Univeristée Paris Sud - FIIL

Rapport de stage 2020
Problème du voyageur canadien

Joé DE OLIVEIRA

Table des matières

Chapitre 1

Présentation du sujet

Ce stage est centré autour du problème du voyageur Canadien, introduit par Yannakakis et Papadimitriou en 1991[4], et sur la manière d'utiliser un Monte-Carlo-Tree-search, introduit par Rémi Coulom en 2006[2], pour chercher des solutions efficaces à ce problème. Il a été effectué sous la direction de Joanna Tomasik et d'Arpad Rimmel.

1.1 Problème du voyageur Canadien

Le problème du voyageur Canadien ou Canadian traveller's problem (CTP), est une généralisation du problème du plus court chemin sur des graphes partiellement observables, ou les liens ont une chance d'être fermés.

Le CTP est donc un problème d'optimisation : le but est de trouver le meilleur chemin possible avec les informations incomplètes dont nous disposons.

Il existe de nombreuses variantes de ce problème, pour ce stage nous allons nous concentrer sur la variante suivante :

-Le succès d'une stratégie est déterminée en fonction du ratio entre le chemin trouvé et le plus court chemin sur un graphe découvert.

-La chance qu'un noeud soit ouvert ou fermé est déterminée de manière stochastique, avec une probabilité connue a l'avance.

-on considère qu'on "voit" l'état des liens adjacents au noeud actuel

-un lien est soit ouvert, soit fermé, sans état intermédiaire.

-une fois l'état d'un lien connu, on considère qu'il ne change pas.

Voici un aperçu d'un graphe pouvant être utilisé pour notre problème de CTP, Nous somme actuellement sur le noeud courant A, nous voyons donc les liens adjacents vers B et C, le lien [A,B] est fermé ; nous ne savons pas si les autres noeuds sont ouverts, mais nous avons accès à la probabilité qu'ils le soient, par exemple, le lien [B,C] a 70% de chance d'être fermé.

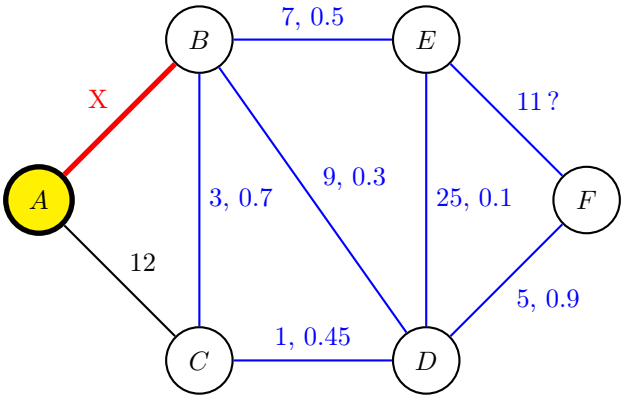

FIGURE 1.1 – visualisation d'un CTP

1.2 Objectif du stage

Le but de ce stage a été de réimplémenter les méthodes utilisées dans le papier de Patrick Eyerich, Thomas Keller et Malte Helmer : High-quality policies for the canadian traveler's problem[3], et de chercher des pistes pour améliorer ces résultats.

Chapitre 2

Monte Carlo Tree Search

2.1 Definition

La méthode que le papier utilise pour résoudre les CTP est basé sur les Monte-Carlo-Tree-Search(MCTS), dans cette partie nous allons définir ce qu'est un algorithme de MCTS, et dans la suivante nous verrons comment l'article sur lequel nous nous basons[3] l'a adapté pour les besoins du problème.

Le MCTS est un algorithme de prise de décision, il est souvent utilisé en machine learning pour les jeux, notamment le go[5], comme son nom l'indique, cet algorithme utilise un arbre de posibilitée, la racine représente l'état initial, et ses enfants sont les états possibles après un coup.

MCTS est un algorithme any-time, on lui donne un temps maximum d'exécution et/ou un nombre maximum d'iteration et MCTS rend un coup à jouer à la fin de ce temps.

