TCG HW2 Report

B02902071 陳柏堯

1. How to run?
   a. Compile the main search program

      $make

      ```
      b02902071@linux2 [~/Course/TCG/TCG_hw2/othello_ver20161123] make
      g++ search_uct.cpp -std=c++11 -O2 -Wall -o b02902071
      ```

   b. Run the program

      $./b02902071

      ```
      b02902071@linux2 [~/Course/TCG/TCG_hw2/othello_ver20161123] ./b02902071
      genmove
      ```

   c. Compile all the search programs [search(random), search_ucb, search_uct, search_ucts, search_progressivepruning]

      $make search

      ```
      b02902071@linux2 [~/Course/TCG/TCG_hw2/othello_ver20161123] make search
      g++ search.cpp -std=c++11 -O2 -Wall -o search
      g++ search_ucb.cpp -std=c++11 -O2 -Wall -o search_ucb
      g++ search_uct.cpp -std=c++11 -O2 -Wall -o search_uct
      g++ search_progressivepruning.cpp -std=c++11 -O2 -Wall -o search_progressivepruning
      g++ search_ucts.cpp -std=c++11 -O2 -Wall -o search_ucts
      ```

   d. Compile judge & UI

      $make judge

      $make ui

      ```
      b02902071@linux2 [~/Course/TCG/TCG_hw2/othello_ver20161123] make judge
      g++ -std=c++11 judge.cpp -o judge
      b02902071@linux2 [~/Course/TCG/TCG_hw2/othello_ver20161123] make ui
      g++ -std=c++11 simple_http_UI.cpp -o simple_http_UI
      ```

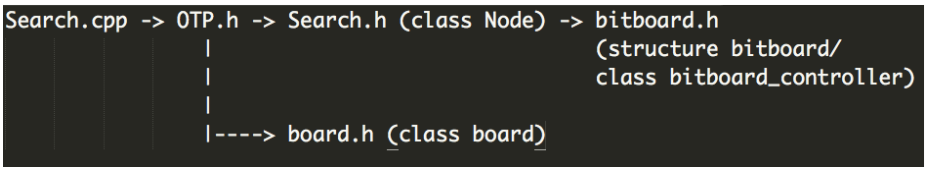
   e. Clean binary files

      $make clean

      ```
      b02902071@linux2 [~/Course/TCG/TCG_hw2/othello_ver20161123] make clean
      rm -f ./simple_http_UI || true
      rm -f ./judge || true
      rm -f ./search || true
      rm -f ./search_ucb || true
      rm -f ./search_uct || true
      rm -f ./search_progressivepruning || true
      rm -f ./search_ucts || true
      rm -f ./b02902071 || true
      ```

2. Implementation

   ＊ Files Structure



a. UCB

location: SearchNode.h (Line: 336)

按照投影片實作

b. UCT

Location: SearchNode.h (Line: 294, 137)

按照投影片的 select, expansion, simulation, backpropagation 實作

每一個 Node 都會創建自己的 Child Node 並且向下搜尋，選擇 Children 中對自己 win rate 最高的 child（反過來說，對於那個 child win rate 最低）

c. Progressive pruning (pp)

Location: SearchNode.h (Line: 198, 250)

實作了兩個版本的 progressive pruning：

・Version 1 (void doProgressivePruning_Bonomial())

每一個 node 的所有 child 作 simulation，其平均值就是勝率，也就是 WinNum/SimulationNum，標準差公式是 sqrt(p(1-p)), 先找所有 child 中最大的 MI，再去一個一個和 MI 比較。比較差的 child 被砍掉。

發現此版本幾乎砍不到枝，因此實作第 2 版本

・Version 2 (void doProgressivePruning())

每一個 node 的所有 child 的 child 作 Simulation，則這些 child 的平均值是該 child 的所有 child 的勝率平均值，標準差是該 child 的所有 child 的勝率的標準差。若某一個 child 的所有 child 的勝率都很低，則會比其他該 node 的 child 還要糟糕，基於這點來砍。

d. UCT with saving root (OTP.h Line:125) (UCTS)

基於已實作的 UCT 版本，在每次 genmove 之後存下選擇的那個 move 的分樹(root)，等到下次 genmove 的時候就去看之前已經算過的樹的 node，砍掉沒有走過去的分數，留下上一次 genmove 搜索的樹，繼續往下搜索。

