Mickaël Bettinelli, Michel Occello, Damien Genthial

ABSG : une architecture d'agents d'inspiration sociale pour le problème de formation de coalitions

Volume 4, nº 2 (2023), p. 9-40.

DOI not yet assigned

La Revue Ouverte d'Intelligence Artificielle est membre du Centre Mersenne pour l'édition scientifique ouverte
www.centre-mersenne.org
e-ISSN : 2967-9672

Revue Ouverte d'Intelligence Artificielle
Volume 4, n° 2, 2023, 9-40

ABSG : une architecture d'agents d'inspiration sociale pour le problème de formation de coalitions

Mickaël Bettinelli[a], Michel Occello[b], Damien Genthial[b]

[a] Université Savoie Mont Blanc, LISTIC, Annecy, France
E-mail : mickael.bettinelli@univ-smb.fr

[b] Grenoble Alpes University, LCIS, 26000 Valence, France
E-mail : michel.occello@univ-grenoble-alpes.fr, damien.genthial@lcis.grenoble-inp.fr.

Résumé. — Nous proposons une nouvelle architecture d'agent d'inspiration sociale adaptée à un système d'aide à la décision pour résoudre un problème de génération de structures de coalitions distribué avec chevauchements pour la conception de produits dans le cadre de l'économie circulaire. Cette architecture centrée agent permet aux agents de savoir avec quelles accointances former une coalition de manière à designer des produits répondant au mieux à un besoin utilisateur. Le mécanisme cognitif utilisé par l'architecture ABSG s'inspire des principes de l'attraction des sciences humaines et sociales.

Mots-clés. — Formation de coalitions, système multi-agent, architecture d'agent.

1. Introduction

L'économie circulaire se concentre sur la réduction des déchets à travers la mise en place de stratégies de réutilisation dans un système de production. Le programme transdisciplinaire CIRCULAR (ANR-15-IDEX-02) est un projet qui développe les technologies et les conditions nécessaires pour rendre les nouveaux systèmes industriels circulaires capables de transformer des produits en fin de vie en nouveaux produits. Dans [4], nous avons introduit un nouveau système d'aide à la décision rendant possible l'économie circulaire à travers l'usage de stratégies de réutilisation telles que le *remanufacturing* et le *repurposing*. Pour cela, le système d'aide à la décision regroupe des composants permettant de concevoir les produits requis par un utilisateur. Le système dépend d'un ensemble de composants variables en nombre et en caractéristiques (par exemple, le prix, les dimensions, la forme, etc.). Il doit être capable de gérer ces caractéristiques liées aux composants usés afin de mettre ces derniers en relation pour concevoir un produit. Cependant, il ne s'agit pas d'optimiser un ensemble de quantités fixes, mais de faire émerger de nouvelles structures en nombre variable à partir des caractéristiques de chaque composant. Ainsi, le système peut présenter

plusieurs solutions et les composants peuvent participer à plusieurs conceptions de produits simultanément, éventuellement en conflit lorsque les composants utilisés ne sont pas suffisamment nombreux pour toutes les fabriquer. En outre, certaines solutions peuvent être incomplètes ou de niveaux de qualité différents. Le système est dynamique, ouvert, à grande échelle et ses dynamiques dépendent du comportement intrinsèque des composants, de leur capacité à évoluer avec les connaissances expertes injectées (caractéristiques physiques, avec qui ils peuvent ou non s'assembler, etc.) et des interventions des fournisseurs de produits sur le système (changement de prix d'un produit, modification de sa qualité, etc.).

Il est alors complexe d'identifier un état global du système et d'en déduire un modèle de résolution global. Le problème est donc de décomposer de manière adéquate la modélisation d'un système composé d'une multitude d'agents autonomes. Nous nous tournons naturellement vers une méthode de résolution de problèmes incrémentale régie par une optimisation continue de l'interaction entre les éléments locaux : les systèmes multi-agents. Pour ce faire, nous avatarisons les composants en agents virtuels [25]. L'objectif des agents est décidé par l'utilisateur lorsqu'il demande au système de concevoir un produit. Ce faisant, il donne aux agents leurs désirs. Dans ce travail, nous définissons un désir comme étant une caractéristique recherchée par un agent chez un autre agent (*e.g.* un agent peut souhaiter faire partie d'une coalition avec des accointances possédant un faible prix). Ce problème de formation de groupe est appelé dans la littérature « problème de formation de coalitions » (CF) [30]. Comme les agents sont des entités autonomes, nous les considérons comme des humains qui ont des désirs, des croyances et des objectifs. C'est pourquoi nous nous inspirons des sciences humaines et sociales (SHS) et de la dynamique de groupe pour aborder notre problème de CF. Cependant, il s'agit d'un vaste problème comportant de nombreuses variantes. Comme nous le verrons dans les sections suivantes, nous nous concentrons plus spécifiquement sur le problème de la génération des structures de coalitions (CSG).

Nous présentons ABSG (Attraction Based Structures Generation), une architecture d'agent d'inspiration sociale conçue pour traiter un problème de formation de coalitions décentralisé dans un système ouvert et variable. La partie 2 fait une brève présentation de la littérature sur le sujet. La partie 3 définit le problème. La partie 4 présente les propriétés et la prise de décision de notre architecture d'agent. Nous présentons l'opérationalisation de notre architecture dans la partie 5, les métriques d'évaluation dans la partie 6 et discutons de nos résultats expérimentaux dans la partie 7.

2. Revue de littérature

Les approches classiques de l'art sont peu adaptées à un problème de CSG ouvert et variable. Nombre d'entre elles sont très efficaces pour trouver des solutions optimales dans des systèmes composés de peu d'agents ou encore pour trouver des solutions approximatives sur des systèmes à grande échelle dans des ensembles fixes d'agents. Il n'existe cependant pas à notre connaissance de méthodes pour répondre à notre problème de CSG ouvert, variable avec un partitionnement chevauchant des coalitions.

Or, contrairement à ces méthodes, la grande variabilité de notre système ne nous permet pas de considérer un ensemble d'agents fixe.

Nous détaillons ci-dessous, de manière non exhaustive, les approches et méthodes les plus communes se rapprochant de notre problématique.

2.1. Clustering

Par définition, une coalition est par nature orientée par un objectif. Elle est dynamique et son existence est de courte durée. En revanche, les sous-ensembles de données construits par un algorithme de *clustering* n'ont pas vocation à la dynamicité. Les données sont le plus souvent fixes et les groupes ne sont pas construits pour répondre à un objectif mais simplement de manière à regrouper des données similaires.

Cependant, étant donné la ressemblance de ces approches, certain travaux essaient de faire le pont entre le partitionnement de données et la formation de coalitions [10]. Farinelli *et al.* [11] proposent un algorithme de formation de coalition basé sur une approche de partitionnement hiérarchique. Cet algorithme, nommé Coalition Link (C-LINK), permet de résoudre un problème de formation de coalitions à grande échelle (jusqu'à 2700 agents) de manière non optimale en un temps raisonnable. La fonction d'association utilisée est basée sur le gain, défini comme la différence entre la valeur d'une coalition composée des agents impliqués et la somme de leurs valeurs individuelles.

Les agents forment ou rejoignent des coalitions uniquement si leur gain est positif. Les résultats montrent que la méthode est efficace lorsque les agents sont fortement connectés les uns aux autres mais qu'elle devient peu performante comparée à des algorithmes *branch and bound* [28] lorsque les agents sont peu connectés entre eux. Ce résultat s'explique par le fait que lorsque les agents ont peu de connexions les uns avec les autres, les algorithmes *branch and bound* travaillent sur un plus petit espace de recherche et peuvent alors trouver de meilleures solutions. De plus, la méthode étant basée sur un algorithme de clustering hiérarchique, elle rend les membres d'une coalition incapables de réévaluer leur gain à rester dans la coalition. Si un agent devait entrer dans le système ou s'il devait être modifié, une approche de clustering hiérarchique ne permettrait pas de prendre en compte le nouvel état du système pour adapter les coalitions. Cette méthode ne peut donc pas être utilisée dans un système ouvert ni être utilisée avec des agents variables.

2.2. Intelligence distribuée

Dans l'intelligence distribuée, la coordination des agents revêt différentes formes, l'objectif peut être la formation d'essaims stables dans lesquels la position de chaque agent n'est pas connue à l'avance [3] et converge vers une position finale, ou alors la mise en place par les agents de formations préconçues par le concepteur du système. Par exemple, dans [14], Veysel Gazi utilise une méthode de contrôle non linéaire pour faire en sorte que les agents de l'essaim forment une figure géométrique.

Le travail [35] utilise des agents homogènes pour former un unique essaim. Un champ de potentiel, défini sous forme d'une fonction d'attraction/répulsion régit les interactions que les agents ont entre eux et permet la stabilisation de l'essaim. Mais l'intelligence distribuée se concentre aussi sur la formation de plusieurs essaims constitués d'agents hétérogènes. Ce concept est très proche de la formation de coalitions sans chevauchement. Kumar *et al.* présentent une méthode parvenant à doter les agents d'un essaim d'un comportement ségrégatif [19]. De la même manière que Shi *et al.* dans [35], leur méthode utilise un champ de potentiel appliqué sur chaque agent permettant de les agréger en plusieurs essaims stables. Cependant Kumar *et al.* s'inspirent de la ségrégation de cellules biologiques pour que les agents de leur système soient capables de se diviser en plusieurs essaims. Pour cela, les agents sont dotés d'un type, par exemple A et B. La fonction de potentiel utilisée par les agents prend en compte la distance entre les types des agents tel que, par exemple, $d^{AA} = d^{BB}$ mais $d^{BB} < d^{AB}$. Cette distance permet ensuite d'adapter la portée du champ d'attraction et de répulsion en fonction du type des agents dont on calcule le mouvement.

Santos *et al.* étendent le travail de Kumar *et al.* pour faire de la ségrégation à plus grande échelle avec 150 agents et 15 types [34]. Les auteurs notent que la fonction de potentiel de [19] peut faire tendre les agents vers un minimum local les empêchant parfois d'atteindre leur groupe. Ils modifient l'équation de Kumar *et al.* en y ajoutant un terme quadratique permettant de supprimer le minimum local. Sa suppression permet ensuite aux agents de continuer leur mouvement vers leur groupe. Inácio *et al.* utilisent une stratégie inspirée de l'optimisation par essaims particulaires (PSO) pour la ségrégation d'agents hétérogènes [16]. De la même manière que dans [19], les agents possèdent un type leur servant à identifier les individus avec lesquels former un essaim. Mais contrairement à l'étude de Kumar *et al.*, les agents ont une portée de capteurs limitée. ORCA [36] est un algorithme permettant à une multitude d'agents mobiles de se déplacer dans un espace en évitant les collisions. Cette stratégie est très similaire à [19] mais montre qu'il est possible de l'utiliser pour un passage à plus grande échelle avec jusqu'à 150 agents et 15 coalitions.

Pour résumer, la ségrégation d'agents hétérogènes consiste principalement à spatialement séparer en plusieurs d'essaims un ensemble d'agents dont le type est connu à l'avance. Cependant, elle semble limitée si l'on souhaite créer une multitude d'essaims sur la base de multiples critères. Bien que similaire, notre problème de formation de coalitions ne consiste pas à trier des agents selon un type mais selon un ensemble de caractéristiques qui semblent difficilement transposables en un unique type. Nos agents sont beaucoup plus riches, ils possèdent une multitude de caractéristiques et de désirs et chaque agent doit adapter ses dynamiques en fonction des spécificités des autres agents. En outre, les travaux de ce domaine utilisent des environnements situés sur lesquels se déplacent les agents. Un tel environnement ne serait pas adapté à notre cadre applicatif qui n'a pas ce besoin de situer des agents, mais plutôt de les faire communiquer sans aucune notion d'espace.

