MoCaNA, un agent de négociation automatique utilisant la recherche arborescente de Monte-Carlo

Cédric L R Buron, Zahia Guessoum, Sylvain Ductor, Olivier Roussel

Cédric L.R. Buron, Zahia Guessoum, Sylvain Ductor, Olivier Roussel

MoCaNA, un agent de négociation automatique utilisant la recherche arborescente de Monte-Carlo

Volume 3, nº 5-6 (2022), p. 645-669.

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 : pending

Revue Ouverte d'Intelligence Artificielle
Volume 3, n° 5-6, 2022, 645-669

MoCaNA, un agent de négociation automatique utilisant la recherche arborescente de Monte-Carlo

Cédric L.R. Buron[a], Zahia Guessoum[b,c], Sylvain Ductor[d], Olivier Roussel[e]

[a] KLaIM team, L@bisen, Yncrea Ouest, 33 Q. Chemin du Champ de Manœuvres, Carquefou, France
E-mail : cedric.buron@isen-ouest.yncrea.fr

[b] CReSTIC EA 3804, Université de Reims Champagne Ardennes, France

[c] Lip6 UMR 7606, Sorbonne Université, Paris, France
E-mail : zahia.guessoum.lip6.fr

[d] Greentea.Cloud
E-mail : sylvain.ductor@greenlux.io

[e] Kyriba Corp, San Diego, USA
E-mail : oroussel@kyriba.com.

Résumé. — La négociation automatique suscite un intérêt croissant dans la recherche en intelligence artificielle. Bien que de nombreuses hypothèses aient été explorées, les résultats proposés ne sont pas satisfaisants vis-à-vis de certaines applications. C'est notamment le cas de l'affacturage, qui propose un intéressant défi de par ses spécificités, notamment l'impossibilité de borner la négociation en termes de temps, mais aussi la taille des domaines de négociation, potentiellement infinis. Les méthodes de Monte-Carlo constituent un outil intéressant pour aborder cette application en raison de leur efficacité face à ce type d'hypothèses.

Dans cet article, nous décrivons un agent de négociation automatique, le *Monte-Carlo Negotiating Agent* (MoCaNA) dont la stratégie d'offre s'appuie sur la recherche arborescente de Monte-Carlo. MoCaNA est doté de méthodes de modélisation du comportement de l'opposant. Il est capable de négocier sur des domaines de négociation incluant des attributs discrets et continus, linéaires ou non, dans un contexte où aucune date butoir n'est spécifiée. Nous confrontons MoCaNA aux agents de l'ANAC 2014 et à d'autres agents capables de négocier sans date butoir, sur des domaines de négociation différents. Il se montre capable de surpasser ou égaler tous les agents dans un domaine sans date butoir et la majorité des finalistes de l'ANAC dans un domaine avec date butoir.

Mots-clés. — Monte-Carlo, Négociation Automatique, Agent.

1. Introduction

La négociation est une forme d'interaction dans laquelle un groupe d'agents ayant des conflits d'intérêts et un désir de coopérer essaient de trouver un accord mutuellement acceptable sur un objet de négociation [27]. Ils explorent, à cette fin, les

solutions selon un protocole prédéfini. La question de l'automatisation de la négociation, bien connue dans les domaines économiques depuis l'avènement des applications de e-commerce, a reçu une attention toute particulière dans le champ de l'intelligence artificielle et des systèmes multi-agents [23]. De nombreux mécanismes de négociation ont été proposés [31] tels que les enchères, les réseaux contractuels et les équilibres généraux. Le cadre de la négociation structure les séquences d'échange autorisées et est caractérisé par divers aspects portant notamment sur l'ensemble des participants (*e.g.,* bilatéral ou multilatéral), les préférences des agents (*e.g.,* linéaires ou non), les attributs négociés (*e.g.,* discrets ou continus). Dans chacun de ces protocoles, chaque agent utilise sa propre stratégie pour déterminer s'il accepte ou non une offre et pour constituer les offres qu'il propose. À notre connaissance, dans le cas du marchandage (en anglais *bargaining*), la plupart des stratégies proposées supposent que le protocole impose une date butoir, en temps ou en nombre de tours (voir les résultats de l'ANAC [2]). Néanmoins, dans de nombreuses applications quotidiennes, par exemple lorsque nous cherchons à acquérir un bien de consommation non critique, il est commun que le protocole ne soit associé à aucune date butoir, invalidant ainsi l'utilisation de ces stratégies.

Dans cet article, nous présentons MoCaNA (**Mo**nte-**Ca**rlo **N**egotiating **A**gent), un agent de négociation automatique conçu pour le protocole de marchandage (*bargaining*) dans le contexte de l'affacturage (voir Section 2). Dans ce protocole, deux agents s'échangent à tour de rôle des offres de manière à trouver un accord. La négociation s'interrompt lorsqu'un des deux opposants accepte ou rejette l'offre qui lui est faite par son opposant. Pour pouvoir répondre aux contraintes de ce domaine applicatif, MoCaNA a pour particularité de ne faire aucune présupposition sur : 1) l'existence d'une date butoir définie dans le cadre du protocole, 2) le caractère continu ou discret de l'espace de négociation ni 3) la linéarité des préférences. Notre approche consiste, d'une part, à formaliser le marchandage comme un jeu tel qu'étudié dans le domaine de la théorie des jeux. Étant donné le haut facteur de branchement de ce jeu, nous avons choisi d'exploiter la recherche arborescente de Monte-Carlo (MCTS) au vu de ses très bons résultats dans de nombreux jeux similaires tels que le Go [32]. D'autre part, nous avons spécialisé notre heuristique en y intégrant des techniques de modélisation de l'opposant issue de l'apprentissage automatique.

À notre connaissance, seuls De Jonge et Zhang [20] ont utilisé MCTS dans le cadre de la négociation automatique. Ils ont traité des domaines de négociations restreints, et où la fonction d'utilité de l'opposant est facilement inférable, comme la répartition d'un dollar entre deux joueurs. Nous nous concentrons sur des domaines plus complexes, où la fonction d'utilité de l'opposant ne peut être facilement inférée et où l'espace de négociation peut être grand, voire infini.

Cet article est organisé comme suit : la Section 2 présente l'application pour laquelle notre agent a été conçu ; la Section 3 introduit les travaux de la littérature traitant de négociation automatique et des méthodes de Monte-Carlo. Nous proposons ensuite une modélisation de la négociation comme un jeu dans la Section 4. La Section 5 décrit la stratégie d'offre, la stratégie d'acceptation et la modélisation d'opposant de MoCaNA. Nous présentons les résultats de la confrontation de MoCaNA aux finalistes de l'ANAC

2014 et à un certain nombre d'agents capables de négocier sans date butoir dans la Section 6. Nous résumons ces contributions et donnons des pistes d'amélioration de notre travail dans la Section 7.

2. Application visée

Ce travail s'inscrit dans le cadre d'une plate-forme *d'affacturage*. L'affacturage se distingue des contextes classiques de négociation du fait de spécificités conceptuelles auxquelles la littérature n'a pas encore répondu à notre connaissance.

Lorsqu'une entreprise vend des biens ou des services à une autre entreprise, elle émet une facture. L'entreprise vendeuse est appelée *fournisseur* et le client est appelé *donneur d'ordre*. Le donneur d'ordre ne paie généralement pas le fournisseur immédiatement, mais sous un délai qui peut être de plusieurs semaines[(1)]. Durant ce délai, le *fonds de roulement* du fournisseur est réduit ; sa capacité à produire, à honorer ses futures commandes ou à payer ses propres fournisseurs est donc affectée. Cela peut même, dans certains cas l'amener à une faillite pour défaut de paiement.

L'affacturage est une réponse commune à ce problème. Une société de financement – banque ou fonds d'investissement – appelée « *factor* », accepte de payer immédiatement les factures du fournisseur, mais en dessous du montant dû par le donneur d'ordre, dit montant nominal. Du point de vue du *factor*, les factures peuvent être vues comme des investissements à court terme, dont le risque de défaut de paiement dépend de la réputation de celui qui doit les payer – le donneur d'ordre – et non du fournisseur lui-même. Le fournisseur a généralement recours à l'affacturage lorsque le donneur d'ordre est beaucoup plus important que lui ; le risque est alors plus faible du point de vue du *factor* et le taux d'intérêt plus abordable pour le fournisseur.