2.1.1 Les quatre étapes

L'algorithme fonctionne à partir de 4 étapes, que nous pouvons voir sur la figure ci-contre, qui représente une iteration,

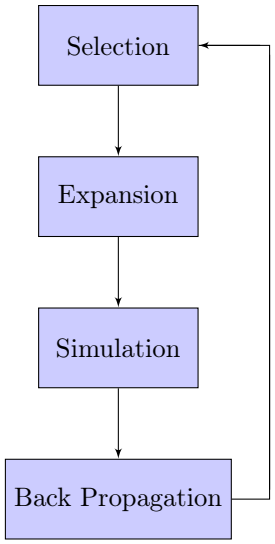

FIGURE 2.1 – Les 4 étapes de MCTS

La Sélection :

la sélection est l'étape ou la question de l'exploration contre l'exploitation se pose, en effet, dans ce type de problème, il existe toujours un compromis à trouver entre l'exploitation d'un coup qui paraît prometteur, et l'exploration pour trouver un potentiel meilleur coup, il existe de nombreuses manières de gérer ce compromis, mais la manière classique est celle de l'Upper Confidence Bound(UCT),

$$\frac{wi}{ni} + c\sqrt{\frac{ln(n)}{ni}}$$

wi : nombre de victoire du noeud
ni : Nombre de visite du noeud
n : nombre d'itération <=> nombre de visites de la racine
c : paramètre d'exploration (nombre magique, classiquement $\sqrt{2}$)

La sélection consiste à descendre dans l'arbre de décision, (en choisissant un coup grace à la formule UCT), jusqu'à arriver à un noeud qui représente une partie terminée ou un noeud où il existe des coups possibles qui n'ont pas été ajouté à l'arbre de décision.

L'Expansion :

à partir du noeud obtenu avec la sélection, si nous ne sommes pas à un état final, nous prenons un coup possible qui n'est pas présent dans l'arbre, et l'ajoutons à l'arbre.

La Simulation :

à partir du coup ajouté à l'arbre, nous lançons une simulation, qui va jouer des coups valides aléatoires (sans les ajouter à l'arbre) jusqu'à arriver à un état final.

La Back Propagation : On observe le résultat de la partie simulée, et on se sert du résultat pour mettre à jour les noeuds de l'arbre, en remontant à partir du noeud ajouté dans l'expansion.

2.2 Version du Papier

Le papier qui nous sert de base[3] utilise une version modifiée du MCTS pour résoudre le problème du voyageur canadien, dans cette partie nous allons voir les changements apportés.

2.2.1 Jeu à un joueur

Le premier changement à effectuer est le passage d'un MCTS classique, qui gère des jeux à deux joueurs, à une version qui permet de simuler des "jeux" à un joueur,

pour cela, il suffit d'un changement dans la back-propagation, au lieu de mettre à jour les noeuds en fonction du joueur qu'ils représentent, on les met simplement tous à jour de la même manière.

2.2.2 Jeu à score

Il faut ensuite permettre d'évaluer des jeux dont le résultat n'est pas binaire,

là encore, il suffit de modifier la back-propagation pour retourner un score au lieu d'un résultat binaire, et d'évaluer la qualité d'un coup non plus avec le pourcentage de victoires, mais avec le score moyen.

2.2.3 CTP en tant que jeu

Maintenant que notre MCTS est adapté pour des problèmes comme celui du CTP, nous définissons le CTP en tant que jeu ;

un jeu de CTP est représenté par un graphe, un état de départ et un état d'arrivée. nous pouvons effectuer 3 actions sur cet objet : - donner un état et un weather pour obtenir une liste des coups possibles, il est à noter que le papier définit les coups possibles d'une manière particulière : au lieu de les définir de manière naïve, comme la liste des noeuds adjacents à notre noeud actuel, ils sont définis comme la liste des noeuds non visités accessible depuis l'ensemble des noeuds déja visité, un coup étant donc la suite de déplacements représentant le plus court chemin pour aller du noeud actuel au noeud voulu, en passant uniquement par des noeuds déja visité. Cette méthode a l'avantage de borner le nombre de déplacement, car on finira toujours par explorer tout les noeuds, et l'inconvénient d'etre plus complexe à calculer. - donner un état et un coup à jouer et obtenir un nouvel état. - donner un état et obtenir une évaluation de cet état, c'est a dire savoir si la partie est terminée (autrement dit si nous somme arrivés à notre destination) et obtenir le score si c'est le cas.

