Un programme de go a battu un joueur professionnel

Tristan Cazenave, Arnaud Knippel

Le jeu de go était la cible des programmeurs depuis la défaite de Kasparov contre Deep Blue aux échecs en 1997. Les méthodes utilisées pour les échecs [1] se montrant insuffisantes pour le go, il a fallu plusieurs innovations majeures, dont l'introduction de méthodes stochastiques dans les années 2000, puis plus récemment les techniques les plus récentes d'apprentissage statistique à base de réseaux de neurones profonds, pour relever le défi. La société Google a réuni une équipe de chercheurs et programmeurs qui a conçu le code AlphaGo. Le résultat est sans appel: l'ordinateur a gagné 5-0 contre Fan Hui, le triple champion d'Europe français [2, 3].

Le jeu de go [4] est un jeu combinatoire à deux joueurs qui se pratique sur une grille carrée de taille 19x19 (mais on joue aussi couramment sur des grilles de taille réduite, 9x9 ou 13x13). La combinatoire du jeu de go explose lorsque la taille de la grille augmente, et le go est classé parmi les problèmes NP-difficiles [17], et même PSPACE-difficiles. Jusqu'aux années 2000, les programmes de go étaient très faibles, même sur un plateau 9x9. Les programmes informatiques s'appuyaient principalement sur des algorithmes dérivés des méthodes de Branch and Bound, mais il manquait, contrairement au jeu d'échecs, une méthode d'évaluation efficace de la position (autre que de parcourir toute l'arborescence des coups possibles) permettant de réduire la combinatoire [13, 12]. Cet obstacle a été levé par l'utilisation de méthodes stochastiques pour évaluer la position [10, 14] : des parties aléatoires (les *playouts*) sont générées à partir de la position qu'on veut évaluer, et permettent de former un indicateur statistique (typiquement la moyenne des scores de ces parties aléatoires, en comptant 1 pour une victoire et 0 pour une défaite). Ce mécanisme d'évaluation est intégré à une méthode d'exploration arborescente nommée MTCS pour Monte Carlo Tree Search [7, 8, 9, 15]. Les résultats

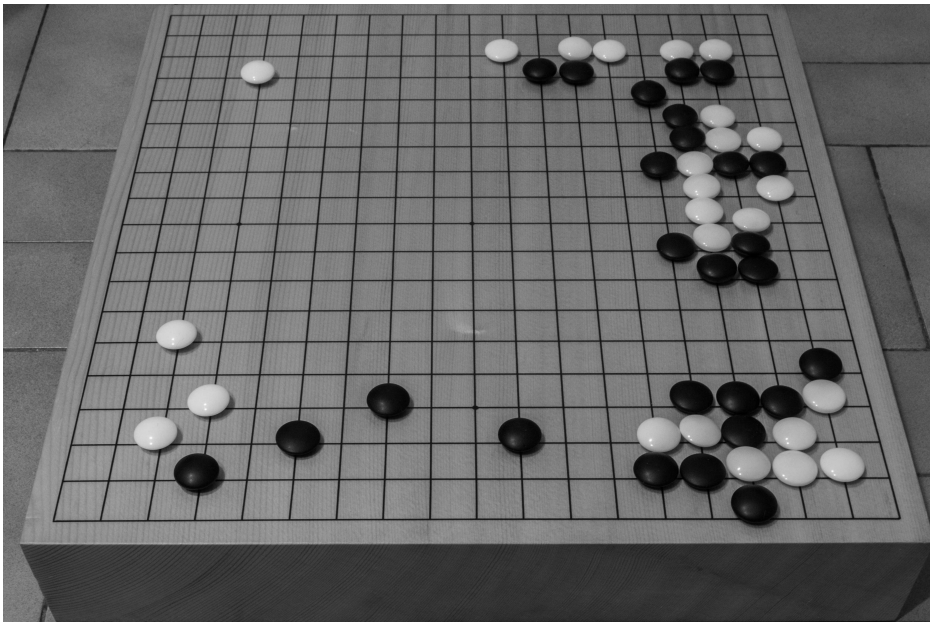

Figure 1: *Jeu de go traditionnel en bois avec pierres en pâte de verre. Le jeu commence avec un plateau vide et chaque joueur à son tour pose une pierre jusque'à ce que les territoires formés ne puissent plus être agrandis ou réduits. On compte ensuite les surfaces contrôlées, et Blanc reçoit une compensation de 7,5 points pour ne pas avoir commencé. (Photographie de Jean-Pierre Lalo)*

