應用遺傳演算法於遊戲搜尋樹之研究
A Study on Applying Genetic Algorithms to Game Search Trees

黃科元
Ke-Yuan Hwang
義守大學資訊工程研究所
Institute of Information Engineering
I-Shou University, Kaohsiung

洪宗貝
Tzung-Pei Hong
義守大學資訊管理學系
Dept. of Information Management
I-Shou University, Kaohsiung
tphong@csa500.isu.edu.tw

林文揚
Wen-Yang Lin
義守大學資訊管理學系
Dept. of Information Management
I-Shou University, Kaohsiung
wylin@csa500.isu.edu.tw

摘要

在本篇論文中，我們提出一遺傳遊戲搜尋樹演算法，解決傳統搜尋演算法無法深層搜尋的缺點。實驗證實我們所提出的演算法，在限定時間內可增加搜尋準確度。

關鍵字：遊戲搜尋樹，遺傳演算法，評估值。

Abstract

In this paper, we propose a genetic search algorithm to solve the problem of traditional game tree searching. Experiments show that the accuracy using our algorithm can be improved within a limited amount of time.

Keywords: game search tree, genetic algorithm, evaluation value.

壹、緒論

在人工智慧的研究課題中，最早也是最有趣的一支就是電腦下棋。自從 1950 年 Shannon 首先提出"對局機(Chess-Playing Machine)"[9]用於西洋棋後，電腦下棋的研究便從未間歇過。許多文獻及新方法也不斷的被提出，像是在 1981 年台大電機所的張耀騰先生所研究人工智慧在象棋上的應用[2]，1985 年台大資研所所提的開局資料庫[1]，一直到最近由 IBM 電腦公司所發展出的深藍系統(Deep Blue)，更是已經可以與世界西洋棋冠軍相匹敵。

在電腦下棋中最主要的關鍵就是遊戲搜尋樹[6]。遊戲搜尋樹主要是模擬人們的思考，預設對手所有可能下法，然後在對手可能下的每一步中，找出我們應對之步，接著再從我們的每一應對之步，預設對手的可能下法。依此類推以長出一棵遊戲搜尋樹，然後從此樹中找出我們最好的下法來下。此遊戲搜尋樹可以表達出人類下棋時的思考方式。

近年來遺傳演算法[4]被廣泛的應用於利用電腦來求近似解。其利用達爾文的適者生存，不適者淘汰的觀念，在每代中保留較佳的解，而逐步將不適用的解淘汰掉，因此所求的解理論上將會一代比一代更好。雖然所求的解並非最佳解，但通常若是代數足夠多將可相當逼近最佳解，且可以節省許多的計算時間和計算空間。因此在本文中我們嘗試利用遺傳演算法於遊戲樹之搜尋。我們主要是針對一人遊戲樹，利用遺傳演算法來改進深度優先逐步加深搜尋法[3]（Depth-First Iterative-Deepening）。雖然其所求的解並非最佳解，但是它所能搜尋的層次，在相同的時間下比傳統的遊戲搜尋法所能搜尋的層次更深。而通常一個遊戲樹，搜尋的層次愈深，所求出的解愈具有參考意義。因此我們所提的方法在相同的時間下，比傳統方法所求的最佳解更正確。

本論文其餘章節的安排如下。在第二節中，我們首先對遊戲搜尋樹的觀念作一說明，並介紹遺傳演算法的基本觀念及如何利用遺傳演算法來解決問題。在第三節中，我們提出了一人遊戲樹的遺傳演算法，我們設計了一編碼方式，將一人遊戲樹編成基因，再用遺傳演算法來解決一人遊戲樹的問題。在第四節中，說明我們實驗的方式，並對實驗的結果作一比較與分析，以了解我們的方法的優劣。最後，在第五節中是我們的結論與未來的研究方向。

貳、相關背景

2.1 一人遊戲樹

一人遊戲樹主要是適用於只要一人(或電腦)就可以玩的遊戲。像拼盤遊戲(Puzzle)、魔術方塊(Magic Square)等都是此類遊戲。此遊戲樹之展開如圖 2-1 所示，其中點 A 代表整棵樹中一個可行解，所以整個遊戲的目的即在求得由根節點到點 A 這條路徑。