2.3. Choix social informatique

Le problème de formation de coalitions du choix social est vu comme un problème d'assortiment (*matching*) dans lequel on peut affecter des agents à d'autres agents mais aussi des agents à des ressources. Cette seconde alternative est très similaire au problème d'allocation de tâches [1]. Le problème de l'assortiment est introduit par David Gale et Lloyd Shapley [13]. Dans ce travail les auteurs proposent un algorithme de mariages stables dans lequel l'objectif est de regrouper des hommes et des femmes deux à deux de manière à respecter au mieux les préférences de tous. Le problème est étendu par le problème hôpitaux/résidents [7, 22]. Il est similaire à celui des mariages stables. Cependant, ce n'est plus un problème liant deux éléments de deux ensembles, mais un élément d'un ensemble à plusieurs éléments d'un autre ensemble. Ce problème peut être rapproché de celui de la formation de coalitions où les hôpitaux seraient des coalitions d'un nombre d'individus prédéfini. Contrairement au problème classique de formation de coalitions, ce ne sont pas les membres qui veulent s'associer à d'autres membres mais les groupes qui possèdent des préférences sur les futurs membres.

Delorme *et al.* étudient une extension du problème hôpitaux/résidents dans laquelle les médecins peuvent postuler en couple à des hôpitaux [9]. Cette contrainte rapproche le problème hôpitaux/résidents du problème de formation de coalitions. Il permet aux médecins de décider de rejoindre une organisation sous condition d'être avec une personne spécifique. Une autre version du problème, appelée travailleurs/entreprises, consiste à rendre possible la multiple assignation des individus à des organisations. Chaque organisation possède des membres faisant aussi partie d'autres organisations. Le problème se rapproche alors d'un problème de formation de coalitions avec chevauchement des coalitions [18].

Pour conclure, les différentes versions du problème de formation de coalitions de cette discipline montrent quelques différences avec celle sur laquelle nous travaillons. Dans notre version, les groupes sont construits autour des besoins des agents et non de ceux d'une coalition comme c'est le cas dans le problème hôpitaux/résidents ou travailleurs/entreprises.

2.4. Théorie des jeux coopératifs

Dans les jeux coopératifs classiques les joueurs forment le plus souvent des coalitions disjointes en coopérant uniquement avec les membres de leur coalition. Dans [33], les auteurs proposent un algorithme d'allocation de tâches pour résoudre un problème dans lequel des agents doivent collecter des données auprès de services. Ces services peuvent par exemple fonctionner sur des appareils embarqués. L'objectif est de former des coalitions d'agents et de services afin de collecter leurs données en fonction des préférences des joueurs. Les agents peuvent servir de relais pour assurer la transmission des données ou de collecteurs qui se déplacent de service en service pour récupérer les informations. Les joueurs (agents et services) forment des coalitions et les adaptent en fonction de leurs préférences pour les autres agents ou services jusqu'à atteindre un

équilibre de Nash. Afin de gérer des changements dans l'environnement comme le déploiement d'un nouveau service ou la suppression d'un ancien, l'algorithme recalcule à partir de zéro une nouvelle structure de coalitions régulièrement.

Dans certaines situations les joueurs peuvent avoir besoin de participer à plusieurs coalitions [2]. [39] se concentre sur la formation de coalitions avec chevauchement. Ils présentent une formalisation pour modéliser et analyser des communications dans un scénario d'allocation de ressources radio. Dans le cas où des agents peuvent participer à plusieurs coalitions, ceux-ci n'ont plus besoin de quitter leur ancien groupe pour en rejoindre un nouveau. La notion de stabilité doit donc être adaptée. Pour cette raison, les auteurs proposent une nouvelle notion de la stabilité dans laquelle les joueurs quittant une coalition peuvent garder leur récompense si les membres restants ne la quittent pas non plus. L'algorithme des auteurs calcule pour chaque structure de coalition stable la valeur maximale que les joueurs peuvent générer. La complexité de cet algorithme est $O(N^s)$, S étant le nombre de mouvements possibles que les joueurs peuvent faire (sortir ou rentrer dans une coalition) et N le nombre de joueurs.

Les travaux de formation de coalitions dans les jeux coopératifs se concentrent sur la recherche d'un équilibre lors de la création des coalitions. L'équilibre est atteint lorsque les préférences des membres des coalitions sont satisfaites. L'approche des jeux coopératifs a le même inconvénient que le choix social : elle est peu adaptée à un cas applicatif changeant. Bien que les auteurs de [33] prennent en considération la possibilité d'un environnement ouvert, leurs agents ne se réorganisent pas depuis la dernière structure de coalition trouvée. Le système recherche les coalitions comme s'il le faisait pour la première fois ce qui pourrait être inadapté à un passage à l'échelle.

2.5. Programmation dynamique

Le premier algorithme de programmation dynamique pour un problème de partitionnement d'un ensemble a été proposé par [40]. L'objectif de ce problème est d'agréger des entités de manière à ce que chacune d'entre elles soit exactement dans un groupe tout en minimisant le coût des agrégations. On peut visualiser l'ensemble des solutions à ce problème comme un graphe dans lequel chaque nœud est un ensemble de coalitions possédant une valeur et chaque liaison indique que l'on peut passer d'un nœud à l'autre en effectuant uniquement une modification sur les solutions liées (fusion ou séparation d'une coalition par exemple). L'algorithme de Yeh, nommé DP, évalue toutes les divisions de coalitions possibles afin de déterminer quelles coalitions valent le coût d'être séparées. Cet algorithme a une complexité $O(3^n)$, n étant le nombre d'entités à traiter, ce qui le rend inutilisable dans une application à grande échelle. De plus, la superposition des coalitions ne rentre pas dans le cadre de ce travail. Si elle était prise en compte, elle augmenterait significativement l'espace de recherche et rendrait cet algorithme encore plus difficilement exploitable. IP [31], algorithme exact et *anytime* permet de résoudre le CSG calculant les frontières de l'espace de recherche afin de le réduire (*prune*) et limiter sa taille. IP a une complexité plus élevée que celle de DP, $O(n^n)$, mais présente tout de même un temps moyen de résolution du problème

de partitionnement plus court grâce à sa meilleure capacité à réduire les espaces de recherche infructueux.

Michalak *et al.* [23] présentent un algorithme exact et *anytime* appelé ODP-IP pour la résolution du problème de CSG. ODP-IP est l'algorithme exact le plus rapide à ce jour pour résoudre un problème CSG. Il s'appuie sur les algorithmes DP et IP. ODP-IP développe une nouvelle représentation de l'espace de recherche permettant à DP et IP de fonctionner ensemble. Une nouvelle version de DP est créée, appellée IDP, capable de chercher des solutions dans un graphe de partitions en nombres entiers. Le mélange de ces deux algorithmes permet à ODP-IP de possèder les propriétés d'IDP et d'IP, il est anytime, possède la même complexité que DP et est en moyenne aussi rapide que le plus rapide des deux algorithmes. ODP-IP a donc une complexité de $O(3^n)$, n étant le nombre d'agents à traiter. Pour Changder *et al.* [8], une des principales limites d'ODP-IP est son parcours de l'espace de recherche. Les auteurs montrent que dans de nombreux cas, ODP-IP cherche des solutions deux fois dans le même espace à cause des méthodes très différentes de fonctionnement sur lesquelles se basent IDP et IP.

Changder *et al.* [8] proposent l'algorithme hybride, nommé ODSS, pour la résolution du problème de CSG. ODSS s'inspire d'ODP-IP et essaie de limiter le chevauchement produit lors du parcours de l'espace de recherche. Sa complexité dans le pire cas est le minimum entre les complexités de ces deux algorithmes, à savoir $O(3^n)$. L'objectif de ce travail étant de combler les limitations d'ODP-IP, les évaluations d'ODSS comparent les vitesses d'exécution de ces deux algorithmes en utilisant différentes distributions de valeurs associées aux coalitions. Les évaluations montrent qu'ODSS est plus rapide qu'ODP-IP sur de nombreuses distributions.

Ces approches sont très peu dynamiques et ne supportent pas l'ouverture. Par exemple, le changement de valeur d'une seule coalition dans les méthodes de programmation dynamique entraînerait la modification de l'espace de recherche et obligerait l'algorithme à chercher la solution optimale dans un tout nouvel arbre. Le temps d'exécution de ces algorithmes étant exponentiel sur le nombre d'agents, cette méthode n'est pas adaptée à un système d'aide à la décision variable et fonctionnant à grande échelle.

2.6. Architectures d'agents et multi-agents

La méthode SACF [32] ne se revendique pas être une architecture d'agent mais possède un fonctionnement relativement proche. Les agents se servant de la méthode SACF (*Self-Adapting Coalition Formation*) utilisent des heuristiques pour déterminer comment former des coalitions dans leur système. Leur approche est proche de celle que nous présentons dans ce papier puisqu'elle permet de former des coalitions dans un environnement ouvert, mais elle ne prend pas en compte un la variabilité des agents qui composent le système. Nous reviendrons sur cette méthode dans la section 7 pour comparer nos résultats aux leurs.

L'architecture multi-robot RACHNA [37] s'attaque à un problème d'allocation de tâche, relativement proche du problème de génération de structures de coalitions que

nous abordons dans notre travail. Contrairement aux architectures d'agents, l'architecture multi-agent (ou multi-robot) décrit la structure du système multi-agent ainsi que les interactions des agents y participants. Le système RACHNA a pout but d'allouer des tâches à des robots en les faisant enchérir pour une tâche en fonction des services qu'ils peuvent effectuer. Les robots ayant envoyé les meilleurs enchère pour une tâche sont sélectionnés pour la réaliser.

Le système multi-agent présenté dans [26] permet de faire de la composition de service. Les agents sont un moyen d'améliorer l'adaptabilité et la flexibilité des solutions de composition de services. Pour cela, les agents sont divisés en deux catégories, des agents utilisateurs et des agents services. Ceux-ci coopèrent à travers un tableau noir où les agents utilisateurs peuvent déposer des requêtes de composition de service. Les agents services peuvent ensuite s'apparier pour former le service souhaité.

Les méthodes que nous avons présentées ont l'avantage de résister à un environnement dynamique dans lequel on peut ajouter ou retirer des agents. Par exemple, les agents SACF peuvent adapter leurs coalition en fonction des agents présents de manière incrémentale. Ces approches sont également adaptées à la représentation d'agents complexes, comme pour les agents du système RACHNA qui possèdent un ensemble de capacités propres à chacun.

2.7. Synthèse

Dans la majorité des disciplines de cette revue de littérature, les méthodes que nous avons présentées sont très peu dynamique et souvent peu adaptées à un problème nécessitant l'ouverture du système. De plus, La notion d'agent est très proche de celle de la donnée. Un agent est presque uniquement vu comme un identifiant auquel est associé une valeur lui permettant de chercher une coalition minimisant ou maximisant un score. Ce manque d'exprimabilité rend difficile la modélisation d'un objet plus complexe. Notre système d'aide à la décision doit former des groupes de composants décrivables par des caractéristiques physiques (*e.g.* poids, forme, prix, etc.) mais aussi par des règles fonctionnelles (*e.g.* quels sont les composants assemblables). Au contraire, l'approche multi-agent rend possible la modélisation d'un système dynamique et semble adapté à notre cadre applicatif.

La partie suivante définit notre problème de formation de coalition et la façon dont le système multi-agent fonctionne à travers une étude de cas minimaliste.