des actions ayant déjà été effectuées au cours des playouts précédents sont mémorisés. Les actions ayant des scores statistiques élevées sont privilégiées par rapport aux actions ayant de moins bons scores. La recherche Monte-Carlo est améliorée par l'utilisation de patterns dont les poids sont appris à partir de parties de joueurs forts. Ces patterns permettent de sélectionner les meilleurs coups dans l'arbre et d'améliorer les parties aléatoires. Ces idées ont permis de produire des programmes de go de niveau professionnel sur plateau 9x9, mais qui restaient encore en deça du niveau des joueurs professionnels pour la dimension standard 19x19, et les joueurs professionnels de go pensaient encore récemment avoir un répit de 10 ou 20 ans devant eux.

La force d'AlphaGo est d'utiliser des réseaux de neurones profonds [6,

5, 11] pour apprendre à trouver de bons coups au jeu de Go et pour apprendre à évaluer une position. L'apprentissage est réparti en trois phases, réalisées au moyens de réseaux de neurones ayant la même structure en 13 couches. Chaque couche est *convolutionnelle* avec de 128 à 384 filtres suivi d'un *rectifieur linéaire*. Une couche convolutionnelle consiste à passer un filtre 3x3 sur tout le plan d'entrée. C'est une technique surtout utilisée en reconnaissance d'image et elle s'applique bien au jeu de Go qui peut être vu comme une image en deux dimensions. Le rectifieur linéaire est indispensable à l'apprentissage profond. Son rôle est de redonner de la force au signal pour pouvoir modifier les couches profondes lors de l'apprentissage. Dans les trois phases, le problème d'optimisation correspondant à l'entraînement du réseau de neurones est traité par une méthode de gradient stochastique.

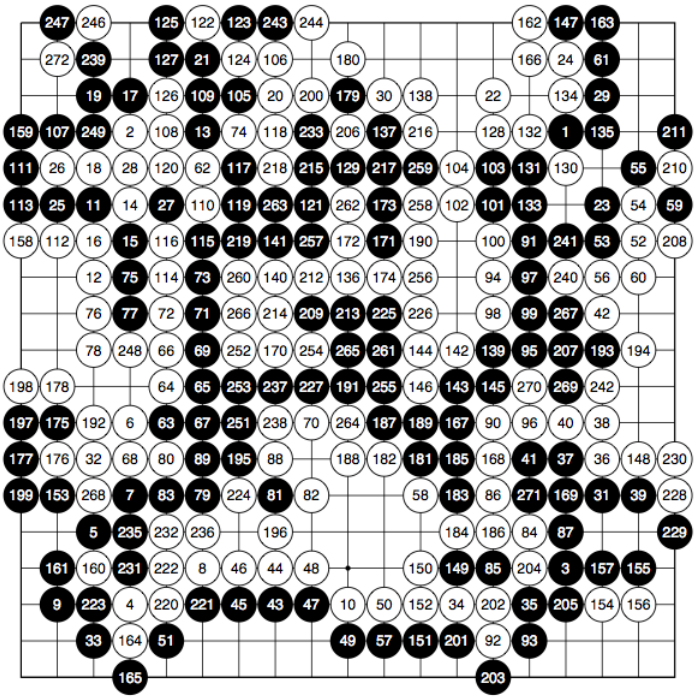

Figure 2: *Première partie du match AlphaGo-Fan Hui. Fan Hui joue Noir et perd de 2,5 points. Les nombres indiquent l'ordre des coups joués. Blanc 234 en 179, Noir 245 en 122, Blanc 250 en 59.*

La première phase consiste à apprendre à retrouver des coups de joueurs experts à partir de parties jouées sur le serveur internet de Go KGS. Trente millions de positions de Go ont été utilisées pour cet apprentissage. A l'issue de cette phase, le taux de prévision du meilleur coup atteint 57.0 % sur une base de test, ce qui est bien meilleur que toutes les approches précédentes qui arrivaient au plus à 44.4 %.