解一人遊戲樹已有相當多的方法[8]，像深度優先搜尋法(Depth-First Search)，廣度優先搜尋法(Breadth-First Search)，逐步加深搜尋法(Depth-First Iterative-Deepening)，最好優先搜尋法(Best-First Search)，啓發式搜尋法(Heuristic Search) 等等，都廣泛地被應用。其共通的問題在於每增加一層搜尋的深度時，所需的時間和空間皆呈指數成長，因此在時間和空間的限制下所能搜尋的層次不足，容易造成錯誤的判斷。

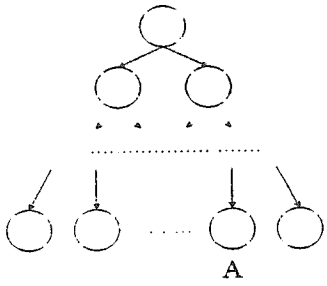

圖 2-1 一人遊戲樹

2.2 遺傳演算法

遺傳演算法是 John Holland[5]於 1975 年首先提出來的，主要是將達爾文的適者生存，不適者淘汰的觀念，用於解決最佳化問題。其利用生物學上的交配和突變的原理來設計電腦求解運算子來產生它的後代，再利用優勝劣敗的原理，來決定它的下一代，因此所求的解，理論上將會一代比一代更好。雖然所求出的解通常是一個近似最佳解，可是所花的時間相對比求最佳解的演算法要少很多，因此被廣泛地應用於各個領域上，如知識擷取、排程、財務等等[7]。下列是一個典型的遺傳演算法。

遺傳演算法
步驟 1: 定義問題的表示方式。
步驟 2: 產生起始的人口，假設含有 N 個個體。
步驟 3: 定義評估函數 F。
步驟 4: 從人口群中隨機或根據評估值選取 M 個，作爲執行遺傳運算的雙親。
步驟 5: 執行交配和突變，產生後代。
步驟 6: 利用評估函數評估所有的個體。
步驟 7: 選取最好的 N 個個體做爲下一代人口。
步驟 8: 假如達到預設的終止條件，則結束；否則跳到步驟 4。繼續執行。

通常的終止條件有下列三種
1. 達到預設的代數。
2. 達到預設的時間。
3. 每一代的評估值已經收斂。

參、 遺傳遊戲樹演算法

對於一人遊戲樹，我們所關心的就是從整棵樹中，找到一連串的走法，再利用這一連串的走法，找到對於自己最有利的解答。譬如說在智慧拼盤中找出如何從啓始盤面走至終止盤面的一條路徑。過去採用傳統的深度搜尋和廣度搜尋，在空間和時間上的需求極大。因此，我們企望利用遺傳演算法的優點，以最短的時間來求得一較佳的走法。要將一人遊戲樹用遺傳演算法來求解，首先要將遊戲樹遺傳化，也就是必須要將遊戲樹轉化成能用遺傳演算法處理的形式，因此須有下列幾個步驟。

1. 編碼：將遊戲樹編碼，也就是要將遊戲樹的結構字串化，如此才能使用交配和突變等遺傳演算法的運算子來運算。以一人遊戲樹而言，我們所設計的編碼方式如圖 3-1 編碼圖所示。

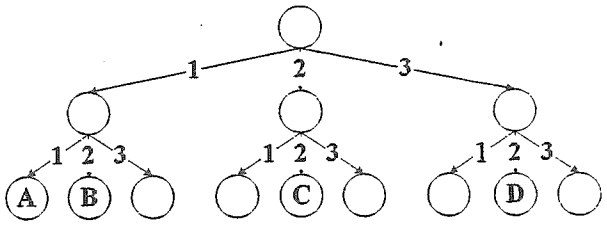

圖 3-1 一人遊戲樹的編碼方式

圖 3-1 表示一棵三層的遊戲樹，而每一個節點有三個分枝，其中的數字就是代表第幾個分枝，所以節點 A 編碼成爲 11，而 11 就形成了一人遺傳遊戲樹方法中的一個可能個體。同理點 B 可編成 12，點 C 可編成 22，而點 D 可編成 32，也都構成可能的個體。因此，所需要搜尋的深度便可決定出編碼的長度；而增加編碼的長度，就可以輕易的求得較深的層次。此乃遊戲樹遺傳化最大的優點—增加個體的長度，並不會大幅增加運算的時間。

