A Blokus Player based on AlphaGO Zero

Jean-baptiste Aujogue Alexis Huet

28 août 2018

1 Description abstraite du problème

Le jeu est décrit à tout instant par un **état** $s \in \mathcal{S}$, et autorise un ensemble d'**actions** $a \in \mathcal{A}$ permettant de passer d'un état s à un autre s'. Ce passage est dénoté $a : s \longrightarrow s'$. Les ensembles $\mathcal{S}$ et $\mathcal{A}$ peuvent être infinis, mais on suppose partout dans ce travail que chaque état s possède un ensemble *fini* $\mathcal{A}(s) \subseteq \mathcal{A}$ d'actions possible. L'ensemble des paires $(s, a) \in \mathcal{S} \times \mathcal{A}(s)$ est noté partout $\mathcal{G}$, et on note que cet ensemble constitue les arrêtes d'un graphe orienté acyclique sur l'ensemble $\mathcal{S}$. Le déroulement du jeu consiste alors à faire concourir deux joueurs pour faire évoluer le jeu dans une suite d'états en suivant les flèches de ce graphe, c'est à dire une *marche* dans $\mathcal{S}$, en partant de l'état vide et jusqu'à atteindre un état *final*, où plus aucun coup n'est possible.

L'objectif d'un Joueur artificiel peut être résumé dans l'idée suivante : étant donné un état $s \in \mathcal{S}$ du jeu à un instant donné, le joueur doit allouer une probabilité $p(.|s)$ sur l'ensemble $\mathcal{A}(s)$ des actions permises, puis sélectionner l'action à effectuer : soit en choisissant celle possédant la probabilité la plus élevée, soit en effectuant un tirage aléatoire suivant cette loi. L'établissement de telles probabilités conditionnelles sur chaque état du jeu constitue toute la difficulté du problème.

2 Modèle pour le jeu de Blokus

2.1 Espace d'états du jeu de Blokus

Un état du jeu de Blokus consiste en une grille 20 par 20 recouverte par des formes géométriques de k couleurs, $k = 2$, 3 ou 4 étant le nombre de joueurs. L'état du jeu à un instant t est entièrement déterminé par la configuration du jeu à cet instant, encodée par

Description d'un état

Pour le jeu de GO, un état consiste globalement en une disposition de jetons blancs et jetons noirs sur un échiquier de taille 19 par 19 (usuellement), ainsi que l'information de la couleur suivante à jouer. Précisément, la méthode de [1] consiste à représenter un état par un ensemble de matrices 19 par 19 suivantes : On considère pour chaque temps t une première matrice X_t contenant la valeur 1 là où un jeton noir est présent, et 0 ailleurs, et une seconde matrice Y_t contenant la valeur 1 là où un jeton blanc est présent, et 0 ailleurs. Ces matrices sont définies comme nulles pour $t < 0$. On considère également une matrice C_t 19 par 19

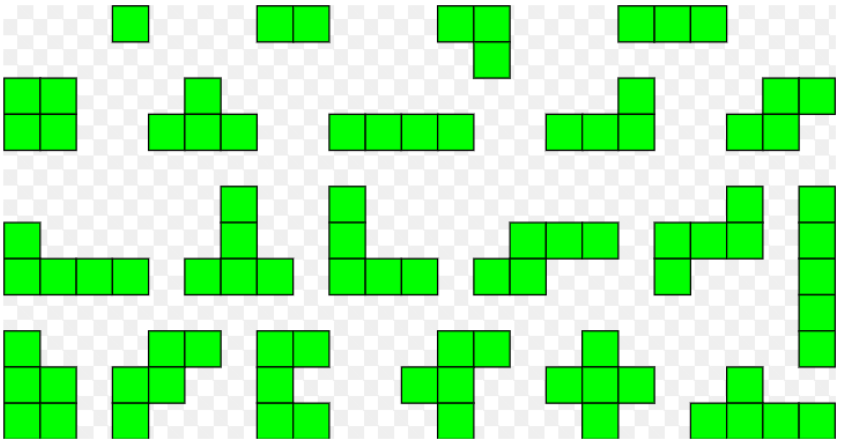

FIGURE 1 – Les différentes pièces que possède un joueur du Blokus en début de partie

identiquement 1 si c'est aux noirs de jouer, 0 si c'est aux blancs. Un état est alors décrit par 17 matrices 19 par 19

$$s_t = (X_t, Y_t, ..., X_{t-7}, Y_{t-7}, C)$$

2.2 Architecture du modèle

Le modèle $\mathcal{M}$ de joueur de Blokus à pour objectif de générer, lorsque le jeu est dans un état $s \in \mathcal{S}$, une probabilité d'action sur l'ensemble des actions a possibles,

$$p(a|s)$$

ainsi qu'une probabilité $v(s) \in [0, 1]$ de victoire du modèle à partir de cet état s. L'architecture du modèle est la suivante :

Architecture du modèle

3 Apprentissage par renforcement

La phase d'entrainement du modèle consiste à faire concourir un modèle contre lui-même, en le faisant jouer à tour de rôle les blancs et les noirs dans un ensemble de parties. Le modèle reste inchangé durant l'intégralité de ces parties, puis se base sur les erreurs faites pour effectuer une étape d'entrainement, fournissant un nouveau modèle. Ce nouveau modèle est comparé à l'ancien, puis vient le remplacer s'il est estimé comme plus performant, et oublié sinon.

