TCG(電腦對局理論) Final Project Report

B02902071 陳柏堯

1. How to run?

- Environment:
  Docker Virtual Machine: Ubuntu14.04

- Makefile: . /code
  $make // make search file
  $make clean // clean search file

- Put “search” under the CreateRoom, EnterRoom ‘s “Search” folder

2. Implementation

- NagaScout
  “search.h” Line 209~277
  利用 alpha 和 beta 來做 null window search, 之後回傳發現有比 m 更大的值得則 re-search, 若 fail-high 則可以砍除下面之後的子樹而直接 return。
  iteravite 一層層下去 evaluate，直到時間結束為止。

- HashTable
  “hashtable.h”
  由於考量 hashtable 製作的大小和方便性，最後選擇用 array 來開，因此 n 取 16, 也就是 entry 大小為 1<<16 大小 65536，然後 n+m 取 128bit 的 hashval 用 bitset 表示，這樣 entry 的 index 就是用 128bit 的 hashkey 來 mod (1<<n)。
  Entry 裡面存有原始的 hashval 用於比對，从而防止 collision 而程式錯亂。

- Bitboard
  “bitboard.h”, “bitboard.hh”
  按照論文 DarkCraft[1]所述，利用位來表示棋盤的每一個子，並且可以用位運算的方式實作 movegen（get_valid_moves() line:146），doMove (update() line:469)等功能。雖然在 movegen 上表現良好，可是連 evaluation 都用位運算的內容來計算，效率的優勢反而消失了，感覺這次沒有把 bitboard 的速度優勢發揮出來。

- Evaluation Function

  (a) Dynamic Piece Value

  “bitboard.h” Line:289~338

  按照論文 Observer[2]所描述，當兵/卒 和 將/帥 同時在場上的時候，就動態地去計算子的價值。而每一個子的價值取決於他的天敵數有多少，這樣就可以讓一些中等價值的子為了保護 將/帥 而犧牲自己去清楚 兵/卒。

  例如 士象車馬兵的分數就根據對應的天敵數來調整分值：

  Score[該子天敵的數目]＝{11000, 4900, 2400, 1100, 500, 240, 110, 50, 20}

  (b) Position Value

  “bitboard.h” Line:375~430

  為了在終盤的時候，不至於和局，也就是最後怕子與子距離太遠導致不吃敵方子，所以加上距離的函數，每個子會根據可以吃的對方子進行加分。加分的方式為曼哈頓距離，後來在競賽中發現曼哈頓距離可能會在最後出現追殺卻一直吃不到，最後循環盤面結束的問題。

  這邊應該要改成歐幾里得距離，應該可以改善的。

- Quiescent search

  ’search.h‘ line:219

  到了 d=0 的深度的時候，若發現下一步還有吃子步，就保留這些吃子步繼續往下搜，直到沒有吃子步為止。

3. Heuristic

- 翻子策略

  走步的搜索我將之分為 move 的搜索（就是移動一個子） 和 flip 的搜索（就是把一個子吃掉另一個子）

  在 move 搜索的 negascout 中找到最好的一步(bestmove)，如果是吃子步，我就直接回傳吃子；如果是移動步，我就再去搜索 flip 的搜索。如果在 flip 的搜索中，找到正分（也就是這麼翻是有好處的）則就 flip 那個子。

  Flip 的搜索方式則是算每一個位置，每一個子翻出的機率乘上翻出它之後搜索一定深度回傳的結果(算期望值)，然後找最好的那個位置回傳回去。

這麼設計的原因是，吃子往往對自己是有利的，而子移動可能不能立即反應好處，還可能是循環前兆，所以此時就去看翻子會不會更好。

- 如果對方剩一子，並且在我某一子的旁邊我能吃它就吃它
  為了防止搜索的過程中不吃子反而繞圈圈。（骯髒 debug）

- 第一層從上一次分數大的 branch 先搜
  Iterative deepening 中，為了讓 negascout 的 βcut 更多，所以在第一層對上一輪的結果做 sorting，由分值大的 branch 先搜。

4. Review

比賽最後拿到總成績 8.5，然而過程中發現程式內部問題很多，如下：

- 審局函數太看重位置，導致大子一直在中間遊走，沒有分力分別去吃角落的子。

- 有一局沒有控制好時間，結果在剩下只需要 1，2 步就能贏的情況下超時輸了，後來就加上一條 if(剩下 100 秒)減少思考時間，就避免了超時。

- bitboard 的層數搜索太慢了，原因一方面是 hashtable 開太小了，所以容易會 conflict 到，另一方面是審局函數中 bitboard 還是要一個一個數 bit 來判斷子力，效率不是很好。

- 翻子策略不夠好，有時會越翻越糟糕。

Reference:

[1] 暗棋程式 DarkCraft 設計與實作 施 2012
[2] 電腦暗棋程式 observer 徐 2014
[3] 電腦暗棋程式 Darkness 的設計與實作 詹 2013