國立台北大學資訊工程學系專題報告

5 五將棋 AI 電腦對局

Minishogi on AI Game Theory Approach

專題組員:王暄喻、陳柏勳、袁浩、張敬謙

專題編號:PRJ-NTPUCSIE-106-010
執行期間:106 年 08 月 至 107 年 07 月

一. 摘要

研究電腦如何下棋、解謎題和進行對局是資訊科學中研究人工智慧的重要一支。繼 AlphaGo 和 AlphaGo Zero 之後,除了繼續研發高棋力的對局程式之外,電腦對局領域也將朝向研究吸引人們學習對弈的方向前進。

為此我們對電腦對局深受吸引,且致力想學習人工智慧基礎知識和對局實作技術甚至精進對局演算法,為此我們挑選了充滿挑戰性的 5 五將棋作為研究電腦對局的媒介。

5 五將棋(Mini-shogi)是本將棋(Japan Shogi)的一種變形,棋盤雖小,但由於打入(drop)和升變(promote)的特殊規則,遊戲複雜度高達 10^{90}。5 五將棋於 2007 年首次在日本舉辦電腦對局比賽,目前已列入國際奧林匹亞賽局競賽的正式項目。由於 5 五將棋著重於中局攻殺且沒有殘局,如何以較少著手獲勝成為關鍵。

二. 簡介

1. 專題之研製背景

相較於其他著名電腦對局遊戲,如象棋、西洋棋、黑白棋等棋類遊戲,電腦 5 五將棋的發展歷史較短。

雖然 5 五將棋的棋盤較小且棋子數較少,但繼承來自日本將棋的規則,比起其他棋類多了打入、升變等玩法與限制,使得其遊戲搜尋樹的分支與複雜度亦相當可觀。因此,如何設計一支具有一定棋力的程式,就成了一個很有趣的議題。

2. 專題研究目標

本次專題研究目標主要以設計和製作一個具有強大棋力的 5 五將棋人工智慧程式,我們把程式命名為 Nyanpass。在專題期間內,我們極力研究和專研精進其程式的棋盤分析能力和運算效能,包括判斷時間和該程式的勝率最大化。

為了看到研究目標較具體的成果我們為自己的研究設定了有兩個鑑定指標分別是戰勝現今網路上面的 5 五將棋 AI 程式和參賽 2018 年 7 月北大主辦的 International Computer Games Association(ICGA)國際電腦對局 5 五將棋項目的賽事,以更了解我們程式的棋力程度。

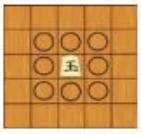
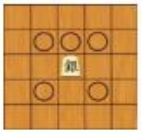
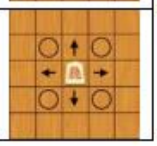
3. 5 五將棋遊戲介紹

棋盤大小為 5x5，所使用的棋子兵種為王將（玉將）、飛車、 角行、金將、銀將和步兵。初始棋局雙方各有 6 顆 棋子（如圖 1），依每個兵種的規則進行遊戲，直到有一方將死對方的王將（玉將）即可獲得勝利。

圖 1 迷你將棋起始盤面

5 五將棋的規則源自於日本將棋，每個兵種的棋子都有其走法規則，當有對方棋子落在己方棋子的可行步範圍裡，則可於俘虜對方棋子（吃子），俘虜的棋子可於下一手後進行打入至棋盤中，當己方棋子前進至對方陣營底線時，可選擇升變為另一兵種，走法與原兵種不同。5 五將棋的棋子走法如下表 1：

表 1 迷你將棋走子規則表

王將（玉將）：遊戲中最重要的棋子，一旦俘虜對手之『王將』，則遊戲結束，並獲得最終的勝利，『王將』的走法規則為每次可行走相鄰八方向一步。	
金將：『金將』的走法為前面三格，左右兩格以及正後方一格。	
銀將：『銀將』的走法為前面三格，左後以及右後，當移動至敵方底線時，可以升級為『全』，走法與金將相同，亦可選擇不升級。	
角行：『角行』的走法為斜前方或斜後方，若沒有其他棋子阻擾，可以不限距離的斜行，當移動至敵方底線時，可以升級為『龍馬』。	
飛車：『飛車』的走法為前後左右方向，若沒有其他棋子阻擾，可以不限距離移動，當移動至敵方底線時，可以升級為『龍王』。	
步兵：每次只能往前一格，移動至對方底線時，升級為『と』，走法與金將相同。	
龍馬：當『角行』移動至對方底線時，可升級為『龍馬』，除了原有走法外，更增加了前後左右方向可移動一格。	
龍王：當『飛車』移動至對方底線時，可升級為『龍王』，除了原有走法外，更增加了斜角方向可移動一格。	