Ce type de financement étant régulier, il y a un fort intérêt à automatiser la négociation entre le fournisseur et le *factor*, telle que l'atteste l'augmentation du nombre de places de marché d'affacturage dans le monde [9]. Cependant, l'affacturage présente plusieurs spécificités. La première est le domaine de la négociation. Plusieurs éléments sont négociés en même temps : le montant nominal à financer et le taux d'intérêt sont les éléments primaires de la négociation. Par ailleurs, à montants nominaux identiques, un *factor* peut demander au fournisseur de vendre les factures de certains donneurs d'ordre en qui il a davantage confiance. Enfin, lorsque plusieurs factures sont disponibles, le nombre de jours de financement prévus peut également être négocié. Nous considérons donc des questions complexes qui combinent à la fois des éléments de différentes natures : continus (le taux d'escompte), numériques (le montant numérique et les jours de financement), et catégoriels (le donneur d'ordre). La deuxième spécificité est liée à l'incertitude et à la disponibilité des ressources. La négociation automatique prend souvent en compte une date limite, qui définit le temps alloué à la négociation. La plupart des stratégies de négociation se basent sur cette pression temporelle pour calculer un taux de concession. Dans notre application, la pression temporelle n'est

[(1)] En France, le délai légal est de 30 jours mais le délai moyen de paiement est de 51 jours [26] du fait d'une asymétrie régulière entre la taille du donneur d'ordre et celle du fournisseur

pas constante au cours de la négociation. Pour le *factor*, elle dépend à la fois de l'argent qu'il a à investir et des opportunités d'investissement présentes. Si le *factor* a beaucoup d'argent et peu d'opportunités, la pression temporelle augmente : le *factor* considère l'argent non investi comme une perte. Au contraire, lorsque les ressources sont limitées et les opportunités nombreuses, le *factor* essaie d'obtenir un meilleur taux d'intérêt et la pression temporelle diminue. Pour le fournisseur, la situation est encore plus imprévisible. La pression temporelle dépend de ses possibilités d'obtenir de nouvelles lignes de crédit (y compris des prêts bancaires) et du temps que prend le donneur d'ordre pour le payer : la négociation pourrait être brutalement interrompue à un moment donné si le paiement de la facture intervient.

Par conséquent, l'affacturage est un contexte de négociation multi-attributs, où les attributs sont hétérogènes (numériques, continus et catégoriels), où les préférences des agents pour les attributs numériques ne sont pas nécessairement linéaires[(2)] et où la pression temporelle est dynamique, propre à chaque agent et non partagée. Ce contexte riche et complexe en fait un défi pour la négociation automatique.

3. Travaux connexes

MoCaNA se trouve au croisement des domaines de la négociation automatique et des techniques de Monte-Carlo appliquées aux jeux. Nous donnons dans cette section une introduction à ces deux domaines.

3.1. Négociation automatique

Les architectures d'agents de négociation automatique impliquent trois fonctionnalités [1] :

- **la stratégie d'offres** définit les offres que l'agent fait à son opposant,
- **la stratégie d'acceptation** définit si l'agent accepte la proposition qui lui a été faite ou s'il fait une contre-offre,
- **la modélisation d'opposants** permet de modéliser certains aspects de l'opposant, comme sa stratégie d'offre, son utilité ou sa stratégie d'acceptation ; elle est utilisée pour améliorer l'efficacité de la stratégie d'offres de l'agent et parfois celle de sa stratégie d'acceptation.

Le critère de négociation sans date butoir est le plus discriminant dans notre cas. À notre connaissance, on ne trouve dans la littérature que deux agents capables de négocier sans date butoir : le RandomWalker [4], le Tit-for-Tat [11]. Le RandomWalker constitue une *baseline* et le Tit-for-Tat, étant un des premiers agents du domaine, n'est pas adaptatif, et peu performant face aux travaux récents qui exploitent la modélisation de l'opposant. Le Nice Tit-for-Tat [5], bien qu'utilisant une stratégie d'offre qui ne prenne pas compte de la date butoir, en a besoin pour sa stratégie d'acceptation.

[(2)] c'est notamment le cas d'un investisseur souhaitant investir un certain montant, mais pas plus ni moins, la fonction d'utilité formant alors un « pic »

L'ensemble des autres agents, notamment ceux qui ont été utilisés dans le cadre des compétitions de l'ANAC [2] sont quant à eux totalement inadaptés au domaine de l'affacturage, puisque leurs stratégies s'appuient sur la date butoir de la négociation de manière essentielle. À notre connaissance, aucun agent de la littérature n'est capable de négocier de façon performante dans le contexte de l'affacturage.

Nous discutons ci-dessous des grandes approches pour chacune de ces trois fonctionnalités et de leurs adéquations avec notre contexte applicatif.

3.1.1. *Stratégies d'offres et d'acceptation*

Les stratégies d'offres utilisées dans le cadre de négociations complexes s'appuient sur deux familles de techniques : les heuristiques couplées au domaine et les algorithmes génétiques. Alors que les algorithmes génétiques se basent systématiquement sur une modélisation de l'opposant impliquant une date butoir, certaines approches heuristiques ne nécessitent quant à elles aucune date butoir. C'est le cas du Random-Walker, qui fait et accepte des offres de manière aléatoire, et qui est utilisé comme témoin dans [4]. C'est aussi le cas de la stratégie Tit-for-Tat, introduite par [11], et qui fait des concessions lorsque l'opposant en fait, et accepte une offre lorsque la concession qu'il fait lui amène une utilité plus faible que celle correspondant à l'offre qu'il allait faire. Enfin, une adaptation de cette stratégie a été proposée par Baarslag *et al.*, le Nice Tit-for-Tat [5]. Cette stratégie, qui repose sur une modélisation de l'utilité de l'opposant, et qui ne fait des concessions qu'en fonction de la distance au point de Nash [24], et dont seule la stratégie d'acceptation fait référence à une date butoir.

On distingue deux principales catégories de stratégies d'acceptation [4] : les stratégies myopes et les stratégies optimales. Les stratégies myopes n'exploitent que les dernières offres de l'opposant et de l'agent. Il peut s'agir par exemple d'accepter toute offre dépassant un certain seuil, toute offre meilleure que la dernière faite par l'agent lui-même ou que la prochaine générée par stratégie d'offre. Les stratégies optimales [4] exploitent un modèle de la stratégie de l'opposant afin de déterminer la probabilité d'obtenir une meilleure offre. Jusqu'à présent, seules les stratégies myopes étaient applicables dans notre contexte, les stratégies optimales proposées dans la littérature s'appuyant toujours sur une connaissance de la date butoir ou son estimation.

3.1.2. *Modélisation de l'opposant*

Baarslag *et al.* [3] proposent une catégorisation des méthodes de modélisation de l'opposant. Elles permettent de modéliser plusieurs composantes de la stratégie de l'opposant, en particulier sa stratégie d'offre et son profil de préférence.

Deux familles de techniques sont adaptées à la modélisation de **stratégies d'offres** adaptatives et non bornées : les réseaux de neurones et les techniques reposant sur l'analyse de séries temporelles. Les approches basées sur l'analyse de séries temporelles sont diverses. Parmi elles, la régression de processus gaussiens s'appuie sur les mouvements précédents pour prédire une densité de probabilité pour le tour suivant. Introduite par Rasmussen et Williams [28], elle a été utilisée avec succès par Williams

Nom	Référence	Description
Random Walker	[4]	Fait des propositions aléatoires ; accepte aléatoirement ;
Tit-for-Tat	[11]	Rend les concessions qu'on lui fait ; accepte quand l'utilité de son offre est moins bonne que la proposition de l'opposant ;
Nice Tit-for-Tat	[5]	Rend les concession qui lui sont faites en proportion de la distance au point de Nash estimé ; accepte selon une stratégie « optimale » reposant sur la connaissance de la date butoir

(a) Stratégies d'offre et d'acceptation sans date butoir

Méthode	Référence	Présupposés
Fréquence	[14]	Attributs discrets, ou définition d'une distance ; les valeurs plus fréquentes sont préférées, les attributs moins souvent modifiés ont un poids plus important
Inférence bayésienne	[18]	Attributs numériques ; l'agent opposant fait des concessions de manière régulières

(b) Modélisation de l'utilité

Méthode	Référence	Avantages/inconvénients
Réseaux de neurones	[10]	Indépendants de la modélisation de l'utilité de l'opposant ; coûteux
Stratégie dépendant de l'utilité	[3]	Peu coûteux/dépend de la modélisation de l'utilité de l'opposant

(c) Modélisation de la stratégie d'acceptation

Méthode	Référence
Réseaux de neurones	[29]
Autorégression	[7]
Régression de processus gaussiens	[34]

(d) Modélisation de la stratégie d'offre

TABLE 3.1 – Composants d'agents de négociation automatique

et al. [34]. L'aspect stochastique de cette méthode permet de couvrir plusieurs propositions possibles de l'adversaire selon la gaussienne prédite par la régression. Cela permet de prendre en compte l'incertitude que nous avons sur le modèle de l'adversaire. C'est cette méthode que nous adoptons.

L'**utilité** d'un opposant est généralement considérée comme étant une somme pondérée des utilités (potentiellement non linéaires) qu'on lui suppose pour chaque attribut (voir [3]). Deux familles de méthodes existent pour estimer, pour chaque attribut, son poids et sa fonction d'utilité supposée. La première s'applique au cas où les attributs sont discrets. Elle considère la fréquence d'apparition de chaque valeur parmi les propositions de l'opposant : les valeurs proposées le plus régulièrement par l'opposant seraient celles qu'il préfère alors que les attributs variant le plus souvent seraient ceux qui ont le moins d'importance pour lui. L'extension au cas continu complexifie le calcul, car il requiert la définition d'une fonction de distance dépendant du domaine. La seconde famille de méthodes repose sur l'apprentissage bayésien. Hindriks et Tykhonov [18] procèdent en générant des hypothèses constituées d'un ordonnancement des attributs, et d'une fonction d'utilité simple (linéaire ou linéaire par morceaux) pour chacun d'entre eux. Le poids associé à chaque attribut est calculé en fonction de l'ordre qui lui est attribué. La fonction d'utilité estimée est la somme des hypothèses pondérée par leur probabilité. Ces deux méthodes nécessitent de faire des présuppositions. La méthode par fréquence est peu adaptée à notre problème. En effet, elle n'est adaptée qu'aux attributs catégoriels, là où notre méthode doit pouvoir gérer des attributs numériques autant que catégoriels. Nous nous appuyons donc sur une méthode qui repose sur l'apprentissage bayésien.