2. 評估函數和評估值：遺傳演算法必須有評估函數，才能決定每一個個體的優劣，所以設計出一個好的評估函數是非常重要的。在此我們以葉節點盤面的評估值作爲此遊戲樹的評估函數。因爲每一種遊戲都會有其考量盤面分數的公式，因此我們只需將葉節點的盤面帶入該遊戲樹的評估公式，就可以算出每一條可能路徑的優劣。

3. 交配：在遺傳演算法中交配和突變運算子的選擇將會影響遺傳演算法的結果，所以選擇合適的遺傳運算子是非常重要的。在一人遊戲樹中，一連串的步法間存在一些相關性，所以常常只要走到某種固定的盤面型態出現，我們便可以採用相同的回應措施，因此在作交配運算時，採用單點子段交配 (one-point substring crossover) 是一種相當合理的想法，因爲某一較佳的子段可能是對某一盤面較佳的回應法則。此外就速度性而言，當我們資料結構採用鏈結的方法，則子段交配只需要鏈結直接指到斷點處即可完成，因此單點子段交配可享有較快的速度。

4. 突變：考慮突變的方法和考慮交配的方法一樣，我們以合理性和速度性來加以考量。在一人遊戲樹中我們可以常常見到某一段幾乎相同的下法，而不同的只是其中一兩步的走法，因此利用單點突變法 (one-point mutation) 正好可以變換那一兩步不同的走法，而其他的突變法則比較不適用於遊戲樹的特性。在速度的考量方面，單點突變法是最快的突變法，因爲只要變更一個位元就可以完成突變動作，而其他的突變法則必須變更數個位元，因此單點突變法有較快的速度。

5. 選擇較佳的子代：一人遊戲樹所針對的只有自己一人而已，不需考慮其他玩家的走法，所以只要保留具有較高的評估值的個體即可；因爲較高的評估值代表對自己有較佳的結果，所以在選擇較佳子代時只要針對較大評估值作考慮。

綜合以上所述，我們的一人遺傳遊戲樹演算法敘述如下：

一人遺傳遊戲樹演算法

步驟 1：將一人遊戲樹編碼。

步驟 2：定義演算法所需的時間和所需要的演進的代數。

步驟 3：定義欲求的深度和評估盤面的函數。

步驟 4：定義交配率和定義突變率。

步驟 5：產生初始的人口數 N。

步驟 6：用交配率選取 M 個雙親，進行交配運算。

步驟 7：用突變率選取 O 個個體，進行突變運算。

步驟 8：利用預定的評估函數，求出所有後代的評估值。

步驟 9：選擇較佳 N 個評估值的個體，作爲下一次演進的人口。

步驟 10：假如達到預設的終止條件，則結束；否則跳到步驟 6。

當演算法終止時，其具有最好評估值的個體所對應的走法即爲所求。

爲更清楚說明上述的演算過程，在此我們以一個 9 方格的拼字遊戲爲例，此實例如圖 3-2 所示，我們希望能從圖 A 能夠拼成圖 B，而拼字的法則只能將相鄰的兩格互換。例如圖 3-2 中 A 圖的 9 可以和 1、2、5、6 這四個點互換，在此我們假設此圖是循環方格，而在移動的過程中，我們希望能夠在有限的時間內，盡可能的移動圖 A，使其到最後能夠達到或接近圖 B。

