HW2 Report: Einstein Würfelt Nicht! (Kari) - 6x6 version

R07922018 資工碩二 王柏翔 | Theory of Computer Games (Fall 2019)

Compile 方式

```
make
./target/agentMCTUnevenTotalTrialPrune # 這是主程式
```

Algorithms & heuristics

這次作業指定使用的是蒙地卡羅樹，以及Progressive Pruning 和 RAVE 其中一個，我選擇的是Progressive Pruning。

蒙地卡羅樹

我的作法是從最初的根節點(現在的盤面)中產生出所有的合法走步(最多18種)當作該節點的子節點，並且每一個結點去做數量不一的隨機走步模擬，每次模擬到分勝負或是平手為止。每個節點的模擬數量根據他是不是特殊走步來分。如果他屬於某些特殊走步的話，獲得的模擬次數會是一般走步的兩倍。而在模擬的時候，隨機的步伐也是對特殊走步會有挑選的先後順序。在全部都計算完之後，向上更新UCB以及pruning分數，再從根節點根據UCB分數最高的節點PV路徑走到頁節點，繼續擴展搜尋樹。在九秒之後，會找到現在勝率最佳的第一層子結點的走步並送出到遊戲程式，等待對手做出決定。UCB 中的 c (探索參數) = 1.18。

特殊走步

1. 優先走步：如果我方走了某個步可以吃掉對方的子，不會被反吃，或是反吃得對方旗子數字比我大，列為優先步數。
2. 劣勢走步：如果到了終局時，對方不達陣，我方的達陣步就會被降低優先度。
3. 必贏步：如果到了終局時，對方達陣且比我方即將達陣數字大，優先走步。

隨機模擬時，如果有優先走步會優先走那一邊，否則選擇一般的走步，再來才選擇劣勢走步，最後沒有走步可以選才會選擇"??"(跳過)。

Progressive pruning (PP)

我目前在progressive pruning使用的分數計算方法是

$$\text{PP-score} = k \cdot \frac{1}{\sqrt{s}}$$

其中s是該局進行到遊戲結束時的步數。$k = -1(\text{我方輸}), 0(\text{平手}), 1(\text{我方贏})$。比起一開始嘗試的(1贏, 0平手, -1輸)是沒有normal distribution的情況，這個PP-score是連續分佈，因此效果比起一開始的離散輸贏好很多，因此後來的PP都是使用這個方法。

相關的參數

- 最小\prune所需trial數: 100 trials
- σ: 2
- R_d: 1

Experiment Results & Findings of My Implementation

GameBoard儲存

- 在GameBoard儲存上，由於原始的code中需要以$O(n^2)$的時間複雜度處尋找方塊所在位置太耗時，所以我在cube身上也儲存現在所在的座標，只要有移動就同時更新，可以將搜尋方塊位置得時間複雜度降至$O(1)$。
- GameBoard中的每個對應參數都有儲存相對於所屬節點初始值，只要將盤面回復到初始值就可以繼續跑下一個模擬，減低「複製盤面」所需的時間。

TreeNode

- 在每個Node上面都會記錄一個GameBoard，以方便跑的時候不用再度複製一次board。
- 在每次長樹時，如果其中一個Node走步是屬於優先走步或是必勝步，則他的搜尋次數會是一班的兩倍，而一次搜尋的總次數會是固定數值$T = 4000$，所以假設下一步合法步數是18種，其中有4個是優先步或是必勝步，那麼除了這四個以外的模擬次數就會是$t = \lfloor \frac{T}{18+4} \rfloor = 181$次，而優先步或是必勝步則是$2t = 362$次。這在後期可行步數比較少的時候可以每種步進行較多次的搜尋，提高精確度。

對戰紀錄

- 我的各版本
  - old: agentMCTUnevenTotalTrial 舊版本，沒有進行progress pruning
    - 舊版本的差異在長樹以及隨機走路時只考慮特殊走步的第一點，二和三不考慮。
  - oldPrune: agentMCTUnevenTotalTrialPrune 舊版本，有進行progressive pruning
  - new: agentMCTUnevenTotalTrial 新版本，沒有進行progressive pruning
  - newPrune: agentMCTUnevenTotalTrialPrune: 新版本，有進行progressive pruning

#	我的版本	對手	共	贏	輸	平手	勝率
1	oldPrune	greedy	50	38	4	8	76%
2	oldPrune	conservative	50	46	2	2	92%
3	newPrune	greedy	50	36	7	7	72%
4	newPrune	conservative	50	48	1	1	96%
5	newPrune	oldPrune	50	30	18	2	60%
6	new	greedy	50	41	3	6	82%
7	new	conservative	50	49	1	0	98%
8	newPrune	old	50	25	24	1	50%
9	new	oldPrune	50	37	10	3	74%
10	newPrune	new	120	23	92	5	19%

- 從測試中可以看出
  - 新版的勝率優過於舊版，代表增加的特殊走步有助於程式對於模擬走步的正確率。
  - 有prune的狀況，雖然依然能夠打過greedy和conservative，但是比較沒有prune的情況，會發現是趨於劣勢的。我猜測這是由於用來prune的計算分數依然不太適合，或是σ和R_d沒有取的很適當的關係。