4. 5 五將棋之特殊規則

(1)打入

當持子方，將俘虜的棋子重新投入至棋盤中， 稱為『打入』，打入時須以未升變前的兵種狀態打入，基本上打入可以放到任何位置，但有幾種情況是無法打入的，若違反將直接導致被判敗：

a. 棋盤上有棋子的位置無法打入。

b. 不能將步兵打入至對方陣營底線。

c. 同一行上已有己方步兵時，不能將步兵打入同一行，若打入則稱為『兩步』。

d. 不可將步兵打入至對方王將前方，且造成 對方王將無法移動的情況，若打入則稱為 『打步詰』。

(2) 升變

棋子進入、離開對方陣營底線，或是在對方陣營底線移動時，可選擇是否升變。直接打入對方陣營底線的棋子，則需等到下一次移動時，才能選擇是否升變。步兵進入對方陣營底線時，則強制升變。

(3) 千日手

對局雙方一直僵持不下，導致同一走法持續循環出現，稱為千日手，

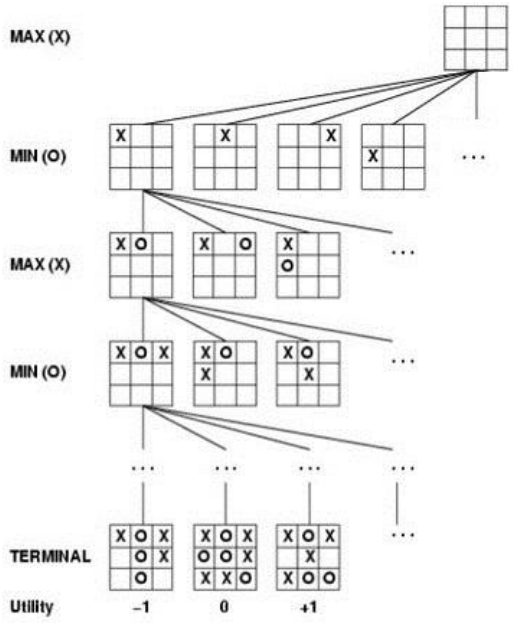
連續循環四次時，先手會被判成敗方。

(4) 千日王手

千日手加上連續王手時(長將)，攻擊方必須改變走法，不然攻擊方將會被判成敗方。

三. 專題進行方式

基於5五將棋是雙人完全資訊零和遊戲這個特性，我們利用我方走一步、敵方走一步的概念建立搜尋樹(圖2)，用以找出走向勝利的主要路徑，做為專題初步的研究方向。由於在5五將棋上，平均一個盤面就有平均35種移動和打入的方式，平均一局要40手才會結束，這搜尋的數量不是現在電腦可以負荷的。



(圖2) 棋局搜尋樹，讓雙方輪流移動，每次選擇對自己最有利的移動

那麼，我們退一步，不全部搜尋，搜尋固定層數，以評估函式來評價一個盤面的好壞，找出底層中分數最高的盤面，向上延伸，每次只選擇對移動方最有利的方式，這種搜尋樹可以用兩種方式提升棋力，一是評估的準確度，二是搜尋的深度，前者需要對遊戲有很深的了解或是用強者們的棋譜分析，我們都是5五將棋的新手，而且又非常冷門，無法找到有參考性的棋譜，所以我們把研究重心擺在增加深度上面。

在計算資源固定下，要增加深度就只能減少寬度，而且不影響搜尋結果，所以我們就採用了經典的對局演算法Alpha-beta剪枝為基礎，再逐步往上增設演算法以增加樹的深度和搜尋效能提升勝率。

