HAL
open science

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

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

To cite this version:

Cédric L R Buron, Zahia Guessoum, Sylvain Ductor, Olivier Roussel. MoCaNA, un agent de négociation automatique utilisant la recherche arborescente de Monte-Carlo. Vingt-sixièmes Journées Francophones sur les Systèmes Multi-Agents, Oct 2018, Métabief, France. hal-01895773

HAL Id: hal-01895773

https://hal.science/hal-01895773

Submitted on 15 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Cédric L.R. Buron [*1,2], Zahia Guessoum [†1,3], Sylvain Ductor [‡4], and Olivier Roussel [§5]

[1] *LIP6, Sorbonne Université, Paris, France*
[2] *Thales Research and Technology, Palaiseau, France*
[3] *CReSTIC, Université de Reims, Reims, France*
[4] *Universidade Estadual do Ceará, Fortaleza, Brésil*
[5] *Kyriba Corp, San Diego, USA*

11 octobre 2018

Résumé

La négociation automatique est un sujet qui suscite un intérêt croissant dans la recherche en IA. Les méthodes de Monte-Carlo ont quant à elles vécu un grand essor, notamment suite à leur utilisation sur les jeux à haut facteur de branchement tel que le go.

Dans cet article, nous décrivons un agent de négociation automatique, Monte-Carlo Negotiating Agent (MoCaNA) dont la stratégie d'offre s'appuie sur la recherche arborescente de Monte Carlo. Nous munissons cet agent de méthodes de modélisation de la stratégie et de l'utilité adverse. MoCaNA est capable de négocier sur des domaines de négociation continus et dans un contexte où aucune borne n'est spécifiée. Nous confrontons MoCaNA aux agents de l'ANAC 2014 et à un *RandomWalker* sur des domaines de négociation différents. Il se montre capable de surpasser le *RandomWalker* dans un domaine sans borne et la majorité des finalistes de l'ANAC dans un domaine avec borne.

Mots-clés : Monte-Carlo, Négociation automatique, Agent

[*] cedric.buron@thalesgroup.com
[†] zahia.guessoum@lip6.fr
[‡] sylvain.ductor@uece.br
[§] olivier.roussel@kyriba.com

Vingt-sixièmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA'18), Métabief, France, p. 85–94, oct. 2018

Abstract

Automated negotiation is a rising topic in Artificial Intelligence research. Monte Carlo methods have got increasing interest, in particular since they have been used with success on games with high branching factor such as go.

In this paper, we describe an Monte Carlo Negotiating Agent (MoCaNA) whose bidding strategy relies on Monte Carlo Tree Search. We provide our agent with opponent modeling tehcniques for bidding strtaegy and utility. MoCaNA can negotiate on continuous negotiating domains and in a context where no bound has been specified. We confront MoCaNA and the finalists of ANAC 2014 and a RandomWalker on different negotiation domains. Our agent ouperforms the RandomWalker in a domain without bound and the majority of the ANAC finalists in a domain with a bound.
Keywords: Monte Carlo, Automated Negotiation, Agent

1 Introduction

La négociation est une forme d'interaction dans laquelle un groupe d'agents ayant des conflits d'intérêt et un désir de coopérer essaient de trouver un accord mutuellement acceptable sur un objet de négociation. 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.

De nombreux *frameworks* ont été proposés [17] : enchères, réseaux contractuels, équilibres généraux ; ils permettent de négocier selon plusieurs modalités. Les négociations peuvent être caractérisées selon divers aspects portant notamment sur l'ensemble des participants (bilatéral ou multilatéral), les préférences des agents (linéaires ou non), les attributs sur lesquels la négociation porte (discrets ou continus), ou encore les caractéristiques du protocole (borné ou non par le temps ou le nombre de tours). Chaque agent utilise une stratégie pour évaluer l'information reçue et faire des offres. Plusieurs stratégies ont été proposées comme [8, 5, 19]. Elles peuvent être fixes ou adaptatives. Cependant, la plupart de ces stratégies reposent sur une borne connue (en temps ou en tours). Dans les applications quotidiennes, par exemple lorsque nous cherchons à acquérir un bien de consommation non critique, il est commun que le protocole ne propose aucune borne, et que chaque agent ignore le moment où il coupera court à la négociation.