La **stratégie d'acceptation** d'un opposant peut être apprise de deux manières. On peut d'abord faire la supposition que l'opposant acceptera une offre sous certaines conditions dépendant de sa stratégie d'offre et/ou de sa fonction d'utilité [3]. Dans ce cas, il est possible de la déduire des modèles ci-dessus sans faire de calcul supplémentaire. En l'absence de cette supposition ou de modélisation des éléments précédemment décrits, [10] propose d'utiliser des réseaux de neurones. Cela demande cependant un calcul supplémentaire, potentiellement coûteux. L'approche que nous proposons dans le cadre de cet article demande une évaluation de la stratégie d'acceptation lors des simulations de Monte-Carlo, ce qui rend cette dernière méthode inutilisable dans les faits, parce que trop lourde. Nous nous appuyons donc sur la seule autre méthode permettant de ne pas prendre en compte la date butoir, en faisant la supposition que les autres agents ont une stratégie myope, comme notre propre agent.

Malgré un grand nombre d'approches différentes, aucune de celles qui ont été exploitées dans les cadres habituels de négociation automatique, *e.g.* les compétitions de l'ANAC n'est adéquate pour le problème de l'affacturage, notamment en raison de leurs stratégies qui se basent sur l'exploitation de la connaissance d'une date butoir, connaissance impossible dans notre cas d'application. Nous proposons donc ci-dessous de pallier ce problème en développant une autre approche, basée sur les méthodes de Monte-Carlo, que nous enrichissons avec les méthodes de modélisation de l'opposant qui sont compatibles avec ces contraintes.

3.2. Méthodes de Monte-Carlo

Les méthodes de Monte-Carlo sont régulièrement utilisées comme méthodes de recherche dans les jeux. Elles ont gagné en popularité suite à leur succès dans les jeux à fort facteur de branchement, notamment le Go. En particulier, AlphaGo [32], qui a vaincu l'un des meilleurs joueurs au monde, utilise les méthodes de Monte-Carlo couplées à de l'apprentissage profond.

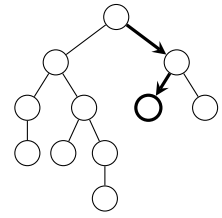
Les méthodes de Monte-Carlo procèdent en attribuant à chaque mouvement possible un score calculé au terme d'une ou de plusieurs simulations. Rémi Coulom [8] propose une méthode hybridant la construction d'un arbre de jeu avec une approche Monte-Carlo. Chaque nœud de l'arbre de jeu garde en mémoire les scores obtenus lors des simulations où il a été joué, ainsi que le nombre de simulations faites dans la branche partant du nœud. Cela permet de garder en mémoire le score moyen du nœud, et ainsi déterminer s'il correspond à un coup intéressant. Pendant l'exploration de cet arbre, les branches privilégiées sont celles qui ont été peu explorées et celles qui ont les plus hauts scores dans les simulations. L'importance donnée à chacun de ces critère permet de trancher entre les aspects d'exploration pour les branches peu explorées et d'exploitation pour les branches ayant le meilleur score, et/ou celles qui ont été le moins simulées ; cela permet en particulier d'améliorer la précision du score caclulé sur les simulations.

Cette méthode, nommée recherche arborescente de Monte-Carlo (*MCTS*[(3)]) a été améliorée par de nombreuses extensions [6]. MCTS se décompose en quatre parties, répétées jusqu'à ce qu'un critère –nombre de simulations ou temps– soit atteint. (1) La première étape consiste à parcourir l'arbre déjà constitué selon une stratégie prédéfinie. À chaque nœud, on décide s'il faut sélectionner une branche de niveau inférieur ou en explorer une nouvelle. La sélection se fait selon un critère prenant en compte le score des nœuds et le nombre de fois où ils ont été explorés (2) Dans le cas où une nouvelle branche est explorée, l'arbre se voit pourvu d'un nouveau nœud, généré selon une stratégie d'expansion. (3) Ensuite, on simule le jeu selon une politique prédéfinie, intégrant généralement de l'aléatoire, jusqu'à un état final. (4) On calcule les scores et on les rétropropage sur les nœuds de l'arbre ayant été explorés. Ce procédé est représenté sur la Figure 3.1.

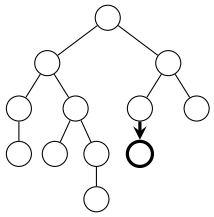
4. Jeux et négociation

La recherche arborescente de Monte-Carlo est une méthode qui s'applique aux jeux extensifs. Dans cette section, nous montrons comment il est possible d'assimiler la négociation à ces jeux. Nous commençons par associer chaque aspect de la négociation à un élément d'un jeu. Nous décrivons ensuite les particularités de la négociation, qui empêchent l'utilisation des stratégies les plus répandues dans MCTS. Nous concluons cette section en donnant d'autres formalismes possibles pour la négociation et en expliquant les relations qu'ils entretiennent avec celui que nous avons choisi.

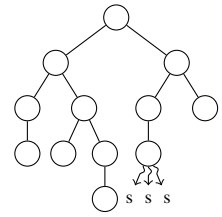
(3) *Monte Carlo Tree Search*



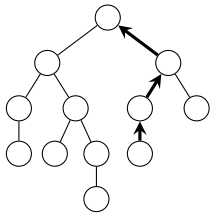
(a) Sélection de nœuds



(b) Expansion d'un nouveau nœud



(c) Simulation de parties



(d) Rétropropagation du résultat des simulations

FIGURE 3.1 – Les quatre étapes de Monte Carlo Tree Search, adapté de [6]

4.1. NOTRE FORMALISME

Un jeu sous forme extensive [25] est composé d'un ensemble de joueurs, de la description des historiques de jeu possibles, d'une fonction indiquant le tour de chaque joueur et d'un profil de préférence. En s'appuyant sur cette définition, il est possible de définir un marchandage $\mathcal{B}$ comme un jeu sous forme extensive.

DÉFINITION 4.1 (*Marchandage*). — *Un jeu de marchandage est un quintuplet* $\mathcal{B} = \langle A, \mathcal{M}, H, T, protocol \rangle$ *où :*

(1) $A = \{1, 2\}$ *est l'ensemble des joueurs participant au jeu,*
(2) $\mathcal{M}$ *est l'ensemble des messages bien formés du type* $\alpha(c)$ *où* α *est l'acte de langage (*`propose`*,* `accept`*,* `reject`*) et c le contenu du message, i.e. une liste de couples* (k, v) *où k est la clé d'un attribut du domaine de négociation et v la valeur correspondante,*
(3) H *est l'historique, c.-à-d. les séquences de messages en conformité avec la fonction tour de parole et le protocole,*
(4) $T : H \rightarrow A$ *la fonction de tour de parole. Si la taille de l'historique h est nulle ou paire alors* $T(h) = 1$. *Sinon* $T(h) = 2$,
(5) $protocol : H \rightarrow 2^{\mathcal{M}}$ *la fonction correspondant au protocole qui détermine les coups légaux qui sont autorisés pour étendre l'historique.*

En outre, les agents sont dotés d'une fonction d'utilité u_i qui associe son utilité pour chaque marchandage ayant atteint son terme. Cette fonction associe une utilité à chaque accord possible trouvé, ainsi qu'au cas de rejet et au cas d'historique infini.

Notons que ces deux dernières valeurs peuvent être différentes, par exemple si l'agent considère qu'il alloue des ressources à la négociation. Dans la suite de cet article, on appelle également cette fonction « profil de préférence » comme cela est fait dans la littérature de la négocaition automatique.

La négociation comporte certaines particularités, qui ont des conséquences sur les algorithmes qui peuvent être utilisés pour la traiter. Il s'agit tout d'abord d'un jeu à somme non nulle : les agents essaient de trouver un accord qui soit profitable à tous les deux. Dans les espaces larges, trouver un accord qui leur confère une haute utilité à tous les deux peut être particulièrement complexe, et cela peut avoir d'importantes consquences sur les algorithmes utilisés, en particulier quand des accords amènent une utilité significativement plus haute que l'utilité de réserve *i.e.* lorsque les agents ne trouvent pas d'accord.

La négociation est aussi un jeu à information incomplète, c'est-à-dire que le type des joueurs, ici leur profil de préférence, est inconnu et fait même généralement l'objet d'une modélisation. Ces deux particularités rendent inutilisable l'*Upper Confidence Tree* [21], largement utilisé pour les jeux combinatoires comme le Go car il est fait pour les jeux combinatoires. Enfin, le domaine de la négociation est très particulier. Il peut être catégoriel (*e.g.* couleur) mais également numérique, voire continu. Cela a des conséquences sur l'exploration de l'arbre, en particulier sur le critère décidant de l'expansion d'un nouveau nœud. Malgré ces difficultés d'adaptation, les succès qu'a remportés MCTS dans les jeux complexes semblent indiquer qu'il pourrait s'avérer être une bonne stratégie pour la négociation dans un contexte réaliste.

