國立台北大學資訊工程學系專題報告

基於 ISMCTS 與 N-Tuple Networks 之幽靈棋 AI 研究

A Study on Geister AI Based on ISMCTS and N-Tuple Networks

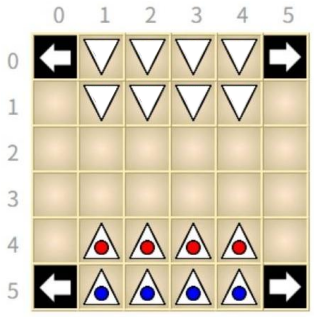
專題組員:楊淳亘、蘇亭伃、張可欣、周子馨

專題編號: PRJ-NTPUCSIE-113-011
執行期間: 113 年 7 月 至 114 年 6 月

1. 摘要

Geister（幽靈棋，日文：ガイスター）是一款由 Milton Bradley、Bütehorn Spiele、Drei Magier Spiele、Schmidt Spiele 等人設計製作的策略類桌遊，於 1982 年發售至今已有逾 40 年歷史[8]，且於日本發展出相關 AI 競賽ガイスターAI 大会（GAT）[1]。本研究旨在製作其 AI（Artificial Intelligence），並參加 2026 GAT 大會。由於 Geister 遊玩時將處於未知對手資訊的狀況，預計將以擅長不完全資訊（Incomplete Information）遊戲的 Information Set Monte Carlo Tree Search（簡稱 ISMCTS）演算法 [4] 與能快速判斷盤面狀況的 N-tuple Networks 作為 AI 的主要決策方針，以提升對戰能力。

2. 簡介



圖一：幽靈棋棋盤

Geister 是一款 6×6 棋盤的雙人策略遊戲（圖一），雙方各有 4 紅 4 藍共 8 隻幽靈，無法得知對手的配置。每回合可選一隻幽靈向上下左右移動一步，若移動到敵方位置則可吃掉並查看顏色。藍幽靈成功從對方角落逃脫，或吃光對方藍幽靈即獲勝；若誤吃光對方紅幽靈則敗北。由於資訊不完全，策略判斷與 AI 訓練比一般棋類更具挑戰。

幽靈棋屬於不完全資訊遊戲，傳統演算法如 Minimax、Alpha-Beta 剪枝與蒙特卡羅樹搜尋（Monte-Carlo Tree Search，簡稱 MCTS）演算法在處理隱藏情報上效果有限。因此，我們採用專為不完全資訊遊戲設計的 Information Set MCTS（簡稱 ISMCTS），並結合 N-tuple Networks 的快速評估與高適應性特點，提升 AI 在策略判斷與對手推測上的能力。此研究旨在強化 AI 於不完全資訊棋類的應用表現，為相關技術發展提供經驗與貢獻。

目標是製作一個幽靈棋的棋類 AI，並挑戰日本的幽靈棋人工智慧競賽ガイスターAI 大会（GAT2026）。

[1] ガイスターAI 大会@GAT2024.

3. 專題進行方式

3.1 成員配置與職責

本計畫成員分成兩組進行實作與分工合作，以提高開發效率並加深對各模組的理解與應用。第一組為 N-Tuple Networks 組，主要負責研究並實作 N-Tuple Networks 的特徵學習方法，包含棋盤狀態的編碼設計、特徵取樣方式、權重更新機制與策略評估函數的建立，致力於提升 AI 在面對各種局面時的快速判斷與策略選擇能力。

第二組為平台整合與 ISMCTS 組，前期主要負責將幽靈棋的介面對接至教授的對戰平台，使 AI 能與其他參賽者進行模擬對弈與測試；後期則深入研究 ISMCTS 演算法，設計適用於幽靈棋的不完全資訊搜尋結構，並與 N-Tuple Networks 組進行成果整合，提升整體 AI 的推理與決策表現。

兩組成員會定期進行交流與測試，根據實際對弈結果進行調整。

3.2 研究流程

本研究的流程大致分為兩個階段。第一階段主要針對 N-Tuple Networks 進行研究與訓練，採用自我對戰方式進行學習。初期對手為隨機走子的 Random Player，用以確認基礎策略評估能力；中期則改以具備完全資訊，更有策略的 Flat Monte Carlo（上帝視角）對手作為訓練評估；後期則以 GAT2020 冠軍 AI Naotti2020 作為評估對手，藉此驗證訓練成果是否具備競爭力。與此同時，平台組同步負責將幽靈棋對接至教授提供的對戰平台，確保 AI 能順利參與實際對局與測試。

