Ce document à pour but de présenter par le biais d'une veille technologique l'algorithme de **Recherche Arborescente Monte-Carlo (RAMC)** ou **Monte Carlo tree search (MCTS)** en anglais utilisé par exemple dans le jeu "**TOTAL WAR: ROME II**" de la société de jeux vidéo Creative Assembly.

Durant cette documentation présentant la veille technologique nous verrons :

- Premièrement l'origine de cet algorithme.
- Deuxièmement son utilisation (principe, application, utilité et l'exemple d'une implémentation si possible).
- Puis troisièmement nous verrons les avantages de l'utilisation d'un tel algorithme mais également ces désavantages généraux.

I. Origine de l'algorithme RAMC

La **Recherche Arborescente Monte-Carlo** provient de la ***Méthode de Monte-Carlo*** qui désigne une famille de méthodes algorithmiques inventée en 1947 par le physicien Nicholas Metropolis. Ces algorithmes visent à calculer une valeur numérique approchée en utilisant des procédés dit aléatoires ou plus simplement expliqués par l'utilisation de techniques probabilistes.

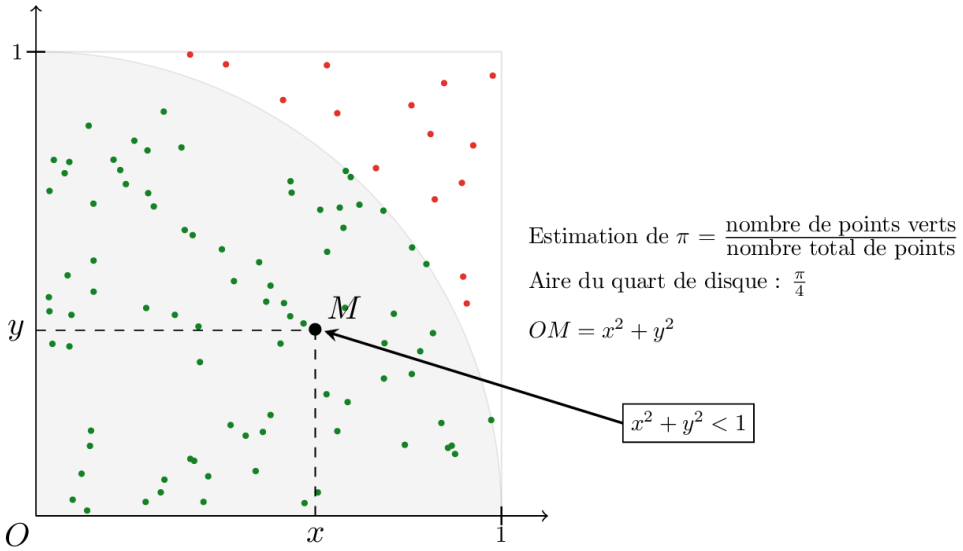

*Figure 1 : Exemple **Méthode Monte-Carlo** pour l'estimation de π*

Quant à la méthode *RAMC*, descendent de la famille de la *Méthode Monte-Carlo,* date de l'année 2006 et a été découverte indépendamment par plusieurs chercheurs en intelligence artificielle dont Rémi Coulom, informaticien français, qui fut celui qui décrit l'application de cet algorithme et l'inventeur de son nom. Cet algorithme est dit **heuristique** notamment dans les prises de décisions employés dans les jeux qu'ils soient vidéos, société ou réels.

II. Utilisation du RAMC

1. Principe

Pour expliquer son principe nous utiliserons le graphique suivant :

