La Recherche Monte-Carlo

Tristan Cazenave, LAMSADE, 2008-2018

Mots-clé : Intelligence Artificielle, Jeux, Optimisation

La recherche Monte-Carlo est une méthode très en vogue pour les jeux et l'optimisation. L'illustration la plus parlante est le logiciel Alpha Zéro de Google DeepMind qui a permis de dépasser le niveau des meilleurs joueurs professionnels au jeu de Go mais qui a aussi été appliqué avec succès aux échecs et au Shogi (les échecs japonais), battant les meilleurs programmes de ces jeux en combinant recherche Monte-Carlo et apprentissage profond. Alpha Zéro utilise des réseaux de neurones résiduels (Tristan Cazenave, 2017). Une autre application ayant trait aux jeux est le GGP (General Game Playing). L'université de Stanford organise tous les ans une compétition de GGP réunissant de nombreux programmes généraux de jeu. Le principe de la compétition est de faire jouer des intelligences artificielles entre elles à des jeux qu'elles ne connaissent pas à l'avance. Tous les gagnants de la compétition depuis 2007 utilisent la recherche Monte-Carlo. Le Lamsade est depuis longtemps à la pointe de ce domaine. Jean Méhat et Tristan Cazenave ont été champions du monde de GGP en 2009 et 2010. La recherche Monte-Carlo est une méthode très générale qui donne de bons résultats même si on a peu de connaissances sur le problème à résoudre.

La recherche Monte-Carlo ne se limite cependant pas aux jeux. De nombreux problèmes d'optimisation ont été abordés en suivant cette méthode. Les premières applications ont été sur des puzzles comme le Sudoku ou le Morpion Solitaire (pour ce dernier les méthodes de Monte-Carlo ont permis de battre largement le record du monde humain en 2011, un record qui datait de plus de 30 ans quand Pierre Berloquin avait organisé un concours de résolution dans la rubrique jeux du magazine Science et Vie dans les année 70). La recherche Monte-Carlo a battu des records du monde pour d'autres problèmes comme par exemple la recherche de 'serpents' dans des hypercubes. Le but de cette recherche est de trouver un chemin dans un hypercube dont tous les sommets ont au plus deux voisins aussi dans le chemin. Ce problème a des applications en théorie des codes, en réseaux et en électricité. Pour un cube de dimension trois ou quatre le chemin optimal se trouve très rapidement, toutefois pour des hypercubes de dimension douze ou treize, on ne connaît pas les solutions optimales car il y a trop de possibilités. La recherche Monte-Carlo qui donne de bonnes solutions sans toutefois en garantir l'optimalité est alors appropriée.

Il existe plusieurs algorithmes de recherche Monte-Carlo. Les plus adaptés pour les jeux sont les algorithmes dérivés d'UCT (pour Upper Confidence bounds applied to Trees). Chaque état visité

par l'algorithme est mémorisé ainsi que le nombre de fois qu'il a été visité et le nombre de parties aléatoires qui ont été gagnées en partant de cet état. Le principe de base de la recherche Monte-Carlo est de faire des statistiques sur les parties aléatoires (d'où le terme Monte-Carlo évoquant le célèbre casino). Chaque état visité maintient pour chaque coup possible le score moyen des parties aléatoires commençant par ce coup. Le principe d'UCT est de balancer exploration et exploitation lors du choix du prochain coup à simuler. L'exploitation consiste à favoriser les coups ayant les meilleurs scores moyens alors que l'exploration consiste à essayer les coups ayant de moins bonnes moyennes mais ayant été peu explorés. Pour résoudre le dilemme exploration/exploitation UCT ajoute un terme de regret au score moyen qui est une formule dénommée UCB ayant de bonnes propriétés théoriques et appelée une formule de bandit (d'après les machines à sous appelées bandits manchots en anglais).