2.2.4 Calcul du weather

Afin que notre simulation du CTP soit pertinente, il nous faut des "weathers" aléatoires, un "weather" est une distribution possible des liens de notre graphe ; nous ajoutons donc une étape MCTS, avant la sélection, ou nous calculons un "weather" valide : c'est un "weather" dans lequel il existe un chemin entre notre point de départ et notre point d'arrivée.

2.2.5 Formule de selection

la formule de sélection est ensuite modifiée, pour l'adapter à un jeu avec score :

$$(b * \sqrt{\frac{ln(n)}{ni}}) - si$$

b : paramètre d'exploration (ici le score moyen de la racine)
ni : Nombre de visites du noeud
n : nombre d'itération <=> nombre de visites de la racine
si : score moyen du noeud

2.2.6 Version compacte

Le papier[3] n'utilise pas une version classique du MCTS, en effet, les étapes de sélection d'expansion et de simulation ont été fusionnée dans une seule étape.

Dans cette version, la sélection, au lieu de s'arrêter quand elle arrive à une partie ou il existe des noeuds non ajoutés à l'arbre, elle ajoute immédiatement ce noeud et continue jusqu'à arriver à un état final, enlevant ainsi le besoin d'une méthode d'expansion, qui est incluse dans la sélection, et de simulation, qui n'est plus nécessaire car on arrive toujours à un état final dans l'arbre de décision.

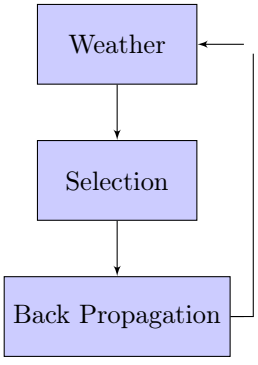

FIGURE 2.2 – Les étapes du MCTS modifié

2.2.7 Version optimistique

Avec les changements apportés, nous avons obtenu l'une des 2 versions
du papier : la version dite "Blind", celle-ci est peu efficace, afin d'obtenir la version "optimistique" qui présente les meilleurs résultats du papier, il nous reste un changement à apporter : l'estimation optimistique.

Nous utilisons une méthode d'approximation pour simuler des itérations du MCTS : quand nous appliquons la formule de sélection, nous agissons comme s'il y avait eu un nombre x d'iterations supplémentaires, dont le score est celui de la méthode d'approximation, le papier estime qu'un nombre entre 5 et 80 simulations d'iterations est raisonable, nous nous somme fixé sur 20. Ce nombre permet d'influencer asser fortement les premières iterations, mais a un impact faible quand suffisamment d'iterations ont été effectué, ce qui permet de partir sur une meilleure base, mais de continuer à s'améliorer par la suite.

nous utilisons également ces simulations de sélection quand nous somme face à des noeuds non explorés : au lieu de choisir au hasard, nous choisissons le noeud avec le meilleur score estimé.

la méthode d'approximation utilisé est la suivante : nous prenons tous les liens ouvert et indéterminé dans notre graphe et nous calculons le plus court chemin à partir de ceux-ci, ce plus court chemin est notre estimation.

Chapitre 3

Variantes

3.1 Les variables

Nous avons identifié 3 variables qui changent le fonctionnement de notre problème de manière intéressante :

La méthode de sélection de coup :

il y a en premier lieu les 2 méthodes définies plus tôt : la méthode Blind et la méthode optimistique, à cela, nous allons ajouter une méthode optimistique "lourde", qui, au lieu de calculer le cout optimistique une fois par sélection, le fera a chaque changement de profondeur durant la sélection.

La définition d'un successeur :

Le papier définit les successeurs comme les noeuds non visités accessibles depuis au moins un des noeuds déjà visité, nous pouvons aussi le définir plus simplement, comme la liste des voisins accessible depuis le noeud actuel.

Le type d'actions que représente un arbre de Monte-Carlo :

Dans le cas où les successeurs sont définis comme dans le papier, il est possible que, pour accéder à un successeur, nous devions effectuer une suite de mouvements, il y a 2 manières de représenter ceci dans un arbre : ajouter chaque coup un par un dans l'arbre, ce qui est la version par défaut, ou décider qu'un ensemble de coups ne représente qu'une seule action dans l'arbre.

3.2 Les variantes

En utilisant ces différentes variables ensemble ont obtiens des variantes, nous allons passer rapidement sur chacune d'entre elles afin de voire lesquelle sont pertinentes.