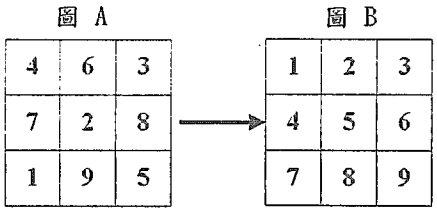

圖 3-2 拼字遊戲範例

步驟 1：首先我們針對所解的問題設計適當的編碼方式，在此我們根據可能的移動來編碼。以拼字遊戲而言，我們所設計的編碼方式如表 3-1 所示。我們把圖 A 中 4 所在的那格和 6 所在的那格互換的方式編成 1，依此類推我們將所有可能的移動方式編碼成 18 種走法，再加上不做任何移動的編碼，總共有 19 個移動方式。

步驟 2：我們預定時間爲 10 秒和代數爲 50 代。

表 3-1 拼字遊戲的編碼法則

交換點 A	交換點 B	編碼	交換點 A	交換點 B	編碼
4	6	1	4	7	10
6	3	2	7	1	11
3	4	3	1	4	12
7	8	4	6	8	13
8	2	5	8	9	14
2	7	6	9	6	15
1	9	7	3	2	16
9	5	8	2	5	17
5	1	9	5	3	18
當編碼爲 0 時代表不做任何移動					

步驟 3：接著我們再決定我們所要計算的層次，假設我們暫訂搜尋層次爲 15 層，那麼每一個個體的長度便是由 15 個單位所組成，其中每一個單位存放 0～18 中的一個可能值，而一個個體所代表的意義便是走 15 步的走法，例如有一個個體爲 (14,4,8,0,18,17,10,12,4,6,17,17,17,14,16, 0) 那它所代表的意義便是如圖 3-3 所示。

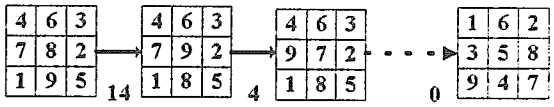

圖 3-3 走法的表示

我們利用盤面的數字差來當評估函數，因此評估函數定義如下：

$$F = 18 - \sum_{i=0}^{i=3} \sum_{j=0}^{j=3} D(R_{i,j} - N_{i,j})$$

其中 $R_{i,j}$ 是(i,j)這個位置最後所想要的正確數字，而 $N_{i,j}$ 是(i, j)這個位置現在的數字。而 $D(R_{i,j} - N_{i,j})$ 是求得兩點的位置距離差，例如圖 3-2 的圖 A 中，點 9 和點 6 位置差爲 1，因爲兩者只要一步就可以移到，而點 9 和點 2 則位置差爲 2。我們由位置的差值，就可以得知和正確位置的差異，所以只要越接近正確的的盤面，我們可以得到的分數就越高。

步驟 4：我們定義交配率 50%，突變率 10%。

步驟 5：我們隨機產生初始的 10 個個體，其個體如表 3-2 所示。

步驟 6：我們隨機選取其中兩個個體，例如若我們選取(14, 4, 8, 0,18,17,10,12, 4, 6,17,17,17,14,16)和(10, 0, 1, 6, 3, 2, 2, 0, 5, 0, 8,15,12, 2, 2)兩個個體來交配，假設所選的交配點在第 11 個位置，再利用子段交配的方式產生子代，我們可以得到(10, 0, 1, 6, 3, 2, 2, 0, 5, 0, 8,17,17,14,16)和(14, 4, 8, 0,18,17,10,12, 4, 6,17,15,12, 2, 2)兩個後代。依照這個方法我們所產生的後代如表 3-3 所示。

表 3-2 初始化的人口

NO	個體表示	評估值
1	14, 4, 8, 0,18,17,10,12, 4, 6,17,17,17,14,16	8
2	0, 4, 2, 3, 7, 2,12, 0, 3, 6,13,13,14, 1, 5	5
3	7,11, 8,12, 8, 0, 3, 9, 1, 2,11,13, 9, 3, 2	7
4	14, 2, 1,10, 0,18,17, 5,15,13, 4,11, 7, 9, 6	3
5	15,18,12,15,13,17, 7, 0,18,15, 7,16,17, 2, 3	8
6	10, 0, 1, 6, 3, 2, 2, 0, 5, 0, 8, 5, 2, 2, 2	9
7	8, 7, 1, 8, 9,13,18,13, 1,16, 9, 7, 8, 8,14	5
8	16, 9, 2, 1, 4, 8, 7,10,10,16, 4,12,18, 7,13	12
9	16, 5,13,16,15, 8,17, 5,14, 0,18,10,14,13,10	7
10	17,11, 6, 7, 8, 4, 9, 7,12, 9,18, 1, 1, 6,17	7

