國立交通大學

資訊科學與工程研究所

碩士論文

電腦象棋審局評分自動調整系統

The Automatically Tuning System of Evaluation Function for Computer Chinese Chess

研 究 生：高暐倫

指導教授：吳毅成　教授

中 華 民 國 九 十 八 年 八 月

電腦象棋審局評分自動調整系統

The Automatically Tuning System of Evaluation Function for Computer Chinese Chess

研 究 生：高暐倫　　Student：Wei-Lun Kao

指導教授：吳毅成　　Advisor：I-Chen Wu

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

August 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年八月

電腦象棋審局評分自動調整系統

研究生：高暐倫　　指導教授：吳毅成

國立交通大學 資訊科學與工程研究所

摘要

象棋是世界上最受歡迎的棋類遊戲之一，電腦象棋在電腦人工智慧上的地位也很重要，電腦象棋的審局是影響電腦象棋棋力的重要因素，然而審局是由大量參數組合計算而成，每個參數都需要設定分數值，傳統的做法是以人工不斷地調整審局參數值，費時費工且難以精準。

本篇論文研究象棋審局，利用基因演算法和統計方法，設計一套新的參數自動優化系統，自動調整審局的參數值，並針對棋局的不同階段做參數的調整。從實驗結果看來，經由此系統可以增進電腦象棋的棋力。

The Automatically Tuning System of Evaluation Function for Computer Chinese Chess.

Student: Wei-Lun Kao Advisor: I-Chen Wu

Department of Computer Science and Information Engineering

National Chiao Tung University

ABSTRACT

Chinese Chess is one of the most popular game in the world, computer Chinese Chess is important in computer artificial intelligence, computer evaluation function is a key factor for computer Chinese Chess, but the evaluation function is computed by a lot of parameters, each parameter is needed to be set a value, the traditional method is to tune values of evaluation parameters continuously by people, it is time-consuming, laborious, and difficult to be precise.

This thesis researches the evaluation function for computer Chinese Chess, we design a new automatically tuning system of evaluation function by genetic algorithm and statistical method, it tunes values of evaluation parameters automatically, and divides into different stages by chess game. The experiment results show the system increases the power of our computer Chinese Chess program.

誌謝

首先要感謝我的指導教授，吳毅成博士，由於他不厭其煩的細心指導，這篇論文才得以順利完成。

此外特別要感謝曾汶傑學長和林宏軒同學，他們在研究的過程中給予我許多寶貴的意見和指導，以及同實驗室裡一起奮鬥的夥伴宜智、靖平、益嘉、冠翬、柏甫和學弟妹們的協助。當然，還有很多替我加油鼓勵的朋友們，在我的研究期間，給了我相當多的關懷與鼓勵，陪我度過這段最值得回憶的學生生活。

最後，我要感謝我的父母、姊姊，在我的求學生涯中給了我最大的支持和照顧。謹以此論文，獻給我最摯愛的家人。

目錄

圖表目錄

第一章、緒論

象棋是華人世界流行的兩人對弈棋類遊戲。棋盤是九條直線和十條橫線相交而成，分為紅方和黑方，一般紅方為先手，棋局進行後雙方輪流走子。雙方各有十六顆棋子，分別是一隻將(帥)、兩隻士(仕)、兩隻象(相)、 兩隻車(俥)、兩隻馬(傌)、兩隻包(炮)、五隻卒(兵)共七種子力。

一般情況，當一方的將(帥)被吃掉或是沒有棋步可以動，即為輸棋。不過除了一般情況，還有一些因為棋規而會有判定勝敗的情況，目前象棋有幾種不同的棋規，國際上最常用的是亞洲棋規[25]。

象棋的歷史悠久，在戰國時期就已有文獻記錄象棋的存在，象棋的文化地位相當重要。下象棋可以訓練定性、專注力、判斷力、思考力…等，是個適合男女老幼的益智休閒遊戲。英國 BGN 智力運動網路公司曾做過一項統計，現今，全世界約有兩億多人會下象棋，象棋的魅力不容小覷。

電腦棋類[18]對局在人工智慧的領域中，是非常重要的研究指標。就棋類的複雜度而言，象棋比圍棋、日本將棋的複雜度低，比西洋棋的複雜度高，越複雜的棋類，電腦程式的棋力越難超過人腦。

西元一九九七年是人工智慧界一個重要的里程碑，由 IBM 團隊開發的西洋棋程式深藍(Deep Blue)，打敗人類世界西洋棋棋王卡斯帕羅夫(Kasparov)，震驚世界，也帶來人工智慧研究的風潮。

不過時至今日，電腦象棋[10][19][20][26][27]雖已足以抗衡特級大師，但仍未打敗人類象棋棋王，相信不久的將來，電腦象棋將能超越人類。

由前面所介紹，可以知道電腦象棋的研究是相當重要的，在電腦人工智慧領域上扮演舉足輕重的角色。增進電腦象棋的棋力，打敗人類棋王，將是非常重大的突破。

1.1 電腦象棋設計

一般電腦象棋設計會把棋局分成三個主要階段，開局、中局和殘局，在不同的階段會使用不同的策略來下棋。

開局是指象棋剛開始布局的階段，一般指前面十幾步棋，此時雙方著重於把自己的子力移動到有利位置，調整成戰鬥陣型，以利於中殘局的攻殺。由於開局的階段分支還不會太多，人類已研究出許多常走且不差的開局。電腦象棋的設計，則是利用這些人類研究出來的開局，或是用統計棋譜的方式，將開局的棋步和資料存到資料庫中，製作成電腦象棋開局庫[2][8][12][13][24]，讓電腦象棋在開局階段直接使用，不但可以節省思考時間，也能走成不錯的開局。

中局是指開局結束到盤面上剩餘子力不多的情況，雙方著重於吃掉對方的王或是把中局引導到自己滿意的殘局，中局盤面相當複雜，也沒有足夠的棋譜和人類經驗可以製作成資料庫，需要程式的思考，找出最好的棋步。程式思考主要是靠搜尋(Search)和審局(Evaluation)，審局[1][22]通常使用審局函數，將盤面的優劣轉換成分數;搜尋[7][11][21]通常使用 Alpha-Beta 搜尋，可以判斷到各個棋步走十幾步後的各個盤面。搜尋和審局是相輔相成、互相影響的，審局越準越好，搜尋越深也越好，不過審局函式的判斷越多，就越花時間，搜尋的深度也會因此減少，因此必須在審局精準度和搜尋深度之間做平衡和取捨。

殘局是指盤面上剩餘子力不多的情況，棋局不一定會進入殘局，很有可能在中局就結束。殘局由於子力很少，通常需要特殊的走法和盤面控制，且要走很多步後，才能分出勝負，例如「海底撈月」、「孤

兵擒王」…等，靠程式的審局和搜尋很難下出真正該走的棋步。殘局通常會加入殘局知識和殘局庫，殘局知識是人類棋手針對各種剩餘子力，個別探討該種盤面的優劣及該如何下；殘局庫[3][4][5][6][9][15][23]則是更進一步，用窮舉的方式，把所有子力所有盤面的資訊事先算出來，做成資料庫，稱為殘局庫，讓電腦象棋在該盤面就立刻得出勝敗結果並知道該走什麼，完全不用程式思考，目前殘局庫是利用回溯分析(Retrograde Analysis)方法產生。

除了棋局的階段外，電腦象棋還有很多種技巧加強棋力，例如背景思考、平行化、時間控制…等，很多的部分都可以加強改進。在電腦象棋的各個部分中，電腦象棋的審局又是特別的困難，審局是人類棋手的強項，但電腦卻一直很難審的精準，沒有很好的解決方法，有待自我學習[14][28][29]的研究。

審局是把盤面的優劣轉換成數字，用分數來比較各盤面的好壞，是影響電腦象棋程式強弱的重要因素。許舜欽教授在「電腦象棋程式的設計與製作」中，提到審局最常見的主要因素有子力的多寡、位置的重要性、棋子的靈活度、棋子之間相互保護與威脅的情形。

