande taille.

6.3 Low Memory Algorithm

Dans cette section, nous décrivons notre algorithme pour la génération de structures de coalitions. Notre algorithme est conçu en trois phases principales. La première est une phase de prétraitement qui consiste à diviser l'ensemble des agents en deux sous-ensembles en fonction des valeurs des coalitions singleton. Les sous-ensembles résultants sont ensuite utilisés dans la phase 2 pour générer les valeurs des coalitions maintenues. La troisième phase consiste à résoudre les sous-problèmes qui en résultent pour calculer CS^+, une solution quasi optimale. Nous détaillons chacune de ces phases dans les sous-sections suivantes.

6.3.1 Phase 1 : Prétraitement

La première phase consiste à générer et à stocker les valeurs des coalitions contenant un agent (c'est-à-dire les coalitions singleton). De plus, dans cette phase, les agents sont répartis en deux groupes de taille respectivement $\left\lfloor \frac{n}{2} \right\rfloor$ et $n - \left\lfloor \frac{n}{2} \right\rfloor$. Cette répartition peut se faire soit de manière aléatoire, soit en utilisant un autre algorithme pour assurer une bonne séparation basée sur les valeurs des coalitions singleton.

Par exemple, pour $n = 5$ agents, notre algorithme stocke les valeurs des cinq coalitions singleton, c'est-à-dire $\{a_1\}$, $\{a_2\}$, $\{a_3\}$, $\{a_4\}$ et $\{a_5\}$. Ensuite, il sépare les agents en deux groupes : G_1 et G_2. Par exemple, $G_1 = \{a_1, a_2\}$ et $G_2 = \{a_3, a_4, a_5\}$.

6.3.2 Phase 2 : Génération des valeurs des coalitions

La deuxième phase consiste à générer et à stocker les valeurs des coalitions appartenant à chaque groupe d'agents. Cette seconde phase crée ainsi deux sous-problèmes de taille $\left\lfloor \frac{n}{2} \right\rfloor$ et $n - \left\lfloor \frac{n}{2} \right\rfloor$: CSG_1 et CSG_2.

Ici, les agents appartenant à G_1 forment le premier sous-problème CSG_1 de taille $\left\lfloor \frac{n}{2} \right\rfloor = 2$ et les agents appartenant à G_2 forment le second sous-problème CSG_2 de taille $n - \left\lfloor \frac{n}{2} \right\rfloor = 3$. Ainsi, seule la valeur de la coalition $\{a_1, a_2\}$ est stockée pour le sous-problème CSG_1 et seules les valeurs des coalitions $\{a_3, a_4\}$, $\{a_3, a_5\}$, $\{a_4, a_5\}$ et $\{a_3, a_4, a_5\}$ sont stockés pour le sous-problème CSG_2. Les coalitions restantes qui n'appartiennent ni à CSG_1 ni à CSG_2 (par exemple $\{a_1, a_4\}$) ne sont donc pas stockées.

Cela réduit les besoins en mémoire d'un problème de taille n (c'est-à-dire $2^n - 1$ coalitions) aux besoins en mémoire des deux sous-problèmes CSG_1 et CSG_2 de taille $\left\lfloor \frac{n}{2} \right\rfloor$ et $n - \left\lfloor \frac{n}{2} \right\rfloor$ respectivement qui correspondent à $2^{\lfloor \frac{n}{2} \rfloor} + 2^{n - \lfloor \frac{n}{2} \rfloor} - 2$ coalitions.

Ainsi, pour un problème de 50 agents, notre algorithme ne nécessite que le double de l'espace mémoire requis pour un problème de 25 ($\left\lceil \frac{n}{2} \right\rceil$) agents. Ce qui en fait un espace mémoire un peu plus grand que 2 Go (contrairement à un espace plus grand que 32 millions de Go pour un algorithme exact).

6.3.3 Phase 3 : Résolution des sous-problèmes

Cette phase représente la phase d'exécution qui résout les sous-problèmes qui résultent de la phase 2. Chaque sous-problème est ainsi résolu à l'aide de n'importe quel algorithme particulier, même celui que nous proposons. Pour cette étude, nous utilisons l'algorithme exact le plus rapide pour résoudre le problème de génération de structures de coalitions, ODP-IP, pour résoudre les deux sous-problèmes. La valeur de la meilleure structure de coalitions trouvée par notre algorithme pour le problème de taille n est alors la somme des valeurs des deux structures de coalitions optimales trouvées par l'algorithme ODP-IP pour chaque sous-problème.

La Figure 6.1 illustre le fonctionnement de notre algorithme avec 5 agents.