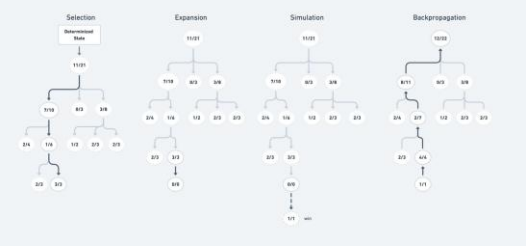
第二階段聚焦於 ISMCTS 的研究與實作，並將其與 N-Tuple Networks 進行整合，建立一套同時具備推理能力與快速評估能力的幽靈棋 AI。整合完成後，繼續以 Random Player、Flat Monte Carlo、Naotti2020 為基準對手進行對戰測試，用以評估整體系統在不完全資訊環境下的決策效果與策略穩定性，作為最終成果驗證依據。

3.3 開發工具

本專題程式的幽靈棋 AI 用 C++進行開發，對戰平台使用 Java 撰寫，開發工具為 Visual Studio Code。

3.4 Information Set MCTS (ISMCTS)

傳統的蒙地卡羅樹搜尋（Monte Carlo Tree Search, MCTS）主要應用於完全資訊遊戲，如圍棋與西洋棋，所有遊戲資訊均對玩家公開[1]。然而，在如撲克牌與橋牌等不完全資訊遊戲中，部分資訊對玩家隱藏，導致傳統 MCTS 無法直接應用於此類環境。ISMCTS 透過在每次模擬過程中隨機生成可能的資訊集合（即不完全資訊的潛在配置），並於該資訊集合上運行 MCTS，以模擬決策過程[6]。在此架構下，ISMCTS 於模擬初始階段根據已知資訊與遊戲規則，隨機確定一個完整遊戲狀態（稱為「確定化」），並基於該狀態執行 MCTS。最終，模擬結果將回傳至原始資訊集合，以最佳化決策策略，全部流程如圖二所示。



圖二：ISMCTS 流程圖

1. 確定化（Determinization）：從資訊集中的所有可能狀態中隨機選擇一個具體的遊戲狀態，將不確定性轉化為確定性。在幽靈棋實作中，根據已知資訊，將對手的未知棋盤進行隨機配置，得到完整資訊棋盤。
2. 選擇(Selection)：從根節點開始，沿著樹選擇具有最高上置信區間（Upper Confidence Bound, UCB）值的子節點，直到達到與確定化狀態相容的子節點。
3. 擴展（Expansion）：在選擇的葉節點上，若該節點不是終局狀態且未完全擴展，則從可能的動作中選擇一個未被探索的動作，生成新的子節點並添加到樹中。
4. 模擬(Simulation)：從擴展的節點開始，根據預設策略進行隨機模擬，直到達到終局狀態。
5. 回傳（Backpropagation）：將模擬結果反向傳播至路徑上的所有節點，更新它們的訪問次數和勝利次數。

在有限時間內不斷重複以上步驟，並使用 UCT(Upper Confidence Bounds to Trees）中的 UCB 公式計算各節點分數[5][10]，公式如下：

$$UCB_i = \frac{w_i}{n_i} + C \times \sqrt{\frac{\ln N}{n_i}} \qquad (1)$$

其中，w_i 為節點 i 的平均勝率或評估值；C 為探索參數，用於平衡探索（Exploration）與利用（Exploitation）；N 為父節點的總訪問次數；n_i 為節點 i 的訪問次數。

3.5 N-Tuple Networks

N-Tuple Networks 是一種有效的特徵提取與評估方法，廣泛應用於棋盤遊戲的人工智慧開發中。其核心概念在於從遊戲盤面中選取特定位置的子集（即 N 個格子），將這些子集視為特徵，並為每個可能的特徵組合分配一個評估值。這種方法能夠有效降低狀態空間的維度，從而提升學習與預測的效率。

在 N-Tuple Networks 中，啟發式值公式（Heuristic Value Function）$V(b)$ 的主要用途是評估遊戲盤面或狀態的優劣程度，用以制定策略、預測結果，以及在複雜遊戲環境中取得成功。公式如下[2]：

$$V(b) = \frac{1}{|\varphi(b)|} \cdot \varphi(b) \cdot \theta \qquad (2)$$

其中，$\varphi(b)$為一個 n 維（即為所有特徵的總數）的二元向量，記錄了在盤面 b 中出現了哪些預先定義的特徵；$|\varphi(b)|$是代表$\varphi(b)$中 1 的個數，也就是盤面 b 中出現的特徵總數。θ 是一個 n 維（即為所有特徵的總數）的權重向量，代表了每個特徵對於盤面價值的影響程度。特徵和權重的設定需要根據具體問題進行定義和學習。