3. Définition du système multi-agent

Pour résoudre notre problème de formation de coalitions, nous considérons un ensemble d'agents ABSG dans un système multi-agent ouvert où les agents peuvent entrer ou sortir à tout moment. Les agents ont leurs propres caractéristiques et désirs qui peuvent également être modifiés à tout moment. Au lancement du système, chaque individu ignore les caractéristiques et les désirs des autres agents. Leur objectif est de former des coalitions avec les agents qui répondent le mieux à leurs désirs. Les agents

n'ont pas conscience de l'objectif global du système, par exemple former une batterie électrique. Mais chaque agent a son propre objectif individuel qui est un sous-ensemble de l'objectif global. Par exemple, un agent *cellule* de batterie souhaite s'apparier avec un agent *module de cellules*. Décomposer l'objectif global en objectifs individuels a pour avantage de réduire l'espace de recherche d'appariements de chaque agent et ainsi réduire le nombre de messages émis dans le système.

Pour former des coalitions, les agents peuvent interagir et échanger leurs caractéristiques et leurs désirs. Ils utilisent ces informations pour évaluer leurs accointances en calculant une valeur d'attraction. L'attraction représente la mesure dans laquelle un agent est attiré par une de ses accointances en fonction de plusieurs facteurs (le traitement de l'attraction est expliqué dans les parties suivantes). Plus la valeur est faible, plus l'attrait est important. Les agents ont la possibilité de former, de rejoindre ou de quitter des coalitions en fonction de la valeur d'attraction qu'ils ont pour les autres agents et pour leurs coalitions.

Dans le problème de CSG, nous considérons un ensemble d'agents N tel que Agents $= \{a_1, \ldots, a_n\}$ dans un système ouvert où les agents peuvent entrer ou sortir à tout moment. Soit CS l'ensemble des coalitions réalisées par les agents de telle sorte que $CS = \{Co_1, Co_2, \ldots, Co_n\}$. Une coalition Co_i est définie comme un tuple : $Co_i = \langle i, A, s\rangle$ où i est l'identifiant du groupe, $A \subset Agents$ et $|A| \geqslant 2$ et s est un agent leader avec $s \in A$. Un leader est un coordinateur de coalition qui donne l'autorisation aux autres membres de la modifier lorsque ceux-ci veulent proposer à un membre extérieur de la rejoindre ou s'ils souhaitent la quitter. La représentation des coalitions n'étant pas centralisée, son rôle permet aux membres de maintenir une représentation correcte de leur coalition. C et D sont des ensembles de caractéristiques et de désirs tels que $C = \{c_1, c_2, \ldots, c_m\}$, $D = \{d_1, d_2, \ldots, d_m\}$ et $|C| = |D|$ car une caractéristique c_i est la satisfaction d'un désir d_i.

Chaque agent est défini comme un tuple : $a = \langle C^a, D^a, Acq^a, V^a, Coal^a, V^{ac}\rangle$ où :

- C^a est un ensemble de caractéristiques : $C^a = \{c_1^a, c_2^a, \ldots, c_n^a\}$ tel que $C^a \subset C$;
- D^a est un ensemble de désirs : $D^a = \{d_1^a, d_2^a, \ldots, d_n^a\}$ appartenant à l'agent tel que $D^a \subset D$;
- Acq^a est une liste d'agents. Elle représente ses accointances ;
- V^a est la liste des valeurs d'attraction liées à ses accointances telle que la valeur d'attraction pour l'accointance Acq_i^a est V_i^a et $V_i^a \geqslant 0$;
- $Coal^a$ est la liste des coalitions auxquelles l'agent appartient ;
- V^{ac} est la liste des valeurs d'attraction liées à ses coalitions de sorte que la valeur d'attraction pour la coalition $Coal_j^a$ est V_j^{ac} et $V_j^{ac} \geqslant 0$.

Les agents choisissent de former ou de rejoindre une coalition avec d'autres agents qui répondent à leurs désirs de telle sorte que $C^j \subset D^i$. Les agents sont égoïstes, ils essaient d'atteindre leur propre but. Nous partons du principe que la connaissance des agents est exacte, que les communications ne peuvent pas être perturbées et que les agents sont dans un réseau entièrement connecté.

À titre d'illustration, la figure 3.1 montre un exemple de l'auto-organisation attendue du système multi-agents (SMA) en fonction d'un produit demandé dans le cas d'une batterie électrique. Nous pouvons voir une description structurelle d'une batterie électrique comme produit demandé en haut à gauche de la figure. Les caractéristiques du produit sont indiquées en dessous. Chaque agent obtient de la demande de l'utilisateur son objectif individuel, par exemple un *système de refroidissement* a le désir de s'associer avec un *boitier* d'une taille spécifique. Sur le côté droit de la figure est montrée une vue du SMA. Elle montre deux groupes déjà constitués. Le groupe en trait plein manque un *boitier*. L'agent *packaging* cherche à combler ce besoin en connectant et en ajoutant à son groupe le *boitier* le mieux adapté à la description du produit demandé. Un autre groupe s'est formé en parallèle, on peut voir que le groupe en pointillés est un groupe complet.

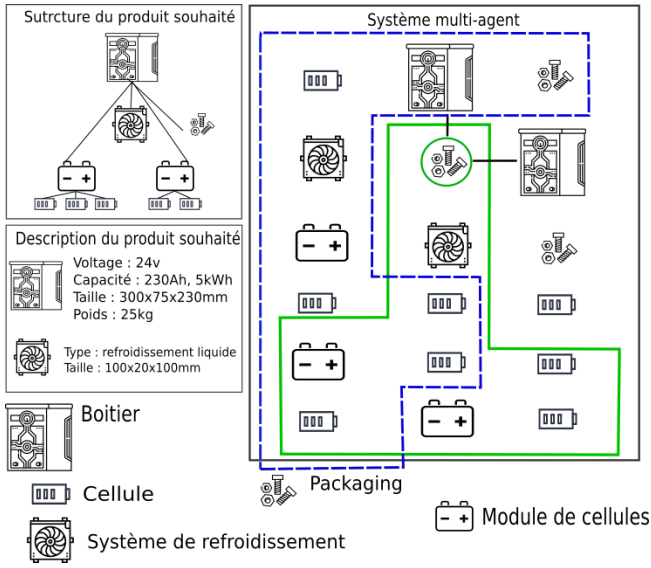


FIGURE 3.1 – Exemple de génération de structures

4. ARCHITECTURE D'AGENT

ABSG est une architecture basée sur Soar [20], une architecture d'agent cognitif possédant déjà certain modules dont nous avons besoin pour ABSG tel que des mémoires et un module décisionnel. La figure 4.1 montre un aperçu de l'architecture d'agent ABSG, composée de trois principaux modules : (1) les mémoires, (2) le module social et (3) le module décisionnel. Les sections 4.1, 4.2 et 4.3 détaillent ces modules. Les agents ABSG fonctionnent à plusieurs, ils ont besoin d'un protocole d'interaction pour se comprendre. Les actions représentent les messages qu'un agent peut envoyer. Les perceptions sont les messages reçus des autres agents. La partie 4.4 explique comment les agents interagissent grâce au module de communication.

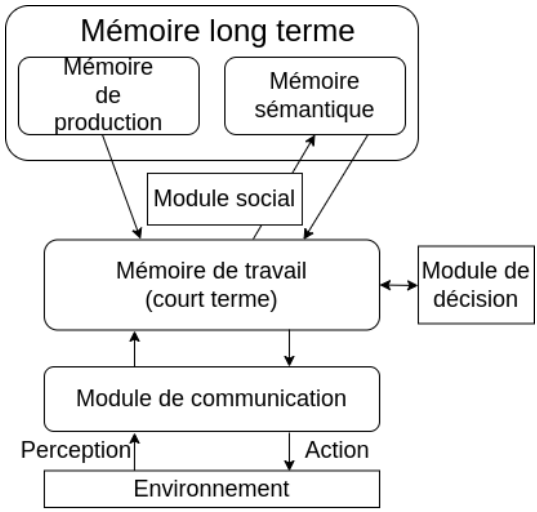


FIGURE 4.1 – Architecture ABSG

4.1. MÉMOIRES

Le système d'aide à la décision fonctionne dans un contexte très variable où les agents, et leur nombre, peuvent être modifiés à tout moment. Notre architecture doit être capable de prendre en compte les modifications concernant l'agent lui-même mais aussi celles de ses accointances. Afin de prendre en compte leur propre état et celui de leurs accointances, les agents ABSG doivent pouvoir construire une représentation d'eux-mêmes et de leur environnement. Dès lors, l'architecture ABSG doit avoir des capacités de représentation des désirs et des caractéristiques de l'agent qui l'implémente ainsi que ceux et celles de ses accointances (*e.g.* comment les contacter, quelles sont leurs caractéristiques, etc.). Un agent ABSG doit aussi être capable d'utiliser ces informations lorsqu'il effectue une action, par exemple envoyer un message à une accointance. Ces connaissances peuvent être stockées dans trois types de mémoires :

- une mémoire de production : c'est une mémoire à long terme stockant les règles de prise de décision des agents ABSG ;
- une mémoire sémantique : c'est une mémoire à long terme stockant des faits sur l'environnement de l'agent. Elle sert par exemple aux agents ABSG à mémoriser les caractéristiques de leurs accointances ainsi que l'avis qu'ils en ont. La mémoire sémantique se met à jour via la mémoire de travail à chaque réception d'un nouveau message.
- une mémoire de travail : c'est une mémoire à court terme stockant les variables temporaires utilisées par le module de prise de décision. Elle est par exemple utilisée pour charger et interpréter les messages envoyés par les autres agents.

Ces trois types de mémoires permettent aux agents de stocker la représentation de leur environnement et de la prendre en compte pour évaluer et générer des coalitions

avec les meilleures accointances. La section suivante définit la métrique que les agents ABSG utilisent pour évaluer une accointance.

4.2. Module social

4.2.1. *Évaluer les accointances*

En plus de la représentation de la mémoire des connaissances, les agents ABSG doivent interagir avec leurs pairs et déterminer quels agents sont les plus appropriés pour former un groupe. Le rôle du module social est de déterminer l'*attraction* que les agents ont les uns pour les autres. Il utilise les connaissances de la mémoire sémantique sur les accointances d'un agent pour calculer et mettre à jour une valeur d'attraction associée à chacun d'eux. L'attraction utilisée par nos agents est dérivée d'une métrique appelée *attraction interpersonnelle* dans les sciences sociales qui quantifie combien un agent est attiré par une de ses accointances. Elle peut être vue comme une distance entre deux agents : plus la distance est petite, plus les agents sont attirés l'un par l'autre. Cependant, cette distance est souvent asymétrique, un agent peut être fortement attiré par l'une de ses accointances alors que l'autre ne l'est pas du tout. Dans une certaine mesure, cette métrique d'attraction est une opérationnalisation de l'attraction interpersonnelle des sciences sociales pour notre problème.

Dans *Les dynamiques de groupes*, le sous-domaine de la *formation de groupes* se concentre sur les processus qui génèrent des liens d'attraction entre les membres des groupes. Le processus de formation de groupes est un phénomène complexe qui implique de nombreuses dimensions. Parmi ces dimensions, les principes d'attraction jouent un rôle important [12, 15, 27]. Comme nous le verrons dans la section 5, les désirs et caractéristiques des agents sont représentés sous forme de décimaux.