Dans cet article, nous étudions MoCaNA (**Mo**nte-**Ca**rlo **N**egotiating **A**gent), un agent de négociation automatique conçu pour le protocole de *bargaining* (ou marchandage). Dans ce protocole, deux agents s'échangent des offres de manière à trouver un accord. La négociation s'interrompt lorsqu'un des deux adversaires accepte l'offre qui lui est faite par son adversaire ou qu'il la rejette. MoCaNA a pour particularité de ne faire aucune présupposition sur : 1) la linéarité des préférences, 2) le caractère continu ou discret de

l'espace de négociation ni 3) l'imposition d'une borne par le protocole. Il exploite pour cela des connaissances issues des domaines du *General Game Playing* et de *l'Apprentissage Automatique*. Afin de décider de la valeur d'une offre, notre agent s'appuie sur la recherche arborescente de Monte-Carlo (MCTS), une heuristique qui a été utilisée avec succès dans de nombreux jeux comme le go, qu'il combine avec de la modélisation d'adversaires pour plus d'efficacité.

Le choix de cette heuristique est guidé par deux éléments. D'abord, MCTS s'est montré efficace dans des jeux très différents, ce qui lui a valu de nombreux succès en *General Game Playing*. Il est aussi adapté aux jeux à haut facteur de branchement. Il constitue donc une bonne alternative pour les domaines de négociation complexes dont le facteur de branchement à chaque proposition est l'ensemble des propositions possibles.

Cet article est oorganisé comme suit : la section 2 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 3. La section 4 présente les fonctionnalités de MoCaNA. Nous présentons les résultats de la confrontation de MoCaNA aux finalistes de l'ANAC 2014 et au RandomWalker dans la section 5. Nous résumons ces contributions et donnons des pistes d'amélioration de notre travail dans la section 6.

2 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.

2.1 Négociation automatique

Les architectures des agents de négociation automatique impliquent trois fonctionnalités [1] : **la stratégie d'offres** définit les offres que l'agent fait à son adversaire ; **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'adversaires** permet de modéliser certains aspects de l'adversaire, comme sa stratégie d'offre, son utilité ou sa stratégie d'acceptation et est utilisé pour améliorer l'efficacité de la stratégie d'offres de l'agent et parfois celle de sa stratégie d'acceptation.

2.1.1 Stratégie d'offres

Les stratégies d'offres utilisées dans le cadre de négociations complexes s'appuient sur deux familles de techniques, que nous présentons par la suite : les heuristiques couplées au domaine et les algorithmes génétiques.

Les heuristiques s'appuient généralement sur des tactiques [8], qui consistent à faire des concessions selon un élément de la négociation comme le temps écoulé, la rareté de

la denrée négociée, ou les concessions faites par l'adversaire. Il est possible d'adapter ces tactiques ou de sélectionner une offre ayant une utilité gagnant-gagnant s'il en existe une. La majorité de ces stratégies [8] reposent sur la connaissance de la borne, ce qui les rend inapplicables dans notre contexte.

Les algorithmes génétiques [5] sélectionnent les offres qui maximisent une fonction objectif déterminée, les croisent et les modifient de manière à les améliorer. Ces stratégies génèrent un certain nombre d'offres parmi lesquelles l'agent doit choisir celle qui sera envoyée à l'adversaire. Elles ont obtenu de bons résultats, sans toutefois vaincre les agents utilisant des heuristiques.