目前Nyanpass 程式裡所使用的演算法,我們在這裡做個簡單介紹：

1. Alpha-beta剪枝 (Alpha-beta Pruning)

Alpha-beta剪枝是一種搜尋演算法，用以減少極小化極大演算法(Minimax演算法)搜尋樹的節點數。這是一種對抗性搜尋演算法，主要應用於機器遊玩的雙人遊戲並且把所有棋步的可能性都展開來。當演算法評估出某策略的後續走法比之前策略的還差時，就會停止計算該策略的後續發展。該演算法和極小化極大演算法所得結論相同，但剪去了不影響最終決定的分枝。

2. 寧靜搜尋 (Quiescence Search)

在搜尋固定深度下，可能發生自己吃了很多棋而自以為優勢，結果下一步就被將死，所以當葉節點為非寧靜盤面(下一步可以吃子或將軍的盤面)時，適度的向下搜尋，直到進入寧靜盤面。

3. 同型表（Transposition Table）

利用 dynamic programing 的概念，將已經搜尋過的盤面儲存在一個 hash table，之後如果在其他分支遇到該盤面，就能使用先前搜尋的數據。

4. 著手排序（Move-ordering）

簡單來講就是利用經驗法則。考慮如果先前走過的好步，可能也會是下一次的好步而優先搜尋，透過這樣的預測降低搜尋樹的分支。

5. 主要變化路徑搜尋（Principal Variation Search）

PVS 搜尋演算法是 alpha-beta search 改良後的一種演算法，利用 zero window 的想法，加速 alpha beta 切捨的效率。PVS 會先從根節點開始，從最左邊展開節點到深度至 N，並將其回傳值假設為介於 alpha 與 beta 間的最佳節點（PV 節點），因此接下去 的步法將是最佳的，於是利用寬度為 1 的邊界(alpha, alpha+1)去搜尋驗證，若合理則接下來的搜尋的回傳值將不會大於 alpha，若驗證失敗，就必須花費額外的時間以原本的 beta 重搜，並再使用同樣方式去搜尋驗證。

6. 逐層加深（Iterative Deepening）

目的是用在著手排序，再來是在限制時間下可以保證得到答案。

表 2	With Iterative Deepening	Without Iterative Deepening
深度	8	8
搜尋數	395	397
搜尋 node 數	3039734	5536213
重搜 node 數	1052800	1559360
寧靜 node 數	19942480	83525687
重搜比例(%)	34.63461	28.16655
平均分枝	5.64	4.86
平均時間	7.99	35.77

(表 2)比較逐層加深的效果

7. 期望窗格（Aspiration Window）

假設 d 層搜尋的結果會跟 d-1 層的結果極為相近 在搜尋 d 層時使用的 alpha=d-1 層的 value-常數 beta=d-1 層的 value+常數 目的為降低初始 alpha beta 的 window size。

表 3	With Aspiration Window	Without Aspiration Window
深度	8	8
搜尋數	406	406
搜尋 node 數	3975154	3678444
重搜 node 數	2875478	1225270
寧靜 node 數	20023340	28753031
重搜比例(%)	72.336267	33.309465
平均分枝	4.13	5.38
平均時間	7.9	11.68

(表 3)比較期望窗格的效果

四. 演算法的相互輔助

上述的演算法都是需要一起使用，才能發揮其最大效果。

1. PVS 若在完美的著手排序下，可以大幅降低搜尋量。

表 3	加入 PVS、著手排序、期望窗格	未使用
深度	8	8
搜尋數	421	424
搜尋 node 數	3321361	10111755
重搜 node 數	2416794	0

寧靜 node 數	15362758	107631681
重搜比例(%)	72.76517	0
平均分枝	4.54	5.92
平均時間	6.87	49.7

(表 4)比較在都使用逐層加深和同形表，加入 PVS、著手排序、期望窗格的差異

2. 在使用同形表與著手排序下，才能幫助在逐層加深搜尋不同深度時，使用之前得到的結果來加快速度。

表 5	加入同形表、著手排序	未使用
深度	6	6
搜尋數	363	362
搜尋 node 數	334103	424450
重搜 node 數	264547	345878
寧靜 node 數	3718444	9458403
重搜比例(%)	79.18127	81.48851
平均分枝	5.86	5.66
平均時間	1.78	3.27