- principe de similarité (s) : les individus aiment les personnes qui leur ressemblent [38]. Dans notre système, la similarité est une distance entre les désirs de deux agents. D'un point de vue plus technique, ce principe représente la distance euclidienne entre les désirs D d'un agent a_i et ceux d'un agent a_j tel que :

$$d = D_i - D_j \qquad\qquad s = \sqrt{d^T d} \tag{4.1}$$

  avec d^T la transposée du vecteur d. Plus la distance est faible, plus les agents possèdent des désirs similaires et plus il est intéressant de les regrouper afin de former une coalition d'agents ayant des objectifs communs ;
- principe de complémentarité (c) : les individus aiment les personnes dont les qualités sont complémentaires aux leurs. Chaque coalition représente un produit composé de plusieurs agents, qui représentent eux-même des composants physique. Nous souhaitons que les coalitions soient les plus complètes possible de façon à concevoir des produits finis. Pour favoriser leur complétion, nous utilisons ce principe comme la complémentarité des désirs d'association entre deux agents. Autrement dit, des agents qui souhaitent ajouter des *composants* différents dans leur coalition seront un peu plus attirés que ceux

qui souhaitent s'affilier avec les mêmes composants.

$$c = 1 - \frac{n_{diff}}{\max(n_i, n_j)} \tag{4.2}$$

avec n_i et n_j les nombres de composants avec lesquels veulent s'associer des agents i et j dans une coalition et n_{diff} le nombre de composants différents avec lesquels ils souhaitent s'associer ;

- principe d'attractivité physique (a) : les individus sont plus attirés par les personnes qui sont physiquement attractives. Dans notre système, l'attractivité physique est considérée comme l'adéquation entre les caractéristiques d'un individu et les désirs de l'autre. Techniquement parlant, l'attractivité physique est la distance entre les désirs D d'un agent et les caractéristiques C d'une de ses accointances. Plus la distance est faible et plus l'accointance est attirante.

$$d = D_i - C_j \qquad\qquad a = \sqrt{d^T d} \tag{4.3}$$

  Ce principe a comme intérêt de rendre attirants les agents qui possèdent des caractéristiques correspondants aux désirs des autres. Il joue donc un rôle majeur dans le comportement de nos agents.
- principe du minimax (m) : les individus sont attirés par les personnes qui leur offrent une récompense maximale pour un coût minimal. Par exemple, si un agent A demande à un agent B de former un groupe et que l'agent B refuse la proposition alors l'agent A dégrade l'attraction pour cet agent de façon à ne pas lui proposer à nouveau de former un groupe.

$$m = nbR * \epsilon \tag{4.4}$$

  avec nbR le nombre de rejets de demande de formation ou d'affiliation à une coalition qu'une accointance a émis vers un agent et ϵ le taux de dégradation de l'attraction pour chaque refus. La partie 6 explique comment ce taux est choisi dans le cadre de notre expérimentation.

Les agents doivent être capables d'évaluer l'attrait qu'ils ont pour d'autres agents afin de former des groupes dans le système. Pour ce faire, nous construisons l'équation 4.5 qui intègre ces principes.

4.2.2. *Calcul de l'attraction interpersonnelle*

La fonction d'attraction appelée ici v_a est la fonction permettant aux agents d'évaluer leurs accointances pour prendre des décisions. Nous la définissons grâce aux principes de l'attraction de la section précédente : la similarité **(s)**, la complémentarité **(c)**, l'attractivité physique **(a)** et le minimax **(m)**. $v_a(A)$ représente la valeur d'attraction qu'à un agent pour son accointance A.

$$v_a(A) = \frac{\lambda_1 c + \lambda_2 s + \lambda_3 a}{\lambda_1 + \lambda_2 + \lambda_3} + m \tag{4.5}$$

Cette équation est une moyenne pondérée de la complémentarité, similarité et attractivité physique additionnée au minimax. L'attractivité physique étant primordiale

à la conception de produits répondant aux désirs de l'utilisateur, nous proposons de surpondérer ce principe dans l'équation. La complémentarité et la similarité étant au contraire des principes de supports, c'est-à-dire facilitant la formation de coalitions, leur pondération est moins important que celle de l'attractivité physique. La partie 6 revient sur les diverses configurations de la fonction d'attraction, évalue leur efficacité dans la formation de coalitions et propose des pondérations en fonction des cas applicatifs.

4.2.3. *Évaluer les coalitions*

La cohésion de groupe est définie de plusieurs manières dans la littérature des *Dynamiques de groupes*. Cependant, la définition de Lott & Lott [21] est particulièrement intéressante dans notre cas puisqu'elle la défini comme une agrégation de l'attraction qu'un membre du groupe à pour les autres membres, attraction que nous avons opérationnalisé dans la section précédente. Nous nous inspirons donc de cette définition pour calculer l'attraction qu'un agent a pour sa coalition. Nous définissons l'attraction qu'un agent a pour sa coalition comme la moyenne de l'attraction qu'il a pour tous les autres membres avec qui il désire être en groupe. $V_{ac}(C)$ représente l'attraction qu'un agent a pour sa coalition C.

$$v_{ac}(C) = \frac{\sum_{A}^{Acc} v_a(A)}{|Acc|} * (1{,}1 - \frac{|Acc|}{|D|}) \tag{4.6}$$

avec Acc l'ensemble des accointances de l'agent dans la coalition concernée et A l'une de ces accointances. $|D|$ représente le nombre total d'accointances avec lesquelles un agent désire former un groupe. La partie droite de la fonction (après la multiplication) est un coefficient représentant à quel point une coalition est complète du point de vue de l'agent. Plus $|Acc|$ est proche de $|D|$, et plus l'agent a d'accointances avec qui il désire être en groupe dans sa coalition. Ce coefficient permet la valorisation des coalitions les plus remplies pour inciter les agents à y entrer et ainsi obtenir des coalitions les plus complètes possible. De la même manière que pour l'attraction interpersonnelle, plus la valeur de l'attraction est faible, meilleure elle est pour les agents. Notons que l'agent ne prend en compte que l'attraction qu'il a pour ses accointances puisqu'il lui est impossible d'évaluer les agents pour lesquels il n'a pas de désirs. Par exemple, si l'objectif était de concevoir une batterie électrique, un agent *cellule* ne pourrait évaluer un agent *boîtier* car les deux n'étant pas liés dans la structure du produit, ils ne se connaissent pas.

Une fois l'attraction calculée et stockée dans la mémoire sémantique d'un agent ABSG, ce dernier a besoin de l'utiliser pour savoir à quelles accointances demander de former une coalition. La section suivante présente la façon dont l'attraction est utilisée dans la prise de décision et détaille le cycle de décision de l'architecture ABSG.

4.3. Module de décision

Les agents ABSG ont besoin d'un module de décision leur permettant de sélectionner leur prochaine action pour former ou quitter des coalitions. Dans cette section, nous présentons les diverses règles de comportement stockées dans la mémoire de

production des agents ainsi que leur cycle de décision général et définissons une limite permettant aux agents de savoir quelles accointances sont bonnes ou non pour former une coalition.

4.3.1. *Prise de décision*

La prise de décision de l'architecture ABSG est basée sur deux facteurs : (1) les règles de comportement issues de la mémoire de production, (2) les valeurs d'attraction affectées aux accointances ou aux groupes. Les comportements (présentés dans la table 4.1) sont un ensemble de règles qui indiquent comment un agent doit agir dans des situations spécifiques. Ils sont similaires à une déclaration « si/alors ».

Règles	Description
Règle 1	Si l'agent ne connaît pas les caractéristiques d'une de ses accointances, alors il les lui demande.
Règle 2	Si l'agent reçoit une demande de caractéristiques, alors il les envoie à l'accointance qui les demande.
Règle 3	Si un agent change de caractéristiques, alors il renvoie les nouvelles à toutes ses accointances.
Règle 4	Si l'agent a une valeur d'attraction inférieure à un seuil α pour une de ses accointances et qu'il n'est pas déjà dans un groupe avec elle, alors il lui envoie une requête de formation de groupe.
Règle 5	Si l'agent reçoit une demande de formation de groupe et que la valeur de l'attraction pour l'émetteur de la demande est inférieure au seuil α, alors il accepte la requête, sinon il la refuse.
Règle 6	Si l'accointance possédant la plus faible valeur d'attraction n'est pas dans le groupe qui possède la plus faible valeur d'attraction de l'agent, alors il lui envoie une requête à son accointance pour lui demander de rejoindre son groupe.
Règle 7	Si l'agent reçoit une demande pour rejoindre un groupe et que son attraction pour l'accointance émettrice est inférieure à un seuil β, alors il accepte, sinon il refuse.
Règle 8	Si l'attraction de l'agent pour un groupe est supérieure à un seuil γ, alors il le quitte en notifiant tous les autres membres de son départ.

TABLE 4.1 – Règles de comportement instanciées dans l'architecture ABSG

Il est important que les agents priorisent l'échange de leur caractéristiques pour adapter leur prise de décision. Par conséquent les règles les plus prioritaires sont les 1, 2 et 3. Ensuite la priorité est à la formation de groupe. Les agents ne créent pas de coalitions avec les accointances qui sont déjà dans un de leur groupe afin d'éviter de faire des groupes redondants dans lesquels les membres sont exactement les mêmes.

Donner la priorité à la règle de formation avant celle de l'affiliation permet donc aux agents de former des groupes avant que les membres ne les remplisse avec de nouveaux membres. Les agents ABSG peuvent faire partie de plusieurs coalitions, ce qui permet au système d'aide à la décision d'essayer plus de solutions. Bien que le chevauchement des coalitions paraisse pratique, il serait inutile de former des coalitions possédant exactement les mêmes membres. C'est pourquoi la règle 4 spécifie de ne pas former de groupes avec une accointance déjà présente dans une coalition. En revanche, un agent est libre d'ajouter n'importe quel membre à une de ses coalitions s'il estime qu'il répond à ses besoins. Enfin, la règle 8 spécifie qu'un agent a la possibilité de quitter une coalition si son attraction pour elle n'est pas suffisamment bonne.

Il est important que le système se stabilise sur une structure de coalitions à proposer à l'utilisateur. Pour cela, les agents tiennent un registre des accointances auxquelles ils ont demandé de former un groupe. Un agent ne peut demander qu'une seule fois à l'une de ses accointances de former un groupe. Cependant, il s'autorise à redemander à chaque fois que cette accointance modifie l'une de ses caractéristiques ou l'un de ses désirs.

4.3.2. *Seuils de sélectivité*

Dans cette section, nous définissons les seuils α, β et γ utilisés dans les règles de la table 4.1. Plus les seuils sont élevés et plus ils laissent aux agents l'opportunité de créer de coalitions. Au contraire, plus ils sont bas, moins les agents sont enclins à former des groupes. Nous proposons une fonction de seuil (équation 6.5) qui correspond à une fonction de normalisation prenant en compte le nombre d'accointances des agents pour calculer α, β et γ. En effet, plus un agent a d'accointances, plus il a de chance d'en trouver de très intéressantes pour former une coalition. Il peut donc se permettre d'être plus sélectif. Au contraire, moins il a d'accointances, plus il y a de risque qu'une forte sélectivité sur la qualité de l'attraction pour une accointance mène l'agent à s'isoler sans former de groupes.

Ces seuils influent sur le comportement des agents à travers les règles explicitées dans la section précédente. La section suivante présente le cycle de décision de l'architecture ABSG.

4.3.3. *Cycle de décision*

L'architecture ABSG étant basée sur Soar, leurs cycles de décision sont très similaires. Les agents ABSG doivent former des groupes dans un système dynamique où leurs accointances peuvent changer de caractéristiques à n'importe quel moment. La valeur de l'attraction doit donc être mise à jour régulièrement pour chaque accointance. Ainsi, un agent ABSG possède dans son cycle de décision une étape supplémentaire par rapport à un agent Soar lui permettant de charger les messages reçus dans la mémoire sémantique et de mettre à jour l'attraction pour ses groupes et ses pairs. Cette phase supplémentaire est appelée « phase de socialisation ». Le cycle de décision est divisé en quatre phases que nous présentons ci-dessous :

- la phase d'élaboration : l'agent charge dans la mémoire de travail toutes les règles de production qui peuvent être déclenchées ;
- la phase de décision : l'agent choisit une règle à lancer. Si plusieurs règles peuvent être lancées, il choisit la règle de plus haute priorité qui est l'ordre dans lequel elles sont présentées dans la partie précédente. Toutes les règles chargées au moment de la phase de décision seront exécutées dans cet ordre ;
- la phase d'application : les actions des règles sélectionnées sont exécutées. La mémoire de travail et la mémoire sémantique sont modifiées ;
- la phase de socialisation : les messages reçus par l'agent depuis le dernier cycle sont traités et les valeurs d'attraction des connaissances et des coalitions sont mises à jour en fonction du nouvel état de la mémoire sémantique.