2.1.2 Stratégies d'acceptation

On distingue deux principales catégories de stratégies d'acceptation [3] : les stratégies myopes et les stratégies optimales. Les stratégies myopes reposent sur les offres de l'adversaire et celles de l'agent. Il peut s'agir 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. Il est aussi possible de combiner ces stratégies. Les stratégies optimales [3] exploitent un modèle de la stratégie adverse afin de déterminer la probabilité d'obtenir une meilleure offre.

2.1.3 Modélisation de l'adversaire

La majorité des méthodes de modélisation d'adversaire utilisées en négociation automatique ont été analysées par Baarslag et ses collègues [2]. Elles permettent de modéliser la stratégie d'offre, la stratégie d'acceptation, l'utilité de l'adversaire, la borne qu'il s'est fixé, et son prix de réserve le cas échéant.

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 gaussien est stochastique. Elle s'appuie sur les mouvements précédents pour prédire une densité de probabilité pour le tour suivant. Introduite par Rasmussen et Williams [15], elle a été utilisée avec succès par Williams *et al.* [20]. L'aspect stochastique de cette méthode permet de générer des propositions différentes à chaque étape de simulation de *Monte Carlo Tree Search* (MCTS), selon la gaussienne prédite par la régression. C'est cette méthode que nous adoptons.

L'**utilité** d'un adversaire est généralement considérée comme la somme pondérée de fonctions pour chaque attribut à valeurs dans $[0, 1]$ [2]. Deux familles de méthodes sont utilisées. La première s'appuie sur la fréquence d'apparition de chaque valeur parmi les propositions de l'adversaire. Elle suppose que les valeurs proposées le plus régulièrement par l'adversaire sont celles qu'il préfère, et que les attributs variant le plus souvent sont ceux qui ont le moins d'importance pour lui. Ces approches ne sont cependant valables que dans le cas où le domaine de négociation est discret. En effet, l'extension au cas continu requiert la définition d'une fonction de distance dépendant du domaine. La seconde famille de

méthodes repose sur l'apprentissage bayésien. Elle est adaptée au cas continu. La méthode présentée par Hindriks et Tykhonov [10] repose sur la génération d'hypothèses. Chaque hypothèse est constituée 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.

La **stratégie d'acceptation** d'un adversaire peut être apprise de deux manières. On peut d'abord faire la supposition que l'adversaire acceptera une offre sous certaines conditions dépendant de sa stratégie d'offre et/ou de sa fonction d'utilité [2]. Dans ce cas, il est possible de la déduire des modèles ci-dessus sans faire de calcul supplémentaire. Dans le cas où l'agent ne modélise pas les éléments précédemment décrits, il est aussi possible d'utiliser des réseaux de neurones [7]. Cela demande cependant un calcul supplémentaire, potentiellement coûteux.

2.2 Méthodes de Monte Carlo

Les méthodes de Monte-Carlo sont utilisées comme heuristiques pour les jeux. Elles attribuent à chaque mouvement possible un score calculé au terme d'une ou de plusieurs simulations. Kocsis et Szepevsvári [12] proposent une méthode hybridant la construction d'un arbre de jeu, méthode qui a fait ses preuves, et les méthodes de 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. Pendant l'exploration de cet arbre, les branches privilégiées sont celles qui ont été peu explorées et celles qui ont le plus haut score dans les simulations. Cette méthode, nommée recherche arborescente de Monte-Carlo (*MCTS*[1]) s'est vue améliorée au moyen de nombreuses extensions [4].

MCTS se décompose en quatre parties. (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. (2) Dans le second cas, l'arbre se voit pourvu d'un nouveau nœud, généré selon une stratégie d'expansion. (3) Ensuite, on simule le jeu jusqu'à un état final. (4) On calcule les scores et on les rétropropage sur les nœuds de l'arbre ayant été explorés.

Les méthodes de Monte-Carlo ont gagné en popularité suite à leur succès dans les jeux à fort facteur de branchement, notamment le Go. En particulier, AlphaGo [18], qui a vaincu l'un des meilleurs joueurs au monde, utilise les méthodes de Monte-Carlo couplées à de l'apprentissage profond. À notre connaissance, seuls de Jonge et Zhang [6] ont utilisé MCTS dans le cadre de la négociation automatique. Ils se sont concentrés sur des domaines de négociations petits, et où la fonction d'utilité adverse 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 adverse ne peut être facilement inférée et où l'espace de négociation peut être grand, voire infini.

