啟發演算發演算法
轉珠遊戲最佳路徑搜尋研究

楊永睿,Yong-Rui Yang
Computer Science and Information Engineering
Fu Jen Catholic University
New Taipei,Taiwan
heguorui0402@gmail.com

***Abstract*—轉珠遊戲是一個數學的益智遊戲，讓玩家透果轉動版面中的珠子，使版面可以消除盡可能多的珠子來獲得高分。本實驗比較了三種演算法來嘗試解決轉珠最佳路徑的搜尋問題，實驗結果發現在計算資源小於２秒與最大路徑長為25的情況下GA所獲得的分數均由於IDFS與ACO。**

Keywords*—*轉珠遊戲*、*超啟發演算法*、*IDFS*、*GA*、*ACO

I. 介紹

轉珠遊戲是一個數學的益智遊戲，讓玩家透過挑選版面中的 5x6 珠子中的一顆進行連續交換，以此來改變本面使版面變得整齊，Fig 1 為遊戲操作示意圖，當大於等於 3 顆珠子連成直線後即可消除，消除的珠子越多分數越高。此遊戲為經典數學遊戲的 15Puzzle 的變形，在其基礎上引入了不同的珠子顏色與消除方式以此來增加遊戲的複雜度與可玩性。本研究的目標就是在短時間內找到一條盡可能獲得高分的路徑。

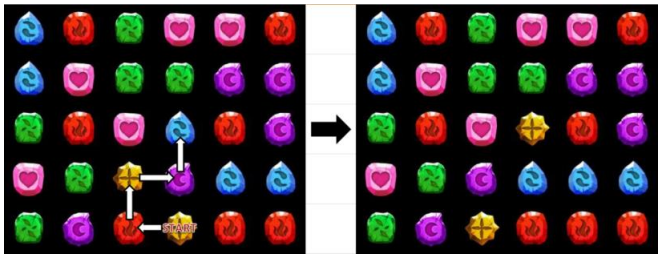

Fig 1. 轉珠遊戲連續移動與結果示意圖

為了解決在短時間內找到一個品質不錯的解因而使用超啟發學習的相關演算法，相關超啟發演算法使用 ChatGPT4[3] 挑選，最後選擇 Iterative Deepening Depth-First Search (IDFS)[4]、Genetic Algorithm(GA)[5]與 Ant Colony Optimization(ACO)[6]，最終實驗結果在 1 秒計算資源與最大路徑長為 25 的限制下。GA 效果最好平均分數可以到達最佳解的 62.6%。

II. 相關工作

A. *Iterative Deepening Depth-First Search(IDFS)*

IDFS 融合了 DFS 與 BFS 的優點，解決了 BFS 需要太多空間與 DFS 需要太多時間與不保證最優解的問題，可以很好的在有限資源下可以找到 15Puzzle 的解[4]。

B. *Genetic Algorithm(GA)*

GA 受到自然中 DNA 啟發的超啟發演算法[5]，屬於一種群體演化，基本運算有自然選擇、交配與突變。可以較容易與其他演算法做結合，在多問題上皆有不錯的性能。

C. *Ant Colony Optimization*

ACO 受到自然界螞蟻覓食啟發的超啟發演算法[6]，屬於一種群體演化，利用螞蟻透過與費洛蒙的互動來找到最佳路徑。基本設計包括螞蟻的移動與費洛蒙的更新，與搜索空間的表示。在組合最佳化問題上有不錯的表現。

D. *開源的轉珠研究*

目前對轉珠問題多為開源專案，以轉珠問題為研究的論文較少，所使用的演算法為 BFS[7][8]、強化學習[9]、Beam Search[10] 。

III. 實驗方法

本段主要為說明實驗演算法挑選過程、問題搜索空間、解的評估方式、演算法設計方式等等。

A. *演算法挑選過程*

本研究使用 ChatGPT4 來輔助挑選實驗使用的演算法，Fig 2 為 ChatGPT4 挑選超啟發演算法流程圖，首先先讓 ChatGPT 挑選 10 個超啟發演算法，再從中挑選挑選出 3 種認為會有不錯效果的演算法，接著就是驗證 ChatGPT 所挑選出來的演算法是否具有合理性與是否存在。最終 ChatGPT 挑選出來的演算法為 GA、ACO 與 GRASP[11]， Fig 3 為 GPT4 對話過程展示三種演算法推薦的原因，但最終考慮到 GRASP 模板太廣泛而改使用 IDFS 做研究。

