# Theory of Computer Games 2022 – Project 3

311551069 余忠旻

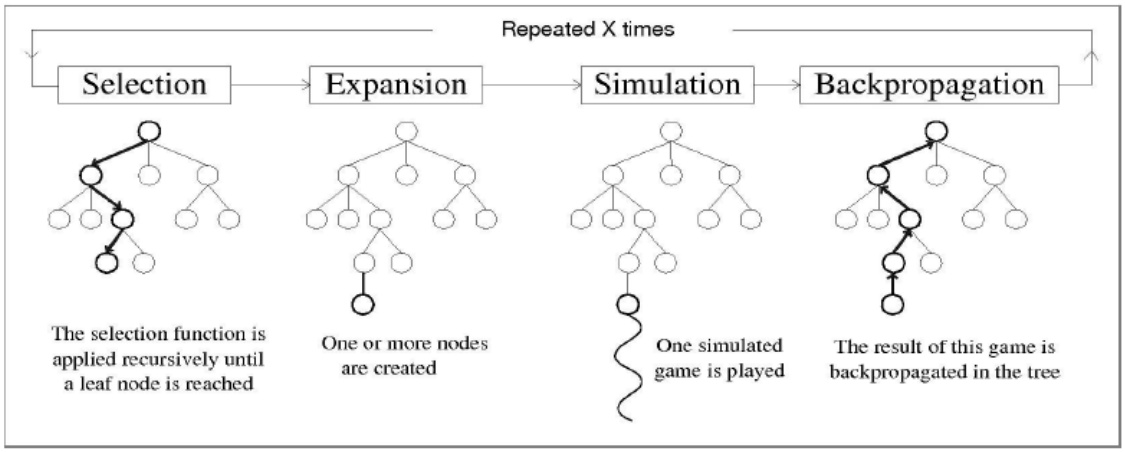
**Method Used**

這次 project 是玩 Hollow NoGo

The player should take actions based on MCTS and should be able to play as both sides.

而 MCTS 會有四個步驟，去模擬棋盤下的過程，再經由不斷的迭代來產生 MCTS Tree，根據 MCTS Tree 來尋找 best value 當作玩家的下一步

MCTS 四個步驟如下:




(1) Selection

此步驟會計算 UCB 值，來不斷的往下搜尋直到抵達 leaf node，再記錄此路徑並決定出最佳的 child node

我判斷 leaf node 則是看這個 node 是不是 fully expanded 以及有沒有剩餘可走步數，而我使用 UCB formula 如下(c=0.5):

$$a^* = \text{argmax}_{a \in legal}\left(Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}\right)$$

(2) Expansion

會透過剛才找到的 leaf node 往下長出一個 expansion node，來繼續下一個步驟。

(3) Simulation

這裡我採用 random 的下法來模擬自己與對手的棋步，一直下到棋盤結束來看是誰獲勝。

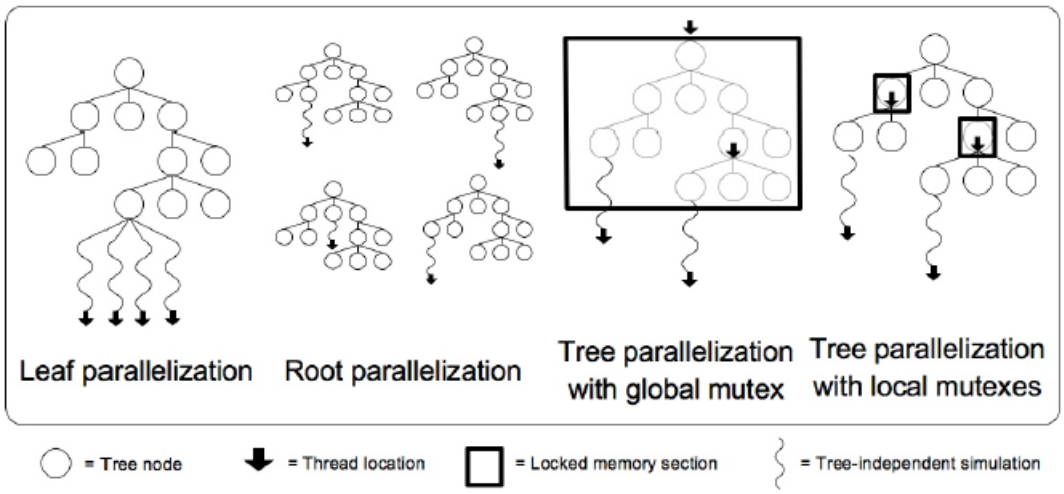
(4) BackPropagation

從 simulation 結果可以得到誰獲勝，來 back propagation 回去，更新路徑的值，如果是我方獲勝，會得到 1 分，反之，對手獲勝，會扣一分。

(5) 這四步驟會重複 N 次(simulation count)，最後 select the best action based on the visit count of child nodes

**Improvement**

Parallelization




在上述各種 parallel MCTS 方法中，我利用 OpenMP 做了 Leaf Parallelization 和 Root Parallelization 的優化，再經過 self-play 來比較，兩者之中我發現 Root Parallelization 的 win rate 有明顯提高，而 Leaf Parallelization 則是不太明顯。

Leaf Parallelization

在 MCTS 的 Simulation 步驟中，playing n games using n different threads，並且將這些結果統計起來，一次性的在 BackPropagation 步驟中做更新。

Root Parallelization

n root nodes using n different threads 來進行 MCTS，最後會將各個 MCTS Tree 的結果統計起來，進行 majority vote，決出最好的下一步。

Unstable-Evaluation Heuristic (UNST)

在 time management 中，可以透過這 strategy 有效提高 win rate

The key concept is that if the most visited move has not the largest score, that means either the most visited move was becoming of low score or a better move was just being found or a potential good move was still exploited near the end of the search. For making sure which move is the best, search is performed more time.

prolongs the search by a factor $f_{unstable}$ and The maximum number of loops until the search is terminated is bound by $l_{unstable}$

**Result**

我採用 Root Parallelization 和 UNST strategy 來對打，結果如下:

```
[chungminyu@tcglinux7 311551069]$ ./run-gogui-twogtp.sh
GoGui-TwoGTP Launcher V20221101
========== PLAYERS ==========
P1B: ./nogo --shell --name="Hollow-Black" --black="mcts count=500 rootParallel"
P1W: ./nogo --shell --name="Hollow-White" --white="mcts count=500 rootParallel"
P2B: ./nogo-judge --shell --name="Judge-Weak-Black" --black="weak"
P2W: ./nogo-judge --shell --name="Judge-Weak-White" --white="weak"
============ GAMES ============
Storage: gogui-twogtp-20221126232306
Monitor: ./gogui-twogtp-20221126232306.mon
P1B vs P2W: ##### 5:0
P2B vs P1W: ##### 0:5
========== RESULTS ==========
P1: (5+5)/10 = 100.0%
P2: (0+0)/10 = 0.0%
```

## Reference


1. G. Chaslot, M. Winands, and H. Herik. *Parallel monte-carlo tree search.* In Proceedings of the 6th international conference on Computers and Games, page 71. Springer-Verlag, 2008.

2. Shih-Chieh Huang, Remi Coulom, Shun-Shii Lin, Time Management for Monte-Carlo Tree Search Applied to the Game of Go. TAAI 2010. (also appeared in ICGA Journal)

3. Hendrik Baier, Mark H.M. Winands, Time Management for Monte-Carlo Tree Search in Go, Advances in Computer Games, Vol. 7168 of Lecture Notes in Computer Science , pp. 39-51. Springer Berlin Heidelberg (2012)