4.2. Autres formalismes

Bien que nous utilisions le formalisme des jeux extensifs pour modéliser la négociation, d'autres modélisations sont possibles, notamment en utilisant les jeux bayésiens et les jeux stochastiques.

Les **jeux à information incomplète** peuvent être modélisés en utilisant le formalisme des jeux bayésiens [30]. Ces jeux sont traditionnellement utilisés pour modéliser les jeux à information incomplète, mais supposent en général la connaissance d'une probabilité *a priori* sur les types possibles de l'opposant. Notons que dans le cadre que nous nous sommes fixés, il n'existe pas de telle distribution *a priori*. La prise en compte de l'information révélée par l'opposant au cours de la négociation, qui permet d'établir des probabilités sur le profil de préférence, se fait au cours de la négociation en utilisant la modélisation de l'opposant.

Un autre formalisme possible est la modélisation par un **jeu stochastique** [19]. Les jeux stochastiques sont une généralisation des processus markoviens. Le marchandage peut être vu comme un processus de décision markovien (MDP), où chaque action de l'agent génère une réaction de son opposant, amenant une transition vers un autre état du jeu. Dans ce nouvel état, les actions possibles restent les mêmes, mais l'utilité induite par l'acceptation de la proposition de l'opposant change. L'exploitation de ce formalisme se fait par une exploration des transitions possibles et une pondération par

leur probabilité. MoCaNA cherche à évaluer ces probabilités en utilisant la modélisation de l'opposant et estime l'utilité attendue pour les différentes transitions possibles, *i.e.* les choix de l'opposant, au moyen de MCTS. Notons que cette méthode s'est montrée particulièrement efficace pour la planification dans les MDP, comme le montre notamment [21].

5. MoCaNA

Comme nous l'avons expliqué dans la section précédente, la négociation est un jeu particulier. Il est donc nécessaire d'ajuster les heuristiques pour les jeux à ces particularités. Dans cette section, nous présentons notre agent de négociation automatique exploitant MCTS. L'architecture générale de MoCaNA est présentée dans la Figure 5.1. La stratégie d'offre implémente MCTS et utilise deux modèles de l'opposant, à savoir : un modèle de l'utilité de l'opposant et un de sa stratégie d'offre. Enfin, la stratégie d'acceptation de MoCaNA consiste en une comparaison entre la proposition de l'opposant et la proposition générée par la stratégie d'offre. Ces différents éléments et leurs interactions sont décrits dans la suite de cette section.

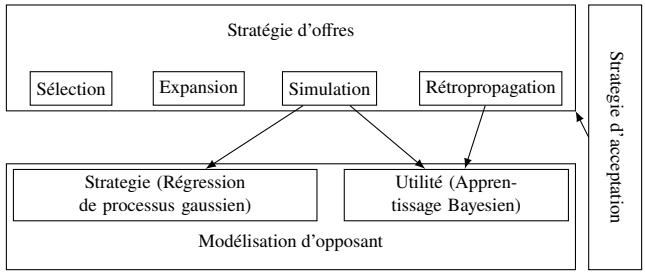


FIGURE 5.1 – Dépendances entre les modules de MoCaNA

5.1. Modélisation d'opposant

Afin d'améliorer l'efficacité de MCTS, il est nécessaire de disposer d'un modèle de la stratégie d'offre de l'opposant ainsi qu'un modèle de son profil de préférence.

5.1.1. *Modélisation de la stratégie d'offre*

Le but est de prédire une future proposition faite par l'opposant au tour x_* de la négociation. Nous utilisons la régression de processus gaussiens [28] qui permet de générer une gaussienne centrée en la valeur prédite par l'algorithme et dont l'écart type représente l'incertitude du modèle.

On commence par calculer la matrice de covariance K, qui représente la proximité entre les tours $(x_i)_{i\in[\![1,n]\!]}$ de la séquence, en nous reposant sur une fonction de covariance, aussi appelée *kernel* k.

Soit

$$K = \begin{pmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{pmatrix} \tag{5.1}$$

On calcule ensuite la distance entre le tour dont on veut estimer la proposition x_* et les tours précédents :

$$K_* = (k(x_*, x_1), \dots, k(x_*, x_n)) \tag{5.2}$$

La régression de processus gaussiens fait la supposition que chacune de ces valeurs est la composante d'une gaussienne multivariée. On en déduit :

$$\overline{y_*} = K_* K^{-1} \mathbf{y} \tag{5.3}$$

$$\sigma_*^2 = \mathrm{Var}(y_*) = K_{**} - K_* K^{-1} K_*^\top \tag{5.4}$$

où $K_{**} = k(x_*, x_*)$. Nous identifions y_* à une variable aléatoire gaussienne de moyenne $\overline{y_*}$ et d'écart type σ_*.

Une des composantes les plus importantes de la régression de processus gaussiens est la détermination du noyau (en anglais *kernel*). Les plus communs sont les fonctions à base radiale, les fonctions *rational quadratic*, le noyau (*kernel*) de Matérn et le noyau sinus quadratique exponentielle (*kernel exponential sine squared*) [28]. Ces noaux servent à déterminer la distance entre deux tours de négociation. Nous avons testé ces quatre noyaux sur plusieurs négociations entre agents. La Table 5.1 donne les résultats de la régression de processus gaussiens pour chacun de ces noyaux. Nous avons fait ces tests dans le contexte de l'ANAC 2014, où les préférences ne sont pas linéaires, et avec les finalistes de l'ANAC. Nous avons généré aléatoirement 25 sessions, et avons modélisé les deux agents, obtenant ainsi 50 modélisations au total. Chaque offre de chaque série est prédite en fonction des précédentes et sert à prédire les suivantes. La Table 5.1 montre la distance euclidienne moyenne entre les propositions faites et le résultat de l'apprentissage. Plus la valeur est basse, plus la prédiction est proche de la véritable série.

Noyau	Dist. moyenne
Fonction à base radiale	43,288
Rational quadratic	**17,766**
Matérn	43,228
Exp. sine squared	22,292

Table 5.1 – Distance moyenne entre les propositions et les prédictions de la régression de processus gaussiens selon le noyau

Le noyau donnant le meilleur résultat est le *Rational quadratic*. C'est donc celui-ci que notre agent utilise.

5.1.2. *Modèle du profil de préférence*

La méthode de modélisation du profil de préférence de l'opposant que nous utilisons est tirée de [18]. Cette méthode approxime l'utilité de l'opposant par une somme pondérée de fonctions triangulaires. Une fonction t de $[a, b] \subset \mathbb{R}$ dans $[0, 1]$ est dite triangulaire si et seulement si :

- t est affine, et $t(a) = 0$ et $t(b) = 1$ ou $t(a) = 1$ et $t(b) = 0$ ou
- il existe c dans $[a, b]$ tel que t est affine sur $[a, c]$ et sur $[c, b]$, $t(a) = 0$, $t(b) = 0$, $t(c) = 1$.

La méthode se divise en deux étapes. D'abord, un certain nombre d'hypothèses sont générées. Ces hypothèses sont composées d'une somme pondérée de fonctions triangulaires. Chaque attribut se voit attribuer un poids et une fonction triangulaire.

L'utilité estimée d'un opposant est la somme de ces hypothèses pondérée par la probabilité de chacune. Cette méthode a pour avantage de ne pas faire de supposition sur la stratégie de l'opposant, mais elle suppose qu'il fait des concessions à un rythme globalement linéaire.

Bien que cette méthode prenne l'hypothèse que l'agent fait une concession à un rythme constant, cette supposition reste faible par rapport aux méthodes basées sur les fréquences.

5.1.3. *Modèle de stratégie d'acceptation*

La stratégie d'acceptation des agents est modélisée de manière très simple : on considère qu'un agent accepte la dernière proposition faite par son opposant si et seulement si son utilité est meilleure que l'utilité générée par la stratégie d'offre. Cette modélisation, évoquée par [3], est peu coûteuse et permet de ne pas alourdir les calculs déjà nombreux de l'agent.

5.2. Stratégie d'offres basées sur MCTS

Les méthodes de Monte-Carlo sont très adaptatives, et ont obtenu de bons résultats dans différents jeux, y compris des jeux à grand facteur de branchement. Dans cette section, nous introduisons une stratégie d'offre basée sur MCTS.

5.2.1. *MCTS sans élagage*

Comme nous l'avons expliqué, MCTS est un algorithme général, qui repose sur un certain nombre de stratégies. À chaque fois que l'agent doit prendre une décision, il génère un nouvel arbre et l'explore en utilisant MCTS. L'implémentation la plus commune de MCTS est l'*Upper Confidence Tree* (UCT). Cette méthode a notamment donné de très bons résultats pour le Go. Néanmoins, lors de la phase de sélection, cette implémentation étend un nouveau nœud dès lors que tous les enfants d'un arbre n'ont pas été étendus. Or, dans le cas d'un domaine de négociation contenant un attribut

infini, le nombre de fils possibles pour un nœud est infini, puisqu'il peut prendre toutes les valeurs possibles pour cet attribut. L'algorithme étendra donc indéfiniment de nouveaux nœuds sans jamais explorer l'arbre en profondeur. Il en va de même si l'arbre est très large ou si le nombre de noeuds possibles est plus grand que le nombre de simulations.