表 3-3 交配所產生的後代

NO	來源	個體表示	F
1	交配	10, 0, 1, 6, 3, 2, 2, 0, 5, 0, 8,17,17,14,16	12
2	交配	0, 4, 2, 3, 7, 2,12, 0, 3, 6,13,10,14,13,10	10
3	交配	16, 5,13,16,18,17,10,12, 4, 6,17,17,17,14,16	9
4	交配	16, 5,13,16,13,17, 7, 0,18,15, 7,16,17, 2, 3	9
5	交配	15,18, 2, 1, 4, 8, 7,10,10,16, 4,12,18, 7,13	8
6	交配	14, 4, 8, 0,18,17,10,12, 4, 6,17,15,12, 2, 2	8
7	交配	16, 5,13,16,15, 8,17, 5,14, 0,18,13,14, 1, 5	8
8	交配	14, 4, 8, 0,15, 8,17, 5,14, 0,18,10,14,13,10	8
9	交配	15,18,12,15,15, 8,17, 5,14, 0,18,10,14,13,10	7
10	交配	16, 9,12,15,13,17, 7, 0,18,15, 7,16,17, 2, 3	7

表 3-4突變所產生的後代

NO	來源	個體表示	F
1	突變	7,11, 8,12, 8, 0, 3, 9, 8, 2,11,13, 9, 3, 2	8
2	突變	5,18,12,15,13,17, 7, 0,18,15,0,16,17, 2, 3	8
3	突變	6, 5, 3,16,15, 8,17, 5,14, 0,18,10,14,13,10	8
4	突變	6, 5,13,16,16, 8,17, 5,14, 0,18,10,14,13,10	8
5	突變	4, 2, 7,10, 0,18,17, 5,15,13, 4,11, 7, 9, 6	8

表 3- 5 選取 10 個最佳的後代

NO	個體表示	F
1	16, 9, 2, 1, 4, 8, 7,10,10,16, 4,12,18, 7,13	12
2	15,18, 2, 1, 4, 8, 7,10,10,16, 4,12,18, 7,13	10
3	10, 0, 1, 6, 3, 2, 2, 0, 5, 0, 8,15,12, 2, 2	9
4	10, 0, 1, 6, 3, 2, 2, 0, 5, 0, 8,17,17,14,16	9
5	14, 4, 8, 0,18,17,10,12, 4, 6,17,17,17,14,16	8
6	1,18,12,15,13,17, 7, 0,18,15, 7,16,17, 2, 3	8
7	15,18,12,15,13,17, 7, 0,18,15, 0,16,17, 2, 3	8
8	16, 5,13,16,16, 8,17, 5,14, 0,18,10,14,13,10	8
9	7,11, 8,12, 8, 0, 3, 9, 1, 2,11,13, 9, 3, 2	7
10	6, 5,13,16,15, 8,17, 5,14, 0,18,10,14,13,10	7

步驟 7: 我們隨機選取一個個體，利用單點突變的方法產生子代，例如若我們選取(15,18,12,15,13,17, 7, 0,18,15, 7,16,17, 2, 3)這個個體來突變，假設突變點在第 10 個位置，則可產生子代(15,18,12,15,13,17, 7, 0,18,15, 0,16,17, 2, 3)。依照這個方法，我們可以產生可能的後代。步驟 7 所產生的後代，如表 3-4 所示。