1 Blind Simple
Cette variante est la variante la plus légère, car le gros du temps d'exécution est dépensé dans le calcul des plus cours chemin, qui sont calculés, soit pour le calcul d'optimistique, soit pour le calcul des successeurs.

2 Blind Smart Single
C'est l'une des 2 variantes du papier, elle n'est pas efficace et on peut considérer que toute variante qui obtient un score pire que cette version n'est pas pertinente.

3 Blind Smart Batch
Cette variante, ainsi que la 6 devraient nous donner un aperçu de la viabilité de la variante batch, car on peut les comparer directement avec les variantes du papier.

4 Optimistic Simple
Cette variante n'est pas pertinente, la sélection peut occasionnellement ne pas se terminer si elle tombe sur des cas similaires à la figure ci-dessous :

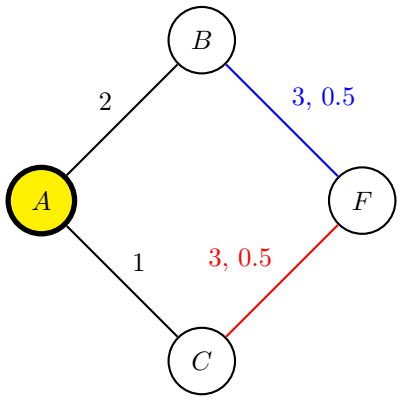

FIGURE 3.1 – exemple de graphe pouvant causer un problème

ici, si la sélection décide d'aller de A vers C, elle découvrira que le chemin de C a F est fermé, et retournera vers A à la prochaine étape de la sélection, cependant, rien ne lui indiquera de ne pas retourner vers C à l'étape suivante, car nous n'avons pas mis à jour le calcul optimistique qui décide où aller et C est encore un successeur valide de A.

5 Optimistic Smart Single
C'est l'une des 2 variantes initiale, celle-ci représente les meilleurs résultats du papier, c'est la variante à dépasser.

6 Optimistic Smart Batch
Cette variante, ainsi que la 3 devraient nous donner un aperçu de la viabilité de la variante batch, car on peut les comparer directement avec les variantes du papier.

7 Optimistic Lourd Simple
Cette variante est la seule variante optimistique à utiliser le déplacement simple, a cause de la non-viabilité de la variante 4.

8 Optimistic Lourd Smart Single
Cette variante utilise les version les plus "avancée" de 2 de nos variables. on peut donc supposer que les résultats par iteration seront bons.

9 Blind Smart Batch
Cette variante utilise les version les plus "avancée" de 2 de nos variables. on peut donc supposer que les résultats par iteration seront bons.

Nous allons maintenant observer les résultats de ces différentes variantes.

Chapitre 4

Résultats

4.1 résultats et analyse

Nous avons testé nos variantes sur des graphes Delaunay[1] de taille 20, avec en premier lieu des tests avec un temps donné, puis des tests avec un objectif d'iteration à atteindre.

Les tests en temps sont effectué sur 2,5 et 20 secondes ; 310 instances de ces tests ont été effectuées.

Les tests en iteration sont effectué sur 100,1000 et 10000 iterations ; 109 instances de ces tests ont été effectuées.

Sur ces tests, visible sur les figures 4.2 et 4.3, on peut voir quatre groupes de performance se former : -Le blind simple -Les 2 blind smart -Les version optimistique smart -L'optimistique simple heavy

Nous constatons également que la différence entre les variantes batch et single est minime, avec un possible léger avantage pour single.

La méthode "optimisc simple heavy" semble être la plus prometteuse, sur la figure suivante nous pouvons la comparer plus simplement avec la version du papier,