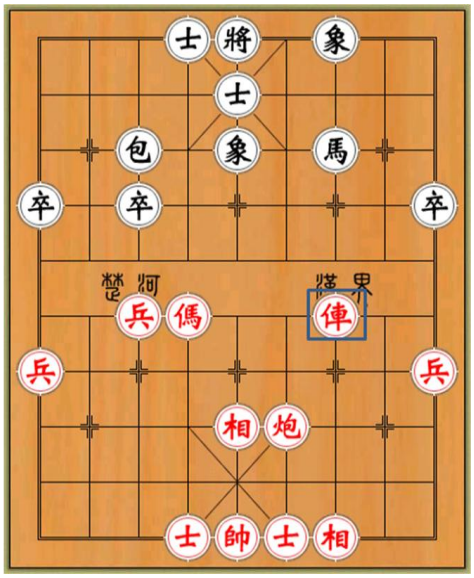


圖 1-1 子力的多寡

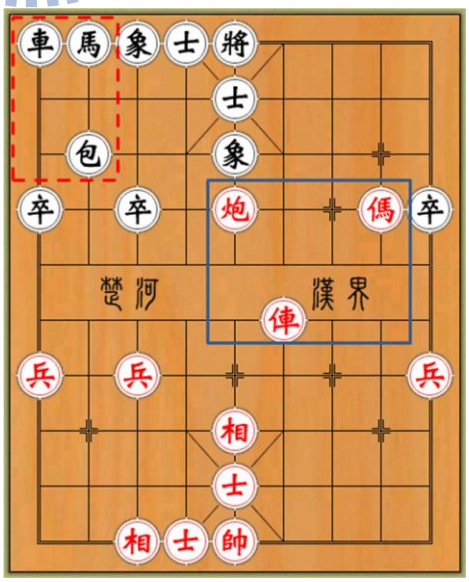


圖 1-2 位置的重要性

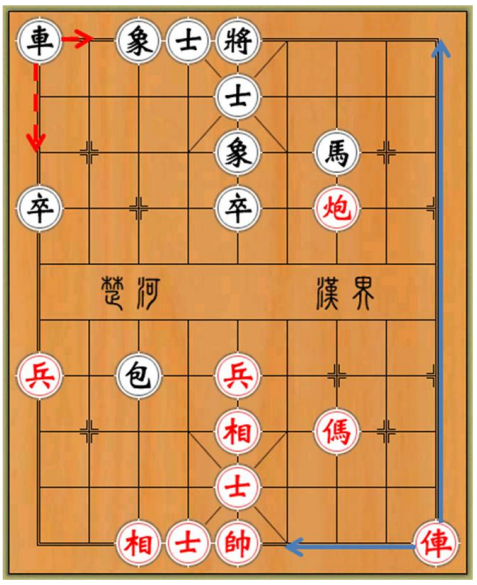


圖 1-3 棋子的靈活度

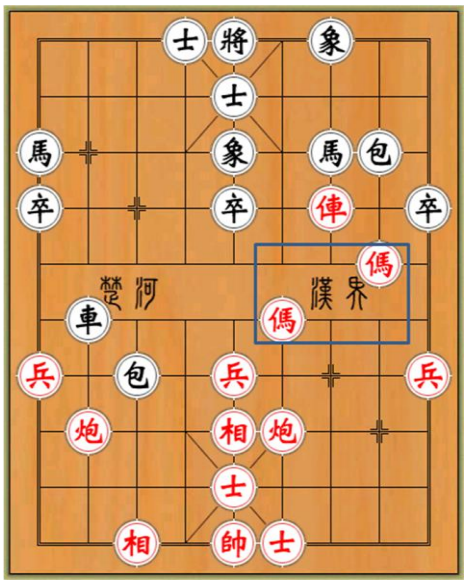


圖 1-4 棋子之間相互保護和威脅

子力是審局最基本的方法，子力多就比較好，而不同的棋子價值也不同，如圖 1-1，紅方多一隻俥，很明顯佔優。除了子力的多寡外，棋子所佔的位置也非常重要，巡河車、臥槽馬、沉底包…等，都是不錯的位置，如圖 1-2，雙方子力相同，但紅方俥傌炮都在好位置，攻擊力很強，而黑方車馬包都在原位，位置很差，紅佔優。棋子的靈活度也很重要，如圖 1-3，紅黑雙方的車馬位置都一樣，甚至位置也一樣，不過紅方的俥傌明顯比黑方車馬能走的步多很多，相當靈活，紅佔優。至於棋子之間相互保護和威脅，直接用圖 1-4 舉例，紅方連環傌互相保護，不會被黑車攻擊，而黑方七路馬沒有保護，直接受到紅俥攻擊，這盤面還是紅優。

1.2 研究動機及研究目的

電腦象棋的審局是由大量參數所組合記算而成，我們把這些參數分成子力位置表和其他參數來說明。

子力位置表是各子力在棋盤上不同位置所佔的分數，共有將士象車馬包卒七種子力，圖 1-5 和圖 1-6 是馬和卒的子力位置表[19]，可

以看出各位置的分數有所不同。

4	8	16	12	4	12	16	8	4
4	10	28	16	8	16	28	10	4
12	14	16	20	18	20	16	14	12
8	24	18	24	20	24	18	24	8
6	16	14	18	16	18	14	16	6
4	12	16	14	12	14	16	12	4
2	6	8	6	10	6	8	6	2
4	2	8	8	4	8	8	2	4
0	2	4	4	-2	4	4	2	0
0	-4	0	0	0	0	0	-4	0

圖 1-5 馬的子力位置表

0	3	6	9	12	9	6	3	0
18	36	56	80	120	80	56	36	18
14	26	42	60	80	60	42	26	14
10	20	30	34	40	34	30	20	10
6	12	18	18	20	18	18	12	6
2	0	8	0	8	0	8	0	2
0	0	-2	0	4	0	-2	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

圖 1-6 卒的子力位置表

我們直接拿 1.1.4 的圖 1-2 當例子，人類棋手看得出來，紅方俥傌炮的位置比黑方車馬包的位置好，我們來看看電腦由子力位置分是否能看得出來紅優。車的位置分看圖 1-7，紅方俥 20 分，黑方車經過上下對稱後是-2 分；馬的位置分看圖 1-8，紅方傌 24 分，黑方馬-4 分；包的位置分看圖 1-9，紅方炮 10 分，黑方包 0 分。依子力位置表，紅方的子力位置分就比黑方多 60 分，紅方佔優，所以良好的子力位置表對審局影響很大。

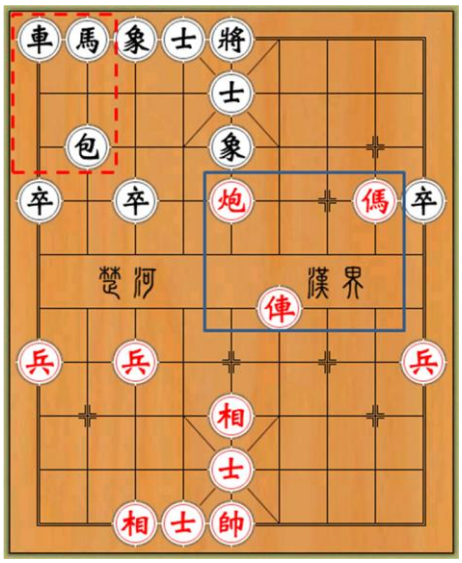


圖 1-2 位置的重要性

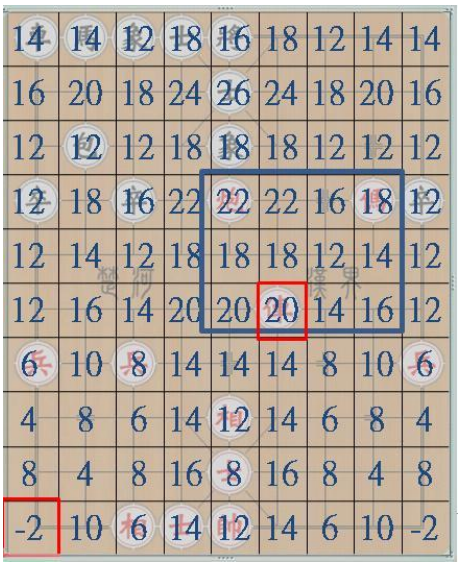
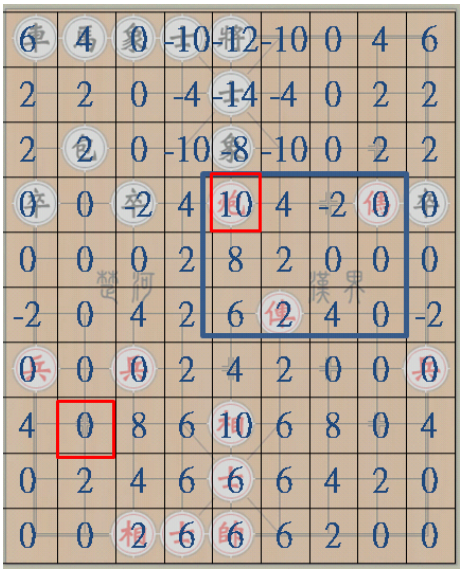


圖 1-7 車的子力位置表

4	8	16	12	4	12	16	8	4
4	10	28	16	8	16	28	10	4
12	14	16	20	18	20	16	14	12
8	24	18	24	20	24	18	24	8
6	16	14	18	16	18	14	16	6
4	12	16	14	12	14	16	12	4
2	6	8	6	10	6	8	6	2
4	2	8	8	4	8	8	2	4
0	2	4	4	-2	4	4	2	0
0	-4	0	0	0	0	0	-4	0