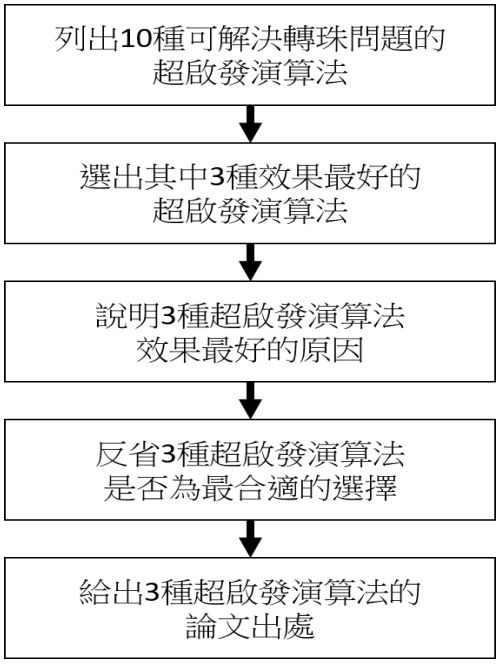

Fig 2. ChatGPT4挑選超啟發演算法流程圖

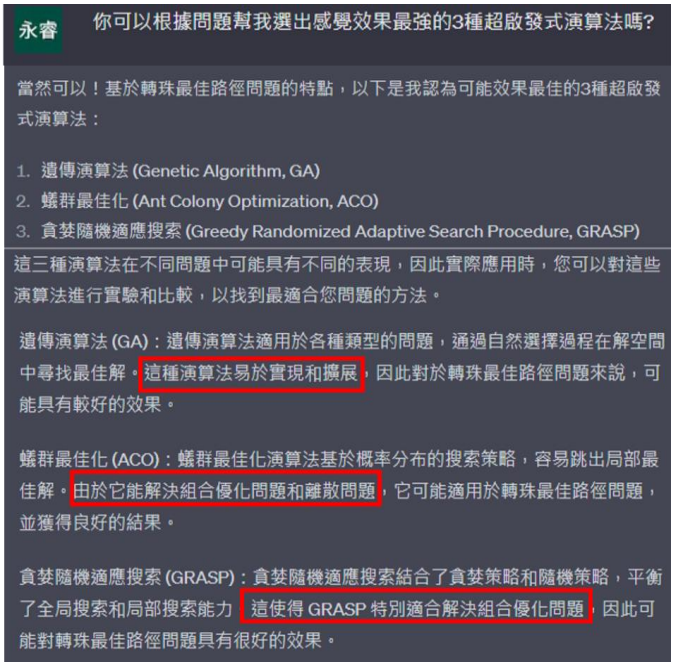

Fig 3. ChatGPT4推薦3種超啟發演算法對話過程

B. 表示法

版面的表示法，版面大小為 5x6，共有 30 顆珠子，珠子顏色種類共有 6 種，版面的表達方式為一個一個長度為 30，由 6 種英文字母所組成的字串。解的表示法，由起始位置與路徑所構成，一個長度為 L+2 的字串，其中 L 為路徑長，前 2 碼為起始點的座標，2 碼後面為路徑，路徑由一個 0 到 3 組成長度為 L 的字串。

C. 版面化簡

化簡版面有兩個假設，一是不同類別珠子不影響最終的解品質，二是不同的初始排序也不影響最終的解品質。版面經過統計各類珠子數量後依照數量由高到低排序最後再重新分類，最後化簡共有 1206 種版面。

D. 解的搜索空間

一個解由起始位置與路徑所組成，其複雜度為

$$O(L) = 30 \times 4 \times 3^{L-1} \quad (1)$$

其中30為起始位置的所有選擇，4為起始點的方向選擇，3為不回頭的方向選擇，L為路徑長。搜索空間與路徑為指數關係。

E. 解的評估方式

本實驗用來評估解的分數由可消除部分、剩餘潛在可消除分數與剩餘版面相連分數三個部分所構成，可消除部分分數用來評估目前路徑實際消除所獲得的分數，公式為

$$\left(1+\frac{combos-1}{4}\right) \times \sum_{i=1}^{combos}\left(1+\frac{orb(i)-3}{4}\right) \quad (2)$$