Dans le contexte de la négociation, il est donc nécessaire de définir une implémentation de MCTS différente :

Sélection: L'objectif est de trancher le dilemme exploration/exploitation, c'est à dire de trouver un critère permettant à la fois de sélectionner les branches qui ont été peu explorées (exploration) et celles qui ont mené à la plus haute utilité lors des simulations (exploitation). Lors de l'exploration du nœud c de parent p, un nouveau nœud est dérivé –c'est à dire qu'on génère un nouveau fils de c– lorsque

$$n_p^{\alpha} \geqslant n_c \tag{5.5}$$

où n_p est le nombre de fois où le parent a été simulé, n_c est le nombre de fils qu'il a, et α est un paramètre du modèle. Ce critère est celui du *progressive widening* ; il permet de faire une exploration qui ne soit pas uniquement en profondeur : si un critère a été beaucoup exploré sans donner lieu à de nouvelles expansions, le terme de gauche de l'inégalité grandit, et on finit par générer un nouveau fils. S'il n'y a pas d'expansion, on sélectionne un nœud fils de c ; en particulier, on sélectionne le nœud i qui maximise :

$$W_i = \frac{s_i}{n_i + 1} + C \times n^{\alpha} \sqrt{\frac{\ln(n)}{n_i + 1}} \tag{5.6}$$

où n est le nombre total de simulations faites jusqu'alors, s_i est le score du nœud i, n_i le nombre de fois où i a été exploré et C est également un paramètre du modèle. La valeur de α est calculée de manière à explorer un arbre suffisamment profond en un temps raisonnable au vu du cas d'application (voir la Section 6.2). Ce critère permet de par le premier terme de sélectionner en priorité les branches ayant un score moyen élevé et de par le second de privilégier les branches peu sélectionnées, puisque le nombre de sélections de la branche est dans le dénumérateur de ce terme.

Expansion: La valeur d'un nœud étendu est quant à elle choisie de manière aléatoire, avec une distribution uniforme sur le domaine de négociation.

Simulation: La simulation part du nœud étendu et poursuit une négociation entre l'opposant et MoCaNA jusqu'à ce qu'un accord soit trouvé ou qu'un des agents refuse définitivement une offre. Nous utilisons le modèle de stratégie de l'opposant basée sur la régression de processus gaussiens et présentée dans la Section 5.1.1, afin de rendre les simulations plus pertinentes.

Rétropropagation: L'étape de rétropropagation utilise aussi le modèle de l'utilité de l'opposant. Pour chaque nœud ayant été exploré dans l'arbre (y compris le nœud étendu), on incrémente le nombre d'explorations du nœud, et on corrige l'utilité moyenne de l'agent pour ce nœud, en fonction de l'accord trouvé.

On met aussi à jour l'utilité moyenne de l'opposant selon la modélisation d'utilité basée sur l'inférence bayésienne, présentée dans la Section 5.1.2.

Enfin, MCTS fait des simulations et calcule l'utilité attendue de toute la négociation. Il a tendance à sous-estimer la valeur de la proposition qu'il ressort, et la probabilité que l'opposant l'accepte. Nous renvoyons donc l'offre b^* parmi les fils de la racine maximisant à la fois l'utilité de l'offre faite par l'agent lui même, et l'utilité attendue à la fin de la négociation. Plus formellement, on maximise :

$$b^* = \text{argmax}_{b \in racine.fils} \left(\frac{utility(b) + score(b)}{2} \right) \tag{5.7}$$

où $utility(b)$ est l'utilité de l'offre b pour l'agent, et *score* est le score calculé par MCTS (identique au s_i de l'Equation (5.6)).

5.2.2. *Élagage*

Afin de n'explorer que les nœuds donnant une utilité importante à l'agent, il est possible d'utiliser des connaissances sur le jeu pour élaguer certaines parties de l'arbre et ne pas les développer. Nous considérons deux cas, un élagage dit « fixe » et un élagage dit « variable ».

Dans le cas de l'**élagage fixe**, nous fixons à notre agent une utilité minimale attendue, *i.e.* une utilité de réserve. Au sein de la simulation, lorsque MoCaNA produit une proposition, l'utilité de cette proposition est calculée en utilisant le profil d'utilité de MoCaNA. Si cette utilité est inférieure à la valeur seuil, la branche est supprimée.

Lorsque l'agent n'est pas capable de déterminer un prix de réserve, il est possible de procéder à un **élagage variable** dépendant des propositions de l'opposant. Dans ce cas, l'agent garde en mémoire la proposition de l'opposant ayant pour lui la meilleure utilité. Toute proposition générée par MoCaNA dans une simulation ayant une utilité inférieure est élaguée. Avec cette politique, l'agent ne fait que des propositions meilleures que celles que son opposant a faites.

5.3. Architecture logicielle

Pour évaluer notre agent, nous utilisons le *framework* Genius [22], qui permet de créer des sessions de négociation et des tournois. Notre agent a été développé en Java. Nous l'avons conçu de façon à ce qu'il soit modulaire, et à ce qu'il soit découplé du *framework* sur lequel les expérimentations ont été réalisées. Nous avons pour cela construit un module permettant à notre agent de s'interfacer avec Genius. L'agent est organisé sous la forme d'interfaces implémentées par les classes de chaque module, et qui sont traduites pour Genius le cas échéant. On retrouve dans l'architecture de MoCaNA les différents éléments décrits ci-dessus : la stratégie d'offre incluant MCTS et les concepts allant avec (*e.g.* notions d'arbre, de nœud), un module de modélisation de la stratégie de l'opposant et un modèle de l'utilité. Nous n'avons pas implémenté de module pour la modélisation de la stratégie d'acceptation. Les différents

modules et composants de notre implémentation sont représentés dans la Figure 5.2. Les enveloppes (✉) représentent les messages envoyés et reçus par l'agent.

Les simulations de MCTS étant très gourmandes en termes de puissance de calcul, nous les avons parallélisées. L'aspect stochastique de notre modélisation de stratégie d'offre assure que deux simulations lancées du même point n'auront pas le même résultat, tout en privilégiant les valeurs les plus probables. La régression de processus gaussien, qui repose sur le calcul matriciel, a été développée en utilisant la librairie JAMA [17]. Les paramètres du noyau de cette méthode sont quant à eux optimisés en utilisant la librairie Apache Commons Math [33].

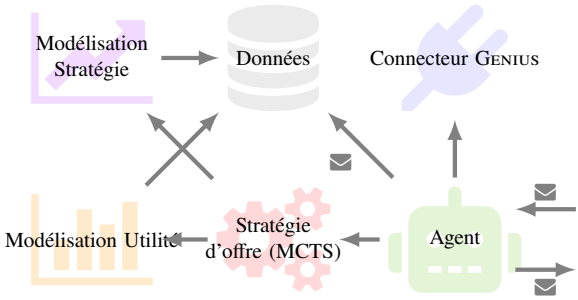


Figure 5.2 – Architecture logicielle de MoCaNA

6. Expérimentations

Nous confrontons MoCaNA à trois agents négociant sans date butoir ainsi qu'aux agents de l'ANAC 2014. Dans cette section, nous présentons d'abord ces agents. Nous décrivons ensuite notre protocole expérimental, avant de présenter et de commenter nos résultats.

6.1. Description des opposants négociant sans date butoir

Dans cette section, nous fournissons une description des agents auxquels nous confrontons MoCaNA. Les trois seuls agents capables de négocier sans date butoir sont le RandomWalker, l'agent Tit-for-Tat et sa variante plus évoluée, le Nice Tit-for-Tat.

6.1.1. *RandomWalker*

Le RandomWalker est décrit par Baarslag dans [4] et fait des propositions aléatoires.

6.1.2. *Tit-for-Tat*

L'agent *Tit-for-Tat* a été décrit pour la première fois par Faratin *et al.* dans [11]. Cet agent fait une concession chaque fois que son opposant en fait une lui-même. Plusieurs

implémentations possibles sont présentées dans l'article original. Ici, lorsqu'il a reçu moins de 2 propositions, l'agent fait la proposition la plus intéressante de son point de vue (il génère 10 000 propositions aléatoires et choisit la meilleure pour sa fonction d'utilité). Pour les autres propositions, l'agent regarde les deux dernières propositions de son opposant et calcule la concession faite en termes d'utilité de son point de vue. Cette concession peut être positive ou négative. Il ajoute cette concession à l'utilité de sa dernière proposition. Il recherche ensuite une proposition dont l'utilité est la plus proche de cette valeur cible. Dans notre implémentation, 10 000 propositions sont générées à cette fin.