圖 1-8 馬的子力位置表



6	4	0	-10	-12	-10	0	4	6
2	2	0	-4	-14	-4	0	2	2
2	2	0	-10	-8	-10	0	2	2
0	0	-2	4	10	4	-2	0	0
0	0	0	2	8	2	0	0	0
-2	0	4	2	6	2	4	0	-2
0	0	0	2	4	2	0	0	0
4	0	8	6	10	6	8	0	4
0	2	4	6	6	6	4	2	0
0	0	2	6	6	6	2	0	0

圖 1-9　包的子力位置表

依據棋局各階段的不同，子力位置表也會有所不同，在「電腦象棋程式位置評分表之研究」[16]中，子力位置表分成開局、中局、殘局三個階段，分界是透過棋子的威力值的多寡判定，定義棋子的威力值，車是 2、馬是 1、包是 1，當棋子的威力值總和等於 16 時為開局，棋子的威力值總和介於 8 和 15 之間為中局，棋子的威力值總和小於等於 7 時為殘局。如圖 1-10、1-11、1-12，分別是車的開局、中局、殘局子力位置表，可以看出同樣子力且同樣位置，在不同階段的分數有所不同。

子力位置表的參數很多，未分階段的子力位置表，就有 7 種子力，90 個位置，共 630 個參數，分三階段的子力位置表，就有 1890 個參數，階段分的越多，參數也就越多。

0	5	2	1	0	1	2	5	0
-1	-2	0	2	0	2	0	-2	-1
0	-1	2	-2	0	-2	2	-1	0
2	8	11	7	2	7	11	8	2
-1	-3	1	2	0	2	1	-3	-1
-4	10	9	8	-1	8	9	10	-4
1	-5	0	0	1	0	0	-5	1
-17	-9	0	-2	0	-2	0	-9	-17
-3	-13	-4	10	-3	10	-4	-13	-3
-13	49	-1	3	0	3	-1	49	-13

圖 1-10 車的開局子力位置表

0	10	15	5	0	5	15	10	0
-4	-2	2	3	0	3	2	-2	-4
-2	-2	6	-6	5	-6	6	-2	-2
12	5	17	16	13	16	17	5	12
-3	-2	-1	7	4	7	-1	-2	-3
-7	6	13	8	2	8	13	6	-7
-1	4	2	7	6	7	2	4	-1
-3	-8	-3	-4	-1	-4	-3	-8	-3
-5	-10	-6	-8	2	-8	-6	-10	-5
-5	-10	-3	-2	0	-2	-3	-10	-5

圖 1-11 車的中局子力位置表

2	8	8	4	-1	4	8	8	2
-3	-1	0	-1	0	-1	0	-1	-3
-4	-5	0	-3	3	-3	0	-5	-4
16	-1	3	10	10	10	3	-1	16
0	-4	-3	2	4	2	-3	-4	0
-3	-2	2	2	3	2	2	-2	-3
-1	0	0	1	4	1	0	0	-1
-3	-5	-4	-2	0	-2	-4	-5	-3
-3	-6	-5	-7	-1	-7	-5	-6	-3
-3	-6	-4	-3	-1	-3	-4	-6	-3

圖 1-12 車的殘局子力位置表

其他參數是指子力位置表外的其他參數，在 1.1.4 有提到幾個審局的常見因素，需要將那些因素轉換成許多參數，子力的多寡，如車 1000 分、馬 450 分…等;棋子的靈活度，如馬靈活度 5 分、車靈活度 3 分…等;棋子間的保護和威脅，如連環馬 24 分、擔子包 16 分…等;一些特殊情形，如車鎮窩心馬 128 分…等。各式各樣的情況都會有參數值要調整，若分階段參數將會更多。

由以上介紹，可以知道電腦審局需要設定大量參數的值，簡單的審局也要設定幾十、幾百個參數值，複雜的審局更要設定成千上萬個參數值。問題是大量參數的值要如何設定?

傳統的做法是找專家，透過請教象棋高手或看棋譜，將參數值一個個地人工設定。這個方法有很多的缺點:

1. 象棋高手和程式設計師往往是不同人，象棋高手不懂電腦程式，程式設計師也不精通象棋，兩者之間的溝通和對審局的理解，都有很大的隔閡，設計出來的審局就難以精準。
2. 各個參數值在不同階段會有所不同，且有大量的參數值需要設定，以人工方式來設定非常容易有疏漏。
3. 審局的法則非常多且互相影響，要轉換成程式的規則和分數十分不易，往往需要一修再修，每次的修改都要花費大量時間和人力。因此，傳統的做法費時費工且難以精準。

為了解決這個問題，鄭明政[16]提出以統計的方法來解決，用程式分析棋譜的方式建立子力位置表，這篇論文探討如何從象棋高手所下的棋局譜中，找出在開局、中局、殘局三種不同階段的下棋規律，並將這些統計資料轉換成子力位置表分數，改良了傳統人工修改子力位置表的方法。不過這個方法還是有統計值轉換成分數值的問題，這個轉換並不一定精準，這個子力位置表也不適用於各種版本的電腦象棋程式。

除了統計的方法，王驕[17]等人提出以基因演算法的方法來解決，用基因演算法做參數優化，透過離線自我學習訓練得到最佳參數組合。不過這篇文章沒有提到參數依照棋局分不同階段，也沒提到參加基因演化的起始參數組合該如何設定。

本研究的主要目的是，設計一套新的參數自動優化系統，結合統計和基因演算法，並將參數依棋局分階段，使電腦象棋的審局更加精準，且當電腦象棋程式改版，也能利用這套系統，快速精確地找到適合該版本的參數組合。

我們應用此系統於本實驗室研發的「棋謀」程式上，改進棋謀裡

審局部分的參數數值，使該程式棋力增強。棋謀在 2009 年的第十四屆國際電腦奧林匹亞(Computer Olympiad)中，獲得電腦象棋競賽之銅牌。

1.3 論文大綱

本篇論文在第二章介紹本文的研究方法和系統架構，分基因演算法系統和棋譜管理系統來說明，描述整個系統如何設計，以及整個系統運作的流程。在第三章為研究結果和實驗，說明整個系統對電腦象棋程式棋力的改進，以及此系統參數設定對參數優化速度的影響。最後第四章提出我們的結論與未來展望。

第二章、研究方法及系統架構

本章第一節介紹整個系統架構，簡略介紹各個部分的功用。第二節介紹基因演算法系統，說明參數如何基因配對以及如何測試測資及過濾。第三節介紹棋譜管理系統，說明如何產生第一代起始參數組合以及如何產生測資。

2.1 系統架構

圖 2-1 是本系統的架構圖，本系統程式有基因演算法(Genetic Algorithm)系統、棋譜管理系統、開局庫系統，外部程式是棋謀，檔案有大量棋譜、測資(Benchmark)和演化的各種參數組合。

整個系統是基因演算法系統中的產生模組讀入需要演化的起始參數組合，變化後產生各種參數組合，把這些參數組合經過過濾模組，利用棋謀程式跑測資，留下結果較好的幾個參數組合，這些參數組合並且成為下一代演化的起始參數組合，繼續演化，不斷地尋找更好的參數組合。而棋譜管理系統的作用是讀入大量棋譜，利用程式分析棋譜，從統計的方式來產生第一代起始參數組合及測資。至於開局庫系統，是因為測資並不容易直接從棋譜管理系統得到，必須先建立好開局庫系統，再從開局庫系統中擷取資料當測資。詳細的介紹會在之後的小節一一解釋。