其中$combos$為連擊數總數，$orb(i)$:第 i 連擊的相連珠子數。該部分分數下界為 0，上界為 32.5。剩餘潛在可消除分數用來評估未來路徑尚可消除所獲得的最大分數，公式為

$$32.5 - \left(1+\frac{combos-1}{4}\right) \times \sum_{i=1}^{combos}\left(1+\frac{orb(i)-3}{4}\right) \quad (3)$$

其中$combos$為潛在的連擊數總數，$orb(i)$:第 i 連擊的相連珠子數。該部分分數下界為 0，上界為 32.5。剩餘版面相連分數用來評估剩餘版面相連程度所獲得的分數，公式為

$$\sum_{i=1}^{blocks}\left(\frac{block(i)}{3}\right)^2 \div \text{剩餘珠子的總數} \quad (4)$$

其中blocks為剩餘版面相連區塊總數，block(i):第i相連區塊的相連總數。平均量化方式為版面分數除以當前版面所能達到的最佳分數。

F. IDFS 演算法設計

IDFS 為 DFS 的迭代演算法，本研究將再次迭代 IDFS 來擴展搜尋最佳路徑的深度。

G. GA 演算法設計

- 個體表達：使用與解表達方式相同的編碼。
- 自然選擇：使用輪盤法。
- 更新群體方式為子代完全取代
- 交配方式：使用單點交叉的方式，且交叉點不會切到起始點的位置。
- 突變方式：起始點會隨機突變到周圍9格的位置，路徑突變使用均勻突變，另外再加入隨機的長度突變使路徑長度隨機加長或減短範圍落在-10到10之間。

H. ACO 演算法設計

本實驗使用 MMAS 為 ACO 的改良演算法，差別在於給費洛蒙添加上限與下限。MMAS 所蒐尋的空間為 30 個節點的聯通圖，隨機放置螞蟻，螞蟻每走固定步數就更新費洛蒙，走超過最大路徑螞蟻就會被重製，當一段時間沒有更新最佳解的時候就會重置費洛蒙矩陣。

IV. 實驗結果

實驗一到實驗四，基本設定為最大路徑長為 25，CPU 使用 i5-13500，使用的計算資源為 1 秒，使用 100 個隨機版面評估演算法性能。

A. 實驗一:演算法參數優化

- *IDFS參數優化*

IDFS 實驗優化的參數為最大深度，本實驗比較了最大深度為 7、10、13 的結果， Table 1 展示了不同最大深度與迭代次數的參數比較結果，結果顯示最大深度為 13，迭代 2 次效果最好。

Table 1. IDFS參數比較結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
(7,4)	33.518	0.586	0.397	17.71
(10,3)	33.831	0.592	0.94	14.29
(13,2)	35.558	0.622	1.005	12.05

- *GA參數優化*

GA 實驗優化的參數為群體大小，本實驗比較了群體大小為 50、100、200 的結果，Table 2 展示了不同群體大小的比較結果，結果顯示群體大小為 50 效果最好。

Table 2. GA參數比較結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
50	35.773	0.626	1.001	16.376
100	35.631	0.623	1.001	16.614
200	32.763	0.573	1.03	14.886

- *ACO參數優化*

ACO 實驗優化的參數為更新費洛蒙螞蟻走的步，本實驗比較了螞蟻步數為 3、6、9 的結果， Table 3 展示了不同螞蟻步數的比較結果，結果螞蟻步數為 9 效果最好。

Table 3. ACO參數比較結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
3	33.897	0.593	1.006	20.13
6	34.147	0.597	1.005	20.55
9	34.481	0.603	1	20.526

B. 實驗二:演算法性能比較

實驗二比較了三種經過參數優化後的演算法在100種版面與1206種版面的平均評估結果，Table 4與

Table 5，從結果可以看到在只評估 100 種版面時 GA 效果最好，評估 1206 種版面時 IDFS 效果最好。

Table 4. 100種版面平均結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
IDFS	35.558	0.622	1.005	12.05
GA	35.773	0.626	1.001	16.376
MMAS	34.481	0.603	1	20.526