6.1.3. *Nice Tit-for-Tat*

Le *Nice Tit-for-Tat* est d'une certaine manière une version plus évoluée du *Tit-for-Tat*. Il a été décrit par Baarslag *et al.* dans [5]. L'objectif de ce travail est de se conformer aux domaines où un accord mutuel est possible. L'agent utilise alors la même modélisation de l'utilité de l'opposant que notre agent et s'en sert pour estimer le point de Nash du cadre, c'est-à-dire l'accord maximisant le produit des utilités des agents. Le taux de concession est calculé entre la première et la dernière offre de l'opposant comme le pourcentage d'utilité concédée de la distance entre sa première offre et le point de Nash. La concession correspondante est faite du point de vue de l'agent. Parmi les offres équivalentes, le modèle d'utilité est également utilisé afin de choisir la meilleure offre pour l'opposant parmi les offres équivalentes pour l'agent lui-même. La seule différence entre la version utilisée dans nos expériences et l'agent proposé dans [5] est la stratégie d'acceptation. En effet, la stratégie proposée dans la version originale qui est présentée dans [4] dépend de la date butoir de la négociation afin de prendre en compte la pression temporelle. Ici, nous fournissons au *Nice Tit-for-Tat* une version simplifiée de sa stratégie d'acceptation, qui correspond à la même stratégie d'acceptation que notre agent.

6.2. Résultats

Nous utilisons le domaine de négociation de l'ANAC 2014 décrit par [12] car c'est celui qui correspond le mieux à notre contexte[(4)]. Les attributs sont entiers[(5)], et varient de 1 à 10. Plusieurs variantes ont été proposées allant de 10 attributs à 50 attributs. Nous nous focalisons sur le contexte à 10 attributs. Nous n'appliquons aucun taux de réduction (en anglais *discount rate*) à l'accord au cours de la négociation. Notre choix se base pour cela sur le fait que ce *discount rate* est généralement lié à la présence d'une date butoir, au-delà de laquelle les accords ont tous une utilité négative ou nulle. L'utilité est une somme pondérée de fonctions non linéaires sur chaque attribut. L'utilité de réserve des agents (dans le cas où ils ne trouvent pas d'accord) est 0, c'est-à-dire la

[(4)]Les compétitions de 2015 et de 2016 se sont concentrées respectivement sur la négociation multilatérale, et sur les *smart grids* et les sessions suivantes se sont concentrées sur la négociation humain-agent, le jeu de Diplomacy et la négociation multi-latérale

[(5)]Genius permet de négocier sur des attributs entiers ou discrets, mais il ne permet pas encore de négocier sur des attributs continus.

Opposant	sans élagage	élagage fixe	élagage variable
AgentM	0,35	0,15	0,10
DoNA	0,05	0	0,10
Gangster	0,45	0,55	0,55
Whale	0,45	0,45	0,25
Group2	0,45	0,45	0,45
kGAgent	0,20	0	0,05
AgentYk	0,15	0,10	0,15
BraveCat	0,95	0,95	1

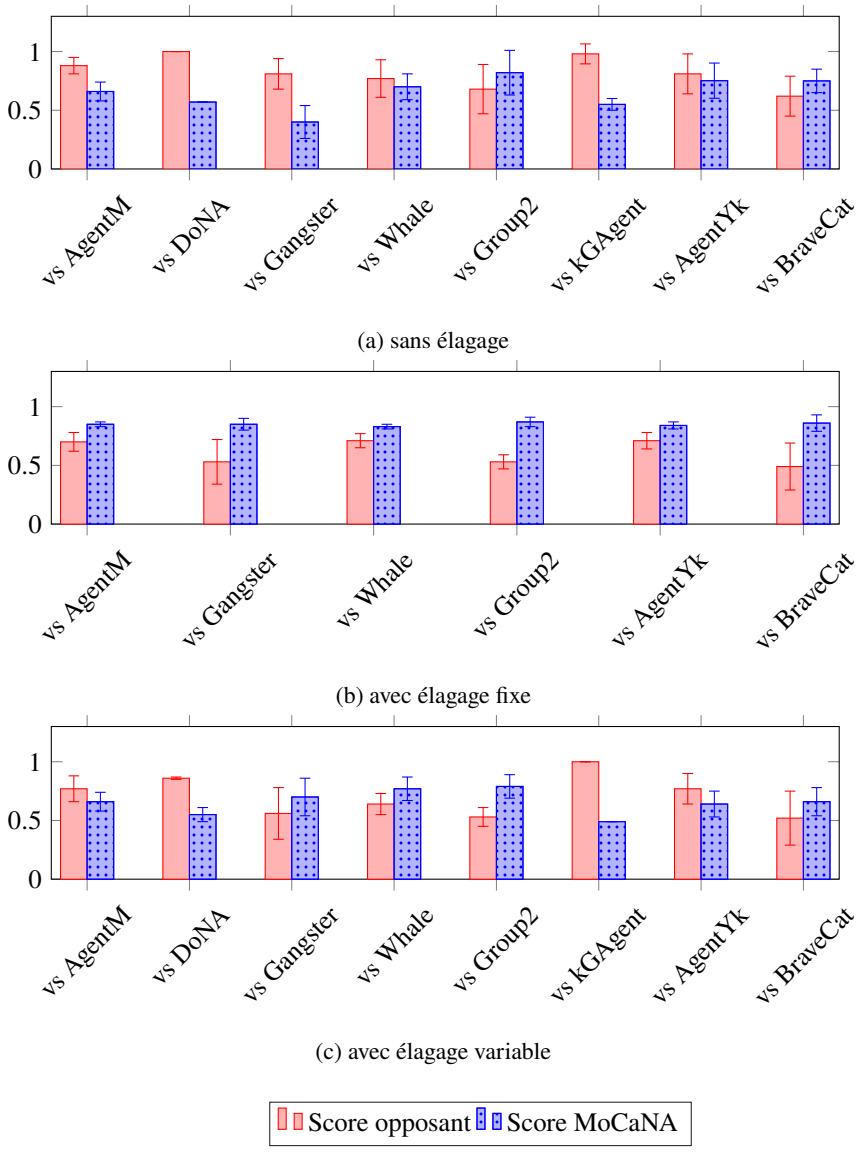
TABLE 6.1 – Taux d'acceptation des négociations entre MoCaNA et les finalistes d'ANAC 2014

valeur minimale. Les profils de préférence sont des fonctions d'utilité définies par [12]. Ces profils sont définis à travers des valeurs d'utilité (entre 0 et 1) sur des hypercubes du domaine de négociation, ici un espace de 10 attributs dont les valeurs varient entre 1 et 10. Deux profils ont été définis pour la compétition, c'est à ceux-là que nous nous référons dans cette section.

Nous distinguons deux suites d'expérimentations : la première (section 6.3.1), face aux agents de l'ANAC 2014, où la date butoir est présente et connue des opposants, et la deuxième (section 6.3.2), face aux agents de la section 6.1, où la date butoir est assez loin pour être considérée comme inexistante. Nous fixons dans la première la date butoir de chaque session de négociation à 1 heure. Après ce temps, la négociation s'interrompt et les agents obtiennent leur utilité de réserve, c'est-à-dire 0. Nous calibrons notre modèle de manière empirique. Lorsque MOCaNA peut choisir parmi 200 propositions, il s'en trouve généralement une qui procure à la fois une haute utilité pour lui et son opposant. Afin que l'arbre soit également exploré en profondeur, nous lui laissons assez de temps pour qu'il génère en moyenne 50 000 simulations. Ainsi, pour faire une proposition, notre agent prend 3 minutes ; le temps de calcul des autres agents est négligeable, inférieur à 1 seconde. Nous obtenons pour la valeur d'α dans l'équation du *progressive widening* (voir l'Equation (5.5)) $\alpha = 0{,}489$. En ce qui concerne le contexte sans date butoir, nous utilisons le même domaine de négociation, mais où nous fixons une date butoir assez grande pour qu'elle ne soit pas atteinte.

6.2.1. *ANAC 2014*

La Figure 6.1 montre les résultats de notre agent face aux finalistes de l'ANAC 2014, décrits par [13], ordonnés selon leurs résultats à l'ANAC 2014 dans la catégorie utilité individuelle. Les scores représentent l'utilité moyenne de notre agent et de son opposant sur 20 négociations, 10 avec chaque profil. Plus le score est haut, plus l'utilité de l'agent est élevée et mieux l'agent a réussi la négociation. Les barres représentent l'écart type de la série. L'utilité moyenne est calculée en ne prenant en compte que les négociations



(a) sans élagage

(b) avec élagage fixe

(c) avec élagage variable

FIGURE 6.1 – Utilité moyenne des finalistes de l'ANAC 2014 face à MoCaNA

qui ont réussi. La Table 6.1 représente le taux d'accord atteint par les agents dans les différents contextes, c'est-à-dire le nombre de fois où les agents parviennent à un accord divisé par le nombre total de négociations. La Figure 6.1b ne contient que 6 colonnes, la négociation avec DoNA et kGAgent n'ayant jamais abouti.

Le taux d'accord très faible peut être expliqué par notre stratégie, qui ne s'accorde pas du tout à la distance à la date butoir, puisqu'elle ne la prend pas en compte. Cette caractéristique l'empêche de ressentir la pression du temps (en anglais *time pressure*), c'est à dire la pression que les agents ont à trouver un accord rapidement qui est utilisée par les autres agents pour décider s'ils concèdent beaucoup ou non. Plus de 50 % des négociations se soldent par un échec (contre 45 % pour les autres agents).