步驟 8: 利用評估函數 F 來求出評估值，得到的評估值列於表 3-3 後面。

步驟 9: 我們可以從表 3-2，表 3-3 和表 3-4 中篩選出最佳的 10 個個體做爲下一代的人口後代，如表 3-5。

步驟 10: 在經過八代的演進，已經收斂，而得到(16, 9, 2, 1, 4, 8, 7, 10,16, 4,12, 13, 7, 11, 4) ，我們將此個體依照順序轉換拼字遊戲，最後我們可以轉換成如圖 3-4。

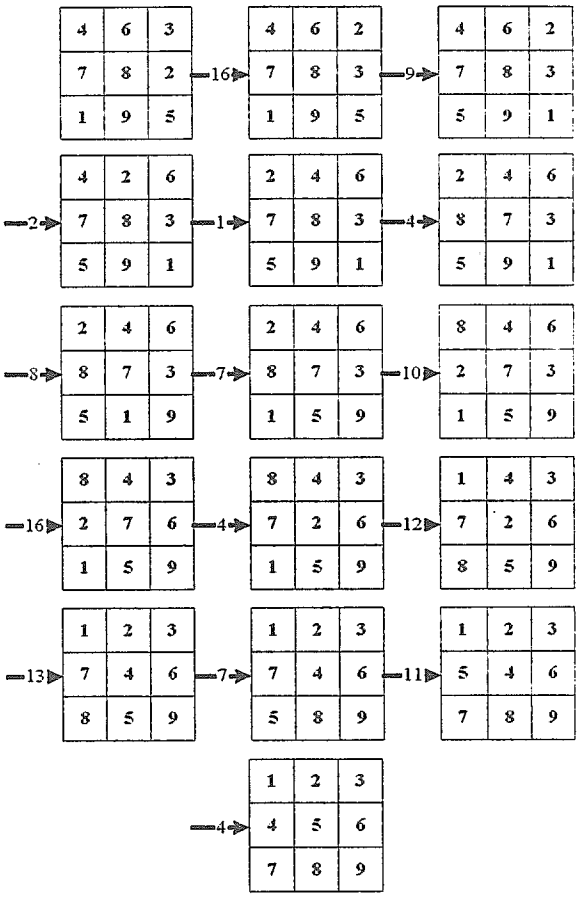

圖 3-4 完成圖

肆、實驗分析

我們使用 3*3 拼字遊戲來實驗比較遺傳遊戲樹演算法和傳統的深度優先搜尋法的優劣。我們主要建立在 IBM-6X86 150+的機器上，利用 C 語言來執行。主要考慮時間和所選路徑的好壞來評定。我們首先使用傳統的深度優先搜尋法(DFS)和遺傳演算法(GA)對不同遊戲樹層次做時間的比較。在遺傳演算法中，所採用的代數是 40 代，人口數則有 10、15、20、25、30、35 六種，實驗結果如圖 4-1 所示。由此圖可以得知，傳統的深度優先演算法因爲呈指數成長，因此到了搜尋第五層後已經要花相當多的時間，而遺傳演

算法的時間和層次間的關係主要是呈現線性關係，因此層次的增加，並不會造成時間上太多的負擔。

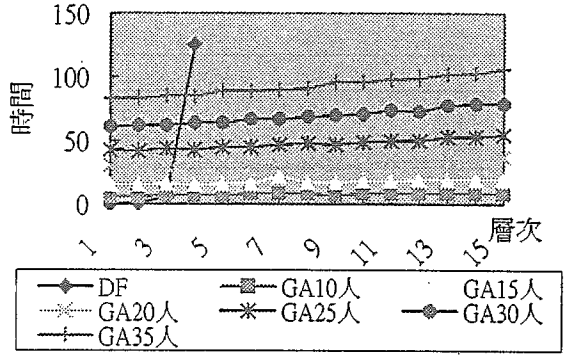

圖 4-1代數固定下(40 代)，時間和層次在不同人口數的關係圖

接著我們考慮人口數和時間的關係。所採用的代數爲 40 代，人口數分別是 10、15、20、25、30 和 35，並分別對 3、6、9、12 和 15 五種層次作實驗比較，實驗結果如圖 4-2 所示。由此圖中，我們可以清楚得知，遺傳演算法的時間和人口數的關係亦是呈現線性關係，人數的增加，並不會造成時間上太多的負擔。

