Theory of Computer Games 2022 – Project 4 report 109550031 李旻融

Method

這次一樣是使用 MCTS 來實作，一次 MCTS 包含四個階段:

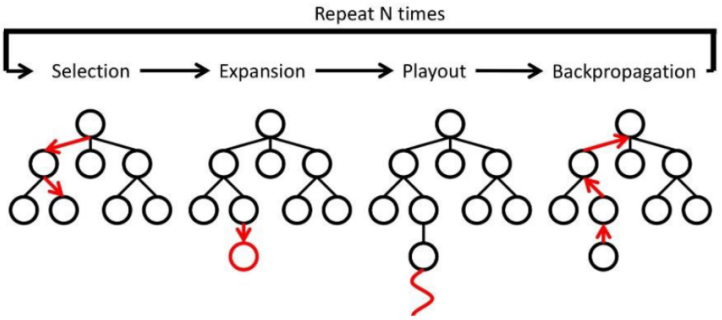
1. Selection: 從根節點開始向下層選擇要移至的位置，直到葉節點為止。

2. Expansion: 長出下一層合法步。

3. Simulation: 雙方輪流隨機地下完這盤棋，得到的結果即是這個節點好壞的估值。

4. Backpropagation: 向上更新回去。

在到達結束條件後，會選擇根節點第一層 child nodes 裡面造訪次數最多的，作為實際下一步要下的動作。



Improvement

- two-player paradigm

  在雙人對局的 MCTS 裡，如果遇上對手的動作，計算勝率要反過來計算，改為計算對手的勝率，而不是自己的勝率。

$$winrate_{opponent} = 1 - winrate_{me}$$

- RAVE

$$Score_i = UCB_i = \frac{W_i}{N_i} + C_{bias} * \sqrt{\frac{log_{10} N_p}{N_i}}$$

W_i : The total win playouts of *i*.
N_i : The total playouts of *i*.
C_{bias}: Constant.
N_p : The total playouts of *p*.

上圖是 UCB 的計算方式, 但是在 Nogo 這種遊戲, 一個動作的值其實不太會受到其它動作的影響, 同一個動作現在做和晚點做的報酬不會差太多, 晚點做的報酬拿來當成現在做的報酬是合理的, 所以我使用了 RAVE 來增進表現。

舉例來說，我執黑子，且這回合 A1 這個位置可以下，下次又輪到我的回合時 A1 假設還能下, 那這兩次 A1 的期望值應該不會差太多，所以如果我晚點才下 A1，它的報酬也能在當前回合使用。

RAVE 是計算時把狀態下的只要有走過的全部算進去, Q 指的是一般的平均報酬, ~Q 指的是利用 RAVE 的平均報酬。但是用 RAVE 估的值有時候會出錯，所以我設定了一個權重 β, 如果樹越大, RAVE 的比重會逐漸上升。

以下為計算 value 的新公式:

$$Q_\star(s,a) = \big(1 - \beta(s,a)\big)\, Q(s,a) + \beta(s,a)\, \tilde{Q}(s,a)$$

我使用 Hand-selected schedule 來定義 β, 當 N(s) 趨近 k 時, β 會趨近於 1/2。

$$\beta(s,a) = \sqrt{\frac{k}{3N(s)+k}}$$

where k specifies the number of simulations

- time management

  使用一個去陣列去決定每一步可以花多少時間，我認為在中場是最關鍵的時刻，所以我給中場較多時間去思考，至於遊戲一開始和最後，相對沒那麼重要，所以時間也給的少一些。

  double time_management[36] = { 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0,
  5.0, 5.0, 9.0, 9.0, 9.0, 9.0, 9.0, 9.0,10.0, 10.0,
  10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0,
  10.0, 10.0, 9.0, 9.0, 9.0, 5.0, 5.0, 1.0 }