Ce fonctionnement individuel dote les agents ABSG d'un comportement adapté à la formation de coalitions. Mais pour réellement les former, il leur est nécessaire d'interagir les uns avec les autres à travers un langage commun et un protocole de communication. La partie suivante les présente.

4.4. Module de communication

4.4.1. *La structure des messages*

Les interactions sont essentielles à la formation des coalitions. Les messages permettent aux agents d'échanger leurs caractéristiques et leurs désirs, d'envoyer des requêtes afin de former des groupes, *etc*. Ils utilisent une réduction du langage de communication FIPA-ACL pour communiquer que nous présentons ci-dessous. Le langage FIPA-ACL est originellement très complet et les agents ABSG n'ont pas besoin d'une structure aussi détaillée pour interagir. Un message généré par le module de communication d'ABSG est composé des paramètres suivants :

- *sender* : l'identifiant de l'agent ayant envoyé le message ;
- *receiver* : l'identifiant de son destinataire ;
- *performative* : caractérise le type de communication. Par exemple une demande de formation de groupes, ou une réponse à une demande d'affiliation. Les performatifs utilisés sont détaillés dans la suite de cette partie ;
- *ontology* : sert de contexte au message donnant un sens à son contenu ;
- *content* : le corps du message, contient toutes les données que l'agent veut transmettre. Par exemple dans le cas d'une demande d'affiliation à une coalition, l'identifiant du groupe concerné, les identifiants de ses membres, *etc*.

4.4.2. *Description des paramètres*

Deux des paramètres de la structure d'un message utilisé par les agents de notre système peuvent contenir une multitude de valeurs. Nous les présentons et justifions ci-dessous.

Ontologies. L'ontologie donne un contexte permettant de comprendre le contenu du message. Dans le cas des messages générés par notre architecture, l'ontologie peut prendre 3 valeurs :

- *agent* : désigne les interactions dont le sujet est l'agent. Toutes les interactions permettant de demander ou transmettre les caractéristiques et les désirs des agents utilisent cette ontologie ;
- *coalition* : désigne les interactions qui ont pour sujet la formation de coalition. Par exemple, une demande de formation de coalition utilise cette ontologie ;
- *scheduling* : désigne toutes les interactions permettant aux agents de se coordonner. Par exemple lorsqu'un agent souhaite ajouter un membre dans sa coalition.

Performatifs. Le performatif exprime l'action du message. Les performatifs manipulés par l'architecture ABSG sont les suivants :

- *proposes* : est utilisé lorsqu'un agent propose à son accointance de former une nouvelle coalition ou d'en rejoindre une existante ;
- *requests* : la requête est utilisée pour les interactions de coordination, par exemple quand un agent demande l'autorisation à ceux de sa coalition d'ajouter un nouveau membre. Les processus de coordination sont détaillés dans la section suivante traitant des protocoles de communication ;
- *queries* : est utilisé lorsqu'un agent souhaite connaître les caractéristiques et désirs d'une de ses accointances ;
- *informs* : est utilisé en réponse au performatif précédent pour renseigner les informations demandées à l'agent concerné ;
- *accept_nproposal* : ce performatif est l'une des deux réponses possibles au performatif *proposes* lorsqu'un agent en invite un autre dans une coalition ;
- *reject_proposal* : le refus est la seconde réponse envisageable au performatif *proposes* en cas de demande de formation ou d'affiliation à une coalition.

Afin de mieux visualiser les liens entre ces performatifs et leur enchaînement dans une conversation entre deux agents, nous pouvons présenter le protocole de performatif de la manière suivante :

- $reception(proposes) = \{accept\_proposal, reject\_proposal\}$
- $reception(requests) = \{ informs\}$
- $reception(queries) = \{ informs\}$
- $reception(informs) = \emptyset$
- $reception(accept\_proposal) = \emptyset$
- $reception(reject\_proposal) = \emptyset$

Nous pouvons lire chaque élément de cette liste comme : *à la réception d'un message contenant un performatif P, l'agent peut répondre par l'un des performatifs de l'ensemble suivant* $\{P1, P2, \ldots\}$. Dans ce protocole, l'ensemble $\emptyset$ désigne un ensemble vide et la fin de la conversation.

Après avoir présenté la structure des messages utilisés par les agents ABSG et détaillé leurs paramètres, nous présentons dans la section suivante le protocole de communication suivi par chaque agent. Ce protocole peut être vu comme la structure de la conversation que les agents doivent suivre pour travailler ensemble.

4.4.3. *Protocole de communication*

Les figures 4.2, 4.3, 4.4 et 4.5 illustrent des instances de protocoles internes d'un agent dans des scénarios de formation de groupe, de demande d'informations, de demande d'affiliation à un groupe et d'abandon de groupes. L'état initial est l'état numéro 1 et l'état final est indiqué par un double cercle. Les marqueurs *!* sont les messages émis, *?* les messages attendus et *x* un paramètre échangé entre les protagonistes. Dans le cas des figures 4.2, 4.4, 4.5 et 4.6, le paramètre *x* représente le numéro du groupe formé, rejoint ou quitté. Dans le cas de la figure 4.3 il représente les caractéristiques physiques et les désirs de l'agent. Ce protocole est très proche du protocole *FIPA Contract Net* [29], la seule différence étant que nos agents n'ont pas la nécessité de faire un appel d'offre puisque qu'ils contactent directement l'accointance avec laquelle ils veulent traiter.

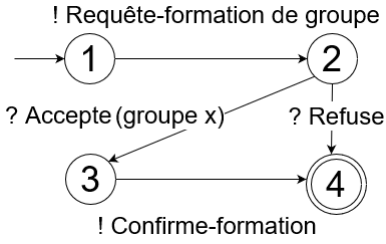


FIGURE 4.2 – Instance du protocole de proposition de formation de groupe

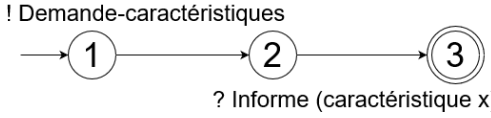


FIGURE 4.3 – Instance du protocole de demande d'information

Puisqu'il n'existe pas de représentation centralisée des coalitions et que chaque membre a la charge de sa propre représentation de la coalition, il est nécessaire aux agents de se synchroniser pour modifier la composition d'un groupe. Il faut par exemple éviter qu'un membre ajoute un nouvel agent au groupe tandis qu'un autre membre le quitte. Par défaut, le nouvel entrant suppose que personne n'a quitté le groupe entre le moment où il a reçu l'invitation et le moment où il l'a acceptée. Modifier la coalition pendant ce laps de temps provoquerait une incohérence parmi les représentations de la coalition des membres. Pour ce coordonner, les coalitions possèdent un mécanisme de *leader* permettant de coordonner les membres d'une coalition lorsqu'ils souhaitent la

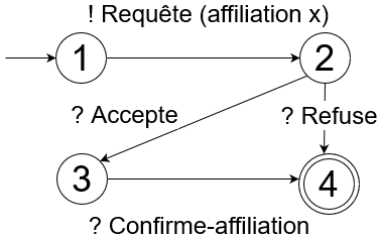


FIGURE 4.4 – Instance du protocole de demande d'affiliation à un groupe

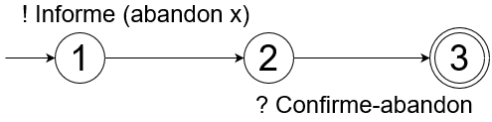


FIGURE 4.5 – Instance du protocole d'abandon de groupe

modifier. L'agent à l'origine de la formation du groupe est automatiquement considéré comme son *leader*. Le *leader* initialise un sémaphore sur lequel il a le contrôle. Son rôle est d'indiquer aux autres membres quand ils peuvent ou non modifier la coalition. Il accorde son autorisation à un membre du groupe par l'envoi d'un jeton. Un agent souhaitant ajouter un nouveau membre au groupe doit utiliser le protocole de demande de jeton illustré dans la figure 4.6. Notons que le transfert du jeton ne peut être refusé ce qui évite aux agents d'une coalition les situations de blocage. Lorsque les agents ont fini de modifier le groupe, le jeton est renvoyé au *leader* qui peut alors le transmettre à d'autres membres. Le rôle de *leader* et le sémaphore qui lui est rattaché sont associés à un groupe. Un agent peut donc être le *leader* de plusieurs coalitions qui possèdent toutes leur propre sémaphore. Lorsque le *leader* d'un groupe souhaite le quitter, il nomme son successeur et transmet son identifiant à tous les membres du groupe.

Dans les protocoles suivants, les agents sont soumis à un message de confirmation permettant à l'initiateur de la conversation de savoir que leur requête à bien été traitée par son destinataire. Le mécanisme de confirmation est particulièrement important puisque l'expéditeur du message n'exécute son action qu'une fois certain que son message a bien été traité. Par exemple dans le cas d'un abandon de coalition, il notifie tous les membres du groupe et ne le quitte réellement que lorsque tous les destinataires ont renvoyé la confirmation. Ce mécanisme de confirmation a pour intérêt de permettre au leader d'être certain que tous les membres de sa coalition ont pris en compte la modification du groupe au moment où il récupère le jeton de l'accointance initiatrice du changement.

5. Opérationnalisation de l'architecture d'agent

L'architecture a été présentée de manière théorique dans la section 4 mais nécessite quelques précisions pour son utilisation pratique. Pour compléter notre présentation, il

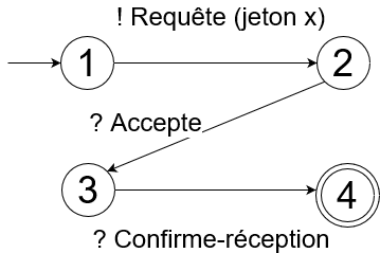


FIGURE 4.6 – Instance du protocole de demande de jeton

est nécessaire de décrire comment les connaissances sont représentées dans la mémoire des agents. Les agents se représentent leurs caractéristiques et désirs à l'aide de deux tableaux de décimaux, le tableau contenant les caractéristiques et celui contenant les désirs. Le nombre décimal d'un agent peut par exemple correspondre à sa couleur. Dans le cas où l'ensemble des couleurs que l'agent peut avoir est {*bleu*, *rouge*, *vert*} alors la caractéristique couleur pourrait prendre les valeurs 0, 0,5 ou 1 où chaque décimal correspond à une unique couleur. Pour les caractéristiques quantitatives, comme le poids d'un contrepoids de lave-linge, la valeur peut être normalisée grâce à l'équation :

$$n = \frac{x}{maxin - minin} \tag{5.1}$$

Avec *maxin* et *minin* les intervalles de valeurs possibles que peut prendre la valeur à normaliser x. En outre, cette normalisation a l'avantage de limiter le biais du calcul de l'attraction en comparant des caractéristiques qui évoluent sur des intervalles et des échelles de valeur différentes. Dans ces expériences, nous instancions les désirs et les caractéristiques de nos agents sous la forme de tableaux de décimaux normalisés entre 0 et 1 permettant de limiter cet effet.