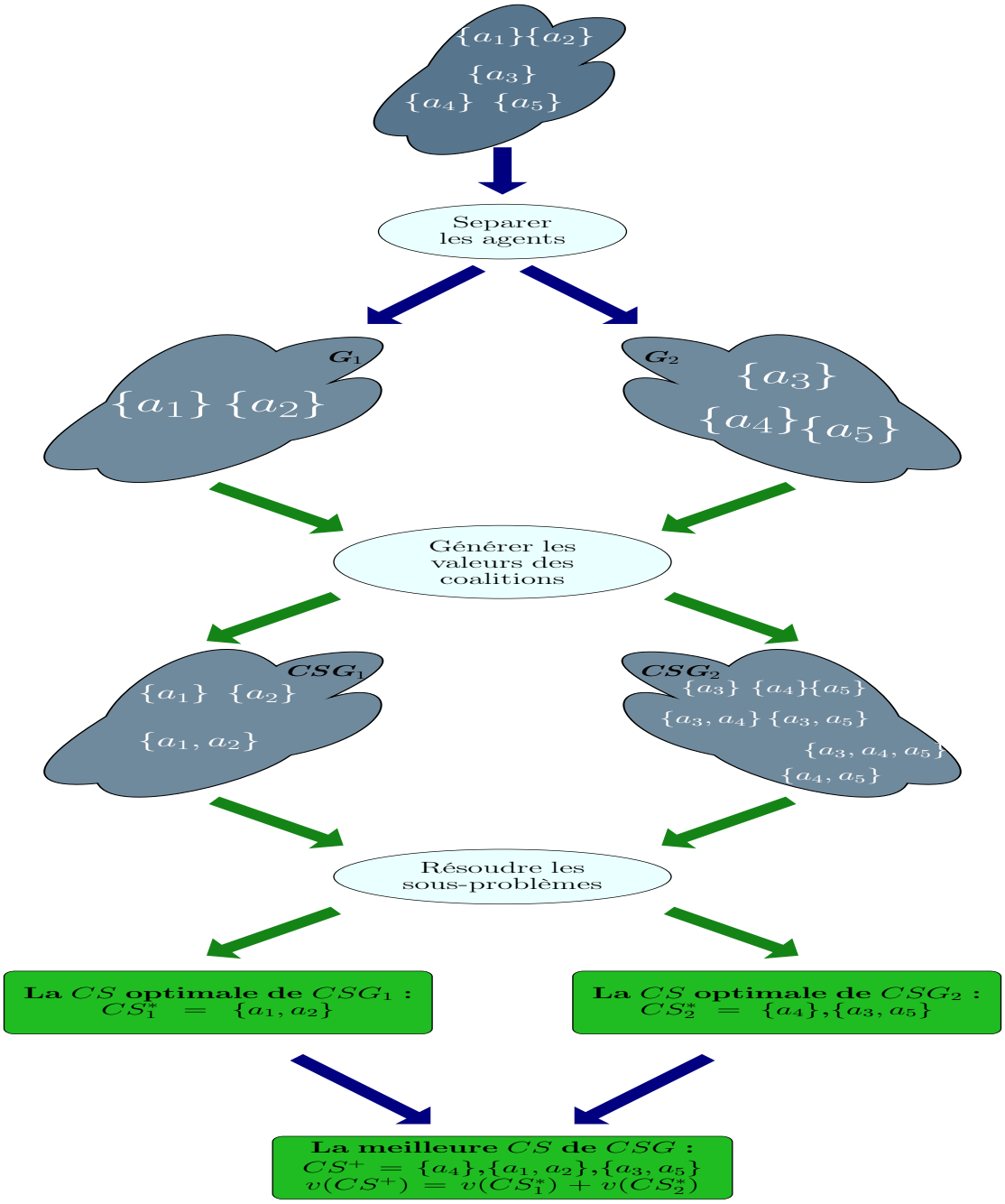


FIGURE 6.1 – Une illustration des trois phases de notre algorithme.

Pour mieux visualiser le résultat des différentes phases de notre algorithme, nous introduisons le graphe des partitions (voir la Figure 6.2).