3.6 N-tuple Networks 與 ISMCTS 演算法運用於 Geister 遊戲

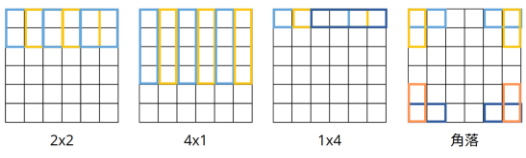
Geister 遊戲是一款具有不完全資訊的策略遊戲，玩家需在有限的資訊下進行推理與決策。N-Tuple Networks 能夠從有限的盤面資訊中提取關鍵特徵，而 ISMCTS 則在決策過程中考慮了資訊的不完全性，能夠更真實地模擬對手行為。

3.6.1 N-tuple Networks 於遊戲的應用

在過去的研究中，N-tuple Networks 已被成功應用於多種棋盤遊戲。例如，在奧賽羅（Othello）遊戲中[9]，N-tuple Networks 應用於棋盤特徵提取，並使用較短的 N-Tuple Networks（如大小為 2 的 N-Tuple Networks），顯著提高了學習效果和遊戲表現。使用僅包含 288 個權重的網路，該方法在奧賽羅聯盟的線上比賽中達到了近 96%的勝率，超過了當時的其他任何玩家。

在愛因斯坦棋(Einstein Würfelt Nicht!）的研究中[5]，作者將遊戲盤面劃分為多個 6-Tuple，利用兩個查找表（Look-Up Table）儲存特徵的勝場數和出現次數，計算特徵勝率。成功運用 N-tuple Networks 來解決遊戲中狀態空間過大的問題，透過將盤面分割成多個小的區域，提取出有效的特徵，並結合蒙地卡羅學習進行訓練，N-tuple Networks 能夠有效地評估盤面優劣。

本研究最後採用 4-Tuple Networks 作為幽靈棋 AI 的盤面評估方式，將盤面劃分為 1×4、2×2、4×1 三種矩形的 Tuple，與為了幽靈棋特性增加的角落 Tuple，共四種，如圖三所示。



圖三：本研究採用之 4-Tuple 形狀

3.6.2 ISMCTS 於遊戲的應用

ISMCTS 已被廣泛應用於各種不完全資訊遊戲中。在一項研究中[6]，作者探討了不完全資訊遊戲中傳統確定化方法（Determinized UCT）存在的策略融合與計算預算分散等缺陷，並創新提出資訊集 MCTS（ISMCTS）系列演算法。該方法透過多次確定化構建單一搜尋樹，使得整體計算資源能夠全盤聚焦，有效解決了策略融合問題。ISMCTS 包括兩種方法，稱之為 SO-ISMCTS 與 MO-ISMCTS，分別進一步修正了玩家在資訊集中區分狀態和對手部分行動時的誤判。該論文之實驗結果顯示，在《魔戒：征服》（LOTR:C）遊戲中，ISMCTS 明顯優於傳統方法；在 Phantom(4,4,4)遊戲中，MO-ISMCTS 表現較為突出。

3.7 當前幽靈棋採用之技術

在幽靈棋 AI 相關研究論文中，使用 DREAM 演算法的幽靈棋 AI[3]最後以平均 80%左右勝率擊敗隨機玩家，使用 Deep CFR（Deep Counterfactual Regret Minimization）的幽靈棋 AI[2]則是以最高 85%左右的勝率打敗隨機玩家。

4. 主要成果與評估

本研究使用 4-Tuple Networks 作為盤面評估方式，利用 1×4、2×2、4×1 三種矩形 Tuple 和角落 Tuple，可以有效將盤面的狀態空間從 1.8×10^{14} 降低為 40,625，有效加快盤面的查找與評估速度。

4-Tuple Networks 與 ISMCTS 結合之後，發展出幾種優秀策略：

1. 誘導對手吃紅棋：棋局剛開始時，會先派出紅棋進攻角落，讓對手以為是藍棋，進而吃我方紅棋。
2. 藍棋逃脫策略：在遊戲後期，我方

已被吃三顆紅棋，敵方會降低吃棋機率，此時將派出我方藍棋往角落進攻，提高藍棋從角落逃脫獲勝的機率。

3. 評估對手棋子的顏色：透過觀察棋局，與 ISMCTS 模擬，可以猜測對手棋的顏色，進而判斷是否吃棋，降低誤吃紅棋與提高吃藍棋的機會。

- 目前對打最高勝率

對手說明：

1. Random Player：隨機走步玩家
2. Flat MC：Flat Monte Carlo，擁有完全資訊(上帝視角)的 AI
3. Naotti2020：GAT2020 的冠軍程式

	Random Player	Flat MC	Naotti 2020
本研究之幽靈棋 AI	99%	82%	50%

表一：本研究程式與各個對手的對打最高勝率

根據表一，可以發現本研究之幽靈棋 AI 跟各個對手對打皆有優秀表現。跟 Random Player 對打成績(99%)已贏過 3.5 小節所提到使用 DREAM 演算法的 80%勝率與 Deep CFR 演算法的 85%勝率。在跟擁有完全資訊的 Flat MC 對戰時，我們有些策略無法使用，因此仍有最高 82%的勝率是非常不錯的。最後，我們的程式已能與 GAT2020 冠軍程式 Naotti2020 不分上下(50%勝率)，十分優秀。