Pour rappel, chaque agent est caractérisé par deux types de données qu'il représente en mémoire comme des tableaux :

- ses caractéristiques : ce sont les caractéristiques de l'agent (*e.g.* poids, volume, prix, etc.) ;
- ses désirs : ce sont les caractéristiques que l'agent aimerait voir chez les membres des groupes auxquels il appartient (*e.g.* volume recherché, prix recherché, etc.).

6. Expériences et analyses

6.1. Métriques d'évaluation

Dans ces expériences, le but des agents est de former des coalitions contenant un nombre spécifique de membres tout en trouvant les meilleurs agents en fonction de leurs désirs et de leurs caractéristiques. Nous avons défini la taille idéale des coalitions à 4 pour les rendre réalisables dans les expériences à plus petite échelle. La valeur d'une

coalition $V(C)$ est une valeur comprise entre 0 et 1. Elle est utilisée comme mesure de la qualité pour un utilisateur extérieur. 0 représente la qualité la plus faible qu'une coalition peut avoir, 1 représente la qualité la plus élevée. Dans notre évaluation, la qualité d'une coalition correspond à la moyenne de la satisfaction moyenne des désirs de chaque agent du groupe. La satisfaction des désirs d'un agent pour les caractéristiques de l'une de ses accointances correspond à la normalisation de la distance entre ses désirs et les caractéristiques de l'accointance :

$$d = \sqrt{s^T s} \qquad s = a_d - a_c \tag{6.1}$$

$$satisfaction(a_d, a_c) = \frac{d}{\sqrt{|a_c|}} \tag{6.2}$$

avec a_d les désirs de l'agent et a_c les caractéristiques de son accointance. La normalisation a pour objectif de ne pas faire augmenter la distance artificiellement en augmentant le nombre de caractéristiques des agents. En effet, la distance entre un désir et une caractéristique est comprise entre 0 et 1, alors qu'une distance entre dix désirs et dix caractéristiques est comprise entre 0 et $\sqrt{10}$. La normalisation évite alors de biaiser la mesure de la qualité pour les agents étant décrits par un grand nombre de caractéristiques. $S(a)$, la satisfaction moyenne d'un agent a pour sa coalition C peut être définie telle que :

$$S_c(a) = \frac{\sum_{m=0}^{|Acc_a|} \big(1 - satisfaction(a_m)\big)}{|Acc_a|} \tag{6.3}$$

avec Acc_a les accointances de a présentes dans la coalition c, $satisfaction(a_m)$ la satisfaction des désirs de l'agent a pour son accointance m et $|Acc_a|$ le nombre d'accointances de a dans la coalition c. La satisfaction moyenne d'une coalition C est ainsi définie comme :

$$V(C) = \sum_{n=0}^{|C-1|} S_c(n) \tag{6.4}$$

avec $S_c(n)$ la satisfaction de l'agent n pour la coalition c et $|C|$ la cardinalité de la coalition C. L'objectif de la somme est à la fois de pouvoir tenir compte de la satisfaction de chaque agent pour ses accointances dans la coalition et à la fois du nombre d'agents dans le groupe, ce qui permet de faire la différence entre une coalition complète et incomplète.

Dans les sections suivantes, nous évaluons cinq critères : (1) la durée des expériences, (2) la qualité des groupes formés, (3) la robustesse à l'ouverture, (4) la robustesse à la variabilité. Nous avons besoin de connaître la meilleure solution pour comparer la qualité de notre résultat à une valeur optimale. Pour cela, un script Python génère des agents afin que nous connaissions la valeur de la meilleure coalition possible et recherche la solution optimale par un algorithme brute-force.

6.2. Conditions d'expérimentations

6.2.1. *Observateur*

L'évaluation de l'ouverture et de la variabilité nécessite une mesure de la qualité des coalitions formées au fil des expériences afin d'observer les effets de l'ajout, de la suppression ou de la modification des caractéristiques des agents sur le système. Pour cela, un processus observateur est lancé en même temps que les agents du système. Celui-ci a pour objectif l'observation du système multi-agent et la collection des coalitions produites. Tous les agents savent intrinsèquement comment communiquer avec l'observateur. Ils le notifient de leur arrivée dans le système au moment de leur exécution et lui envoie une copie de chaque message transmis à leurs accointances. Notons que ces copies ne sont pas prisent en compte dans les expériences de la section suivante. En plus de ces copies, l'observateur demande régulièrement aux agents les coalitions dans lesquelles ils se trouvent. En outre, ce mécanisme de demande d'information est utile du point de vue du système d'aide à la décision puisqu'il permet à celui-ci de proposer des solutions à son utilisateur à n'importe quel moment.

6.2.2. *Paramétrage des agents*

Dans les expériences suivantes les caractéristiques et les désirs des agents sont générés aléatoirement. Les seuils de formation, d'affiliation et de sortie d'une coalition sont calculés en fonction du nombre d'agents dans le système tel que :

$$f_s(x) = \frac{(maxin - x - minout) * (maxout - minout)}{maxin - minin} + minout \tag{6.5}$$

Cette équation permet la normalisation des seuils en fonction du nombre d'accointances d'un agent. Plus un agent a d'accointances, meilleure l'attraction doit être pour que celui-ci accepte une requête de formation de groupe ou d'affiliation. Cette méthode permet de rendre les agents moins exigeants lorsqu'ils ont peu de choix de formation de groupe ou d'affiliation, par exemple dans les systèmes à petite échelle. Au contraire, dans des systèmes à plus grande échelle, les agents ont de multiples possibilités de former ou rejoindre des groupes, diminuer la valeur des seuils permet alors de se concentrer sur les meilleures d'entre elles. En plus de former des coalitions de meilleure qualité, ce mécanisme réduit le temps d'exécution des systèmes les plus larges. Dans ce cas précis, nous affectons *maxin* comme étant le nombre d'accointances maximum que peut avoir un agent puisque, comme nous le verrons dans la partie suivante, la majorité des évaluations sont exécutées avec un maximum de 80 agents. *minin* est initialisé à 0 dans le cas où un agent n'a aucune accointance et *minout* et *maxout* sont respectivement associés aux valeurs 0,35 et 0,45. L'attraction étant normalisée entre 0 et 1 (sans prendre en compte le principe du minimax qui peut la faire dépasser 1), nous estimons qu'une accointance intéressante est une accointance pour laquelle l'attraction est d'environ 40 % de la pire valeur possible. Ce seuil a été fixé empiriquement de façon à faire un compromis d'une part entre une trop faible sélectivité des accointances menant à la formation de multiples coalitions de faible qualité et d'autre part une trop forte sélectivité empêchant toute coalition d'émerger.

maxout représente le seuil maximum de sélectivité et permet aux agents ayant peu d'accointances de réduire leur sélectivité afin de tout de même pouvoir former ou s'intégrer à des coalitions.

Les seuils de sélectivité étant compris entre 0,35 et 0,45, nous fixons ϵ à 0,2 afin de rapidement dépasser *minout* et d'empêcher les agents de trop insister à intégrer une accointance dans leurs coalitions si le système comprend un grand nombre d'agents. L'objectif étant de leur éviter de s'entêter à s'affilier avec un agent réticent si beaucoup d'autres sont disponibles. Au contraire dans le cas où un agent a peu d'accointances et le seuil de sélectivité est plus élevé, $\epsilon = 0{,}2$ lui permettrait d'insister auprès des accointances pour lesquelles il a les meilleures valeurs d'attraction pour les intégrer à plusieurs coalitions. Nous ne prétendons pas avoir proposé la valeur optimale de ϵ en l'initialisant de cette manière, cependant nous suggérons que celle-ci est suffisamment convenable pour permettre le bon fonctionnement de nos agents et permettre au système de proposer des solutions de bonne qualité. Les parties suivantes investigueront plus en détail de l'effet de ϵ et l'influence que ce paramètre a sur le comportement des agents et sur leur capacité à former des coalitions.

Les parties suivantes évaluerons l'approche multi-agent et les capacités de notre système à faire émerger des coalitions dans un cadre théorique. Le comportement des agents étant indépendant du nombre de caractéristiques et de désirs qu'ils manipulent, nous avons arbitrairement choisi de leur en affecter 5 pour l'évaluation.

6.2.3. *Environnement matériel et logiciel*

Cette architecture a été développée à l'aide des langages *C* et *C++* à partir du code source de l'architecture Soar téléchargée sur son dépôt *Github*.

Les agents sont exécutés en parallèle sur la même machine de manière asynchrone et communiquent avec le protocole TCP/IP. Les expériences sont effectuées sur un PC multi-cœur Intel i7-6600U CPU (2,60GHz), 16 Go, 64-bit, Manjaro Linux 20.1.

7. Évaluation de l'approche multi-agent

Les évaluations suivantes présentent la qualité des solutions produites par les agents ABSG. Sur chaque graphique de cette partie, les aires Q1, Q2, Q3 et Q4 représentent chacune un quartile de l'espace de solution triées en fonction de leur qualité. Ainsi, Q1 représente les 25 % meilleurs solutions, Q2, mes 25 % à 50 %, etc. Cette visualisation permet de mieux voir la distribution des solutions générées par notre méthode.

7.1. Comparaison à la littérature

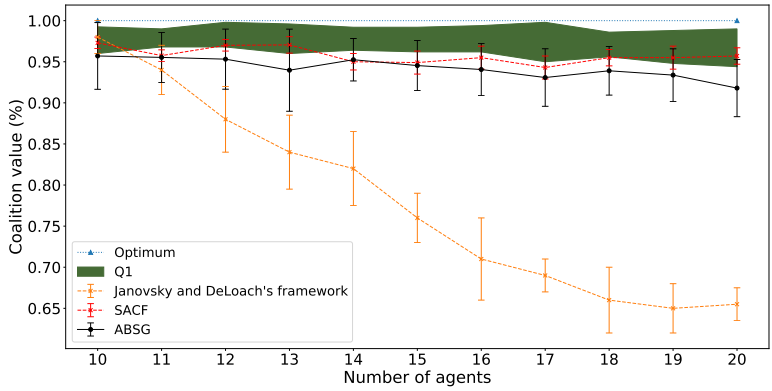


Figure 7.1 – Comparaison avec la littérature de la qualité des structures de coalitions formées en fonction de l'optimum pour des expériences de 10 à 20 agents.

Dans ces expériences, nous comparons la qualité des coalitions de notre approche à celles de l'art. Les courbes représentant les valeurs de structure de coalitions optimales sont obtenues par un algorithme brute-force testant toutes les solutions pour ne sélectionner que la meilleure. Dans ces expériences les agents ne possèdent pas d'objectif commun. Nous générons leurs désirs de manière aléatoire et observons le temps dont le système a besoin pour faire émerger des coalitions.

La figure 7.1 présente les résultats des approches SACF [32] et du framework de Janovsky et Deloach [17].

Dans leur travail, Janovsky et Deloach présentent un framework pour la formation de coalitions à grand échelle. Leur méthode s'inspire d'une approche de clustering hiérarchique afin de construire des structures de coalitions de manière dynamique selon un processus itératif. Contrairement à nous, leur approche n'est pas distribuée et fonctionne exclusivement en monde fermé. Ce framework étant basé sur une approche de clustering, il n'est pas adapté à fonctionner dans un système ouvert et variable. Cependant, il se trouve être plus adapté à des systèmes à très large échelle (jusqu'à 10 000 agents) dans lesquels les agents sont des entités invariables.

SACF est une méthode distribuée fonctionnant en système ouvert dans un environnement situé. Leur approche présente des similitudes à la nôtre puisque leurs agents ont la capacité de créer ou rejoindre des coalitions en fonction de la valeur qu'ils associent à leurs accointances. Bien que les agents SACF soient conçus pour être flexibles, les coalitions générées ne sont pas remises en question en cas de changement dans le système.