1. *Monte Carlo Tree Search*

3 Jeu et négociation

La recherche arborescente de Monte-Carlo est une méthode appliquée aux jeux extensifs. Dans cette section, nous montrons comment il est possible d'assimiler la négociation à un tel jeu. 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.

3.1 Notre formalisme

Un jeu extensif [14] 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 *bargaining* $\mathcal{B}$ comme un jeu sous forme extensive.

Définition 1 (*Bargaining*)
Un bargaining est un triplet $\mathcal{B} = \left(H, A, (u_i)_{i \in [\![1,2]\!]}\right)$ *vérifiant :*

1. ***joueurs :*** *il y a deux joueurs : un acheteur (joueur* 1*) et un vendeur (joueur* 2*),*
2. ***historique :*** *l'historique de la négociation est composé des messages que les agents s'envoient : les propositions faites par les agents, et les* `accept` *et* `reject`*. Les historiques terminaux sont les historiques infinis et les historiques se terminant par* `accept` *ou par* `reject`*. Chaque message est composé d'une paire* (α, c) *où* α *est l'acte de langage (performatif) du message et,* c *est 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. On sépare tout historique en deux, chacun correspondant aux actions d'un joueur :* $h_i = (\alpha, c)_i$
3. ***tour de jeu :*** *la fonction tour fonctionne selon la parité. On suppose que c'est l'acheteur (le joueur 1) qui commence. Ainsi,* $\forall h \in H, joueur(h) = 2 - (|h| \mod 2)$ *où* $|h|$ *est la taille de* h*,*
4. ***préférence :*** *l'ordre sur les situations terminales de chaque joueur est induit par une fonction d'utilité* u_i *qui associe son utilité pour chaque historique terminal. Cette fonction associe une utilité à chaque accord possible trouvé, ainsi qu'au cas de rejet et au cas d'historique infini*[2]*.*

La négociation n'est pas un jeu combinatoire classique, comme peuvent l'être les échecs ou le go. La première différence entre ces jeux et la négociation tient au fait que cette dernière est un jeu à somme non nulle. Les agents cherchent à trouver un accord mutuel qui soit profitable à chacun. Cela est particulièrement vrai dans des domaines complexes,

2. Notons que ces deux dernières valeurs peuvent être différentes, par exemple si l'agent considère qu'il alloue des ressources à la négociation.

ayant de nombreux attributs, où une solution peut être bien meilleure que la solution par défaut où chaque agent reçoit son utilité de réserve, *i.e.* quand les agents ne trouvent pas d'accord.

Ensuite, la négociation est 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 [12]* 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 remporté MCTS dans les jeux complexes semblent indiquer qu'il pourrait s'avérer être une bonne stratégie pour la négociation dans un contexte complexe.

3.2 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 [16]. 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'adversaire. 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'adversaire 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'adversaire.

Un autre formalisme possible est la modélisation par un **jeu stochastique** [11]. Les jeux stochastiques sont une généralisation des processus markoviens. Le *bargaining* peut être vu comme un processus de décision markovien (MDP), où chaque action de l'agent génère une réaction de son adversaire, 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'adversaire 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'adversaire et estime l'utilité attendue pour les différentes transitions possibles, *i.e.* les choix de l'adversaire, au moyen de MCTS. Notons que cette méthode s'est montrée particulièrement efficace pour la planification dans les MDP, comme le montre notamment [12].