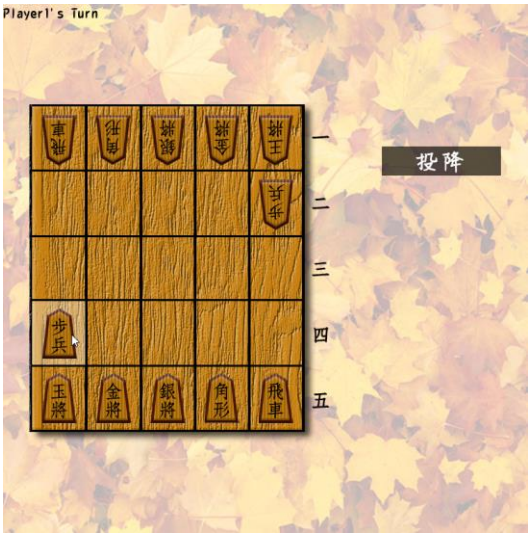
(表 5)比較在都使用逐層加深、期望窗格和 PVS，加入同形表和著手排序的差異

五. 系統介面

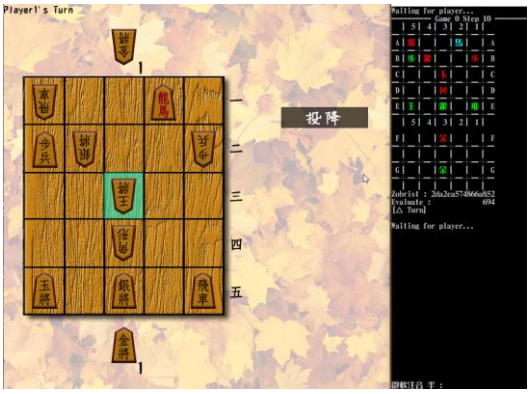
本專題使用支援 C++語言之 SFML 函式庫(Simple and Fast Multimedia Library)為基礎撰寫程式介面，以便呈現與方便操作。

介面為獨立之程式，其與主程式以 Shared memory 方法進行資料傳輸，並以字串處理主程式傳送之訊息，將主程式內容以多媒體畫面呈現出來。

介面提供使用者以滑鼠點按棋子，在主程式之搭配下配合規則能標示可以點按之棋子、顯示該棋子可移動之範圍(綠色方格)、最新一步移動到之位置(淺藍色方格)，以及可要求顯示 AI 位於某盤面下之主要移動路徑(PV)。



(圖 3)介面以及初始盤面



(圖 4)已連接之主程式與介面

(圖 5)以畫面詢問是否升變

六. 結語與展望

對於這次專題的作品我們覺得還有許多發展的空間與棋力進步的空間。直到 ICGA 比賽前都還會極力去做增強棋力方法的研究和學習，希望可以把這程式的能力更往上推一層樓。

七. 銘謝

特別感謝指導教授在各方面的指導和意見甚至寶貴的學習資源分享，才能讓我們有今天的成果與作品，沒有指導教授此專題的進度絕對沒有現在的成果與棋力效能。

另外，也要感謝各位學長姐與同學們給的建設性意見，才能讓我們有多方的參考方向，不斷的改進專題的程式與呈現方式。

八. 參考文獻

[1] 陳志昌,「電腦象棋開局知識庫系統之設計與製作」,碩士論文，國立台灣大學，1998。

[2] 圖 2，
https://pt.slideshare.net/melvinzhang/cg2006

[3] 「技巧ミニ」アピール文書
東京大学大学院法学政治学研究科，出村洋介

[4]5五将棋における評価関数の自動学習，柿木 義一

[5]5 五將棋之打步詰,Uchifuzume in Minishogi, 阮柏鈞

[6] 徐贊昇,「電腦對局理論」, ISBN：9789863502371

[7] Mastering the game of Go without human knowledge, Nature 24270

[8]
https://www.jair.org/index.php/jair

[9]
www.iis.sinica.edu.tw/~tshsu/tcg/2017/index.html

[10] Simple and Fast Multimedia Library，
https://www.sfml-dev.org/