La Figure 6.2 montre que notre algorithme peut atteindre à partir du noeud inférieur chaque noeud connecté au noeud $[5, 5]$ (c'est-à-dire $[\lfloor \frac{n}{2} \rfloor = 5,\ n - \lfloor \frac{n}{2} \rfloor = 5]$) par une série d'arêtes, sauf pour les noeuds rouges que notre algorithme n'évalue pas. Pour le noeud $[5, 5]$, une seule structure de coalitions est évaluée, elle correspond aux deux premiers groupes formés après la phase 2. En-

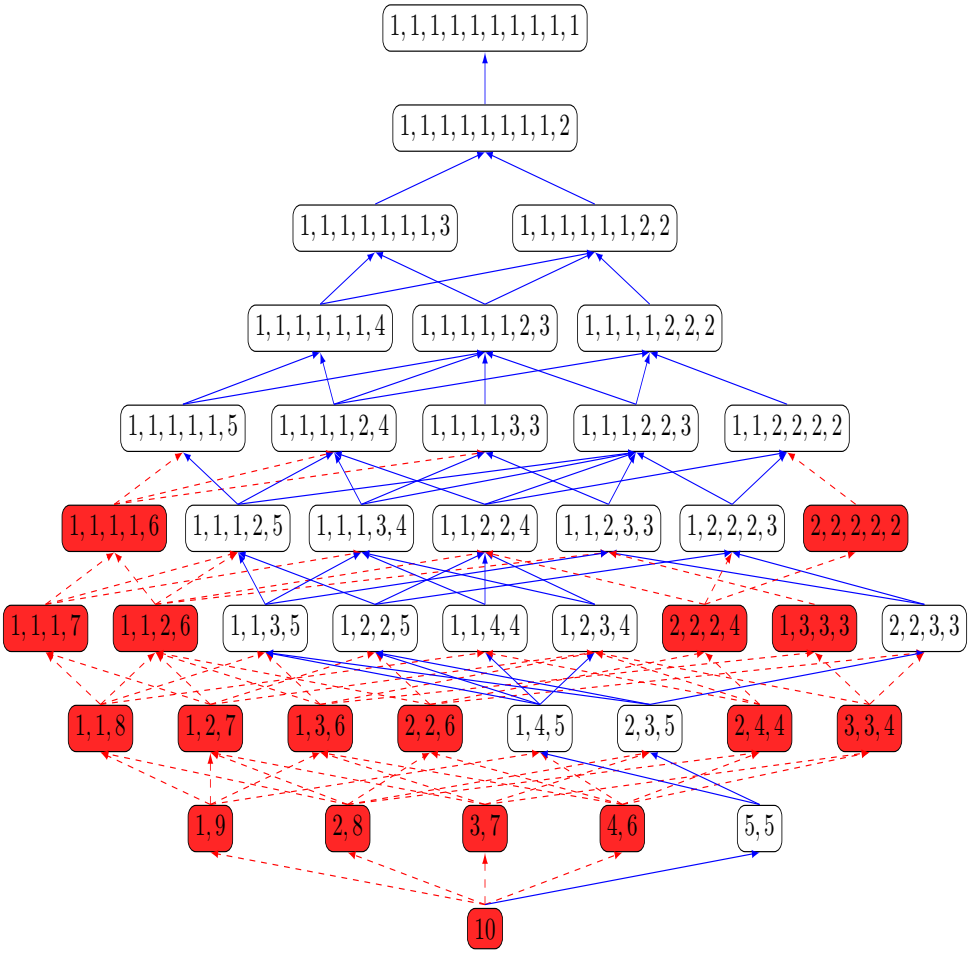


FIGURE 6.2 – Un exemple à dix agents du graphe des partitions. Les noeuds blancs sont partiellement recherchés par notre algorithme alors que les noeuds rouges ne sont pas recherchés par notre algorithme.

suite, pour les sous-espaces de solutions restants, notre algorithme recherche un sous-ensemble de structures de coalitions (en fonction du groupe qui contient chaque agent).

6.4 Technique de répartition des agents

Cette section présente une technique qui peut être utilisée dans la phase 1 de notre algorithme pour réaliser une bonne séparation des agents.

L'idée qui sous-tend notre technique est que pour bien répartir les agents en deux groupes, nous devons nous assurer que chaque groupe formé possède le même potentiel. Par même potentiel, nous entendons que les valeurs des agents (c'est-à-dire les coalitions singleton) dans chaque groupe doivent être équivalentes.

Pour ce faire, les agents doivent être classés en fonction de leurs valeurs. Ensuite, pour les séparer de manière égale, nous procédons comme suit : le premier agent ayant la valeur la plus élevée va dans le deuxième groupe et l'agent ayant la deuxième valeur la plus élevée va dans le premier groupe et ainsi de suite pour le reste des agents.

Par exemple, pour un ensemble de six agents $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ comme dans la figure 6.3. Soit v une fonction caractéristique qui attribue une valeur réelle v(C) à chaque coalition singleton C comme suit : $v(\{a_1\}) = 29, v(\{a_2\}) = 49, v(\{a_3\}) = 35, v(\{a_4\}) = 25, v(\{a_5\}) = 45, v(\{a_6\}) = 39$. Une illustration de la technique de séparation de ces six agents est présentée dans la Figure 6.3.