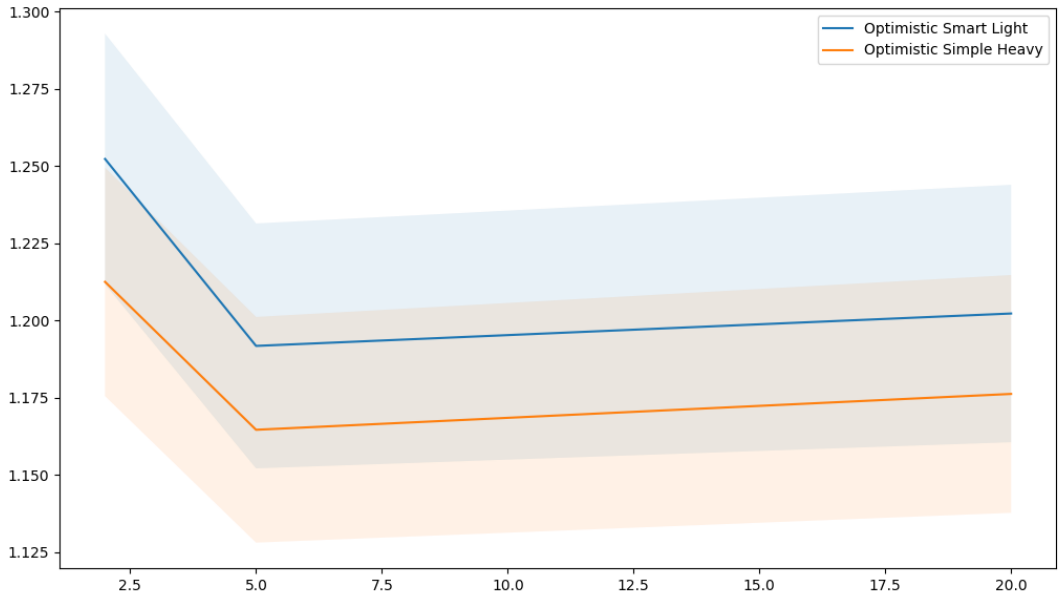

FIGURE 4.1 – comparaison entre la version de [3] et notre potentielle amélioration

Ici nous pouvons voir que, bien que la méthode "optimistic heavy simple" semble être la meilleure, nous n'avons pas encore asser de tests pour en être sur à 95%, avec le nombre de tests actuels, l'intervalle de confiance de cesse de se croiser qu'à 48%, ce qui est insuffisant pour affirmer que la nouvelle variante est meilleure que la variante du papier, nous n'avons malheureusement pas eu le temps nécessaire pour générer asser de test pour être fixé.