Une heuristique qui améliore nettement le niveau des programmes de jeux qui utilisent UCT est l'heuristique RAVE (Rapid Action Value Estimation). Le principe est de faire non seulement pour chaque état visité des statistiques sur les parties aléatoires qui commencent par un coup, mais aussi des statistiques sur les parties aléatoires qui contiennent un coup, même si ce coup est très éloigné de l'état pour lequel on tient à jour les statistiques. RAVE commence par suivre les statistiques sur les parties qui contiennent un coup lorsqu'il y a très peu de parties aléatoires passant par l'état. Puis elle passe graduellement aux statistiques sur les coups commençant les parties aléatoires lorsque le nombre de parties aléatoires comptabilisées pour l'état augmente. Une optimisation très simple qui donne de meilleurs résultats pour de nombreux jeux est GRAVE (pour Generalized RAVE) telle que développée au Lamsade (Tristan Cazenave, 2015). Le principe de GRAVE est de retenir comme statistiques pour un état peu visité les statistiques du dernier état de la partie aléatoire contenant plus qu'un nombre fixé de parties aléatoires déjà comptabilisées.

Il est aussi possible de résoudre des jeux en utilisant la recherche Monte-Carlo. Le principe est de noter les états dont tout le sous-arbre a été exploré et de mémoriser le score optimal résultant de l'exploration de ce sous-arbre. Une amélioration de la rapidité de résolution est possible en effectuant des coupes dans l'arbre de recherche lorsque le nombre de scores possibles d'un jeu est plus grand que deux (les jeux comme le Go qui ne peuvent être que gagnés ou perdus ont deux scores possibles alors que les jeux comme les échecs ont trois scores possibles : gagné, perdu et nulle). L'utilisation de coupes pour améliorer la résolution de jeux a été publiée en 2010 par Abdallah Saffidine et Tristan Cazenave.

Un algorithme de recherche Monte-Carlo plus adapté qu'UCT aux jeux à un joueur et aux problèmes d'optimisation est la recherche Monte-Carlo imbriquée (Tristan Cazenave, 2009). C'est cet algorithme qui a trouvé des records du monde aux 'serpents' dans des hypercubes. Au lieu de maintenir un score moyen pour chaque action possible, la recherche Monte-Carlo imbriquée fait une partie aléatoire par coup possible et retient le coup qui est suivi de la partie aléatoire ayant

obtenu le meilleur score. La recherche est imbriquée car il existe plusieurs niveaux de recherche : chaque niveau de recherche utilisant le niveau en dessous pour faire des parties aléatoires. Il est important pour cet algorithme de mémoriser la meilleure partie aléatoire à chaque niveau et de la suivre tant qu'une partie aléatoire encore meilleure n'a pas été trouvée. Cela balance l'exploration qui consiste à faire des parties aléatoires et l'exploitation qui consiste à suivre la meilleure partie aléatoire du niveau.

Une autre méthode de Monte-Carlo imbriquée est NRPA (Nested Rollout Policy Adaptation). Le principe est d'apprendre une politique de partie aléatoire de façon à améliorer la qualité des parties aléatoires. Pour chaque action possible on associe un poids. Dans une partie aléatoire on choisit une action possible avec une probabilité proportionnelle à l'exponentielle de son poids. A chaque niveau de la recherche imbriquée, la meilleure partie aléatoire est mémorisée. L'apprentissage de la politique consiste à augmenter les poids des actions de la meilleure partie aléatoire de un et à diminuer les poids des actions qui n'ont pas été choisies de leur probabilité d'être choisies. De cette manière la somme des poids reste constante au cours de la recherche.

Les applications de la recherche Monte-Carlo imbriquée ont été nombreuses. Dans les transports on peut citer le voyageur de commerce avec fenêtres de temps (Arpad Rimmel, Fabien Teytaud et Tristan Cazenave, 2011), la régulation de lignes de bus (Tristan Cazenave, Flavien Balbo et Suzanne Pinson, 2009) ainsi que la planification logistique d'une flotte de véhicules. En bio-informatique on peut citer l'alignement de séquences biologiques (ADN, ARN, protéines) ainsi que le problème du pliage inverse de l'ARN. En Génie Logiciel on peut citer la génération automatique de tests ainsi que la vérification de modèles. En optimisation on peut citer le remplissage d'un container avec des objets de formes variées. En génération de programme on peut citer la découverte d'expressions mathématiques dans des domaines variés (Tristan Cazenave, 2010 et 2013), par exemple pour calculer la volatilité implicite d'un sous-jacent en finance (Tristan Cazenave et Sana Ben Hamida, 2015). Pour la résolution de puzzles on peut citer le Sudoku (Tristan Cazenave, 2009), SameGame (Tristan Cazenave, 2009 et 2016, Benjamin Negrevergne et Tristan Cazenave en 2017 pour la parallélisation), le Morpion Solitaire (Tristan Cazenave, 2009), le Kakuro (Tristan Cazenave, 2009) et la conception de mots croisés.