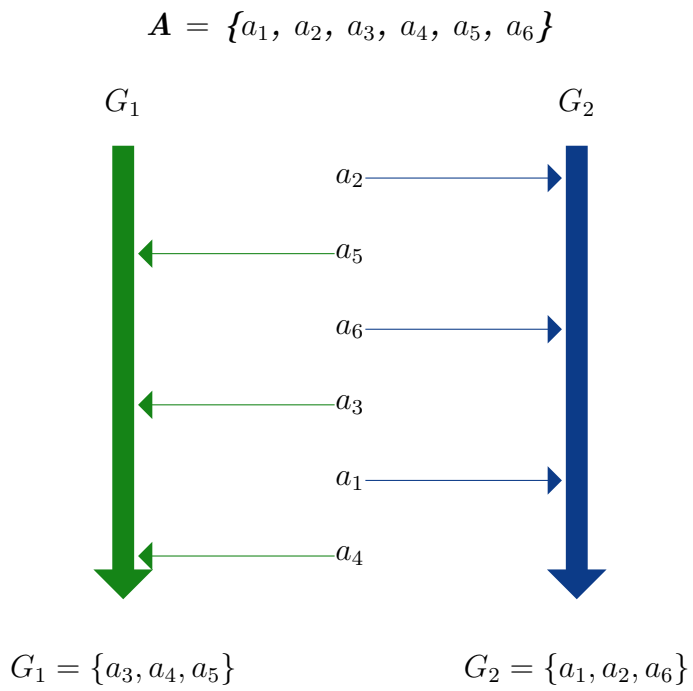


FIGURE 6.3 – Une illustration de la technique de répartition des agents.

6.5 Evaluation de performances

Après avoir décrit toutes les parties de notre algorithme, nous évaluons maintenant son efficacité en comparant la qualité des solutions trouvées par notre algorithme à la qualité de la structure de coalitions optimale.

6.5.1 Performances en termes de qualité de solutions

Il nous est impossible de comparer la qualité des solutions avec des problèmes de plus de 30 ou 40 agents en raison des besoins énormes en mémoire d'un algorithme exact pour obtenir la solution optimale. Au lieu de cela, nous comparons la qualité des solutions de notre algorithme dans des ensembles pour lesquels un algorithme exact est applicable.

La Figure 6.4 montre l'évolution de la qualité de solutions en fonction du nombre d'agents sur différentes distributions de valeurs.

Comme nous pouvons le constater, la qualité de la solution fournie par notre algorithme, sans utiliser un algorithme pour séparer les agents (représenté par **LM**, sur la figure), diffère d'une distribution de valeurs à une autre :

- pour les distributions Agent-based Normal et Normal, la qualité de la solution produite est comprise entre 98 % et 100 %. Avec ces deux distributions, la pire séparation possible des agents résulterait tout de même en une solution dont la qualité dépasserait 93% (ou 97% dans le cas de la distribution Agent-based Normal).

- pour les distributions Agent-based Uniform, Uniform et Beta, la qualité de la solution est supérieure à 86 % (ou 90 % dans le cas bêta).

- pour NDCS, la qualité de solutions est autour de 78%.

La Figure 6.4 montre également la valeur maximale que notre algorithme peut atteindre. Nous pouvons voir ici que la qualité de la meilleure solution pouvant être trouvée par notre algorithme est toujours supérieure à 99,50 %, sauf pour la distribution Agent-based Uniform, où la qualité de la solution est tout de même autour de 95 %. Cela signifie qu'une séparation suffisamment bonne des agents peut conduire à une solution très proche de l'optimum.

6.5.1.1 Efficacité de la technique de séparation

Il est facile de vérifier l'efficacité de la technique de répartition des agents en comparant la qualité des solutions produites avec (**LM** + **T**) et sans la technique de séparation.