La méthode SACF et le framework de Janovsky et DeLoach n'ont pas été réimplémentés. Nous avons comparé leurs conditions expérimentales et les avons réutilisées

afin de pouvoir comparer nos résultats aux leurs. La courbe noire représente la qualité moyenne des coalitions formées par les agents ABSG. On peut la considérer comme la valeur de notre structure de coalitions de manière à mieux nous comparer à la courbe de SACF et du framework de Janovsky et Deloach. Afin de mieux comparer les résultats entre eux, tous sont comparés à la valeur optimale que les systèmes auraient pu trouver dans leur contexte d'exécution. Dans le cas de notre système, puisque nous recherchons les meilleures coalitions possibles pour de l'aide à la décision, l'optimal représente la qualité de la meilleure coalition concevable. Sur la figure 7.1, l'aire Q1 représente la qualité des 25 % meilleures coalitions que les agents ont formées. Comme nous pouvons le constater, ces coalitions sont pour la plupart plus proches de l'optimal que les structures générées par SACF.

Pour notre seconde expérience tous les agents sont lancés en même temps, le décompte du temps commence au moment où l'observateur reçoit le premier message d'un des agents du système. L'observateur estime que le système a convergé vers des solutions et arrête le décompte lorsqu'il ne reçoit plus de nouveaux messages pendant dix secondes.

Les algorithmes de ODP et ODP-IP ont téléchargés sur leur dépot *Github* et ont été exécutés sur la même machine que celle qui exécutait notre système multi-agent de façon à comparer les temps d'exécution. La figure 7.2 compare les temps nécessaires à notre approche pour générer des solutions face à des algorithmes exacts. Ces algorithmes sont très efficaces pour de très petites quantités d'agents. Cependant, leur temps d'exécution dépasse rapidement celui de notre approche à partir de 22 et 25 agents. Plus important, leur complexité en temps est exponentielle et leur consommation de mémoire étant elle aussi exponentielle, il leur est difficile de s'exécuter avec plus de 25 agents. L'échelle rend difficile la mesure de l'évolution du temps d'exécution de notre approche sur cette figure. Du point de vue du temps d'exécution, notre approche semble donc plus pertinente que celles des méthodes exactes de programmation dynamique.

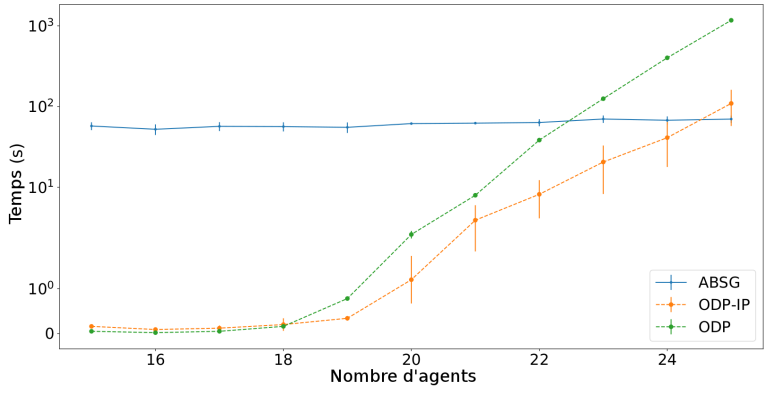


FIGURE 7.2 – Comparaison du temps d'exécution du système avec la littérature

D'autres approches comme celles du clustering [6], ou des jeux hédoniques [24] obtiennent des temps d'exécution plus faibles que les approches présentées dans la figure 7.2. Cependant ces algorithmes sont pour la majorité conçus pour résoudre une instance non variable d'un problème de formation de coalitions de manière non exacte (contrairement à l'approche de programmation dynamique). Contrairement à l'approche multi-agent, celles-ci sont soit centralisées soit assument que les agents connaissent tout de leurs accointances au lancement du système ce qui réduit voire enlève totalement les temps de communication inter-agent. Cependant, comme nous le verrons dans les sections suivantes, occulter l'individualité des agents en les traitant comme des données et leur retirer leur capacité de décision peut également réduire la capacité des systèmes à s'adapter à un environnement ouvert et variable.

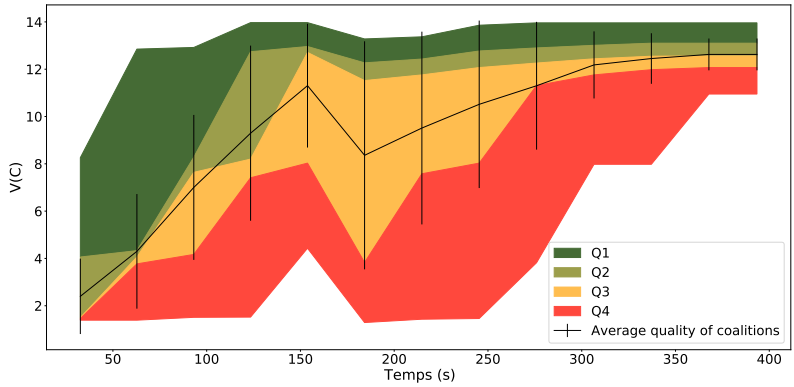
7.2. L'adaptabilité face à un système variable

Dans cette expérience de 60 agents (figure 7.3), nous voulons voir comment le système réagit si les agents changent aléatoirement leurs caractéristiques et leurs désirs. Contrairement aux expériences précédentes, nous ne connaissons pas la valeur optimale qu'une coalition peut avoir. Durant l'expérience, l'observateur demande à tous les agents de modifier les valeurs de leurs caractéristiques et de leurs désirs. Conformément à la règle 8, lorsque les agents reçoivent de nouvelles caractéristiques, ils doivent les envoyer à leurs accointances, ce qui entraîne l'explosion des messages émis (figure 7.3(b)). La qualité des coalitions est affectée par ces modifications (figure 7.3(a) à environ 160 secondes). Lorsqu'ils reçoivent les nouvelles caractéristiques de leurs accointances, les agents mettent à jour toutes leurs valeurs d'attraction et se réorganisent, ce qui augmente la moyenne des qualités de coalition. Le système recherche des solutions de manière incrémentale et peut s'adapter à des modifications de ses agents sans avoir à reconstruire les coalitions en partant de zéro. En outre, notons que le système peut retourner une solution à n'importe quel moment.

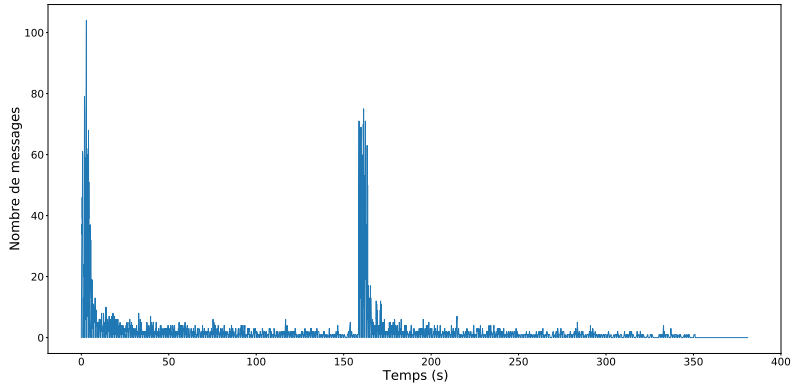
7.3. L'adaptabilité face à un système ouvert

Nous voulons voir comment le système réagit lorsque nous ajoutons ou retirons des agents. La figure 7.4(c) présente la qualité moyenne des coalitions lors de l'exécution d'un système de 50 agents. Du fait de que des agents sont introduits durant le temps d'exécution de l'expérience, il est difficile de prédire à l'avance la qualité de la coalition optimale. C'est pourquoi V(C) est ici exprimé comme une valeur brute et non comme un pourcentage de l'optimal comme dans les expériences précédentes.

Après 100 secondes écoulées, le processus observateur demande à 20 agents aléatoires du système de s'en retirer. Nous ne pouvons observer un effet notable de leur départ sur la qualité des coalitions. Cependant l'ajout de 21 nouveaux agents quelques secondes plus tard semble faire drastiquement diminuer leur qualité aux alentours de 180 secondes. En effet, au moment de leur entrée dans le système, ceux-ci créent de nouvelles petites coalitions qui n'ont pas eu le temps de se développer. L'ajout de ces coalitions dégrade la qualité moyenne des coalitions. Cependant elles se développent avec le temps et la qualité moyenne augmente à nouveau jusqu'à se stabiliser.



(a) Évolution de la qualité des coalitions formées par le système

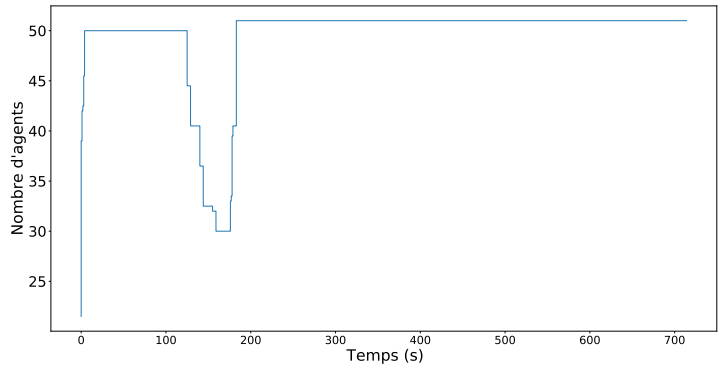


(b) Nombre de messages dans le système

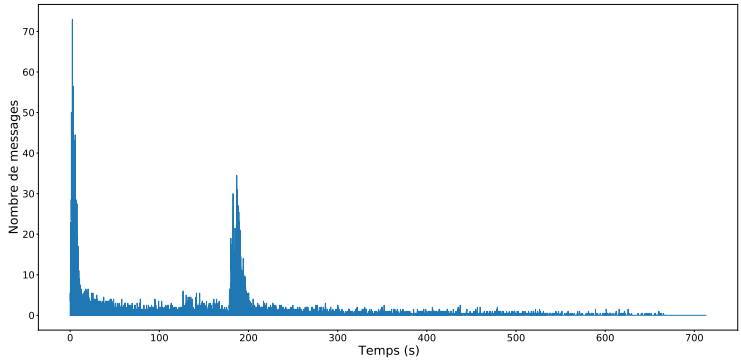
FIGURE 7.3 – Évaluation de la variabilité dans un système composé de 60 agents. Modification aléatoire des caractéristiques et désirs des agents à 150 secondes

La figure 7.4(b) montre le nombre de messages émis durant l'expérience. Comme pour les autres expériences, le premier pic représente le moment où les agents échangent leurs caractéristiques et leurs désirs. Le nombre de messages est ensuite peu élevé jusqu'au moment où les agents quittent le système. À environ 120 secondes une suite de petits pics montrent que des agents préviennent leurs accointances de leur départ de leurs coalitions. Lorsque les agents entrent dans le système un second grand pic apparaît aux alentours de 200 secondes qui est dû aux échanges des caractéristiques et désirs entre les nouveaux agents et ceux qui étaient déjà présents.

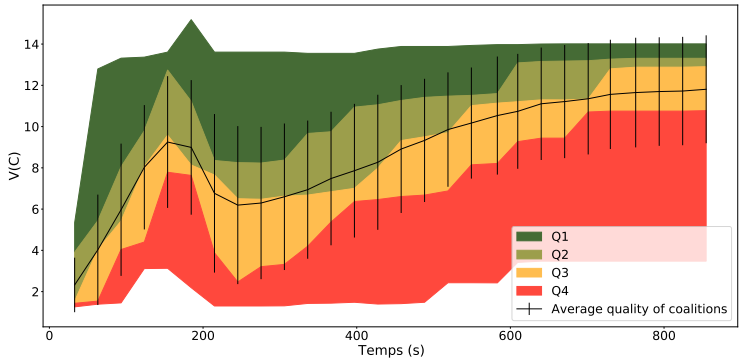
Cette évaluation montre la capacité du système à s'adapter dynamiquement à un environnement ouvert. Les agents se montrent capables de se réorganiser lorsqu'un évènement inattendu apparaît.