4 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 des différents modules de notre agent est présentée dans la Figure 1. La stratégie d'offre implémente MCTS et utilise le module de modélisation d'adversaire, qui comporte deux sous-modules : l'un pour l'utilité de l'adversaire, l'autre pour sa stratégie d'offre. Le dernier module, celui de stratégie d'acceptation, effectue une comparaison entre la proposition de l'adversaire et la proposition générée par la stratégie d'offre. Les différents sous-modules de l'agent ainsi que les interactions entre eux sont décrits dans la suite de cette section.

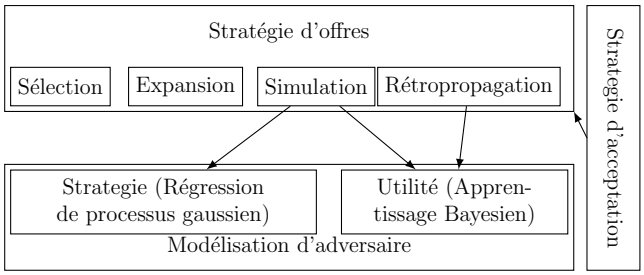

FIGURE 1 – Interaction entre les modules de MoCaNA

4.1 Modélisation d'adversaire

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

4.1.1 Modélisation de la stratégie d'offre

Le but est de prédire une future proposition faites par l'adversaire au tour x_* de la négociation. Nous utilisons la régression de processus gaussiens [15] 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

$$K = \begin{pmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{pmatrix} \tag{1}$$

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

$$K_* = (k(x_*, x_1), \dots, k(x_*, x_n)) \tag{2}$$

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

$$\overline{y_*} = K_* K^{-1} \mathbf{y} \tag{3}$$

$$\sigma_*^2 = \mathrm{Var}(y_*) = K_{**} - K_* K^{-1} K_*^\top \tag{4}$$

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 gaussien est la détermination du *kernel*. Les plus communs sont les fonctions à base radiale, les fonctions *rational quadratic*, le *kernel* de Matérn et le *kernel exponential sine squared*. Ces kernels servent à déterminer la distance entre deux tours de négociation. Nous avons testé ces quatre *kernels* sur plusieurs négociations entre agents. La Table 1 donne les résultats de la régression de processus gaussien pour chacun de ces *kernels*. 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 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.

Kernel	Dist. moyenne
Fonction à base radiale	43.288
Rational quadratic	**17.766**
Matérn	43.228
Exp. sine squared	22.292

Table 1 – Distance moyenne entre les propositions et les prédictions de la régression de processus gaussien selon le *kernel*

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

4.1.2 Modèle du profil de préférence

L'apprentissage bayésien décrit dans [10] suppose seulement que l'agent fait une concession à un rythme constant, une supposition faible par rapport aux autres méthodes.

On approxime l'utilité de l'adversaire 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 adversaire 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'adversaire, mais elle suppose qu'il fait des concessions à un rythme globalement linéaire.

4.1.3 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 adversaire si et seulement si son utilité est meilleure que l'utilité générée par la stratégie d'offre. Cette modélisation, évoquée par [2], est peu coûteuse et permet de ne pas alourdir les calculs déjà nombreux de l'agent.

4.2 Stratégie d'offres basée 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'offres basée sur MCTS.

4.2.1 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. Si l'arbre est très large, si le nombre de nœuds possibles est plus grand que nombre de simulations, la même situation apparaîtra.

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

Sélection Pour cette étape, nous utilisons le *progressive widening*. Le critère du progressive widening peut s'exprimer ainsi : un nouveau nœud est étendu si et seulement si :

$$n_p^\alpha \geq n_c \tag{5}$$

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. S'il n'y a pas d'expansion, nous sélectionnons

le nœud i qui maximise :

$$W_i = \frac{s_i}{n_i + 1} + C \times n^{\alpha} \sqrt{\frac{\ln(n)}{n_i + 1}} \tag{6}$$

où n est le nombre total de simulations faites jusqu'alors, s_i est le score du nœud i et C est également un paramètre du modèle.

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 Nous utilisons le modèle de stratégie de l'adversaire, afin de rendre les simulations plus pertinentes. Nous utilisons la régression de processus gaussien et l'apprentissage bayésien.