Les résultats de la Figure 6.4 montrent clairement que l'utilisation de la technique de séparation améliore les résultats. Cependant, bien que cette technique améliore les résultats, elle ne permet pas d'obtenir une des meilleures séparations possibles. Pour les obtenir, il est peut être nécessaire de stocker les valeurs d'autres coalitions, ce qui augmentera les besoins en mémoire, mais donnera

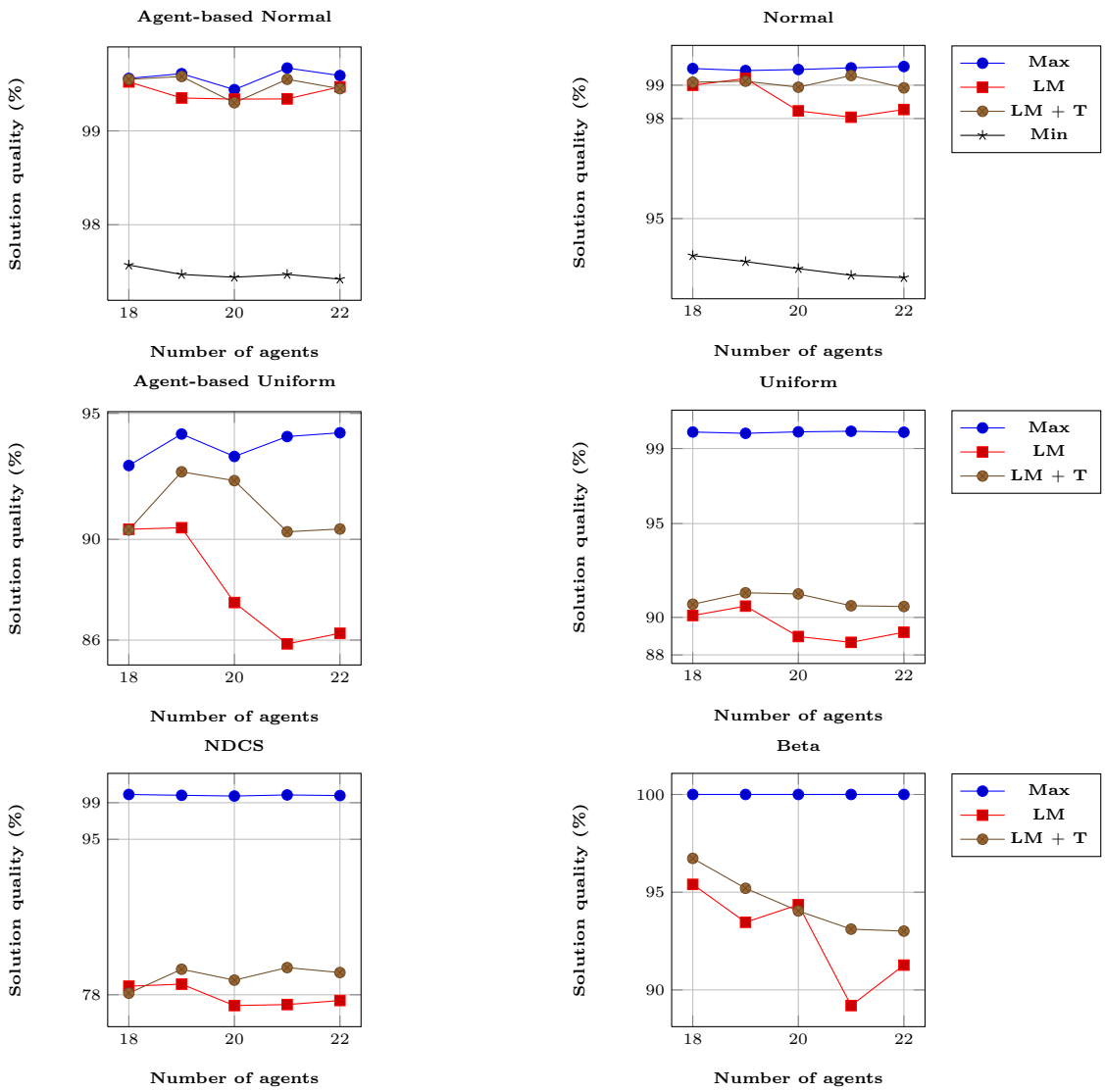


FIGURE 6.4 – Qualité de solutions de notre algorithme.

plus de garantie quant à la qualité des solutions. Néanmoins, cette technique présente une base pour des améliorations futures pour plus de garanties, du fait que les valeurs des coalitions sont générées aléatoirement dans notre étude, appliquées à un problème réel ; les résultats peuvent être meilleurs.

6.5.2 Performances en termes de temps d'exécution

Dans cette section, nous évaluons les performances de notre algorithme en termes de temps d'exécution.

Les deux sous-problèmes de notre algorithme peuvent être exécutés soit séquentiellement, soit en parallèle. Ainsi, le temps d'exécution de notre algorithme pour un problème de taille n correspond à la somme du temps d'exécution d'un problème de taille $\left\lfloor \frac{n}{2} \right\rfloor$ et du temps d'exécution d'un problème de taille $n - \left\lfloor \frac{n}{2} \right\rfloor$ dans le cas séquentiel et il correspond au temps d'exécution d'un problème de taille $n - \left\lfloor \frac{n}{2} \right\rfloor$ au cas où les deux sous-problèmes seraient exécutés en parallèle.