(a) Nombre d'agents dans le système en fonction du temps



(b) Nombre de messages émis dans le temps



(c) Mesure de la qualité des coalitions dans le temps

FIGURE 7.4 – Évaluation de l'ouverture du système par la suppression et l'ajout d'agents à respectivement 100 et 180 secondes

7.4. Synthèse et discussion

Notre approche se positionne entre les méthodes de programmation dynamique qui obtiennent des solutions exactes mais dont le passage à l'échelle est pour le moment impossible, et les algorithmes de clustering [11] qui fonctionnent avec plusieurs centaines – parfois milliers – d'agents dans des temps très courts mais où la qualité des solutions peut être moins importante. Les évaluations ont montré que notre système est capable de trouver des solutions proches de l'optimal malgré un grand nombre d'agents.

L'un des avantages que propose notre approche est sa capacité à prendre en compte des contraintes réalistes comme l'ouverture et la variabilité du système. Comme nous l'avons vu dans les sections 7.2 et 7.3, notre approche permet cette flexibilité et les agents de notre système se montrent capables de se réorganiser de manière incrémentale en cas de modification de l'un d'eux.

8. Conclusion

Ce problème de CSG fait intervenir de multiples contraintes issues d'un contexte applicatif réaliste. Celles-ci rendent difficile l'utilisation de nombreuses approches traditionnellement utilisées pour résoudre ce type de problèmes. Nous avons proposé l'utilisation d'une approche multi-agent inspirée par les dynamiques de groupes. Cette approche possède les propriétés requises pour concevoir notre système d'aide à la décision, à savoir la capacité à s'adapter à un système ouvert et variable et la possibilité de travailler avec des coalitions chevauchantes. En outre l'inspiration des dynamiques de groupes nous permet de proposer une méthode pertinente et innovante fondée sur l'étude de l'humain pour la formation de groupes.

Les évaluations ont montré que notre approche a bien permis la conception d'un système d'aide à la décision ouvert et variable à travers l'élaboration d'une architecture d'agent rendant le SMA capable de s'adapter à un environnement changeant. Les agents ont été capables de s'adapter à l'entrée et la sortie d'agents durant l'exécution du système et ont pu se réorganiser en conséquence. De la même manière, la variabilité a été évaluée et les agents ont su restructurer leurs coalitions lorsque leurs accointances ont modifié leurs caractéristiques. Les évaluations ont montré des performances intéressantes par rapport à la littérature, notamment dans la qualité des solutions proposées. Les performances du système multi-agent ont également été évaluées sur un cas d'étude réaliste pour la conception d'appareils électroménagers construit avec la collaboration d'acteurs industriels. Ces résultats sont présentés dans le manuscrit de thèse [5].

Bibliographie

[1] E. Anshelevich & W. Zhu, « Ordinal approximation for social choice, matching, and facility location problems given candidate positions », *ACM Transactions on Economics and Computation (TEAC)* **9** (2021), nº 2, article no. 9 (24 pages).

[2] M. Bennis, M. Simsek, A. Czylwik, W. Saad, S. Valentin & M. Debbah, « When cellular meets WiFi in wireless small cell networks », *IEEE communications magazine* **51** (2013), nº 6, p. 44-50.

[3] M. Bettinelli, O. Michel & G. Damien, « Coalition Formation Problem: a Group Dynamics Inspired Swarming Method », HAL preprint `https://hal.science/hal-02903531`, 2020.

[4] M. Bettinelli, M. Occello, D. Genthial & D. Brissaud, « A decision support framework for remanufacturing of highly variable products using a collective intelligence approach », *Procedia CIRP* **90** (2020), p. 594-599.

[5] M. Bettinelli, « Une approche d'intelligence collective pour la conception d'un système d'aide à la décision appliqué à l'économie circulaire », thèse de doctorat, Université Grenoble Alpes, 2021.

[6] F. Bistaffa, A. Farinelli, J. Cerquides, J. Rodríguez-Aguilar & S. D. Ramchurn, « Algorithms for graph-constrained coalition formation in the real world », *ACM Transactions on Intelligent Systems and Technology (TIST)* **8** (2017), nº 4, article no. 60 (24 pages).

[7] F. Brandt, V. Conitzer, U. Endriss, J. Lang & A. D. Procaccia, *Handbook of computational social choice*, Cambridge University Press, 2016.

[8] N. Changder, S. Aknine, S. D. Ramchurn & A. Dutta, « ODSS: Efficient Hybridization for Optimal Coalition Structure Generation. », in *AAAI*, vol. 34, 2020, p. 7079-7086.

[9] M. Delorme, S. García, J. Gondzio, J. Kalcsics, D. Manlove & W. Pettersson, « Stability in the hospitals/residents problem with couples and ties: Mathematical models and computational studies », *Omega* **103** (2020), article no. 102386.

[10] A. Dutta, V. Ufimtsev & A. Asaithambi, « Correlation clustering based coalition formation for multi-robot task allocation », in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, p. 906-913.

[11] A. Farinelli, M. Bicego, F. Bistaffa & S. D. Ramchurn, « A hierarchical clustering approach to large-scale near-optimal coalition formation with quality guarantees », *Engineering Applications of Artificial Intelligence* **59** (2017), p. 170-185.

[12] D. R. Forsyth, *Group dynamics*, Cengage Learning, 2010.

[13] D. Gale & L. S. Shapley, « College admissions and the stability of marriage », *The American Mathematical Monthly* **69** (1962), nº 1, p. 9-15.

[14] V. Gazi, « Swarm aggregations using artificial potentials and sliding-mode control », *IEEE Transactions on Robotics* **21** (2005), nº 6, p. 1208-1214.

[15] D. D. Henningsen, M. L. M. Henningsen & P. Booth, « Predicting social and personal attraction in task groups », *Groupwork* **23** (2013), nº 1, p. 73-93.

[16] F. R. Inácio, D. G. Macharet & L. Chaimowicz, « PSO-based strategy for the segregation of heterogeneous robotic swarms », *Journal of Computational Science* **31** (2019), p. 86-94.

[17] P. Janovsky & S. A. DeLoach, « Multi-agent simulation framework for large-scale coalition formation », in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, IEEE, 2016, p. 343-350.

[18] F. Klijn & A. Yazici, « A many-to-many 'rural hospital theorem' », *Journal of Mathematical Economics* **54** (2014), p. 63-73.

[19] M. Kumar, D. P. Garg & V. Kumar, « Segregation of Heterogeneous Units in a Swarm of Robotic Agents », *IEEE Transactions on Automatic Control* **55** (2010), nº 3, p. 743-748.

[20] J. E. Laird & C. B. Congdon, « The Soar User's Manual Version 9.5.0 », Tech. report, Computer Science and Engineering Department, University of Michigan, 2015.

[21] A. J. Lott & B. E. Lott, « Group cohesiveness as interpersonal attraction: A review of relationships with antecedent and consequent variables », *Psychological bulletin* **64** (1965), nº 4, p. 259-309.

[22] D. F. Manlove, « Hospitals/residents problem », in *Encyclopedia of Algorithms*, Springer, Boston, MA, 2008, p. 390-394.

[23] T. Michalak, T. Rahwan, E. Elkind, M. Wooldridge & N. R. Jennings, « A hybrid exact algorithm for complete set partitioning », *Artificial Intelligence* **230** (2016), p. 14-50.

[24] M. Morge & A. Nongaillard, « Affectation distribuée d'individus à des activités avec des préférences additivement séparables », in *Journées Francophones sur les Systèmes Multi-Agents*, Cépaduès édition, 2017, p. 19-28.

[25] M. Mrissa, L. Médini, J.-P. Jamont, N. Le Sommer & J. Laplace, « An avatar architecture for the web of things », *IEEE Internet Computing* **19** (2015), nº 2, p. 30-38.

[26] I. Muller, R. Kowalczyk & P. Braun, « Towards agent-based coalition formation for service composition », in *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IEEE, 2006, p. 73-80.

[27] T. M. Newcomb, « Some varieties of interpersonal attraction », in *Festschrift for Gardner Murphy*, Harper, 1960, p. 171-182.

[28] S. Poikonen, B. Golden & E. A. Wasil, « A branch-and-bound approach to the traveling salesman problem with a drone », *INFORMS Journal on Computing* **31** (2019), nº 2, p. 335-346.

[29] A. Qasim, A. Bader & A. Munawar, « Efficient Contract-Net Protocol For Formal Modeling Of Multi-Agent Systems », *International Journal of Computing and Digital Systems* **10** (2021), p. 805-816.

[30] T. Rahwan, T. P. Michalak, M. Wooldridge & N. R. Jennings, « Coalition structure generation: A survey », *Artificial Intelligence* **229** (2015), p. 139-174.

[31] T. Rahwan, S. D. Ramchurn, N. R. Jennings & A. Giovannucci, « An anytime algorithm for optimal coalition structure generation », *Journal of artificial intelligence research* **34** (2009), p. 521-567.

[32] G. D. O. Ramos, B. Juan C. & A. L. Bazzan, « Self-Adapting Coalition Formation Among Electric Vehicles in Smart Grids », in *2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, 2013, p. 11-20.

[33] W. Saad, Z. Han, T. Basar, M. Debbah & A. Hjorungnes, « Hedonic coalition formation for distributed task allocation among wireless agents », *IEEE Transactions on Mobile Computing* **10** (2010), nº 9, p. 1327-1344.

[34] V. G. Santos & L. Chaimowicz, « Cohesion and segregation in swarm navigation », *Robotica* **32** (2014), nº 2, p. 209-223.

[35] H. Shi & G. Xie, « Collective dynamics of swarms with a new attraction/repulsion function », *Mathematical Problems in Engineering* **2011** (2011), article no. 735248 (14 pages).

[36] J. Van Den Berg, S. J. Guy, M. Lin & D. Manocha, « Reciprocal n-body collision avoidance », in *Robotics research*, Springer, 2011, p. 3-19.

[37] L. Vig & J. A. Adams, « Market-based multi-robot coalition formation », in *Distributed Autonomous Robotic Systems 7*, Springer, 2006, p. 227-236.

[38] E. Walster, V. Aronson, D. Abrahams & L. Rottman, « Importance of physical attractiveness in dating behavior », *Journal of personality and social psychology* **4** (1966), nº 5, p. 508-516.

[39] T. Wang, L. Song, Z. Han & W. Saad, « Overlapping coalition formation games for emerging communication networks », *IEEE Network* **30** (2016), nº 5, p. 46-53.

[40] D. Y. Yeh, « A dynamic programming approach to the complete set partitioning problem », *BIT Numerical Mathematics* **26** (1986), nº 4, p. 467-474.

Abstract. — We propose a new socially inspired agent architecture adapted to a decision support system to solve a distributed problem of coalitions structure generation for product design in the circular economy. This agent-centric architecture allows agents to know with whom to form a coalition. The cognitive mechanism used by the ABSG architecture is inspired by the principles of attraction from the human and social sciences. We argue that a multi-agent paradigm and a social approach are suitable solutions for solving open and dynamic coalition formation problems. We assess this assumption by using a study case from industry

Keywords. — Isotriviality, log-selfishness, Machine.

Manuscrit reçu le 18 mars 2022, accepté le 23 novembre 2022.