

**Group 29**

**Member**

**0613411 陳詠恩**

**0613413 蔡怡君**

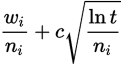
INTRODUCTION TO AI - FINAL GROUP PROJECT

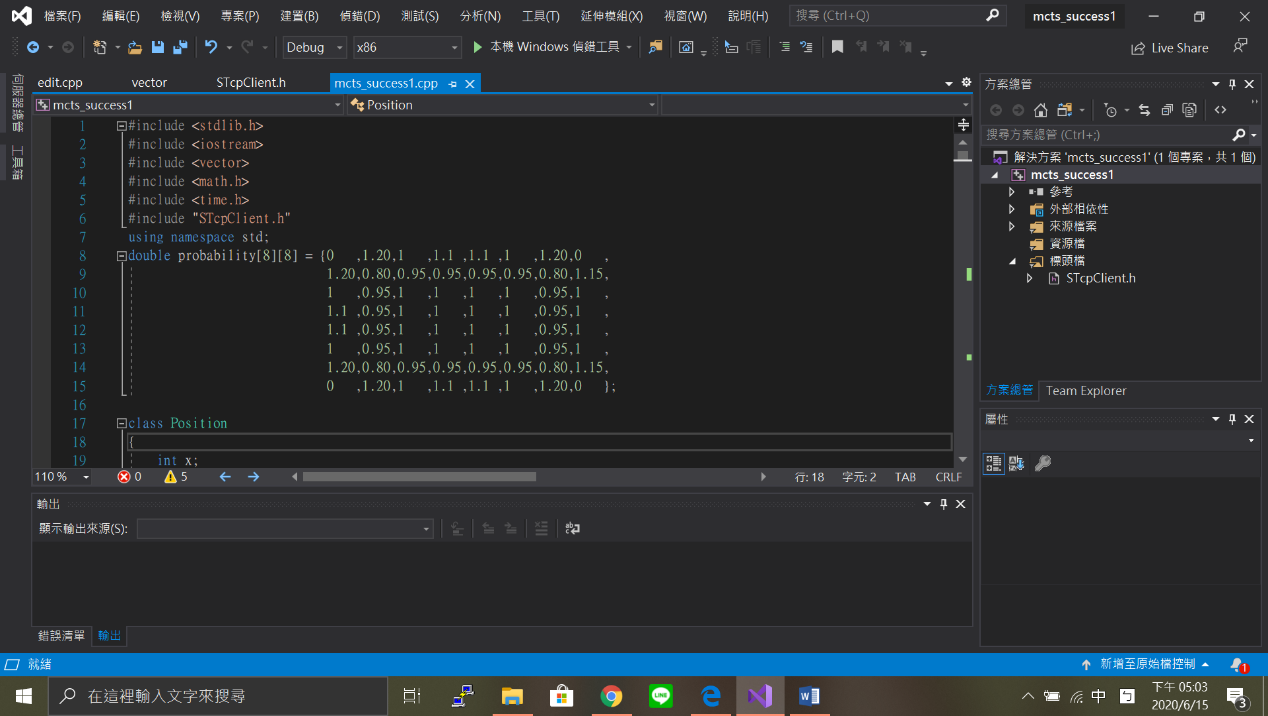
研究方法

利用Monte Carlo tree search ，加上Policy及UCT（Upper Confidence Bounds to Trees）公式，設計一個game-playing agent，並比較參數及policy以獲得最佳勝率。

1. 程式設計
   1. Class & function

|  |  |  |
| --- | --- | --- |
| Class name | variable | function |
| Position | int x;  int y; | Getters and Setters |
| Board | int boardValues[8][8];  int totalMoves; | Getters and Setters  void Print board()  void performMove()  int checkStatus()//檢查遊戲是否結束並算分  vector<Position> getPossiblePositions()  //尋找現在可以下的位置 |
| State | Board board;  int playerNo;// 1 or 2  Position move;// 動了哪一步才到目前的狀態 | Getters and Setters  int getStateOpponent()  vector<State> getAllPossibleState()  State randomPlay()//for simulation step |
| Node | State state;  Node\* parent;  vector<Node> childArray;  int visitCount;  double winScore;  bool root;//is root node | Getters and Setters  Node\* getRandomChildNode()  double getChildScore(int i)  Node\* getChildWithMaxScore() |
| Tree | Node root; | Getters and Setters |
| UCT |  | static double uctValue(int parentVisit, double nodeWinScore, int nodeVisit)  static Node\* findBestNodeWithUCT(Node\* node,int level) |
| MonteCarlo  TreeSearch | double WIN\_SCORE;  //贏了一次加幾分  int opponent;//對手  double t;//time count  int level;  //目前棋盤上的剩餘空格 | Node\* selectPromisingNode(Node\* rootNode,int level)  void expandNode(Node\* node)  int simulateRandomPlayout(Node\* node)  void backPropogation(Node\* nodeToExplore, bool playerwin)  Position findNextMove(Board board, int playerNo,int levels) |

* 1. UCT
* wi = number of wins after the i-th move
* ni = number of simulations after the i-th move
* c = exploration parameter
* t = total number of simulations for the parent node
  1. Policy
     1. Action probabilities weight



* + 1. Heuristic function：行動力分析(對手可以下的位置)

1. 實驗方法

(由最陽春的MCTS Agent與加上Policy 及更改UCT係數後的Agent作對手)

* 1. 操縱變因
     1. UCT的係數：更改exploration parameter觀察結果(預設為√2)
     2. Policy：
        1. Action probabilities weight
        2. Heuristic function
  2. 應變變因：
     1. 勝率

結果分析

Uct最原始係數：1.2

Uct最原始係數：1.4

Uct最原始係數：1.6

Action probabilities weight

新(1.2)/舊(1.15)

1/1

0/2

1/1

1/1

0/2

未來展望：

對原本的下棋規則來說，下棋的方式，不是取決於這次能夠吃多少旗子，而是下完了這次能夠爭取到更多可下的步數(行動力)？但如今的下棋規則改了，6\*6裡頭的都是可下步數，而在實驗方面，還是可以去試試看是否增加可下步數 ( 在角落、邊上 ) ，同時在simulation的時候都以**能夠增加自己的可下步數、阻擋對手的可下步數。**

－主要採取控制中心策略：上網查到减少對手行動力（尤其是没有大大减少自己行動力）的方法是避免翻轉過多的的邊緣子。擁有內部子（即不與空位相鄰的棋子）比擁有邊緣子更好，

在最後剩下五步直接使用MIN-MAX，因為時間足夠跑完所有的可能性，可以高度提升最後五步的獲勝率。