人工智慧期中報告

5 五將棋對局

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 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。

5 五將棋(Minishogi)是本將棋 (Japan Shogi)的一種變型，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 1090。

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。

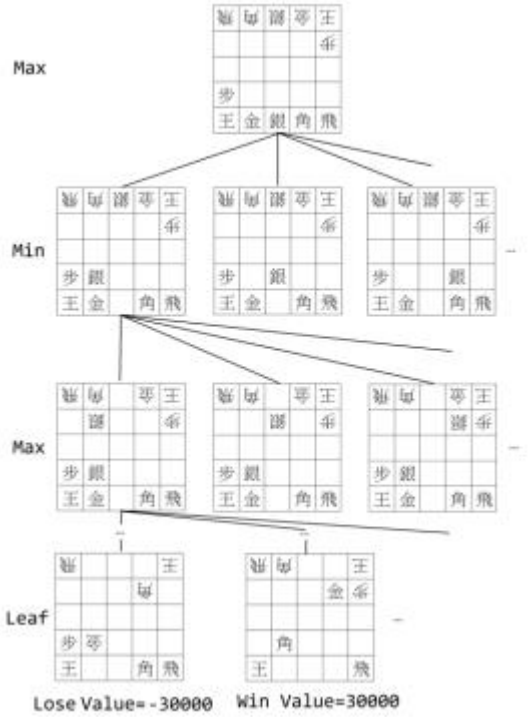
二. 簡介

5 五將棋遊戲介紹棋盤大小為 5×5，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，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。

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。

三. 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，每次選擇對自己最有利的移動， 用以找出走向勝利的主要路徑，做為初步的方向。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。



那麼，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，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。

演算法簡介

1.Alpha-beta 剪枝

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

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

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. 主要變化路徑搜尋

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 α 與 β 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 α，若驗證失敗，就必須花費額外的 時間以原本的 β 重新搜尋，並再使用同樣方式去搜尋驗證。

5. 著手排序

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。圖表是在雙方都使用逐層加深、期望窗格、PVS 和同型表的演算法，比較著手排序的 差異。

	使用著手排序	不使用著手排序
深度	8	8
搜尋次數	457	455
搜尋節點數	3,832,450	5,594,920
重搜節點數	2,867,134	4,511,381
寧靜節點數	17,757,439	34,708,341
重搜比例(%)	74.81	80.63
平均分枝數	6.67	6.84
平均時間(秒)	7.99	17.94

6. 逐層加深

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。圖表是在雙方都使用 PVS、同型表和著手 排序的演算法，比較逐層加身的效果。

	使用逐層加深	不使用逐層加深
深度	8	8
搜尋次數	395	397
搜尋節點數	3,039,734	5,536,213
重搜節點數	1,052,800	1,559,360
寧靜節點數	19,942,480	83,525,687
重搜比例(%)	34.63	28.16
平均分枝數	6.32	6.83
平均時間（秒）	7.99	35.77

7. 期望窗格

假設 d 層搜尋的結果會跟 $d-1$ 層的結果極為相近，則在搜尋 d 層時使用 $(\alpha, \beta) = (v - c, v + c)$，其中 v 是$d-1$ 層的數值、c 是常數。目的為降低初始 (α, β) 的 window size。 圖表是在雙方都使用逐層加深、PVS 和同型表的演算法，比較期望窗格的差異。

	使用期望窗格	不使用期望窗格
深度	8	8
搜尋次數	406	406
搜尋節點數	3,975,154	3,678,444
重搜節點數	2,875,478	1,225,270
寧靜節點數	20,023,340	28,753,031
重搜比例(%)	72.33	33.30
平均分枝數	6.54	6.48
平均時間（秒）	7.9	11.68