Table 5. 1206種版面平均結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
IDFS	41.561	0.723	0.999	11.224
GA	41.141	0.716	1.004	16.601
MMAS	40.689	0.708	1.002	20.741

C. 實驗三: 在不同最大路徑長的演算法性能比較

實驗三為三種不同最大路徑長的結果比較，實驗長度為 25、35、45，25 為對照組結果為 Table 4，35 與 45 為實驗組結果為 Table 6 與

Table 7，可以發現增加路徑長可以讓 GA 與 ACO 增加解的平均分數。IDFS 所獲的最佳路徑長度沒有明顯影響。

Table 6. 最大路徑為35的實驗結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
IDFS	35.59	0.623	1.004	11.85
GA	37.33	0.65	1.002	22.24
MMAS	35.81	0.626	1.001	26.472

Table 7. 最大路徑為45的實驗結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
IDFS	35.498	0.622	1.008	11.75
GA	37.156	0.65	1.003	24.51
MMAS	34.568	0.6	1.013	29.56

D. 實驗四: 在不同計算資源下的演算法性能比較

實驗四為三種不同計算資源下的結果比較，實驗分別測試了1秒、2秒、3秒，1秒為對照組結果為Table 4，2秒與3秒為實驗組結果為Table 8與

Table 9，可以發現增加解的平均分數。實驗結果顯示 IDFS 和 GA 會發生沒有使用完計算資源的狀況發生。

Table 8. 計算資源為2秒的實驗結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
IDFS	36.423	0.638	1.874	14.774
GA	36.845	0.645	1.854	16.838
MMAS	35.369	0.618	2.007	21.306

Table 9. 計算資源為3秒的實驗結果

參數	平均分數	平均量化分數	平均花費時間	平均路進長度
IDFS	36.933	0.646	2.517	16.486
GA	36.82	0.644	2.102	16.776
MMAS	36.412	0.637	3.004	21.444

V. 討論與結論

由最終實驗結果得知在最大路徑長為 25 與計算資源為 2 秒以下時，GA 的效能最好，其次是 IDFS，最後是 MMAS。在不同最大路徑長的情況下 GA 的效能最好。ACO 效果最差的原因可能是因為在設計演算法時缺發對問題的轉換設計導致費洛蒙無法有效引導螞蟻的走向。本研究在設計實驗時未考慮演算法是否能充分利用計算資源與演算法迭代是否收斂，未來研究可以在設計實驗時對於這個問題多加考慮。

參考資料

[1] JABO, “【心得】為新手而生~轉珠遊戲基礎教學(前篇)” 巴哈姆特電玩資訊站, https://forum.gamer.com.tw/C.php?bsn=23805&snA=77313, November 2013 (accessed June. 2023).

[2] JABO, “【心得】為新手而生~轉珠遊戲基礎教學(後篇)” 巴哈姆特電玩資訊站, https://forum.gamer.com.tw/Co.php?bsn=23805&sn=634462, November 2013 (accessed June. 2023).

[3] OpenAI, “ChatGPT” OpenAI, https://chat.openai.com/, May 2023 (accessed June. 2023).

[4] KORF, Richard E. Depth-first iterative-deepening: An optimal admissible tree search. Artificial intelligence, 1985, 27.1: 97-109.

[5] HOLLAND, John H. Genetic algorithms. Scientific american, 1992, 267.1: 66-73.

[6] DORIGO, Marco; DI CARO, Gianni. Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). IEEE, 1999. p. 1470-1477.

[7] senkevinli, “PAD-Auto-Solver,” GitHub, https://github.com/senkevinli/PAD-Auto-Solver, Jan 2021 (accessed June. 2023).

[8] Matt Hargett, “padopt” GitHub, https://github.com/matthargett/padopt, Jun 2016 (accessed June. 2023).

[9] nuwapi, “P-PAD” GitHub, https://github.com/nuwapi/P-PAD, Apr 2019 (accessed June. 2023).

[10] pazusoba, “core” GitHub, https://github.com/pazusoba/core, Apr 2022 (accessed June. 2023).

[11] FEO, Thomas A.; RESENDE, Mauricio GC. A probabilistic heuristic for a computationally difficult set covering problem. Operations research letters, 1989, 8.2: 67-71.