Le principe de l'apprentissage d'une politique de parties aléatoires a été réutilisé pour les jeux. Toutefois la notion de meilleure séquence est mal définie pour les jeux. Au lieu d'apprendre les coups d'une meilleure séquence, PPA (Playout Policy Adaptation) apprend à renforcer les coups du gagnant d'une partie aléatoire (Tristan Cazenave, 2015). Les performances de l'algorithme augmentent nettement lorsqu'au lieu de mettre à jour les poids des coups on met à jour les poids des coups associés à leur environnement direct dans la partie aléatoire (Tristan Cazenave, 2016). Il est aussi bénéfique de mémoriser la politique d'une recherche pour la réutiliser lors de la recherche suivante (Tristan Cazenave et Eustache Diemert, 2017). Au final une recherche Monte-Carlo PPA avec environnement des coups et mémorisation de la politique gagne entre 80 % et

100 % de ses parties contre un UCT standard pour de nombreux jeux, à savoir : Breakthrough, Misere Breakthrough, Knightthrough, Misere Knighthrough, Domineering, Misere Domineering, Atarigo et Nogo. Les gains atteignent 100 % pour Misere Breakthrough et Misere Knighthrough. Ce sont des jeux misère, c'est à dire que la condition de gain est l'opposée de celle du jeu normal. Par exemple à Breakthrough le but est de placer un pion sur la ligne opposée du damier. Le but à Misere Breakthrough est de ne pas placer un pion sur la ligne opposée. Dans les jeux misère comme Misere Breakthrough et Misere Knighthrough il y a de nombreux coups qui font perdre la partie. PPA apprend à éviter ce genre de coups et rend de ce fait les parties aléatoires beaucoup plus significatives.

Enfin, comme j'en parlais au début de cet article les méthodes de recherche Monte-Carlo se marient très bien avec l'apprentissage de réseaux de neurones profonds. Un programme peut apprendre à partir de rien à jouer mieux que n'importe quel être humain à un jeu. Pour cela il entraîne un réseau politique et une réseau évaluateur. Le réseau politique apprend à reproduire la répartition de probabilités sur les coups donnée par la recherche Monte-Carlo alors que le réseau évaluateur apprend à trouver le résultat de la partie. Il est probable que cette manière d'apprendre permettra de résoudre de nombreux problèmes difficiles à l'avenir.

Abstract :

Monte Carlo Search is a fashionable method for games and optimization. For example Alpha Zero taught itself to play the games of Go, Chess and Shogi from scratch using a combination of Monte Carlo Search and Deep Learning. Monte Carlo Search is a general method that can be applied to many games and problems. Each year Stanford organizes a General Game Playing competition where programs play each other at games they have not seen before. Since 2007 all the winners of the competition use Monte Carlo Search. Besides games, many optimization problems have been addressed using Monte Carlo Search. In transportation the Traveling Salesman with Time windows, bus regulation and vehicles routing problems have been addressed. In bioinformatics multiple sequence alignment (for DNA, RNA and proteins) and the RNA inverse folding problem are solved by Monte Carlo Search. In software engineering applications are automatic test generation and heuristic model checking . In optimization packing objects of different sizes in a container has been addressed. In program generation, it can discover mathematical expressions in various domains, for example to generate an expression that approximates the implicit volatility in finance. For puzzles application domains have been Sudoku, SameGame, Morpion Solitaire, Kakuro and generation of crosswords.