Les deux versions de l'élagage amènent une amélioration significative. Avec l'élagage par une valeur fixe, notre agent ne fait et n'accepte que des propositions pour lesquelles il recevrait une utilité de 0,8 au moins. Cela a un effet important sur le résultat des négociations. Celles qui se terminent permettent à notre d'agent d'obtenir une bonne utilité, mais elles sont moins nombreuses que dans le cas précédent, en particulier en ce qui concerne les agents avec lesquels notre agent avait déjà un taux d'accord faible.

L'autre méthode d'élagage est moins dure pour les opposants, puisque l'élagage repose sur les propositions qu'ils font tout au long de la négociation. Ainsi, pour un opposant n'ayant fait que des propositions dont l'utilité est inférieure ou égale à 0,7, on n'élaguera que les propositions dont l'utilité est inférieure ou égale à 0,7. L'agent est en revanche moins complaisant avec ses opposants, et n'accepte pas des offres ayant une forte utilité pour l'opposant et une faible utilité pour lui, ce qui explique la baisse de l'utilité des opposants. L'utilité de notre agent reste comparable mais est moins haute, en revanche, que dans le cas avec un élagage fixe de 0,8. Notre agent bat la moitié de ses compétiteurs.

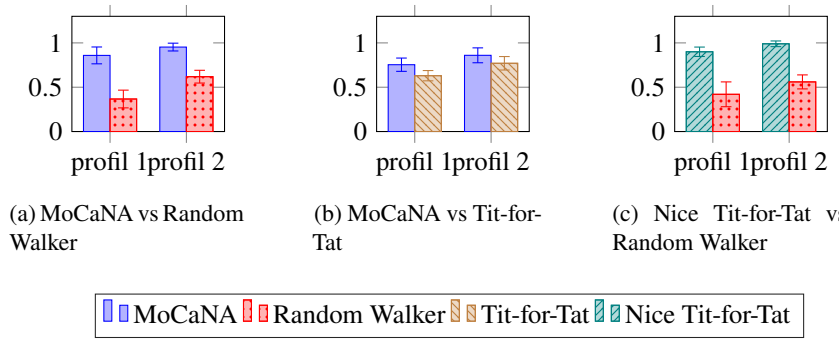
Quelle que soit la technique d'élagage, elle permet aux agents de ne pas explorer les branches inintéressantes, et donc de rechercher les meilleures solutions parmi les solutions acceptables.

6.2.2. *Contexte sans date butoir*

La Figure 6.2 affiche l'utilité des agents lorsqu'ils négocient entre eux à l'aide d'un histogramme. Les résultats sont une moyenne sur 20 sessions de négociation avec chaque profil, les barres d'erreur représentant l'écart type par rapport à la moyenne.

Notons que les deux profils de préférence sont très différents l'un de l'autre, et pas du tout symétriques. Cette spécificité explique le fait que pour tous les agents dans toutes les configurations, l'utilité est toujours plus élevée avec le Profil 2 qu'avec le Profil 1.

Comme le montre Figure 6.2a, notre agent est capable de battre le Random Walker dans toutes les situations, même lorsque son profil de préférence est le Profil 1 et celui du Random Walker le Profil 2. Il est intéressant de noter que les négociations avec le Random Walker sont très courtes avec seulement 3,1 propositions en moyenne : 2,5 quand il a le Profil 1 et 3,7 propositions quand notre agent a le Profil 2. Cette différence peut s'expliquer par le fait qu'il est facile de trouver des accords avec des résultats très élevés pour le Profil 1 (0,9 ou plus) et des résultats élevés pour le Profil 2 (0,6). Dans la plupart des négociations, la première proposition de notre agent est de ce type. Dans



(a) MoCaNA vs Random Walker

(b) MoCaNA vs Tit-for-Tat

(c) Nice Tit-for-Tat vs Random Walker

FIGURE 6.2 – Utilité moyenne des agents ne reposant pas sur une date butoir face à MoCaNA

ce cas, Random Walker est plus susceptible de générer une proposition avec une utilité inférieure à celle proposée par notre agent et l'accepte, générant une utilité de 0,6 pour Random Walker et une session de négociation consistant en une seule proposition.

Négocier avec le *Tit-for-Tat* est plus difficile, comme on peut le voir sur la Figure 6.2b. Notre agent a une utilité plus faible que contre le RandomWalker mais est capable de battre le *Tit-for-Tat*. Le niveau d'attente de *Tit-for-Tat* génère également des négociations beaucoup plus longues : 34,2 propositions en moyenne avec 31,55 propositions lorsqu'il a le Profil 1 et 36,85 propositions lorsque notre agent a le Profil 2. Ce résultat peut être expliqué de la même manière que les résultats de la négociation avec le Random Walker.

La négociation avec le *Nice Tit-for-Tat* ne se termine parfois jamais : les agents continuent à négocier indéfiniment. Notre méthode basée sur MCTS refuse de faire une concession suffisamment importante pour avoir une chance d'être acceptée par le *Nice Tit-for-Tat*, compte tenu de la forte espérance qu'elle a en utilisant le point de Nash. Réciproquement, le *Nice Tit-for-Tat*, sans pression temporelle, et sans adaptation dynamique de sa stratégie d'acceptation, n'accepte pas les propositions de notre agent. En observant l'état interne du *Nice Tit-for-Tat*, nous constatons également que son estimation du point de Nash est incorrecte : il attend une utilité supérieure à la réalité.

Nous proposons à la place une évaluation indirecte en confrontant Nice Tit-for-Tat à Random Walker, dans le même cadre. Les résultats sont représentés sur Figure 6.2c. Les performances des deux agents sont comparables, lorsqu'on prend en compte l'écart-type des séries.

7. CONCLUSION

Dans cet article, nous avons présenté un agent de négociation automatique capable de négocier dans un contexte sans date butoir, ni temporelle ni en termes de nombre de tours, et où le domaine de négociation peut être continu. Après avoir décrit la négociation sous forme de jeu extensif, nous avons décrit une stratégie d'offre s'appuyant

sur une version de MCTS et sur deux méthodes de modélisation de l'opposant : la régression de processus gaussiens pour la modélisation de la stratégie d'offre et l'apprentissage bayésien pour la modélisation du profil de préférence. Pour la régression de processus gaussiens, nous avons testé plusieurs noyaux et retenu le meilleur. Nous avons proposé deux variantes à notre stratégie d'offre s'appuyant sur des élagages des branches.

Aucun agent n'est conçu pour négocier dans un contexte aussi réaliste que celui pour lequel MoCaNA a été conçu. Nous l'avons donc comparé aux finalistes de l'ANAC 2014, dont le contexte est le plus proche de celui pour lequel notre agent est conçu et à trois agents conçus pour négocier sans date butoir, le RandomWalker et deux variations du *Tit-for-Tat*. Les expérimentations ont montré qu'il est difficile d'obtenir un haut taux d'accord pour un agent n'ayant pas conscience de la *time pressure*. Dans les cas où un accord est trouvé, l'élagage améliore grandement la performance de MoCaNA et lui permet de battre la majorité de ses opposants, au prix d'une baisse du taux d'acceptation. Dans le contexte sans date butoir, les résultats expérimentaux sont également prometteurs : MoCaNA surpasse Random Walker et Tit-for-Tat, et obtient des résultats comparables au *Nice Tit-for-Tat*. Ce travail indique que les méthodes issues de l'intelligence artificielle pour les jeux telles que les MCTS peuvent être utilisées avec succès dans la négociation automatisée. Certains éléments tels que la modularité de l'architecture, des stratégies générales telles que celles utilisées dans le domaine du *General Game Playing*, ou encore les méthodes d'apprentissage automatique constituent un point de départ intéressant pour concevoir un agent capable de négocier dans d'autres domaines d'application spécifique lorsque ceux-ci diffèrent de celui généralement traité par l'état de l'art.

Les perspectives de ces travaux comprennent tout d'abord l'application de ces travaux à des domaines plus larges, comme ceux à 50 attributs proposés par l'ANAC. Une autre extension serait l'élargissement du spectre de domaines sur lequel MoCaNA est capable de négocier, notamment en testant des domaines de négociation formés d'attributs soit purement catégoriels ou continus soit mixtes. Une autre perspective consisterait à concevoir une version de l'ANAC sans date butoir. Il serait enfin intéressant de voir dans quelle mesure MoCaNA pourrait intégrer des informations supplémentaires telles que la date butoir, des connaissances sur la nature des attributs afin d'en tirer parti lorsque cela est possible. Nous souhaitons aussi étendre le travail aux protocoles multilatéraux, notamment des protocoles 1:*n* de type enchères ou *n*:*m* de type *many-to-many bargaining*.