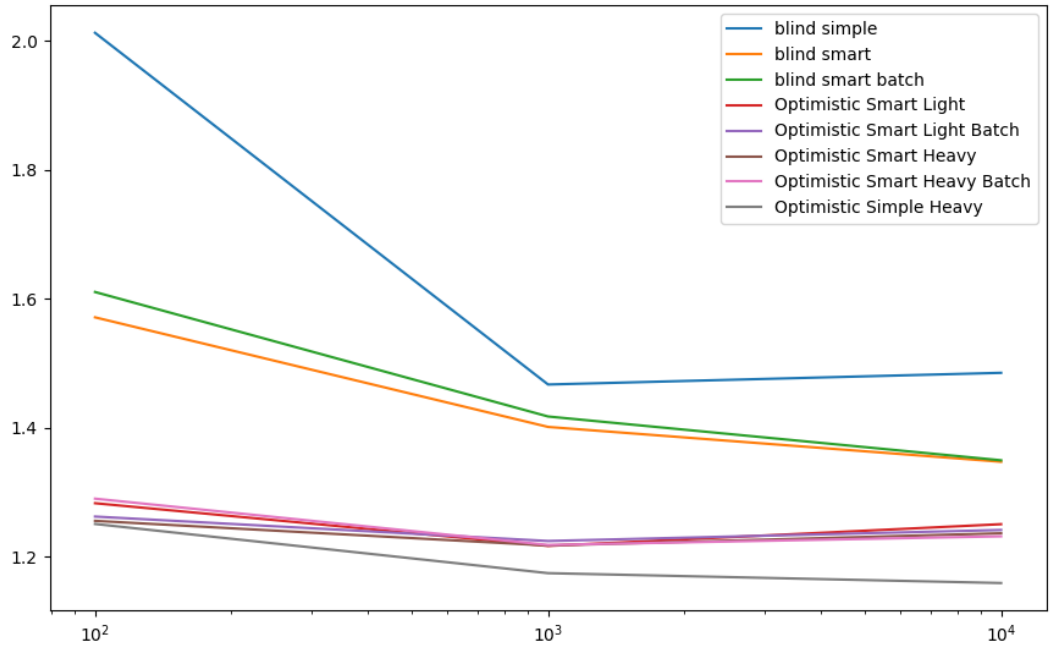

FIGURE 4.2 – résultat en nombre d'iteration

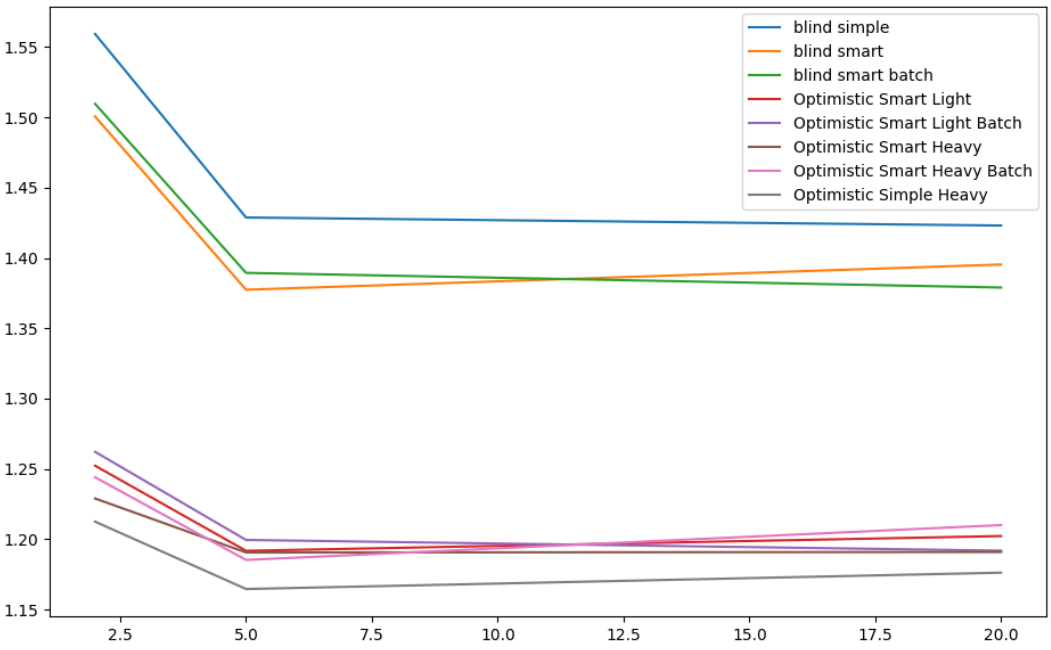

FIGURE 4.3 – résultat en temps d'execution

Chapitre 5

Bilan

Nous avons expérimenté avec plusieurs variantes dans le but de trouver des solutions plus efficaces au problème du voyageur Canadien, bien que n'ayant pas eu le temps d'effectuer asser de tests pour avoir une réponse définitive quant à la supériorité de la variante "Optimistic simple heavy", il n'en reste que les résultats obtenus sont meilleurs en moyenne que ceux de la version du papier que nous avons cherché à améliorer.

La première tâche à effectuer dans une éventuelle continuation serait évidemment de générer plus de test afin de confirmer les résultats. Une fois ceci fait, nous avons deux pistes potentiellement intéressantes : -La réutilisation des arbres de décision générés après chaque coup joué. -Développer de nouvelles variantes à partir d'une version non compacte du MCTS.

Bien qu'ayant, comme tout le monde, du faire face aux problèmes liés au COVID et ayant dû effectuer mon stage en télétravail, je suis satisfait d'avoir trouvé une variante qui semble donner une amélioration par rapport à l'état de l'art actuel.

Le code produit durant ce stage est disponible sur `https://github.com/EtherFrog/StageM2/`

Bibliographie

[1] Delaunay Boris. Sur la sphere vide. a la memmoire de georges voronoi. http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=im&paperid=4937&option_lang=eng, 1934.

[2] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. https://hal.inria.fr/inria-00116992/document, 2006.

[3] Patrick Eyerich, Thomas Keller, and Malte Helmer. High-quality policies for the canadian traveler's problem. https://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/viewFile/1735/1922, 2014.

[4] Papadimitriou C.and Yannakakis M. Shortest paths without a map. https://www.sciencedirect.com/science/article/pii/0304397591902632, 1991.

[5] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree searchoi. https://www.nature.com/articles/nature16961, 2016.