Rétropropagation L'étape de rétropropagation utilise aussi le modèle de l'utilité de l'adversaire, et répercute sur les nœuds l'utilité de l'agent et l'utilité de son adversaire selon la modélisation d'utilité.

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'adversaire l'accepte. Nous renvoyons donc l'offre b^* parmi les fils de la racine maximisant :

$$b^* = \text{argmax}_{b \in racine.fils} \left(\frac{utility(b) + score(b)}{2} \right) \tag{7}$$

4.2.2 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'adversaire. Dans ce cas, l'agent garde en mémoire la proposition de l'adversaire 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 adversaire a faites.

4.3 Architecture logicielle

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

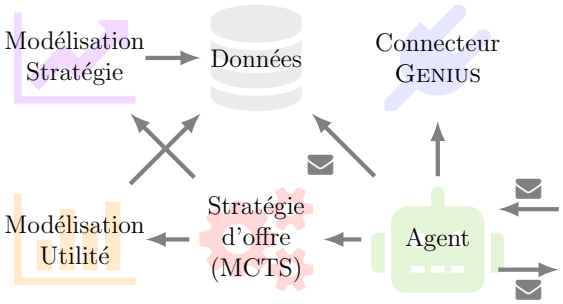

Figure 2 – Architecture logicielle des modules de MoCaNA

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 adverse et un modèle de l'utilité. Nous n'avons pas implémenté de module pour la modélisation de la stratégie d'acceptation. Cette architecture est représentée dans la Figure 2. Les ✉ représentent les messages envoyés et reçus par l'agent.

Les simulations de MCTS étant très gourmandes en terme de puissance de calcul, nous les avons parallélisé. L'aspect stochastique de notre modélisation de stratégie d'offre assure que deus 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. Les paramètres du kernel de cette méthodes sont quant à eux optimisés en utilisant la librairie Apache Commons Math.

5 Expérimentations

Pour évaluer notre agent, nous utilisons le *framework* Genius [13], qui permet de créer des sessions de négociation et des tournois. Nous confrontons MoCaNA au RandomWalker ainsi qu'aux agents de l'ANAC 2014.

5.1 Protocole expérimental

Genius permet de négocier sur des attributs entiers ou discrets, mais pas encore continus. Nous utilisons le domaine de négociation de l'ANAC 2014 décrit par [9]. Les compétitions de 2015 et de 2016 se sont concentrées respectivement sur la négociation multilatérale, et sur les *smart grids*. À notre connaissance, les agents de négociation de l'ANAC 2017 n'ont pas encore été mis à disposition. Les attributs sont entiers, et varient de 0 à 10. Plusieurs variantes ont été proposées allant de 10 attributs à 50 attributs. Nous nous

focalisons sur le contexte à 10 attributs. Le contexte utilisé n'a aucun taux de réduction. 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 valeur minimale.

Nous fixons en outre la borne 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 notre agent peut choisir parmi 200 propositions, il s'en trouve généralement une qui procure à la fois une haute utilité pour lui et son adversaire. 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, chaque pour faire une proposition, notre agent prend 3 minutes. Nous obtenons $\alpha = 0.489$.

5.2 Résultats

Nous distinguons deux contextes de négociation : un contexte où la borne est présente et connue des adversaires, face aux agents de l'ANAC 2014, et un où elle est assez loin pour être considérée comme inexistante, face au *RandomWalker*.

5.2.1 ANAC 2014

La Table 2 montre les résultats de notre agent face aux finalistes de l'ANAC 2014, décrits par [9], ordonnés selon leur résultats à l'ANAC 2014 dans la catégorie utilité individuelle. Les scores représentent l'utilité moyenne de notre agent et de son adversaire moyenné 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. Le nombre suivant le score précédé du signe $\pm$ est l'écart type de la série. L'utilité moyenne est calculée en ne prenant en compte que les négociations qui ont réussi. La dernière colonne représente le taux d'accord atteint par les agents, c'est à dire le nombre de fois où les agents parviennent à un accord divisé par le nombre total de négociations.