3.1 Déroulement d'une partie

Chaque partie consiste brièvement en 3 boucles imbriquées : 1) La partie en elle-même consiste en une séquence de coups joués, 2) chaque coup joué nécessite une séquence de simulations, 3) chaque simulation consiste en une séquence de coups fictifs générée par une *Monte Carlo Tree Search* (MCTS).

Plus précisément, chaque partie se déroule de la façon suivante : En début de partie, on initialise deux valeurs pour chaque couple $(s,a) \in \mathcal{G}$:

1) Un nombre $N(s,a)$ de passages par l'état s suivis par l'action a, initialisé à 0.
2) Une récompense totale $R(s,a)$ initialisée à 0.

La partie consiste alors en une séquence d'états, où chaque passage d'un état s_t à l'état suivant s_{t+1} implique un ensemble de $L^{(1)}$ **simulations**, chacune desquelles venant mettre à jour les valeurs ci-haut. On décrit d'abord comment se déroule une simulation : il s'agit d'une partie fictive entièrement générée par le modèle selon une méthode de **MCTS**, qui démarre depuis l'état actuel s_t et termine lorsque un état de fin de partie $\overline{s}$ est atteint ou bien au bout d'un nombre $L^{(2)}$ d'actions. Ce nombre est la *profondeur* de la MCTS. Le modèle joue successivement pour les noirs et les blancs, effectuant à l'état s l'action $a : s \longrightarrow s'$ qui maximise un terme de confiance

$$\underbrace{\frac{R(s,a)}{N(s,a)}}_{\text{récompense moyenne}} \quad + \quad c_{\text{puct}} p(a|s) \frac{\sqrt{\sum_{b \in \mathcal{A}(s)} N(s,b)}}{1+N(s,a)}$$

où c_{puct} est un coefficient bien choisi, $p(.|s)$ est la probabilité fournie par le modèle $\mathcal{M}$ sur les actions possibles $\mathcal{A}(s)$ depuis l'état s. Lorsque la partie fictive arrive à l'état $\overline{s}$, le modèle $\mathcal{M}$ fournit la probabilité de victoire $v = v(\overline{s})$, et chaque couple (s,a) appliqué lors de cette partie fictive effectue alors la mise à jour

$$N(s,a) \mathrel{+}= 1 \qquad\qquad R(s,a) \mathrel{+}= v$$

Cette mise à jour faite dans une simulation a alors un impact sur la partie fictive de la simulation suivante. Une fois l'ensemble des $L^{(1)}$ simulations effectuées, chaque action possible $a \in \mathcal{A}(s_t)$ a été choisie un nombre $N(s_t,a)$ depuis l'état actuel s_t de la vraie partie, et l'état suivant s_{t+1} est alors le résultat d'une action a_t obtenue par tirage aléatoire selon la probabilité sur $\mathcal{A}(s_t)$ donnée par

$$\pi(a|s_t) := \frac{N(s_t,a)^{\frac{1}{\tau}}}{\sum\limits_{b \in \mathcal{A}(s_t)} N(s_t,b)^{\frac{1}{\tau}}}$$

où $\tau \in [0,1]$ est un "paramètre de température". Une fois atteint l'état suivant s_{t+1}, le modèle effectue alors la même procédure de simulations et génère un état suivant s_{t+2}. On remarque que les mises à jours des termes $N(.)$ et $R(.)$ effectuées lors des multiples simulations entre s_t et s_{t+1} sont conservées, et ont un impact sur les traitements faits entre les états s_{t+1} et s_{t+2}. Cette marche dans $\mathcal{S}$ se poursuit jusqu'à atteindre un état de fin de partie s_{fin}, et la clôture de cette partie permet d'attribuer à chaque état traversé s un score $z(s) = 1$ ou 0, selon si le joueur devant jouer à l'état s a gagné la partie ou non.

On accumule alors dans une mémoire l'ensemble des états par lesquels la partie est passée, avec pour chaque état s traversé le **terme d'erreur**

$$l(s) = (z(s) - v(s))^2 + \pi(.|s)^{\mathrm{T}} \log p(.|s) + c\|\mathcal{M}\|^2$$

où c est un coefficient de pénalisation et $\|\mathcal{M}\|^2$ un terme quantifiant la taille du modèle.

Le terme de température τ est pour alphaGO zero fixé à $\tau = 1$ pour les 30 premiers coups, ie $t = 0, ..., 29$ puis choisi très proche de 0 pour les coups suivants. Ce déroulement de partie ne comprend pas de mécanisme d'abandon du modèle lorsque sa confiance en le fait de gagner la partie passe en dessous d'un certain seuil. Les probabilités conditionnelles $p(.|s_t)$ générées par le modèle $\mathcal{M}$ au passage dans chaque état s_t peuvent également être légèrement bruitées, facilitant l'exploration de nouveaux coups au début de chaque MCTS.

3.2 Procédure de mise à jour du modèle

4 Références

[1] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676) :354, 2017.