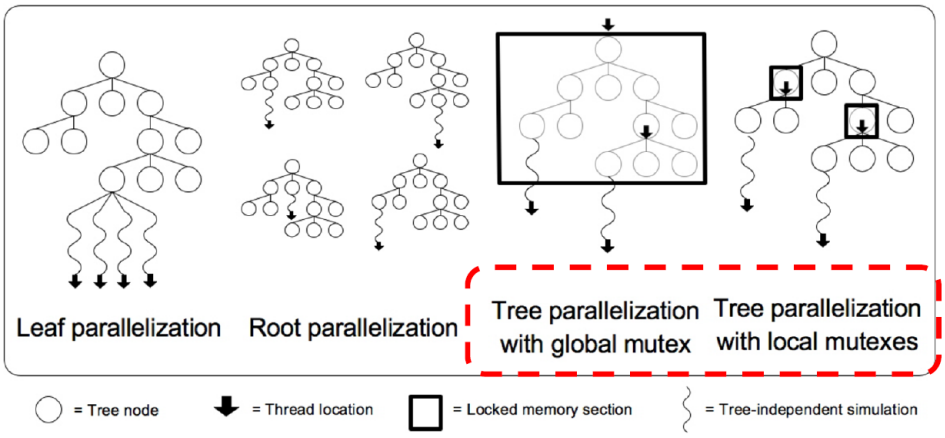
- parallel MCTS

  利用多個 thread 將 MCTS 平行化，共有三種實作方式：

  1. Leaf Parallelization: 只有一個 thread 遍歷整棵樹，每個 thread 分開 simulation

  2. Root Parallelization: 每個 thread 都有自己的樹，最後再結合所有的樹

  3. Tree Parallelization: 每個 thread 共用同棵樹



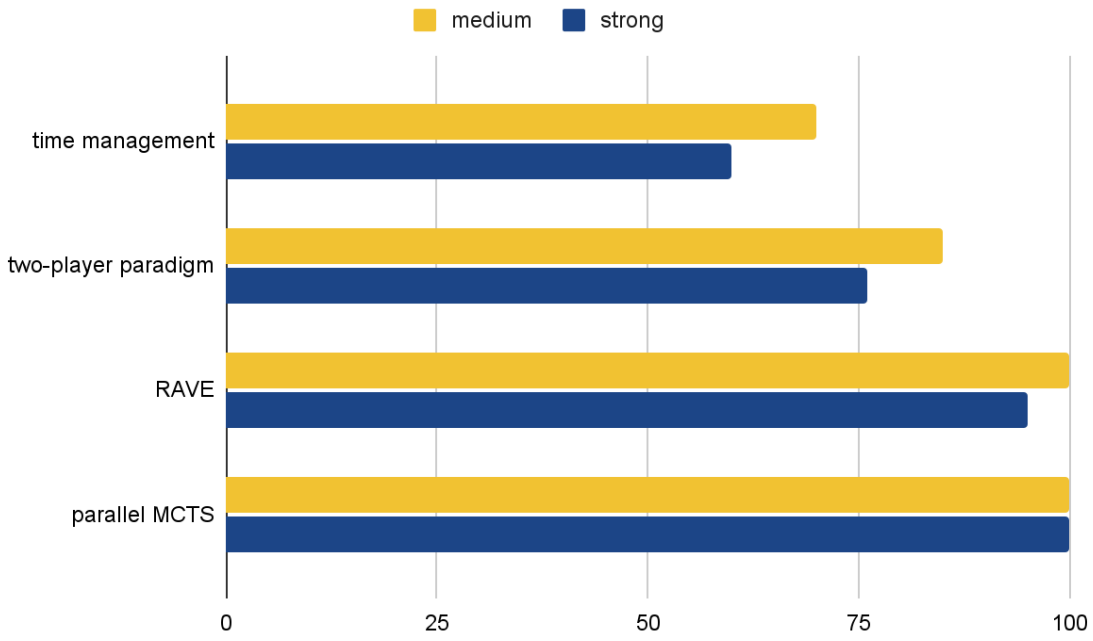
我選擇使用 Root Parallelization，讓每個 thread 都有自己的樹，最後再結合所有的樹，並選擇根 child nodes 裡面造訪次數最多的，作為下一步的動作。Root Parallelization 讓每個 thread 之間資訊有些許流通，也比較易於實作，但缺點是沒有 Tree Parallelization 的資訊流通。

Performance

以下是我實作的順序:

1. time management
2. two-player paradigm
3. RAVE
4. parallel MCTS

下方圖表是依序加入這四項後，對抗 medium/strong baseline 的勝率，可以發現勝率都有持續上升，如果四項都加入後，對抗 strong baseline 也可達勝率 100%。



Difficulty & Solution

一開始在實作這些方法的時候，一直得不到很好的表現，後來發現是上次實作 MCTS 時，有些地方沒考慮到，造成加入這些方法後，依然得不到好表現，後來將這些地方修改完成後，勝率也都提升了。另外，在實作 parallel MCTS 時，遇到了 IA 的問題，找了好久才發現是開多個 thread 的地方弄錯了，後來改成用 omp 就成功修好了。