La seconde phase utilise l'apprentissage par renforcement pour améliorer le réseau. L'apprentissage par renforcement consiste à apprendre de l'environnement à partir de récompenses . Si les récompenses sont positives le comportement est renforcé, sinon il est diminué. Le programme joue des parties contre différentes versions de lui-même et apprend les résultats des parties ainsi jouées pour s'améliorer. Le réseau amélioré gagne après apprentissage 80% des parties contre le réseau original. Il gagne aussi 85% de ses parties contre Pachi [16], un programme Monte-Carlo utilisant 100,000 playouts et jouant au niveau de deuxième dan (un bon niveau de joueur de club).

La troisième phase apprend à prévoir l'issue d'une partie à l'aide d'un réseau évaluateur, qui prend en entrée une position de Go et estime en sortie la probabilité de gain associée à cette position. Trente millions de positions différentes sont générées à partir de parties jouées contre lui-même et le réseau évaluateur apprend le résultat de ces parties.

Les différents réseaux de neurones sont finalement combinés dans un programme qui fait de la recherche Monte-Carlo en utilisant les réseaux qui choisissent les coups pour biaiser les parties aléatoires et le réseau évaluateur pour biaiser les statistiques sur les parties aléatoires.

Le programme est enfin parallélisé sur un cluster de CPU et GPU. Il bat les meilleurs programme de Go 99.8 % du temps et il a battu le champion européen de Go Fan Hui cinq à zéro. Les joueurs professionnels estiment que le niveau d'AlphaGo se rapproche de celui des meilleurs joueurs asiatiques, mais ne la pas encore atteint. Google estime que le programme continue de s'améliorer rapidement et a le potentiel pour battre les meilleurs joueurs professionnels. Cela sera observé lors du prochain match, programmé au mois de mars en Corée, entre AlphaGo et Lee Sedol, considéré par beaucoup comme le meilleur joueur actuel.

References

[1] Campbell, M., Hoane, A. , Hsu, F. Deep Blue. *Artificial Intelligence* , 134:57-83, (2002).

[2] David Silver, Aja Huang et 18 autres auteurs Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–503, (2016).

[3] http://deepmind.com/alpha-go.html
Site web de Google DeepMind sur le programme AlphaGo

[4] http://jeudego.org
Site web pour débutants de la Fédération Française de Go

[5] Maddison, C. J., Huang, A., Sutskever, I. , Silver, D. Move evaluation in Go using deep convolutional neural networks. *3rd International Conference on Learning Representations*, (2015).

[6] LeCun, Y., Bengio, Y. , Hinton, G. Deep learning. *Nature*, 521, 436444 (2015).

[7] Coulom, R. Efficient selectivity and backup operators in Monte-Carlo tree search. *In 5th International Conference on Computers and Games*, 7283 (2006).

[8] Kocsis, L. , Szepesvári, C. Bandit based Monte-Carlo planning. *In 15th European Conference on Machine Learning*, 282293 (2006).

[9] Coulom, R. Computing Elo ratings of move patterns in the game of Go. *ICGA J.*, 30, 198208 (2007).

[10] Bouzy, B. , Helmstetter,B. Monte-Carlo Go developments. *In 10th International Conference on Advances in Computer Games*, 159174 (2003).

[11] Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* , 518: 529533 (2015).

[12] Müller, M. Computer Go. *Artif. Intell.*, 134, 145179 (2002).

[13] B. Bouzy, T. Cazenave. Computer Go: An AI-Oriented Survey *Artificial Intelligence* , 132 (1), pp. 39-103 (2001).

[14] Tesauro, G. , Galperin, G. On-line policy improvement using Monte-Carlo search. *In Advances in Neural Information Processing*, 10681074 (1996).

[15] Gelly, S. , Silver, D. Combining online and offline learning in UCT search. *In 17th International Conference on Machine Learning*, 273280 (2007).

[16] Baudis, P. , Gailly, J.-L. PACHI: State of the Art Open Source Go Program *Lecture Notes in Computer Science*, 7168, 24-38 (2012).

[17] Garey, M. R.; Johnson, D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness. *Victor Klee, ed. A Series of Books in the Mathematical Sciences.* San Francisco, Calif.: W. H. Freeman and Co. (1979).

Figure 3: *Logo du Tournoi International de Go de Paris 2016. Le tournoi de Paris a lieu à Pâques tous les ans et attire parfois plus de 300 joueurs.*