Le tableau 6.1 montre le gain de temps obtenu en utilisant notre algorithme par rapport à l'algorithme ODP-IP. Le gain de temps approche les 100 % dans toutes les distributions.

Distribution	**ODP-IP** (t_1)	**Our algorithm** (t_2)	**Difference** $t_1 - t_2$
Agent-based normal	478.492	0.012	478.480
Agent-based uniform	459.302	0.010	459.292
Uniform	28.465	0.006	28.459
Normal	77.238	0.006	77.232
Beta	29.113	0.005	29.108
NDCS	87.066	0.004	87.062
Modified normal	4.608	0.002	4.606
Modified uniform	69.704	0.003	69.701
Exponential	79.365	0.004	79.361

TABLE 6.1 – Différence de temps entre ODP-IP et notre algorithme en secondes pour 25 agents.

6.6 Conclusion

Dans ce chapitre, nous avons présenté notre troisième algorithme pour la génération de structures de coalitions. Tout d'abord, nous avons décrit quelques problèmes liés aux algorithmes existants, en particulier le problème relatif aux besoins en mémoire qui rend les algorithmes existants difficiles à appliquer. Nous avons ensuite présenté notre algorithme qui ne stock qu'un sous-ensemble de valeurs de coalitions en divisant un problème de taille n en deux sous-problèmes de

taille $\frac{n}{2}$. Les résultats obtenus montrent que notre algorithme est capable de fournir des solutions de haute qualité, tout en minimisant les besoins en mémoire et en assurant un gain de temps avoisinant les 100%, ce qui lui permet d'être idéal pour les problèmes de grande taille.

Conclusion générale et perspectives

La formation de coalitions, forme importante d'interaction dans les systèmes multi-agents, est un processus par lequel un groupe d'agents est réuni pour coordonner et coopérer dans l'exécution d'un ensemble de tâches. Les coalitions ainsi formées peuvent améliorer les performances des agents à titre individuel et/ou du système dans son ensemble, en particulier lorsque les tâches ne peuvent être exécutées par un seul agent ou lorsqu'un groupe d'agents exécute les tâches de manière plus efficace. L'un des problèmes les plus difficiles qui se posent dans le processus de formation de coalitions est celui de la génération de structures de coalitions, qui implique de diviser l'ensemble des agents en coalitions disjointes de manière à maximiser le gain.

Dans ce mémoire, nous avons tout d'abord étudié quelques concepts de base des systèmes multi-agents. Nous avons présenté les notions d'agents, de systèmes multi-agents, et de formation de coalitions d'agents ainsi que ses domaines d'application. Par la suite, nous avons étudié certains travaux de recherche menés pour résoudre le problème de génération de structures de coalitions. Nous avons présenté quelques représentations de l'espace de recherche sur lesquelles les chercheurs se sont basé pour concevoir la plupart des algorithmes existants dans la littérature. Nous avons ainsi constaté que ces représentations ne permettaient pas de différencier entre les structures de coalitions, mais uniquement de les regrouper. Dans ce contexte, nous avons proposé une nouvelle représentation de l'espace de recherche qui permet d'évaluer directement les structures de coalitions sans avoir à parcourir les coalitions qui les composent et ainsi de produire des solutions à tout moment de l'exécution. Notre premier algorithme proposé est basé sur cette nouvelle représentation. En effet, notre heuristique utilise cette représentation pour énumérer partiellement les structures de coalitions et ainsi de parcourir l'espace de recherche de manière efficace. Nous avons ensuite évalué les performances de notre algorithme en le comparant avec l'algorithme le plus rapide pour résoudre le problème de génération de structures de coalitions, ODP-IP. Les résultats ont montré que CSE produit des solutions de haute qualité tout en étant plus rapide que ODP-IP sur les différentes distributions de valeurs. Nous avons ensuite proposé un algorithme imparfait, c'est-à-dire que pour la plupart des cas, notre algorithme fournit la solution optimale. Le principe de cet algorithme est basé sur deux observations sur les valeurs des coalitions. L'analyse de ses performances à montré que l'algorithme n'échoue presque jamais à produire la structure de coalitions optimale et que pour certaines distributions de valeurs, le gain de temps dépasse les 300 secondes pour 25 agents. Notre troisième algorithme est un algorithme conçu spécialement pour les problèmes de grande taille où les besoins en mémoire dépassent la capacité disponible.

