Chinese Checker AI

109550019王培丞 109704011楊璨聰 109550147許竣凱
Github: https://github.com/alexyang0826/Chinese-checkers-AI.git

Abstract

因為有組員小時候由於沒有人陪他玩跳棋感到難過，所以想透過這學期學習到的AI相關知識做出會下跳棋的AI，讓他不再感到孤單，而在組員間討論的過程中，我們認為不同演算法間的效率及正確綠的差距也是一個十分值得探討的議題，因此最終決定以不同演算法及人數間跳棋AI的勝率作為此次報告的主題。

1.introduction

此次期末報告的目標是想要做出能夠合理進行跳棋遊戲的AI，並以不同的演算法和人數作為比較的依據，判斷哪種演算法在何種情境下勝率更高，以及觀察及探討不同狀況下AI間的互動以及影響。

2.environment

此次project以python3.9製作，由於程式需求額外安裝了pygame以及numpy兩個語言庫。

3.methodology

首先，製作一個能夠執行跳棋環境的棋盤，為此我們使用了pygame的功能，並將棋盤以fig1的方式做標號以及呈現，類似於二維迪卡爾座標。值得一提的是，為了解決此標號方式造成的縱軸跟橫軸間距離不對等的問題，會將y軸(橫軸)縮小。製作完棋盤後，藉由先前訂定的座標將每位AI的旗子放置於各自的起點上，並設定每一位AI分別的終點以及不能經過的格子。

其次，為AI設定移動方式的演算法（分別為alphabeta以及greedy）以及相對應的evaluation function（註1），並且每一輪的移動方式均透過當前輪次所執行的move的score來決定順序。實際執行狀況如Fig2所示。

最後，在AI的比較方面，則透過計算每一輪哪位AI最先將全部旗子抵達終點來獲取分數，並透過大量測試得出一個較為精準的比例，藉此得出何種演算法在何種情境的精確度較高。

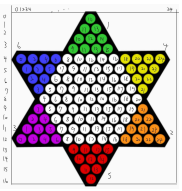


Fig1

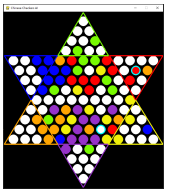


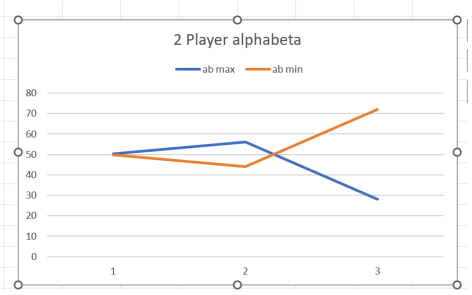
Fig2

註1：
evaluation：在greedy下，對於最遠距離的棋子到目標的距離給不同的weight，測試表現。在alphabeta下，更改不同的depth，同時也改不同的role(MAX/min)來看結果。最終，挑選出最好的algorithm去進行遊戲

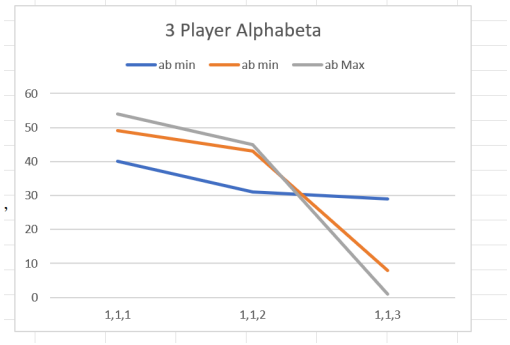
4.experiments

greedy w=0.5	abm d1	p1	p2
0.5	0.5	44	56
greedy w=0.5	abM d1	p1	p2
0.5	0.5	50	50

alphabeta min 贏greedy

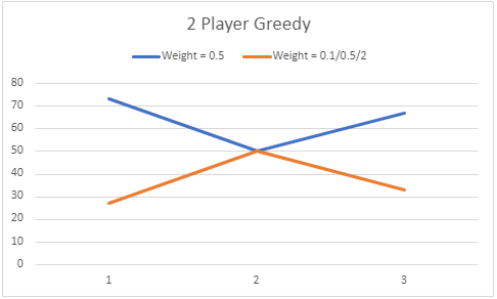


min 贏 max (2人情況下)



,

min 略贏max



weight = 0.5 表現最好(兩人)

greedy1 w	greedy2 w	greedy3 w	p1	p2	p3
0.5	0.5	0.5	22	18	10
0.5	1	1.5	28	16	6

weight = 0.5 表現最好 (三人)

綜合上表可得知

greedy方面，weight 0.5 表現最好且不隨人數改變。但表現上不比minimax。

minimax方面，兩人情況下min較有優勢，但隨著人數增加，由於min能針對的力度會因為人數增加而減少，max會逐漸取得優勢，甚至會反超min。此外，在人少的狀況下depth越高會因為預測過多導致精準度下降。

5.conclusion

由實驗結果可得知在人數較少時，跳棋偏向讓對手優勢最小的遊戲，但隨著人數增加，所需考量的因素變多後，讓自己優勢大才是重點，由此可知不只是演算法的選擇，人數的多寡也是決定最優選項的重要因素。

6.Future work

可以進一步使用DQN實作，並加入能讓玩家實際與電腦對戰的功能。

7.contribution of each member

許竣凱：evalution function 、AI行為邏輯

楊璨聰：操作人性化、自動化、code

王培丞：棋盤製作、AI的初始設定

8.Reference

Github

https://github.com/AndreaVidali/ChineseChekersAI/blob/master/gui.py