5. 結語與展望

本研究成功開發出一套結合 ISMCTS 與 N-Tuple Networks 的幽靈棋 AI，能夠有效應對不完全資訊下的決策挑戰，並在對戰實驗中展現出高度競爭力。

研究過程中，我們驗證了 N-Tuple Networks 在盤面特徵學習與快速評估上的效率，同時也確認了 ISMCTS 在處理資訊不完全環境中的決策潛力。兩者結合後，使 AI 能夠在複雜棋局中快速做出反應，同時具備合理的推測與判斷能力。

未來的研究將專注於對現有演算法的進一步最佳化，包含以下幾個方向：

1. ISMCTS 模擬效率提升：透過更好的確定化方法，提高模擬過程中生成對手配置的合理性，減少無效模擬的比例。
2. N-Tuple 特徵組合調整：重新設計或篩選更具代表性的 Tuple 排列組合，並探索動態特徵選擇機制，以提升盤面評估的準確性。
3. 參數自我調整機制：導入自我調整策略（如根據對局進展自動調整探索參數），使 AI 在不同局勢下展現更佳的策略適應力。

總結而言，本專題已建立穩固的技術基礎，後續將朝著演算法內部的結構調整與策略最佳化方向持續深化，期望在未來比賽中展現更優異的實力，並為不完全資訊 AI 技術的發展貢獻實質成果。

6. 銘謝

感謝指導教授於專題的製作過程中給許多技術上的建議與幫助，引導我們進行方向的同時，保留足夠多的空間讓我們嘗試各種可能的方法。除了定期的討論外，也撥出了額外的時間協助專題的進行。也感謝實驗室的

學長，於製作中遇到問題時與我們進行討論，提供了許多不同面向的想法，也提供很多實作上的幫助。

7. 參考文獻

[1] Browne C.B. et al. (2012). A Survey of Monte Carlo Tree Search Methods. IEEE Transactions on Computational Intelligence and AI in Games, 4(1), 1 - 43.

[2] Chen, C., & Kaneko, T. (2019). Utilizing History Information in Acquiring Strategies for Board Game Geister by Deep Counterfactual Regret Minimization. Proceedings of the Game Programming Workshop 2019, 2019, 20 - 27. https://ipsj.ixsq.nii.ac.jp/records/199967

[3] Chen, C., & Kaneko, T. (2020). Application of DREAM to the Board Game Geister. Proceedings of the Game Programming Workshop 2020, 2020, 111 - 117. https://ipsj.ixsq.nii.ac.jp/records/207663

[4] Chen, J., Tang, S., & Wu, I. (2022). Monte-Carlo Simulation for Mahjong. Journal of Information Science and Engineering, 38(4), 775 - 790. https://doi.org/10.6688/JISE.202207_38(4).0005

[5] Chu, Y. R., Chen, Y., Hsueh, C., & Wu, I. (2017). An Agent for EinStein Würfelt Nicht! Using N-Tuple Networks. 2017 Conference on Technologies and Applications of Artificial Intelligence (TAAI). https://doi.org/10.1109/TAAI.2017.32

[6] Cowling, P. I., Powley, E. J., & Whitehouse, D. (2012). Information Set Monte Carlo Tree Search. IEEE Transactions on Computational Intelligence and AI in Games, 4(2), 120 - 143. https://doi.org/10.1109/TCIAIG.2012.2200894

[7]DeNA. (2024). HandyRL. Github. https://github.com/DeNA/HandyRL

[8] Good & Bad Ghosts. (n.d.). Wikipedia. https://en.wikipedia.org/wiki/Good_%26_Bad_Ghosts

[9] Jaśkowski, W. (2014). Systematic N-Tuple Networks for Position Evaluation: Exceeding 90% in the Othello League. ICGA Journal, 37(2), 85 - 96. https://doi.org/10.48550/arXiv.1406.1509

[10] Kocsis, L. & Szepesvári, C. (2006). Bandit based Monte-Carlo Planning. ECML-06, LNCS/LNAI 4212.

[11] Szubert, M., & Jaśkowski, W. (2014). Temporal Difference Learning of N-Tuple Networks for the Game 2048. 2014 IEEE Conference on Computational Intelligence and Games. https://doi.org/10.1109/CIG.2014.6932907

[12] Yeh, K., Wu, I., Hsueh,

C., Chang, C., Liang, C., & Chiang, H. (2017). Multistage Temporal Difference Learning for 2048-Like Games. IEEE Transactions on Computational Intelligence and AI in Games, 9(4), 369–380. https://doi.org/10.1109/TCIAIG.2016.2593710