Notre algorithme divise un problème de taille n en deux sous-problèmes de taille $\frac{n}{2}$. Les résultats de l'évaluation de performances ont montré que notre algorithme parvient à trouver des solutions quasi-optimales avec un gain de temps avoisinant les 100 %.

En guise de perspectives, un certain nombre d'extensions importantes qui feront l'objet de nos futures recherches peuvent être envisagées. Par exemple, nous pouvons combiner notre représentation de l'espace de recherche avec un autre algorithme de programmation dynamique afin qu'il n'ait pas besoin de parcourir toutes les partitions possibles. Nous envisageons également d'améliorer la qualité des solutions produites par notre troisième algorithme en stockant les valeurs d'autres coalitions, ce qui permettra d'effectuer de meilleures répartitions des agents.

Bibliographie

[1] Adams, J., et al. : Approximate coalition structure generation. In : Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)

[2] Andrews, G.E., Eriksson, K. : Integer partitions. Cambridge University Press (2004)

[3] Arib, S. : Mécanismes de formation de coalitions d'agents dans les processus de planification. Ph.D. thesis, Paris 9 (2015)

[4] Bell, E.T. : Exponential numbers. The American Mathematical Monthly **41**(7), 411–419 (1934)

[5] Changder, N., Aknine, S., Dutta, A. : An imperfect algorithm for coalition structure generation. In : Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 9923–9924 (2019)

[6] Changder, N., Aknine, S., Dutta, A. : Leveraging symmetric relations for approximation coalition structure generation. In : International Conference on Principles and Practice of Multi-Agent Systems. pp. 434–442. Springer (2019)

[7] Dang, V.D., Dash, R.K., Rogers, A., Jennings, N.R. : Overlapping coalition formation for efficient data fusion in multi-sensor networks. In : AAAI. vol. 6, pp. 635–640 (2006)

[8] Dang, V.D., Jennings, N.R. : Generating coalition structures with finite bound from the optimal guarantees. In : Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2. pp. 564–571. IEEE Computer Society (2004)

[9] Faye, P.F.M. : Modèles de formation de coalitions stables dans un contexte ad-hoc et stochastique. Ph.D. thesis (2015)

[10] Ferber, J. : Les systèmes multi-agents : un aperçu général. Techniques et sciences informatiques **16**(8) (1997)

[11] Horling, B., Lesser, V. : A survey of multi-agent organizational paradigms. Knowledge Engineering Review **19**(4), 281–316 (2004)

[12] Karp, R.M. : The probabilistic analysis of combinatorial optimization algorithms. In : Tenth International Symposium on Mathematical Programming. pp. 384–395 (1979)

[13] Keinänen, H. : Simulated annealing for multi-agent coalition formation. In : KES International Symposium on Agent and Multi-Agent Systems : Technologies and Applications. pp. 30–39. Springer (2009)

[14] Klusch, M., Shehory, O. : A polynomial kernel-oriented coalition algorithm for rational information agents. Tokoro, ed pp. 157–164 (1996)

[15] Larson, K.S., Sandholm, T.W. : Anytime coalition structure generation : an average case study. Journal of Experimental & Theoretical Artificial Intelligence **12**(1), 23–42 (2000)

[16] Michalak, T., Rahwan, T., Elkind, E., Wooldridge, M., Jennings, N.R. : A hybrid exact algorithm for complete set partitioning. Artificial Intelligence **230**, 14–50 (2016)

[17] Rahwan, T., Jennings, N.R. : An improved dynamic programming algorithm for coalition structure generation. In : Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3. pp. 1417–1420. International Foundation for Autonomous Agents and Multiagent Systems (2008)

[18] Rahwan, T., Michalak, T., Jennings, N. : A hybrid algorithm for coalition structure generation. In : Twenty-Sixth AAAI Conference on Artificial Intelligence (2012)

[19] Rahwan, T., Ramchurn, S.D., Dang, V.D., Giovannucci, A., Jennings, N.R. : Anytime optimal coalition structure generation. In : AAAI. vol. 7, pp. 1184–1190 (2007)

[20] Rahwan, T., Ramchurn, S.D., Dang, V.D., Jennings, N.R. : Near-optimal anytime coalition structure generation. In : IJCAI. vol. 7, pp. 2365–2371 (2007)

[21] Rahwan, T., Ramchurn, S.D., Jennings, N.R., Giovannucci, A. : An anytime algorithm for optimal coalition structure generation. Journal of artificial intelligence research **34**, 521–567 (2009)

[22] Rothkopf, M.H., Pekeč, A., Harstad, R.M. : Computationally manageable combinational auctions. Management science **44**(8), 1131–1147 (1998)