Le taux d'accord très faible peut être expliqué par notre stratégie, qui ne s'accorde pas du tout à la distance à la borne, puisqu'elle ne la prend pas en compte. Cette caractéristique l'empêche de ressentir la pression du temps (en anglais *time pressure*) 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 adversaires, puisque l'élagage repose sur les propositions qu'ils font tout au long de la négociation. L'utilité de l'agent, en

Adversaire	Score adversaire	Score MoCaNA	Taux d'accord
AgentM	0.88 (±0.07)/0.70(±0.08)/0.77(±0.11)	0.66 (±0.08)/0.85 (±0.02)/0.66(±0.08)	0.35/0.15/0.10
DoNA	1(±0.0)/N.A/0.86(±0.01)	0.57 (±0.0)/N.A/0.55(±0.06)	0.05/0/0.10
Gangster	0.81 (±0.13)/0.53(±0.19)/0.56(±0.22)	0.4 (±0.14)/0.85(±0.05)/0.70(±0.16)	0.45/0.55/0.55
Whale	0.77 (±0.16)/0.71(±0.06)/0.64(±0.09)	0.70 (±0.11)/0.83(±0.02)/0.77(±0.10)	0.45/0.45/0.25
Group2	0.68 (±0.21)/0.53(±0.06)/0.53(±0.08)	0.82 (±0.19)/0.87(±0.04)/0.79(±0.10)	0.45/0.45/0.45
kGAgent	0.98 (±0.085)/N.A/1(±0)	0.55 (±0.05)/N.A/0.49(±0)	0.20/0/0.05
AgentYk	0.81 (±0.17)/0.71(±0.07)/0.77(±0.13)	0.752 (±0.15)/0.84(±0.03)/0.64(±0.11)	0.15/0.10/0.15
BraveCat	0.62(±0.17)/0.49(±0.20)/0.52(±0.23)	0.75 (±0.10)/0.86(±0.07)/0.66(±0.12)	0.95/0.95/1

Table 2 – Résultats de MoCaNA face aux finalistes de l'ANAC 2014 (sans élagage/ élagage fixe/ élagage variable)

revanche, est bien meilleure que dans le cas sans élagage. Elle l'est moins, en revanche, que dans le cas avec un élagage fixe de 0.8. Notre agent bat la majorité 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. Elles sont en cela meilleure que la pondération *a posteriori*.

5.2.2 Contexte sans borne

En ce qui concerne la négociation avec le *Random Walker*, MoCaNA semble beaucoup mieux s'en sortir, même sans élagage. Il obtient un score élevé, $0.703(\pm 0.062)$ contrairement au *Random Walker*[3], qui obtient un score moyen de $0.378(\pm 0.077)$. ce qui confirme que notre agent est gêné par la présence de la borne, et le fait qu'il ne s'appuie pas dessus.

Notre agent est donc un bon négociateur dès lors qu'on ne lui impose pas de borne. La présence de cette dernière coupe la négociation avant qu'il ne puisse arriver à un accord avantageux pour lui. Les négociation réussissant (moins de 50%), sont donc celles pour lesquelles il a fait le plus de concessions, ce qui a pour effet de lui conférer une utilité moindre que celle de son adversaire.

6 Conclusion

Dans cet article, nous avons présenté un agent de négociation automatique capable de négocier dans un contexte où les agents ne se fixent pas de borne, 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'offres s'appuyant sur une version de MCTS et sur deux méthodes de modélisation de l'adversaire. Pour l'une d'entre elles, la régression de processus gaussiens, dont nous avons testé plusieurs *kernels* 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 un RandomWalker. 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 adversaires, au prix d'une baisse du taux d'acceptation.