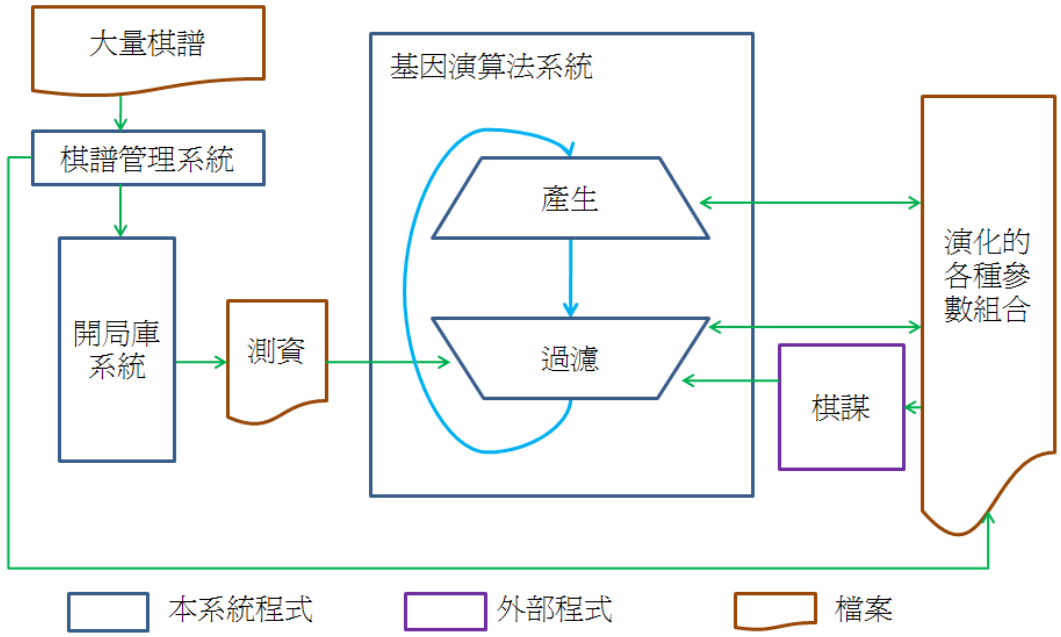


圖 2-1 系統架構圖

2.2 基因演算法系統

2.2.1 先簡介一下整個演化流程，2.2.2 介紹產生模組，2.2.3 介紹過濾模組。

2.2.1 演化流程

我們先定義此基因演算法所要演化的所有參數，我們定義 P 為參數組合，由以下參數所組成，包含子力位置表和其他參數。子力位置表有將士象車馬包卒 7 種子力表，依棋局階段又各有不同表，棋謀程式第 85 版把棋局分成 9 個階段，所以共有 5670(7*90*9)個參數。其他參數，如子力基本分、子力靈活度分、連環馬分、窩心馬分…等，有些分又依棋局會有不同分數，所以其他參數部分棋謀程式第 85 版有 123 個參數。

我們令第 i 代演化時，$P_{m,i}$ 就是第 i 代的第 m 個參數組合，n_b 是每一代開始時的參數組合數，n_g 是每一代產生後的參數組合數，

n_f 是每一代過濾後的參數組合數。在第 i 代時，n_b 個起始的參數組合經過產生模組後，產生出 n_g 個不同的參數組合，將這些參數組合經過過濾模組測試測資並過濾後，留下較好的 n_f 個參數組合，上述流程可以看圖 2-2。

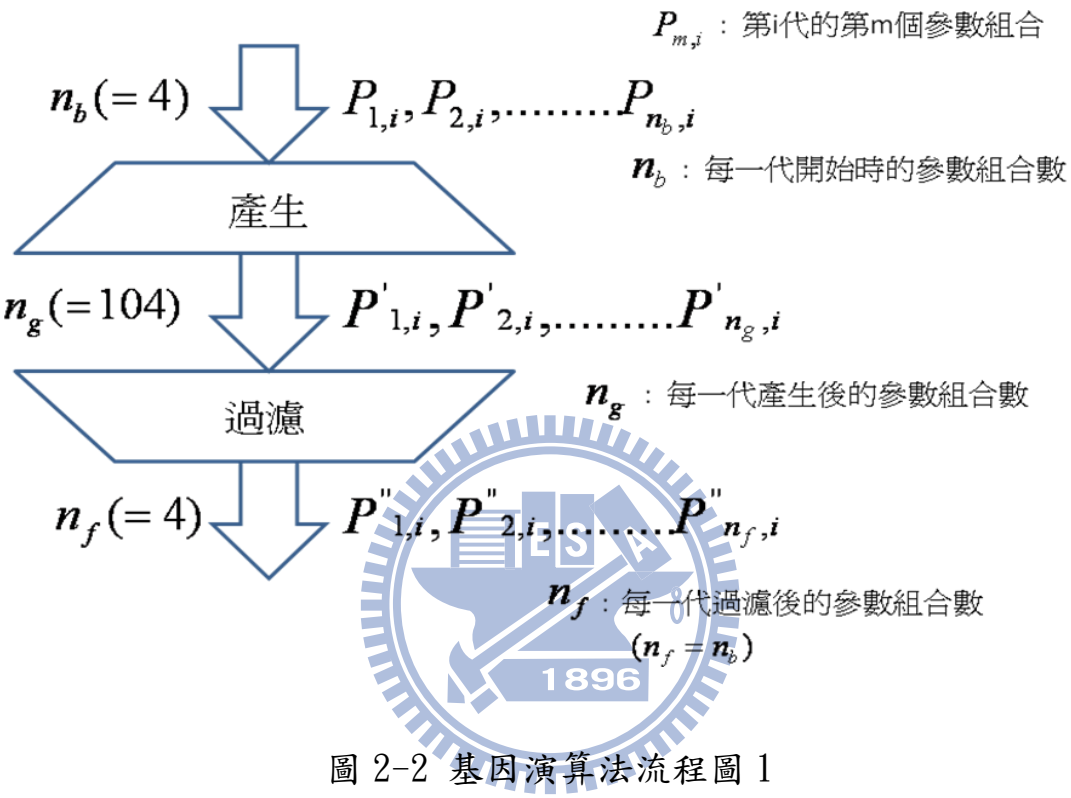


圖 2-2 基因演算法流程圖 1

演化是一代接著一代的，所以每一代過濾後留下的較好 n_f 個參數組合，成為下一代的起始參數組合，也因此 $n_f = n_b$ ，如圖 2-3 所示，第 i+1 代的起始參數等於第 i 代留下的冠軍參數。

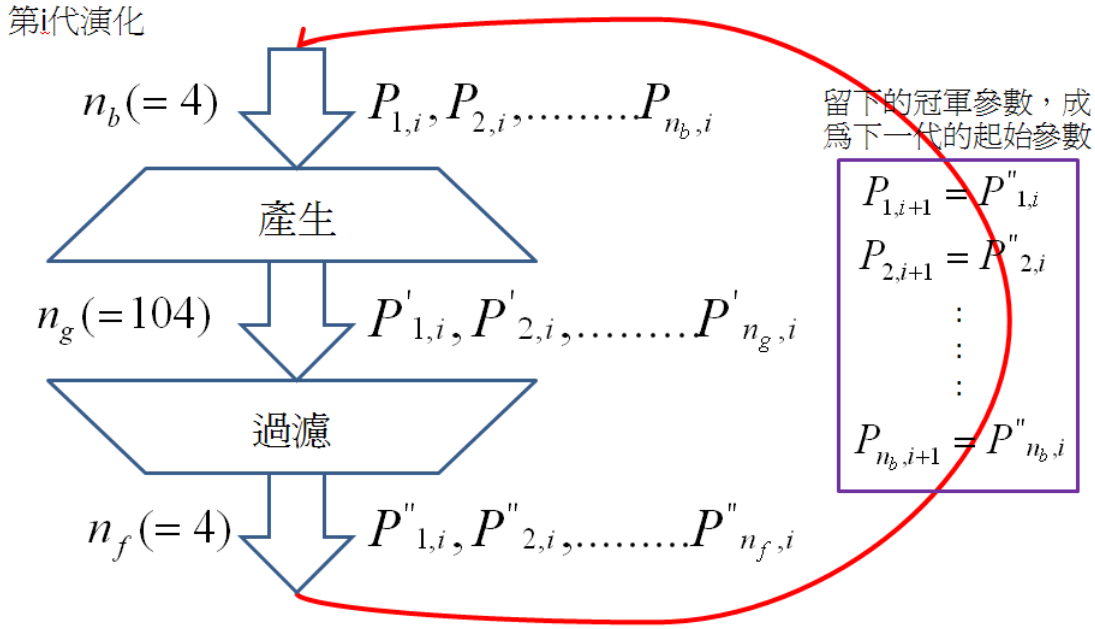


圖 2-3 基因演算法流程圖 2

2.2.2 產生

此部分給它起始的 n_b 個父代評估函數，產生成 n_g 個不同的子代，其中 $n_g > n_b$ 。

由於產生新子代的方式有很多，我們把它分成參數變化方式、參數量、變化量三大項來說明。

參數變化方式有兩種，交配(Crossover)和突變(Mutation)。交配是選兩個父代產生新的子代，以圖 2-4 為例子箭頭左邊兩個是父代參數組合，箭頭右邊是產生的子代參數組合，子代的 V1 和 V2 從第二個父代來，子代的 V3 從第一個父代來。突變是選一個父代變異出新的子代，以圖 2-5 為例，子代的 V1 和 V3 自己變異，V2 則保持不變。

在基因演算法中，交配的機率較高，突變的機率較低，我們先假設交配的機率是 70%，突變的機率是 30%，則假設我們要新產生出 100 個參數組合，約有 70 個是交配產生的，約有 30 個是突變產生的。這

個交配和突變的機率要設定多少較好，在第三章將會有實驗說明和比較。

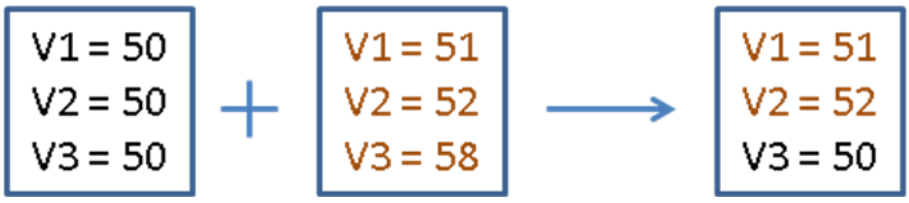


圖 2-4 交配(Crossover)