De plus, il est possible d'améliorer MoCaNA en se concentrant sur MCTS. Il serait par exemple intéressant de voir comment un double élagage, par une valeur fixe et selon les propositions de l'opposant, influerait sur l'utilité et le taux d'accord de MoCaNA. Une autre amélioration, qui permettrait d'augmenter le nombre de simulations de MoCaNA et donc de diminuer le temps de calcul ou d'augmenter sa performance serait d'implémenter des algorithmes comme *All Moves As First* [16] ou la *Rapid Action Value Estimation* [15]. En ce qui cocnerne les méthodes de modélisation de l'adversaire, il pourrait être intéressant, en particulier dans le cadre d'une négociation sans date butoir,

de proposer une méthode de moédlisation de la stratégie adverse qui ne repose pas sur les régularités des propositions de l'adversiare. De même, proposer d'autres types de fonctions que des fonctions traingulaires dans l'apprentissage bayésien permettrait de généraliser ce que la méthode proposée permet de modéliser.

8. Remerciements

Nous remercions la région Île de France qui a financé ces travaux dans le cadre du projet FUI *Risk, Credit Line and Supply Chain Management* ainsi que l'ANRT pour le financement de la thèse qui y a été associé.

Bibliographie

[1] T. Baarslag, « Exploring the Strategy Space of Negotiating Agents: A Framework for Bidding, Learning and Accepting in Automated Negotiation », Thèse, Delft University of Technology, 2016.

[2] T. Baarslag, R. Aydoğan, K. V. Hindriks, K. Fujita, T. Ito & C. M. Jonker, « The automated negotiating agents competition, 2010–2015 », *AI Magazine* **36** (2015), nº 4, p. 115-118.

[3] T. Baarslag, M. J. C. Hendrikx, K. V. Hindriks & C. M. Jonker, « Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques », *Autonomous Agents and Multi-Agent Systems* **20** (2015), nº 1, p. 1-50.

[4] T. Baarslag & K. V. Hindriks, « Accepting Optimally in Automated Negotiation with Incomplete Information », in *AAMAS '13* (Richland, SC), International Foundation for Autonomous Agents and Multiagent Systems, 2013, p. 715-722.

[5] T. Baarslag, K. V. Hindriks & C. Jonker, « A Tit for Tat Negotiation Strategy for Real-Time Bilateral Negotiation », in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, vol. 435, Springer Berlin Heidelberg, 2013, p. 229-233.

[6] C. C. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavernier, D. Perez, S. Samothrakis & S. Colton, « A Survey of Monte Carlo Tree Search Methods », *IEEE Transactions on Computational Intelligence and AI in games* **4** (2012), nº 1, p. 1-43.

[7] S. Chen & G. Weiss, « A Novel Strategy for Efficient Negotiation in Complex Environments », in *Multiagent System Technologies: 10th German Conference Proceedings*, Springer Berlin Heidelberg, 2012, p. 68-82.

[8] R. Coulom, « Efficient selectivity and backup operators in Monte-Carlo tree search », in *International conference on computers and games*, Springer, 2006, p. 72-83.

[9] D. T. Dziuba, « Crowdfunding Platforms in Invoice Trading as Alternative Financial Markets », *Roczniki Kolegium Analiz Ekonomicznych/Szkoła Główna Handlowa* **49** (2018), p. 455-464.

[10] F. Fang, Y. Xin, Y. Xia & X. Haitao, « An Opponent's Negotiation Behavior Model to Facilitate Buyer-seller Negotiations in Supply Chain Management », in *2008 International Symposium on Electronic Commerce and Security*, 2008.

[11] P. Faratin, N. R. Jennings & C. Sierra, « Negotiation decision functions for autonomous agents », *Robotics and Autonomous Systems* **24** (1998), nº 3-4, p. 159-182.

[12] K. Fujita, R. Aydoğan, T. Baarslag, T. Ito & C. Jonker, « The Fifth Automated Negotiating Agents Competition (ANAC 2014) », in *Recent Advances in Agent-based Complex Automated Negotiation* (N. Fukuta, T. Ito, M. Zhang, K. Fujita & V. Robu, éds.), Springer International Publishing, Cham, 2016, p. 211-224.

[13] N. Fukuta, T. Ito, M. Zhang, K. Fujita & V. Robu (éds.), *Recent Advances in Agent-based Complex Automated Negotiation*, Studies in Computational Intelligence, vol. 638, Springer International Publishing, 2016.

[14] N. van Galen Last, « Agent Smith: Opponent Model Estimation in Bilateral Multi-issue Negotiation », in *New Trends in Agent-Based Complex Automated Negotiations* (T. Ito, M. Zhang, V. Robu, S. Fatima & T. Matsuo, éds.), Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, p. 167-174.

[15] S. Gelly & D. Silver, « Monte-Carlo tree search and rapid action value estimation in computer Go », *Artificial Intelligence* **175** (2011), nº 11, p. 1856-1875.

[16] D. P. Helmbold & A. Parker-Wood, « All-Moves-As-First Heuristics in Monte-Carlo Go », in *IC-AI*, Citeseer, 2009, p. 605-610.

[17] J. Hicklin, C. Moler, P. Webb, R. F. Boisvert, B. Miller, R. Pozo & K. Remington, « Jama: A Java matrix package », 2000, `http://math.nist.gov/javanumerics/jama`.

[18] K. Hindriks & D. Tykhonov, « Opponent Modelling in Automated Multi-issue Negotiation Using Bayesian Learning », in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 1, 2008, p. 331-338.

[19] A. Jaśkiewicz & A. S. Nowak, « Non-Zero-Sum Stochastic Games », in *Handbook of Dynamic Game Theory* (T. Basar & G. Zaccour, éds.), Springer International Publishing, Cham, 2016, p. 1-64.

[20] D. de Jonge & D. Zhang, « Automated Negotiations for General Game Playing », in *AAMAS '17* (Richland, SC), 2017, p. 371-379.

[21] L. Kocsis & C. Szepesvári, « Bandit Based Monte-Carlo Planning », in *Machine Learning: ECML 2006* (Berlin, Heidelberg) (J. Fürnkranz, T. Scheffer & M. Spiliopoulou, éds.), Springer Berlin Heidelberg, 2006, p. 282-293.

[22] R. Lin, S. Kraus, T. Baarslag, D. Tykhonov, K. Hindriks & C. M. Jonker, « Genius: an integrated environment for supporting the design of generic automated negotiators », *Computational Intelligence* **30** (2014), nº 1, p. 48-70.

[23] M. Morge, « Négociation bilatérale par concession : un état de l'art », in *Vingt-sixièmes journées francophones sur les systèmes multi-agents*, Cépaduès éditions, 2018, p. 75-84.

[24] J. F. Nash Jr, « The bargaining problem », *Econometrica: Journal of the econometric society* (1950), p. 155-162.

[25] M. J. Osborne & A. Rubinstein, *A course in game theory*, 12 éd., MIT press, 1994.

[26] J.-M. Prost & J.-P. Villetelle, « Rapport annuel des délais de paiement en 2019 », Tech. report, Observatoire des délais de paiement, 07 2020.

[27] D. G. Pruitt, *Negotiation Behavior*, Academic Press, 1981.

[28] C. E. Rasmussen & C. K. I. Williams, *Gaussian processes for machine learning*, MIT Press, 2006.

[29] H. Rau, M.-H. Tsai, C.-W. Chen & W.-J. Shiang, « Learning-based automated negotiation between shipper and forwarder », *Computers & Industrial Engineering* **51** (2006), nº 3, p. 464 - 481, Special Issue on Selected Papers from the 34th. International Conference on Computers and Industrial Engineering (ICC&IE).

[30] D. M. Reeves & M. P. Wellman, « Computing Best-response Strategies in Infinite Games of Incomplete Information », in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* (Arlington, Virginia, United States), UAI '04, AUAI Press, 2004, p. 470-478.

[31] T. W. Sandholm, « Distributed rational decision making », *Multiagent systems: a modern approach to distributed artificial intelligence* (1999), p. 201-258.

[32] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser et al., « Mastering the game of Go with deep neural networks and tree search », *Nature* **529** (2016), nº 7587, p. 484-489.

[33] « The apache commons mathematics library », online, 2016, see `https://commons.apache.org/proper/commons-math/`.

[34] C. R. Williams, V. Robu, E. H. Gerding & N. R. Jennings, « Using Gaussian Processes to Optimise Concession in Complex Negotiations Against Unknown Opponents », in *IJCAI'11*, 2011, p. 432-438.

ABSTRACT. — Automated negotiation is of growing interest in artificial intelligence research. While research has focused numerous contexts, some applications have not benefited from these advances. This is particularly the case for factoring, which offers an interesting challenge due to its specificities, notably the impossibility to limit the negotiation in terms of time, and negotiation on vast, even infinite domains. Monte-Carlo methods are an interesting tool for solving this problem because of their effectiveness with such hypotheses.

In this paper, we introduce a Monte Carlo Negotiating Agent (MoCaNA) whose bidding strategy relies on Monte Carlo Tree Search. We endow MoCaNA with opponent modeling techniques for bidding strategy and utility. MoCaNA can negotiate on continuous domains and in a context where no bound is specified. We confront MoCaNA with both the finalists of ANAC 2014 and to agents that are able to negotiate without bounds on different negotiation domains. MoCaNA outperforms or ties all the agents in a domain without bound and the majority of the ANAC finalists in a domain with a bound.

KEYWORDS. — Monte Carlo, Automated Negotiation, Agent.

Manuscrit reçu le 15 juillet 2021, révisé le 31 janvier 2022, accepté le 23 mars 2022.