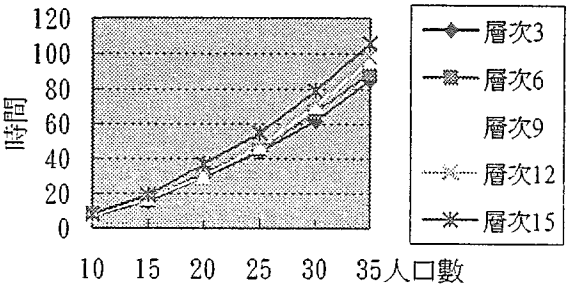

圖 4-2 代數固定下(40 代)，人口數和時間在不同會次的關係圖

接著我們使用傳統的深度優先搜尋法和遺傳演算法對不同遊戲樹層次做準確性的比較。準確性是以每一種演算法最後所得到的盤面和所希望的盤面比較後所得的失誤個數來決定。假如一個數字不在其正確位置，則計算其最快需幾步才能到達指定位置，把所有數字的步數加起來即是其失誤個數。所以最爲正確的是失誤個數爲零。在遺傳演算法中，所採用的代數是 40 代，人口數則有 10、15、20、25、30、35 六種，實驗結果如圖 4-3 所示。

由圖 4-3 中，我們可以清楚得知，傳統的深度優先演算法因爲呈指數成長，因此到了搜尋第五層後已經要花相當多的時間，因此在時間考量下，無法得知結果，而在五層內很明顯的遺傳演算法並不會輸給深度優先演算法，而且在層次越深的計算下，雖然曲線仍然有些震盪，但也越來越爲正確。

接著我們考慮層次和時間和在不同代數下的關係。在遺傳演算法中，我們固定人口數 35，代數則有 10、20、30、40 四種。實驗結果如圖 4-4 所示。由此圖中，我們可以清楚得知，層次愈多，所需花的時間愈多，且成線性成長。

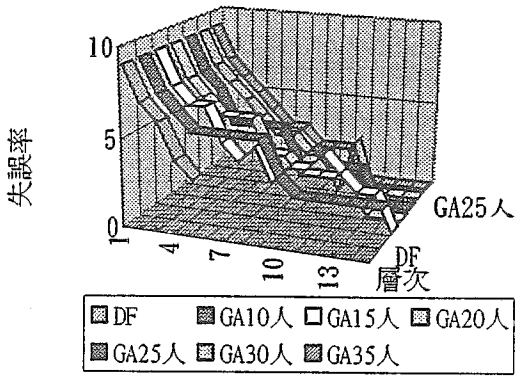

圖 4-3 代數固定下(40 代)，失誤率和層次在不同人口數的關係圖

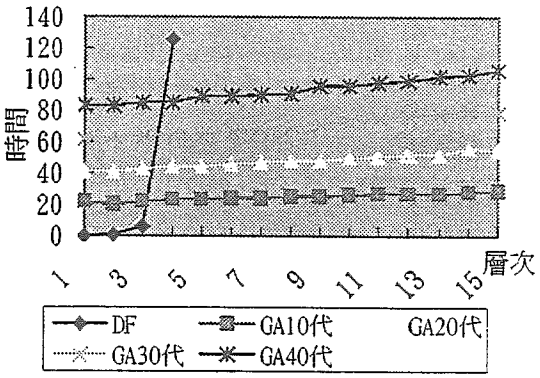

圖 4-4 人口數固定下(35 人)，時間和層次在不同代數的關係圖

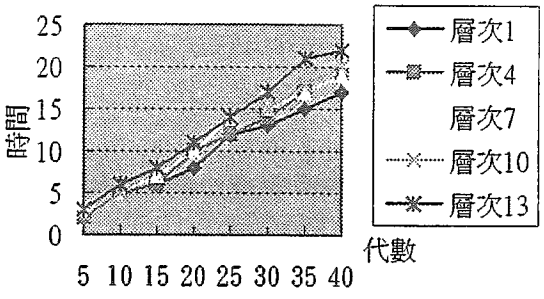

圖 4-5 固定人口數固定下(35 人)，代數和時間在不同層次的關係

接著我們考慮代數和時間的關係。在遺傳演算法中，我們固定人口數 40 人，層次爲 3、6、9、12、15 五種。實驗結果如圖 4-5 所示。由此圖中，可以得知代數愈多，所需花的時間愈多，且成線性成長。