圖 2-5 突變(Mutation)

參數組合中有相當多的參數，一次要修改多少個參數呢?所以我們把要修改的參數量粗略分成兩大類，SP(Smaller Parameters)一次修改較少個參數，LP(Larger Parameters)一次修改較多個參數。以圖 2-6 為例，Initial 中，參數組合有 V1 到 V8 共 8 個參數，修改的參數量是 SP，則有 V2 和 V7 共 2 個參數改變，參數量較少;修改的參數量是 LP，則有 V1、V2、V4、V5、V6、V7 共 6 個參數改變，參數量較多。

當決定好要修改某個參數時，該參數的調整範圍要多少呢?所以我們把要修改參數的調整範圍粗略分成兩大類，SV(Smaller Variation)參數調整範圍較小和 LV(Larger Variation)參數調整範圍較大。以圖 2-7 為例，Initial 中，我們欲修改的參數是 V1，數值是 50，參數調整為 SV，則可能調整到 48 跟 52 之間的範圍，變化量較小;參數調整為 LV，則可能調整到 25 跟 75 之間的範圍，變化量較大。

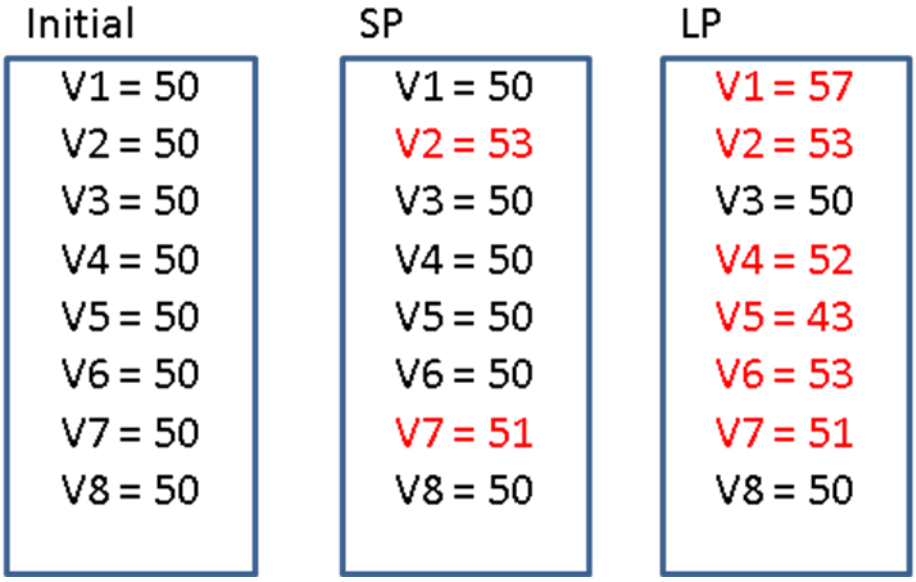


圖 2-6 SP 和 LP

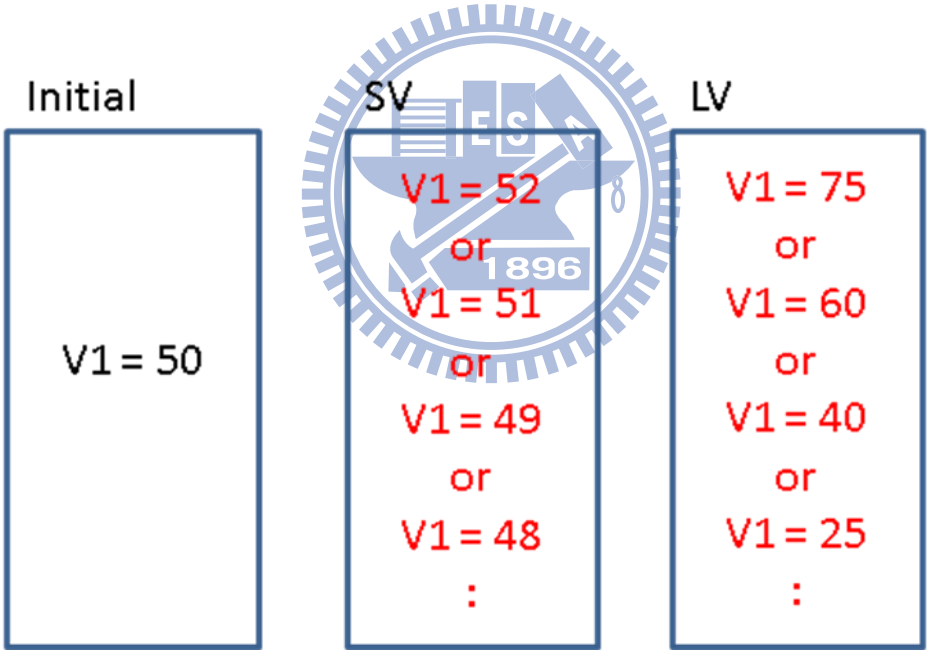


圖 2-7 SV 和 LV

根據參數量和變化量的選擇，我們現在有 4 種參數的修改方式:

1. SPSV(Smaller-Parameters-Smaller-Variation)
2. SPLV(Smaller-Parameters-Larger-Variation)
3. LPSV(Larger-Parameters-Smaller-Variation)
4. LPLV(Larger-Parameters-Larger-Variation)

參數組合中的大量參數，不論是交配還是突變，都要決定要修改多少參數和每個參數修改多少，到底參數量和變化量要如何選擇呢?我們首先考慮到，不適合一次改變太多，因為這樣整個參數組合變化太大，失去遺傳的作用，好的父代的基因留下的太少，不容易找到更好的參數組合。也不適合一次改變太少，因為這樣會使整個演化的搜索過程緩慢，也可能僅找到局部最優就找不到更好的了，整個演化會很久。

根據以上邏輯，我們選擇用 SPLV 和 LPSV。SPLV 是一次修改較少個參數，參數調整範圍較大，如圖 2-8 為例，此例子暫時不考慮交配還是突變，箭頭左邊是父代參數組合，右邊是可能產生的子代參數組合，可以看到右邊的三個參數組合，改變的參數不多，都約 2、3 個，而參數調整的範圍很大，從 25 到 75 都有;LPSV 是一次修改較多參數，參數調整範圍較小，如圖 2-9 為例，可以看到右邊的三個參數組合，改變的參數很多，都約 5.6 個，而參數調整的範圍很小，只看到從 46 到 53。

SPLV 和 LPSV 是否真的會比 SPSV 和 LPLV 好，我們將會在第三章做實驗，來驗證這四種方式的好壞。

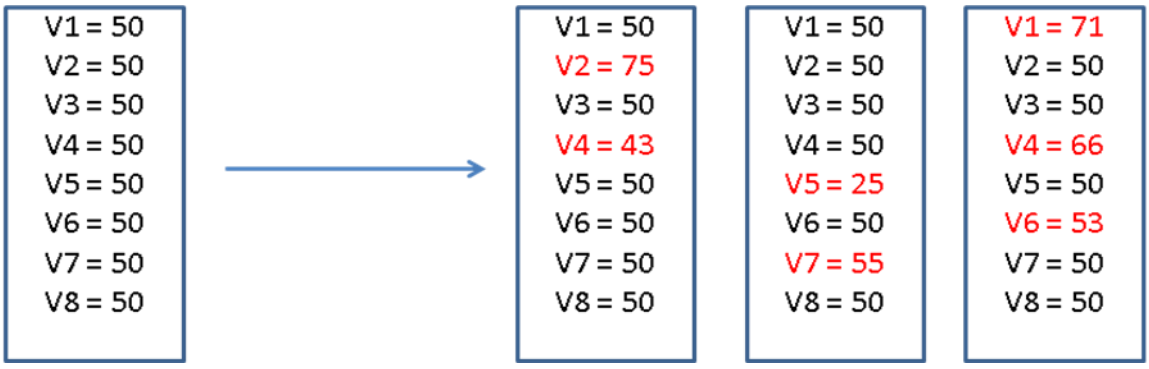


圖 2-8 SPLV

圖 2-9 LPSV

現在來分別介紹交配和突變的實際情形。首先來講交配，以圖 2-10 為例子，A 是欲改變的參數組合，B 是要往哪個目標參數組合改變，LPSV 的 A 中參數有 50%的機率會往 B 的方向調整，SPLV 則是 20%的機率調整，不過圖 2-10 中的調整是完全調整成 B 的參數值，事實上並不一定是這樣，因為我們不希望參數值只有 A 和 B 兩種，要各種可能的參數值都有，即使可能性很低還是都要有機會調整到，這樣才能做到微調的部分。