Les perspectives de ces travaux comprennent l'élargissement du spectre de domaines sur lequel MoCaNA est capable de négocier, notamment à tester des domaines de négociations formés d'attributs catégoriels uniquement, continus uniquement, et mixtes. Un autre type de protocoles à tester réside dans les protocoles multilatéraux, notamment des protocoles $1{:}n$ de type enchères ou $n{:}m$ de type *many-to-many bargaining*. Une autre perspective consisterait à concevoir une version de l'ANAC sans borne. Il serait enfin intéressant de

3. Cet agent est décrit dans [3] et fait des propositions aléatoires.

voir dans quelle mesure MoCaNA pourrait intégrer des informations supplémentaires telles que la borne, des connaissances sur la nature des attributs afin d'en tirer parti lorsque cela est possible.

Il est aussi 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'adversaire, 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 la performance de notre agent serait d'implémenter des algorithmes comme *All Moves As First* ou la *Rapid Action Value Estimation*.

Références

[1] Tim Baarslag. *Exploring the Strategy Space of Negotiating Agents : A Framework for Bidding, Learning and Accepting in Automated Negotiation.* PhD thesis, Delft University of Technology, 2016.

[2] Tim Baarslag, Mark J C Hendrikx, Koen V Hindriks, and Catholijn M Jonker. Learning about the opponent in automated bilateral negotiation : a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, 20(1) :1–50, 2015.

[3] Tim Baarslag and Koen V. Hindriks. Accepting optimally in automated negotiation with incomplete information. In *AAMAS '13*, pages 715–722, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[4] Cameron C Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Taverner, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1) :1–43, 2012.

[5] Dave de Jonge and Carles Sierra. *Recent Advances in Agent-based Complex Automated Negotiation*, volume 638 of *Studies in Computational Intelligence*, chapter GANGSTER : An Automated Negotiator Applying Genetic Algorithms, pages 225–234. Springer International Publishing, 2016.

[6] Dave de Jonge and Dongmo Zhang. Automated negotiations for general game playing. In *AAMAS '17*, pages 371–379, Richland, SC, 2017.

[7] Fang Fang, Ye Xin, Yun Xia, and Xu 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.

[8] Peyman Faratin, Nicholas R Jennings, and Carles Sierra. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3-4) :159–182, 1998.

[9] Naoki Fukuta, Takayuki Ito, Minjie Zhang, Katsuhide Fujita, and Valentin Robu, editors. *Recent Advances in Agent-based Complex Automated Negotiation*, volume 638 of *Studies in Computational Intelligence*. Springer International Publishing, 2016.

[10] Koen Hindriks and Dmytro 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*, volume 1, pages 331–338, 2008.

[11] Anna Jaśkiewicz and Andrzej S. Nowak. *Non-Zero-Sum Stochastic Games*, pages 1–64. Springer International Publishing, Cham, 2016.

[12] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning : ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[13] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M Jonker. Genius : an integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, 30(1) :48–70, 2014.

[14] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 12 edition, 1994.

[15] Carl E Rasmussen and Christopher K I Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[16] Daniel M Reeves and Michael P Wellman. Computing best-response strategies in infinite games of incomplete information. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 470–478, Arlington, Virginia, United States, 2004. AUAI Press.

[17] Tuomas W Sandholm. Distributed rational decision making. *Multiagent systems : a modern approach to distributed artificial intelligence*, pages 201–258, 1999.

[18] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, and al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587) :484–489, 2016.

[19] Bálint Szöllősi-Nagy, David Festen, and Marta M Skarżyńska. *Recent Advances in Agent-based Complex Automated Negotiation*, volume 638 of *Studies in Computational Intelligence*, chapter A Greedy Coordinate Descent Algorithm for High-Dimensional Nonlinear Negotiation, pages 249–260. Springer International Publishing, 2016.

[20] Colin R Williams, Valentin Robu, Enrico H Gerding, and Nicholas R Jennings. Using gaussian processes to optimise concession in complex negotiations against unknown opponents. In *IJCAI'11*, pages 432–438, 2011.