接著我們考慮正確性和代數的關係。在遺傳演算法中，我們固定人口數 35，代數則有 10、15、20、25、30、35 六種。實驗結果如圖 4-6 所示。由此圖可以清楚得知，在每一代數下幾乎層次愈多，準確性愈高，而且代數愈多，準確性也愈高。

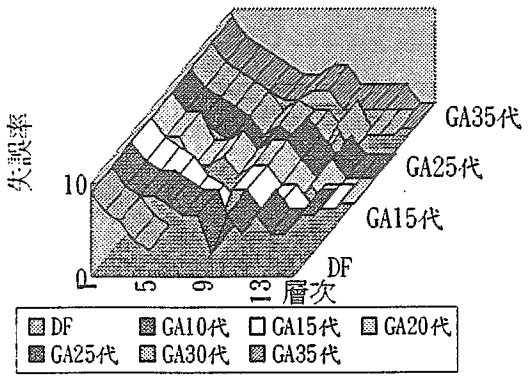

圖 4-6人口數固定(35 人)，失誤率和層次對不同代數的關係圖

最後我們比較，在相同時間內何者較爲正確。實驗結果如圖 4-7 所示。由此圖可以得知，我們所提的方法的確比傳統的方法更爲精確。

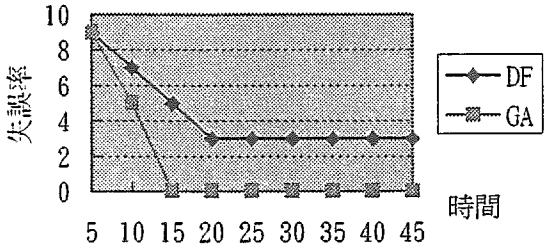

圖 4-7 相同時間內，遺傳演算法和深度優先演算法的比較

伍、結論

在本論文中，我們提出了新的遊戲搜尋樹演算法，利用遺傳演算法的優點來加速遊戲樹的搜尋。我們所提的遺傳演算法在相同時間的限制下可以搜尋的層次較深，且所搜尋出來的解比傳統演算法在較低層次所搜尋出的最佳解還要好。

此外，傳統的深度優先搜尋法可能會重覆的尋找相同的盤面，形成迴路的情形。譬如像拼字遊戲中，將相鄰的兩點交換後，下一個步驟又可以換回來。在遺傳演算法中，透過評估值將較好的後代保留住，而迴路中的路徑將會比其他路徑爲差，所以我們很快就能擺脫在迴路中求解的問題，而能更快求得解答。

在未來的研究方面，我們希望可以將所提的遺傳演算法，實際應用於象棋或圍棋中，以能有深層搜尋的優勢，再配合較佳的知識庫，而能達到下棋高手的境界。

參考文獻

[1] 許舜欽和黃東輝，"智慧型中國象棋程式設計和製作，" 民國 74 年全國計算機會議論文集，1985, pp. 505-509。

[2] 張耀騰，"人工智慧在電腦象棋上的應用，"台灣大學電機研究所碩士論文，1981。

[3] T. Anantharaman, Murray S. Campbell and Feng-hsiung Hsu, "Singular extensions: adding selectivity to brute-force searching," *Artificial Intelligence*, Vol. 43, 1990, pp. 99-109.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Second Edition, MIT Press, 1992.

[6] R. Knight, *Artificial Intelligence*, Second Edition.

[7] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT press, 1996.

[8] H. Sahni, *Data Structures in Pascal*, Third Edition.

[9] C. E. Shonnon, "Programming a computer for playing chess," *Philosophical Magazine*, Vol. 41, March, 1950, pp. 256-275.