參數調整範圍以圖 2-11 為例子，A 值、B 值和改變後的值成線性關係，A 的值欲往 B 方向調整，最正常的調整就是直接變成 B 的值，不過我們也有可能調整到 A 和 B 之間的值，甚至也有可能會往 B 右邊改變或直接反向往 A 左邊改變，這種情形比較奇怪，算是交配中的突變，由於值調整的範圍的合理性不同，我們對每個調整範圍都給一個機率，在 B 附近和 A 到 B 之間的調整範圍的機率較高，其他調整範圍的機率較低。而 SPLV 的調整範圍較大，LPSV 的範圍較小。

至於突變，以圖 2-12 為例子，A 是欲調整的參數組合，LPSV 的 A 中參數有 50%的機率會自我變異，SPLV 則是 20%的機率自我變異。自我變異的方式以圖 2-13 為例子，A 值和改變後的值成倍數關係，比較正常的變異是放大 1.1 倍、縮小 0.9 倍之類，然而也有可能直接放大 2 倍、縮小 0.5 倍之類，我們也是對各種調整範圍給不同的機率，

在 A 附近的機率較高，遠離 A 的機率較低。而 SPLV 的調整範圍較大，LPSV 的範圍較小。

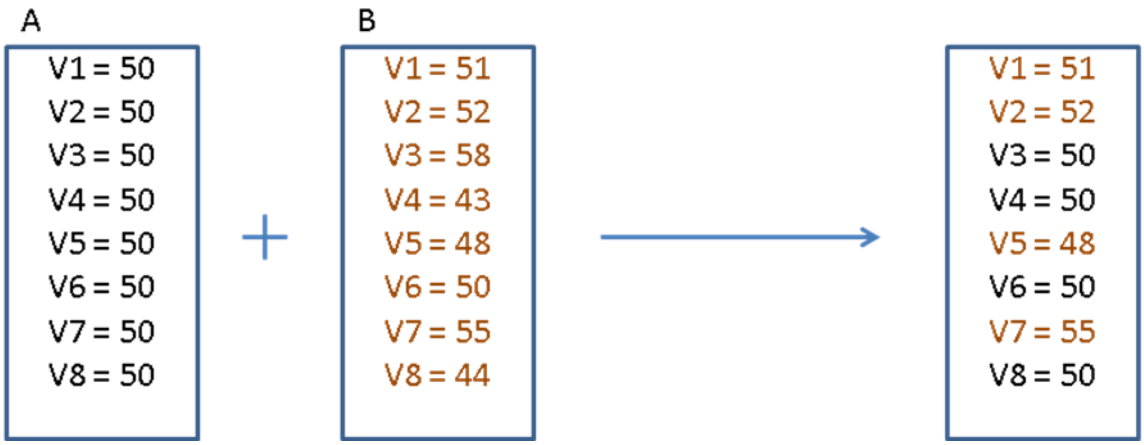


圖 2-10 交配-參數量

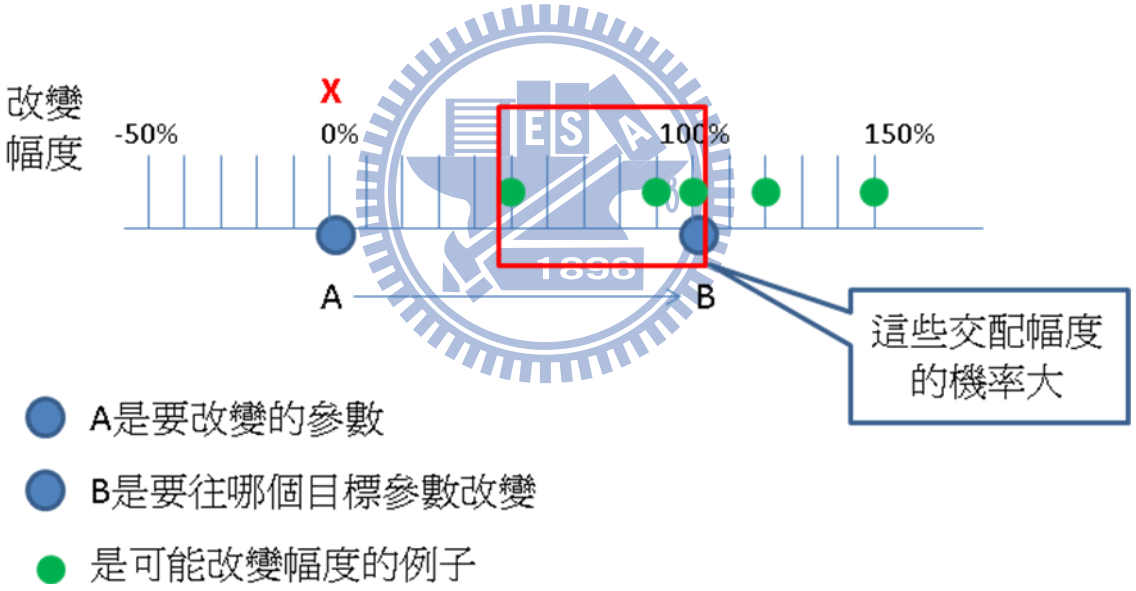


圖 2-11 交配-變化量

圖 2-12 突變-參數量

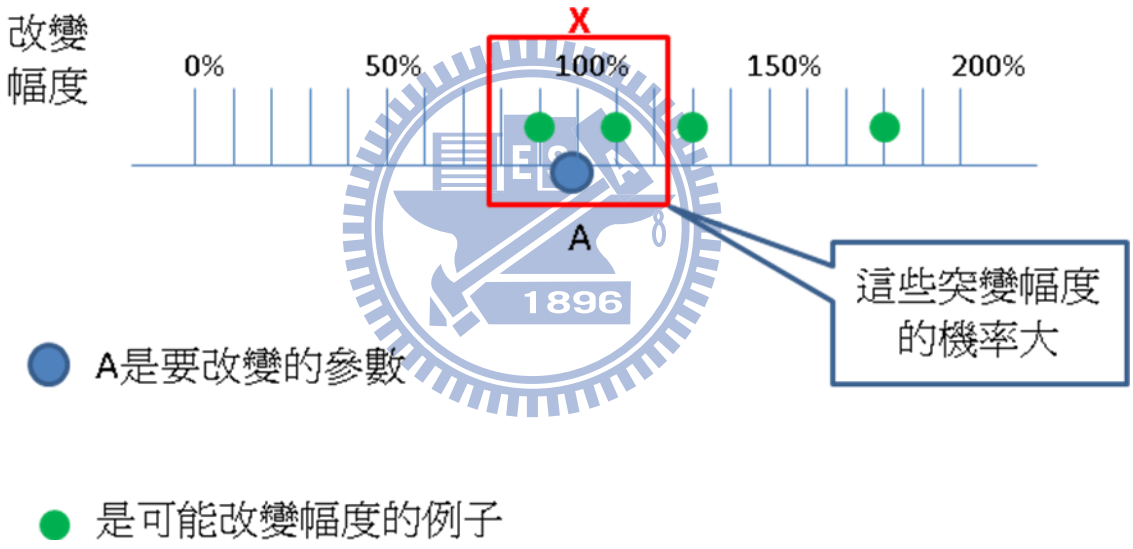


圖 2-13 突變-變化量

先前有提到會一次修改多個參數，在一般情況下，利用機率隨機挑選參數做改變就可以了，然而我們發現子力位置表的參數間其實有些關係，隨機挑選的方式可以做些改變。我們看圖 2-14，車的中局子力位置表，車在上方數來第 4 列、第 6 列，或是從左右數來第 3 行、第 4 行，車的分數相對其他位置較高，這些位置都是人類棋手認為的好位置，那些好位置如果要加分，我們會認為那些區塊分數設定應該同時增加或減少，而不是在同一區塊會有分數加分有分數減分。同樣

地，馬和卒在接近對方帥，容易攻擊到的位置也應該比較高分，棋盤邊緣位置會比較低分。因此，我們在選擇子力位置表的參數時，並不使用隨機挑幾個的方式，而想要用隨機挑選區塊的方式。挑選區塊有個地方要注意，就是雖然同一個區塊，分數的增減要同時，但是區塊應該呈現交錯的情形，也就是說每個參數的分數增減還是會跟別的參數不同，不會永遠跟著某個參數變動。

圖 2-14 子力位置表

我們選擇區塊的方式，是先隨機選一點，然後把該點的可走範圍也包含進去，最後並做左右對稱處理，那些位置就形成一個區塊，如圖 2-15，有著色的地方就是一個區塊。這個選取區塊的方法是否比隨機選幾個參數好，將在第三章做實驗比較。