發現該實作方法並沒有明顯增加強度，因此暫時沒有採用。

3. Technique

Bitboard

保留助教的 Class board，在進入 genmove 之後，將傳入的 board 轉換成 bitboard，對 bitboard 創建 update，random_pick_move, get_valid_moves 等重要方法。

測試之後，在開局時 1 秒可以 simulate 9 萬次左右。

Ref:

http://www.gamedev.net/topic/646988-generating-moves-in-reversi/

http://stackoverflow.com/questions/2001597/how-do-you-randomly-zero-a-bit-in-an-integer

4. Coefficients

UCB c choose 2.718 比較發現理論值 2.718 已經足夠了

NAME:b02902071(c=2.0) QUIT:1 WIN:3 LOSE:7 DRAW:0

NAME:b02902071(c=2.718) QUIT:1 WIN:7 LOSE:3 DRAW:0

NAME:b02902071(c=2.0) QUIT:1 WIN:3 LOSE:6 DRAW:1

NAME:b02902071(c=2.718) QUIT:1 WIN:6 LOSE:3 DRAW:1

NAME:b02902071(c=2.0) QUIT:1 WIN:4 LOSE:6 DRAW:0

NAME:b02902071(c=2.718) QUIT:1 WIN:6 LOSE:4 DRAW:0

NAME:b02902071(c=2.0) QUIT:1 WIN:6 LOSE:4 DRAW:0

NAME:b02902071(c=2.718) QUIT:1 WIN:4 LOSE:6 DRAW:0

Progressive Prunning, Rd = 2; PruningSigma = 0.2; 使用投影片表格中的範圍預設值。

NAME:b02902071(PP Sigma=0.2, Rd=2) QUIT:1 WIN:3 LOSE:6 DRAW:1

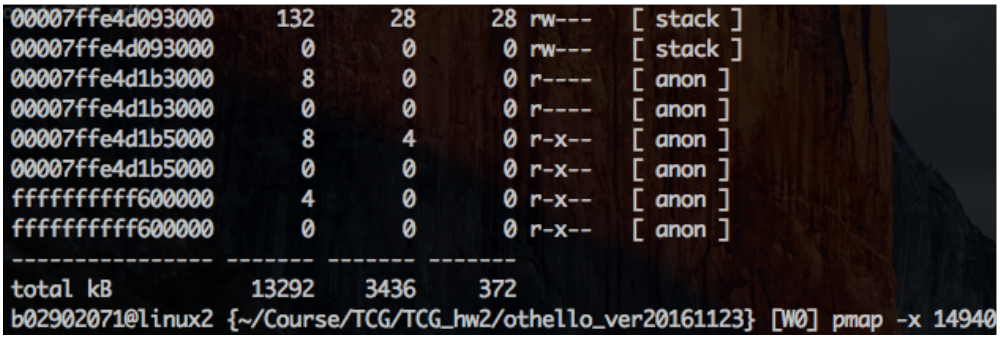
NAME:b02902071(UCT) QUIT:1 WIN:6 LOSE:3 DRAW:1

5. Memory Usage & Disk Usage

Memory

開局時 10 秒內會生成約 1000 node(不到)，一個 node 內有約 15 個 double，2 個 int，1 個 bool，1 個 short，2 個 unsigned long long, 約 147,000Bytes ≈ 150KB

再加上每個 node 內部臨時所建的參數、function 等，大約 300KB.



實際在開局時用 pmap 去檢查，大概使用了 13M 左右的記憶體。

Disk Usage

None, 沒有讀寫 disk。

6. Comparison

UCB VS template 7122

NAME:UCB QUIT:1 WIN:60 LOSE:0 DRAW:0

NAME:template7122 QUIT:1 WIN:0 LOSE:60 DRAW:0

UCT VS template 7122

NAME:UCT QUIT:1 WIN:60 LOSE:0 DRAW:0

NAME:template7122 QUIT:1 WIN:0 LOSE:60 DRAW:0

UCT VS UCB

NAME:UCT QUIT:1 WIN:54 LOSE:3 DRAW:3

NAME:UCB QUIT:1 WIN:3 LOSE:54 DRAW:3

UCTPP (Progress pruning) vs UCT

NAME:UCTPP QUIT:1 WIN:26 LOSE:28 DRAW:6

NAME:UCT QUIT:1 WIN:28 LOSE:26 DRAW:6