Univeristée Paris Sud - FIIL

Rapport de stage 2020
Probleme du voyageur canadien

Joé DE OLIVEIRA

Table des matières

Chapitre 1

Présentation du sujet

Ce stage est centré autour du problème du voyageur Canadien, et comment utiliser un Monte-Carlo-Tree-search pour chercher des solutions efficaces à ce problème.

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.

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.

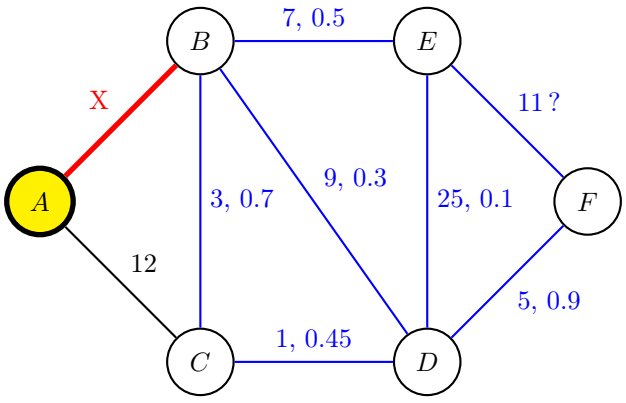
-La chance qu'un noeud soit ouvert ou fermé est déterminée de manière stochastique, avec une probabilité connue a l'avance.

-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.



Plus de Schémas représentant le comportement de notre problème de CTP sont dans l'annexe [WIP]

1.2 Objectif du stage

Le but de ce stage a été de réimplémenter les méthodes utilisées dans le papier[1], et de chercher des pistes pour améliorer ces résultats.

Chapitre 2

Monte Calro 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 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), 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 4 etapes

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

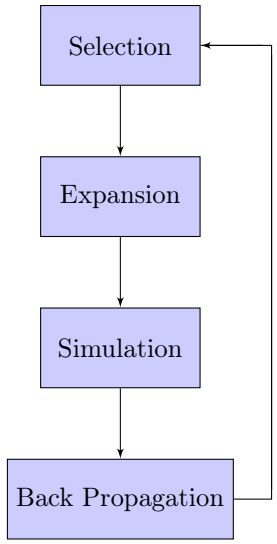


FIGURE 2.1 – Les 4 étapes de MCTS

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

transfomrations :

-passage a 1 joueur

-adaptation pour le "jeu" CTP

-ajout du weather

-version Deep : fusion de selection expantion et simulation

-blind(par défault) et modification pour optimistic

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.

```
(4eme variable ?)
-réutiliser l'arbre pour améliorer les performance?( absurde avec batch?)
```

3.2 Les variantes

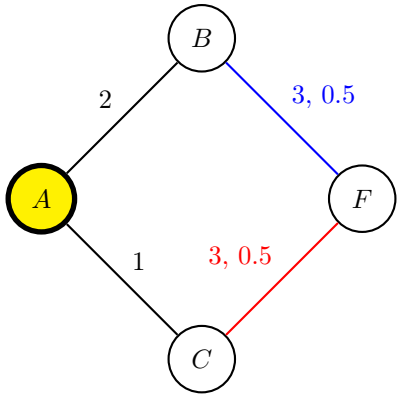
En utilisant ces différentes variables ensemble ont obtiens des variantes, nous allons passer rapidement sur chacune d'entre elles afin de voire lesquelle sont pertinentes.

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 NOT DONE YET
? ? ?

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 :



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 A a 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 NOT DONE YET
? ? ?

7 Optimistic Lourd Simple
Cette variante, ainsi que la 8, sont les 2 variantes les plus lourdes.

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

3 Blind Smart Batch NOT DONE YET
? ? ? . . .

Nous allons maintenant observer les résultats de ces différentes variantes.

Chapitre 4

Résultats

4.1 résultats

```
2 seconds per decision :
sample size : 331
Blind Simple :
278.09192963418724 (1.5507412438495005, 1.4695969920751248, 1.6318854956238762)
Blind Smart :
265.0485931802221 (1.476116484092765, 1.4072446640109153, 1.5449883041746146)
Optimistic Smart Light :
227.5447593400038 (1.2713563054772266, 1.2233944851700405, 1.3193181257844127)
Optimistic Smart Heavy :
225.5361879466939 (1.2575464972592, 1.2134081557906709, 1.3016848387277293)
Optimistic Simple Heavy :
212.41862389751853 (1.1856363897833857, 1.1479746270673248, 1.2232981524994466)

20 seconds per decision :
sample size : 331
Blind Simple :
287.4554451235579 (1.589761338755095, 1.5007241173064074, 1.6787985602037827)
Blind Smart :
268.8998841836993 (1.494157566533144, 1.4193804341197434, 1.5689346989465445)
Optimistic Smart Light :
228.79802677965085 (1.2719763850749912, 1.2261821852952914, 1.317770584854691)
Optimistic Smart Heavy :
227.41432885139798 (1.2616273883353832, 1.2163418515472264, 1.30691292512354)
Optimistic Simple Heavy :
220.0011416269495 (1.2197238634497354, 1.177824507907276, 1.2616232189921948)

-finir et tester les variantes avec batch
-faire et tester variante avec sauvegarde d'arbre?
-plus de test (taille de graph 50,100) (donner plus de temps?)
-compter nb d'iteration moyenne
```

4.2 analyse

```
-Optimistic Simple Heavy semble etre le meilleur (si confirmé -> gain par rapport au papier)
-plus de temps semble baisser la qualité du résultat?
(plus de test a faire, ne va pas dans le sens du papier original)
```

Chapitre 5

Bilan

Bibliographie

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