圖 2-15 子力位置表選取方式

選取區塊的方式，會因為兵種不同，而選取改變的參數量不同，如車改了 27 個參數，卒只改了 8 個參數，這種改變量的不同，也有待研究和實驗。

2.2.3 過濾

此部分是 n_g 個產生後的參數組合，全都都去測試測資，依照測試結果過濾後留下 n_f 個較好的參數組合，其中 $n_g > n_f$。

理論上，我們的目標是增加棋謀程式的強度，較好的過濾方式是讓棋謀程式交互對戰，留下比賽結果較好的版本，然而這個方法非常地花時間。假設我們每次有 100 個要對戰的參賽者，每個參賽者都要相互對戰，由於公平性的問題，每個參賽者必須同一開局紅黑各持一場，又由於一場開局定勝負不太準確，所以設定隨機選 10 種開局來比賽，這樣就總共要比 99000 場($C_2^{100} * 2 * 10$)，若一場對戰花 1 小時，則演化一代全部比完要 99000 小時，等於 4125 天，等於 11.3 年，更何況要演化很多代，這種方法太沒效率。即使是我們把每次要對戰的參賽者縮減到 20 個，所選開局僅選 4 個，一場對戰只花 30 分鐘，演化一代還是要 760 小時，約 31.7 天，時間依舊很久，而且參賽者減少，會導致需要演化更多代，所選開局減少及對戰時間縮短，也使對戰結果跟實際強度的準確性下降。為了解決時間的問題，可以改變對戰方式，或是利用平行化加速，不過不論怎樣改進，演化時間還是偏久，效果有限，需要另外找尋大量減少時間的方法。

過濾的方法，我們不採取程式對戰的過濾方式，改採取測試測資，若依後面會提到的四千多盤測資盤面和測試方法，一個參賽者全部測完只要約 10 分鐘，100 個參賽者演化一代只要約 1000 分鐘，約等於 16 小時，雖然說測試測資的結果和實際程式強度會有較大的落差，不過時間上大幅減少，比交互對戰有效率很多。而測資和實際程式強度的關係，可以獨立於基因演算法系統做加強，針對測資檔案做改進，找尋更加優良準確的測資，並不需要改變本系統，就可以改進本系統參數優化的結果。

我們測試測資的方法是，用棋謀程式思考 500 萬個節點，得出一個最佳步，和測資內存的最佳步做比對，我們把結果分成三種，分別是必勝、猜對、錯誤。必勝是指在 500 萬個節點內就已經算出必勝步了;猜對是指思考 500 萬個節點找不出必勝步，而得出的最佳步和測資內存的棋步是相同的;錯誤則是思考後的最佳步跟測資內存的棋步不同。然後我們把所有測試測資的參賽者結果做排序，排序的方式是先依照錯誤數排序，錯誤數越少越好，若錯誤數相同，則依猜對數排序，猜對數越少越好，也就是必勝數越多越好，若測試結果都一樣，則所用的節點數越少越好。

我們的測資有三個，如表 2-16 所示，第一個是「象棋路邊攤」軟體中的棋局盤面，都是殘局的連將殺棋局，共有 4000 盤，由於大部分能快速找到必勝，全部測完只要約 210 秒就好;第二個是「棄子攻殺法」[30]棋譜中的棋局，屬於中殘階段，這是人工觀察棋譜所尋找出來的盤面，我們挑選了 95 盤做為測資，全部測完約 200 秒;第三個是實戰譜統計產生的棋局盤面，屬於開局中局階段，我們挑了 40 盤做測資，全部測完約 165 秒，如何利用統計挑選測資，我們將會在下一節詳細說明。一個參賽者把這三個測資測試完總共約花 9 到 10 分鐘。補充一下，這裡的棋局階段並沒有明確的定義範圍，是 1.1.3 中介紹的棋局階段。

	棋譜來源	**棋局階段**	**盤面數**	**測試時間**
1	象棋路邊攤	殘局	4000 盤	約 210 秒
2	棄子攻殺法	中局、殘局	95 盤	約 200 秒
3	實戰譜統計產生	開局、中局	40 盤	約 165 秒

表 2-16 三個測資

所以整個演化流程的每一代，都是不斷地變化產生參賽者，每個參賽者去測試測資，過濾出最好的幾個參賽者，進行下一代的演化，不斷循環。

2.3 棋譜管理系統

2.3.1 先介紹一下大量棋譜的來源和分類，2.3.2 介紹如何利用棋譜管理系統產生第一代起始參數組合，2.3.3 介紹如何建立利用棋譜管理系統新開局庫，2.3.4 介紹如何產生測資，2.3.5 介紹如何產生程式對戰起始開局盤面。

2.3.1 大量棋譜

我們的大量棋譜主要有兩個來源，一個是象棋大師實戰譜，一個是遊戲網站的玩家實戰譜。象棋大師實戰譜，我們從 1990 年到 2008 年，取得約 20000 局象棋大師的實戰譜。遊戲網站的玩家實戰譜，我們從「群想」網站取得十幾萬的玩家對局譜。象棋大師實戰譜，因為是象棋大師在比賽時的棋譜，我們相信象棋大師在比賽時不會亂下，所以我們信任這類棋譜。至於遊戲網站的玩家實戰譜，因為玩家的棋力程度參差不齊，所以我們必須做一些挑選，我們信任群想網站上積分 2000 分以上的玩家所下的棋步，不過信任度還是比象棋大師實戰譜低。

棋譜管理系統讀入大量的棋譜，將各種棋譜做格式轉換、整理、統計、分類…等，也可以將棋譜建立成開局庫。

2.3.2 產生第一代起始參數組合

我們在上一節的演化流程中有提到，演化的每代起始參數組合為上一代的冠軍參數組合，不過沒有提到第一代的起始參數組合。第一代的起始參數組合，除了可以直接拿原來程式的參數值，也可以用統計的方式產生，例如子力位置表用鄭明政[16]的方法產生，子力靈活度分、攻擊分…等，也依棋局階段做統計並轉換成分數。

2.3.3 新開局庫

我們的棋譜管理系統，可以建立出一個開局庫，這開局庫是傳統的做法，樹狀結構並利用雜湊(hash)的技巧建立，棋譜來源是象棋大師實戰譜，資料庫包含兩大資料表 positions 和 moves，每個 positions 有 id、盤面…等資訊，每個 moves 有 id、move、勝敗和次數…等資訊，對每一個盤面我們都可以查詢棋譜所出現的棋步，並知道走該步的勝敗機率等資訊。

不過這個傳統的開局庫有一些缺點，我們想要改進這個開局庫，所以我們建立一個開局庫系統，專門處理開局庫相關資料，並建立新的資料庫。為了簡化敘述，我們把傳統方法建立起來的資料庫稱為原資料庫，新方法改進後的資料庫稱為新資料庫。

原資料庫的樹狀結構，在某一盤面會有幾個大師譜的棋步可以選擇，然而有些棋譜步可信度低(例如棋譜數量少於 5 譜)或是棋步太差(例如勝率 10%敗率 70%)，導致我們並不想真的照該棋譜步走。如圖 2-17 所示，圓形和方形都是大師譜中的棋譜步，然而方形的我們不想照著棋譜步走，所以我們象棋程式所採用的開局庫只有圓形部分。

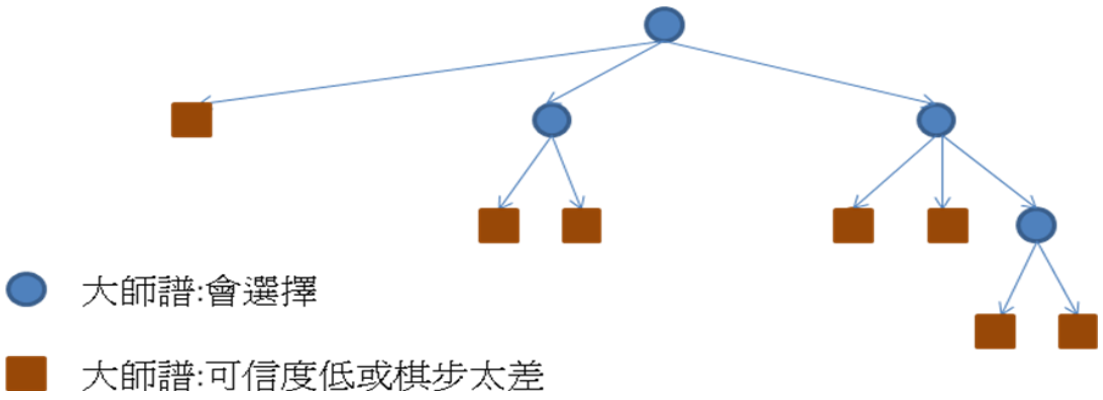


圖 2-17 開局庫-棋譜走法

在實戰中，某盤面情況下，若底下沒有棋譜步或是沒有想走的棋

譜步時，我們程式就會脫譜，必須進行思考。這個思考是在實戰中，不能花費太多時間，否則會壓縮到以後的思考時間。然而既然會拖譜的盤面是固定的，在脫譜的盤面必定就要程式思考，不如我們在平常的時候，就先把這些脫譜時的盤面先行思考出最佳步，而且這時候程式可以思考比較久，沒有時間壓力。如圖 2-18 所示，在某些盤面，底下並沒有會選擇的棋步時，會利用程式思考新產生一步，就是圖中三角形的搜尋步。在實戰中，我們通常會思考約 30 秒，而這裡的搜尋步，我們的思考，會搜尋完超過 60 秒的那一層，這個方式是因為是離線思考，我們會完整地把那一層給搜尋完，不浪費任何時間在某層中途停止。