[23] Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohmé, F. : Coalition structure generation with worst case guarantees. Artificial Intelligence **111**(1), 209–238 (1999)

[24] Sandholm, T.W., Lesser, V.R. : Coalitions among computationally bounded agents. Artificial intelligence **94**(1), 99–138 (1997)

[25] Sen, S., Dutta, P.S. : Searching for optimal coalition structures. In : Proceedings Fourth International Conference on MultiAgent Systems. pp. 287–292. IEEE (2000)

[26] Shehory, O., Kraus, S. : Methods for task allocation via agent coalition formation. Artificial intelligence **101**(1-2), 165–200 (1998)

[27] Tsvetovat, M., Sycara, K. : Customer coalitions in the electronic marketplace. In : Proceedings of the fourth international conference on Autonomous agents. pp. 263–264 (2000)

[28] Yeh, D.Y. : A dynamic programming approach to the complete set partitioning problem. BIT Numerical Mathematics **26**(4), 467–474 (1986)

Résumé

La génération de structures de coalitions est une problématique de recherche importante dans les systèmes multi-agents (SMA). Elle consiste à diviser l'ensemble des agents en sous-ensembles disjoints appelés coalitions, de manière à maximiser le gain. Trouver la structure de coalitions optimale est un problème NP-complet qui est donc difficile à résoudre, même avec des hypothèses assez restrictives. Cela a motivé les chercheurs à développer divers algorithmes déterministes et approches heuristiques pour résoudre efficacement le problème. Dans ce mémoire, nous avons d'abord proposé une nouvelle représentation de l'espace de recherche basée sur le graphe des partitions. Ensuite, nous avons développé une approche heuristique innovante et efficace pour rechercher efficacement l'espace des structures de coalitions afin de trouver une solution proche de l'optimum dans un temps court. Notre technique d'énumération des structures de coalitions (CSE), qui combine une heuristique originale et une nouvelle représentation de l'espace de recherche, offre une nouvelle façon d'explorer le graphe des partitions en calculant directement les structures de coalitions. L'analyse des performances de notre approche montre son efficacité en termes de temps et de qualité des solutions trouvées par rapport à l'algorithme exact le plus rapide, à savoir ODP-IP. Nous avons ensuite développé un algorithme imparfait (ADP) basé sur deux observations sur les valeurs des coalitions. Cet algorithme fournit la solution optimale pour plus de 97% des cas avec un gain de temps pouvant atteindre 300 secondes pour 25 agents. Nous avons ensuite présenté un problème négligé des algorithmes existants, à savoir leurs besoins en mémoire, et proposé notre troisième algorithme pour faire face à ce problème majeur. Notre algorithme fournit des solutions de haute qualité avec un gain de temps qui avoisine les 100 % tout en ne nécessitant que moins de 1 % de la mémoire requise par les algorithmes existants (0,02 % pour 26 agents).

Mots clés : *Génération de structures de coalitions, Formation de coalition, Programmation dynamique, Algorithme imparfait, Graphe des partitions*

Abstract

Coalition Structure Generation (CSG) is an important research issue in Multi-Agent Systems (MAS). It involves partitioning the set of agents into disjoint and exhaustive subsets called coalitions such that the total welfare is maximized. Finding the optimal coalition structure is an NP-complete problem that is therefore computationally challenging, even under quite restrictive assumptions. This has motivated researchers to develop a range of deterministic algorithms and heuristic approaches to solve the CSG problem efficiently. In this paper, we first proposed a novel representation of the search space based on the integer partition graph of coalitions. Then, we developed an innovative and effective heuristic approach to efficiently search the space of coalition structures in order to find an accurate near-optimal one within a short runtime. Our Coalition Structure Enumeration (CSE) technique, which combines an original heuristic and a novel state space representation, provides a new way of exploring the integer partition graph by directly computing the coalition structures and then allowing an effective search. The performance analysis of our approach shows its effectiveness in terms of time and high-quality of the solutions found compared to the fastest exact algorithm for CSG, namely ODP-IP. We then developed an imperfect algorithm (ADP) based on two observations on the coalition values. This algorithm provides the optimal solution for more than 97% of the cases with a time gain that can reach 300 seconds for 25 agents. We then presented one neglected problem with existing algorithms which is their memory requirements and proposed our third algorithm to face this major problem. Our algorithm provides high-quality solutions with a time gain that nears 100% while requiring only less than 1% of the memory required by existing algorithms (0.02% for 26 agents).

Keywords : *Coalition structure generation, Coalition formation, Dynamic programming, Imperfect algorithm, Integer partition graph*