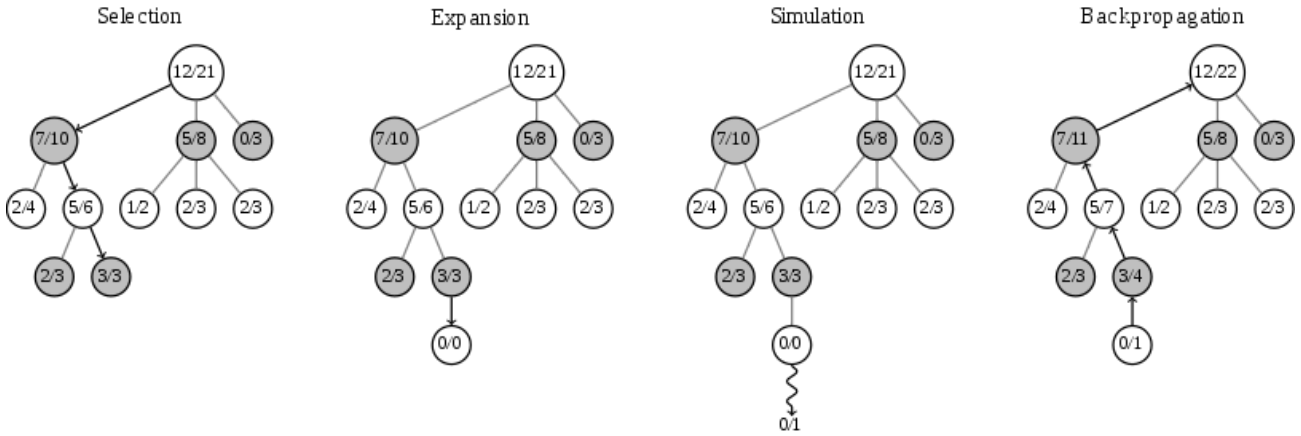

Figure 2 : Graphique montrant les étapes d'un RAMC

Le principe de cet algorithme est d'**explorer l'arbre des possibles**. C'est à dire que l'on part d'une racine correspondant à l'initiale du jeu puis dans chaque nœud représentant une configuration différente on y **explore les différentes possibilitées** à travers ses enfants jusqu'à arriver à ce que l'on appel des feuilles de l'arbre qui symbolise à la fois la configuration finale ou bien un nœud non exploré. On note que dans chacun des nœuds on **stocke deux nombres : le nombre de simulations gagnantes et le total de simulations**.

Lors de l'exploration des possibles on passe par quatre grandes phases :

- La première phase est **la sélection qui** désigne le fait de choisir entre **exploitation** (aller vers un enfant qui a été prouvé comme prometteur) et **exploration** (aller visiter un autre enfant, qui a l'air moins prometteur mais qui pourrait l'être davantage). Ici dans le graphique le choix est le nœud de gauche.

- **L'expansion** à pour but de créer un enfant ou plusieurs suivant les règles du jeu sauf si la feuille se termine de manière décisive. Sur le graphique l'expansion est représentée par l'**ajout de l'enfant marqué 0/0**.

- La **simulation** représente la partie où l'on exécute la suite de la partie au hasard jusqu'à une configuration finale.

- La dernière phase représente la **rétropropagation**. Cela à pour but de simplement **mettre à jour les informations** sur la branche en partant du nœud enfant vers la racine. Dans le graphique comme la partie simulée a donné une défaite alors il suffit simplement d'**incrémenter les simulations totales** sur chacune des branches sinon il aurait fallu également augmenter les simulations gagnantes.

La partie la plus difficile dans l'application de cet algorithme est la première phase : **sélection**. La stratégie appliquée utilise une fonction d'évaluation pour **sélectionner de manière optimale les nœuds** avec la valeur estimée la plus élevée. Cette fonction utilise la

formule de la limite de confiance supérieure (UCB) appliquée aux arbres comme stratégie dans le processus de sélection pour traverser l'arbre. Il équilibre le compromis exploration-exploitation.

$$S_i = \frac{w}{n} + C\sqrt{\frac{\ln N}{n}} = x_i + C\sqrt{\frac{ln(t)}{n_i}}$$

S_i = valeur pour le nœud i.

$x_i = \frac{w}{n}$ = moyenne empirique d'un nœud i => simulations gagnantes au i ème coup divisé par simulations totales au i ème coup.
C = une constante qui représente le paramètre d'exploration.
t = N = nombre total de simulations où le nœud, père de i, a été visité.

L'exploitation est représenté par $\boldsymbol{x_i}$ alors que l'exploration est $\boldsymbol{C}\sqrt{\frac{ln(t)}{n_i}}$.

2. Application et utilité

Maintenant que nous avons vu le principe de l'algorithme voyons dans quel cas il peut être utilisé. Cet algorithme est particulièrement utile dans des jeux où il n'y aucun élément de hasard dans la mécanique du jeu et qui sont au tour par tour avec un nombre fini de coups comme les jeux d'échecs, de go ou encore Puissance 4. Nous pouvons prendre par exemple son application dans le programme **AlphaGo** capable de jouer au jeu de go, où il est utilisé pour battre les meilleurs joueurs de go du monde.

Figure 3 : AlphaGo battant au go l'un des meilleurs joueurs mondiaux Lee Sedol

Attention, il est important de noter que cet algorithme ne se limite pas seulement à des jeux il peut être utilisé dans des situations décrites par des paires état-action et des simulations utilisées pour prévoir les résultats comme par exemple pour résoudre des problèmes de déplacement de pièces dans des ateliers d'assemblage.

III. Avantages/désavantages

1. Avantages

Les avantages d'un tel algorithme sont :

- Sa **mise en oeuvre simple** (petit ensemble de fonctions à implémenter)
- Qu'il est **heuristique**
- Il **enregistre n'importe quel état intermédiaire** et peut être réutilisé

2. Désavantages

Les désavantages notable sont :

- La **quantité de mémoire nécessaire**
- **Possibilité qu'il n'existe qu'une seule branche** et qui peut entraîner par conséquent une défaite (généralement dû à une trop grande quantité de combinaisons)
- **Grand nombre d'itération** pour décider le chemin le plus efficace.

IV. Bibliographie

figure 1 : Exemple Méthode Monte-Carlo pour l'estimation de π
figure 2 : Graphique montrant les étapes d'un RAMC
figure 3 : AlphaGo battant au go l'un des meilleurs joueurs mondiaux Lee Sedol

Wikipédia parlant de Monte-Carlo :

- Monte-Carlo tree search
- Méthode Monte-Carlo
- Recherche arborescente Monte-Carlo

Utilisation dans le jeu Total War : ROME II

Explication de Monte-Carlo