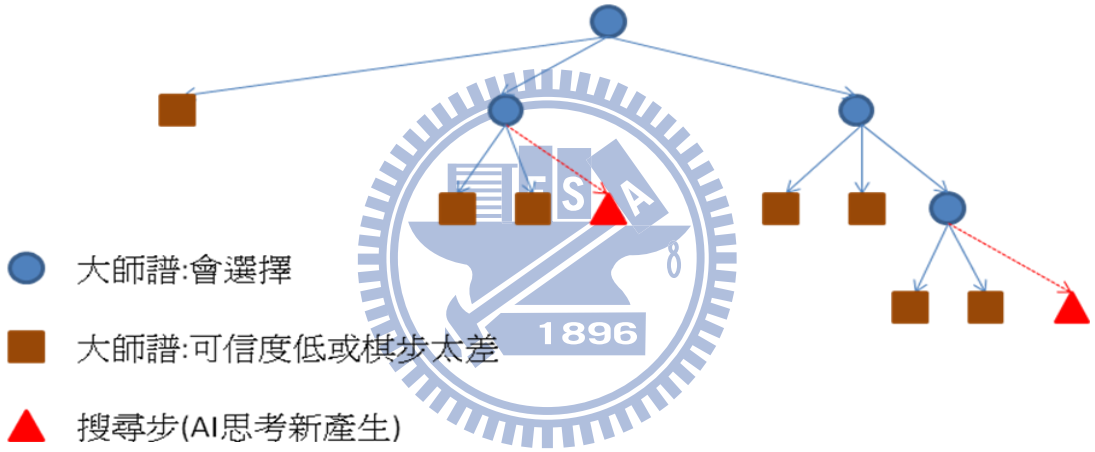


圖 2-18 開局庫-搜尋步

若搜尋步跟棋譜步相同時，整個樹會繼續往下走，若不同則就不再往下走，這個是因為即使我們之前看到某些棋譜步並不想走，但是那可能是該情況下逼不得已的走法，當我們的搜尋步和棋譜步相同時，代表該棋譜步是必要的，我們也沒有其他走法，我們就轉為相信該棋譜步。如圖 2-19 所示，中間的三角形因為沒 match 到棋譜步，所以底下並沒有分支，而右邊的菱形剛好跟棋譜步相同，則底下會展開分支繼續往下走。

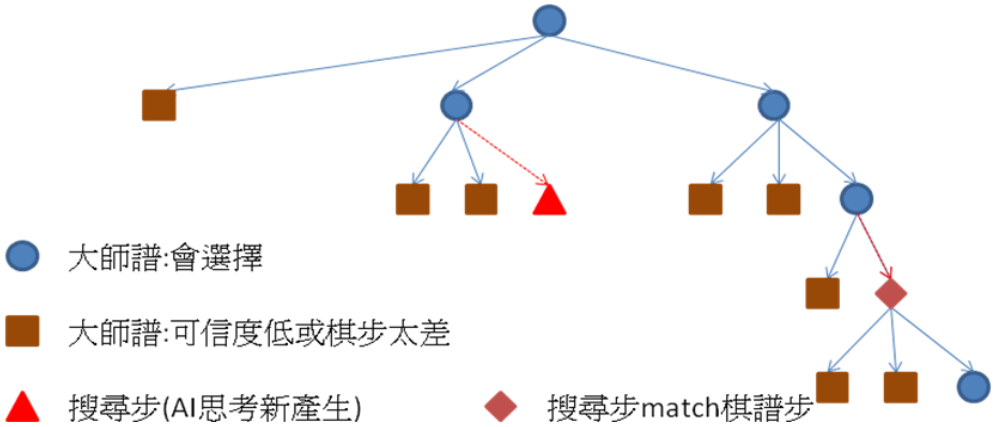


圖 2-19 開局庫-搜尋步 match 棋譜步

傳統的開局庫，持黑持紅是使用相同的開局庫，然而我們的新開局庫分成紅黑兩個開局庫，原因如下。第一個原因，輪到對方時，對方開局庫跟我方不同，不一定照著我們的開局庫走，所以當對方脫離我方開局庫時，我方也就跟著脫譜，不知該走什麼步。新開局庫的改良方式是，輪對方時，對方不只是展開棋譜步，而是展開所有合法步，如圖 2-20 所示，因此不論對方走什麼步，我方都能從新開局庫中找到該走的步。

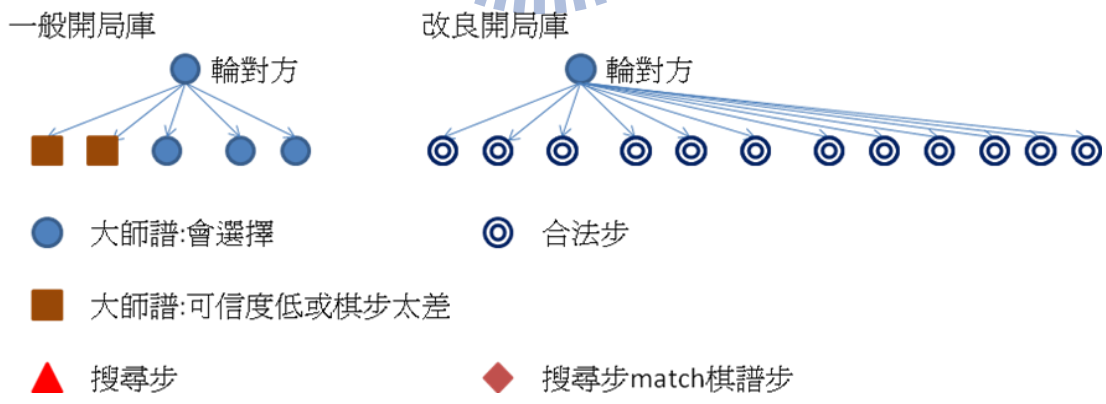


圖 2-20 開局庫-對方脫譜 1

然而所有合法步的分支實在太多，很多棋步完全沒有意義，例如起始盤面帥五進一之類，所以我們要用另一種方法找出對方可能的有效棋步。我們的方法是利用網站譜，也就是在 2.3.1 中提到的，群想

網站 2000 分以上玩家的實戰譜，將那些譜中的棋步，也加入新開局庫對方的棋步中，使對方可能走的棋步更具多樣化，且不會完全沒道理。新開局庫輪對方時就如圖 2-21 所示，會有棋譜步和網站譜步。

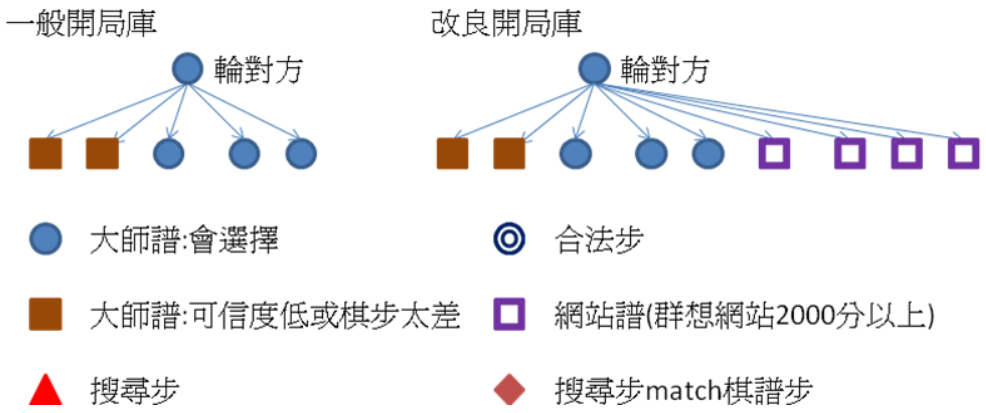


圖 2-21 開局庫-對方脫譜 2

分成紅黑兩個開局庫的第三個原因是，輪自己方時，只會選擇開局庫中的一步去走，所以開局庫的樹不會那麼大，分支只要展開一支就好。如圖 2-22 所示，傳統開局庫會把棋譜步中會選擇的棋步都展開，而新開局庫會先把那些會選擇的做優劣排序，只展開最好的那步，至於開局庫的棋步優劣怎麼選擇，一般來說主要看勝敗和機率跟數量，詳細情形我們不多做講解。

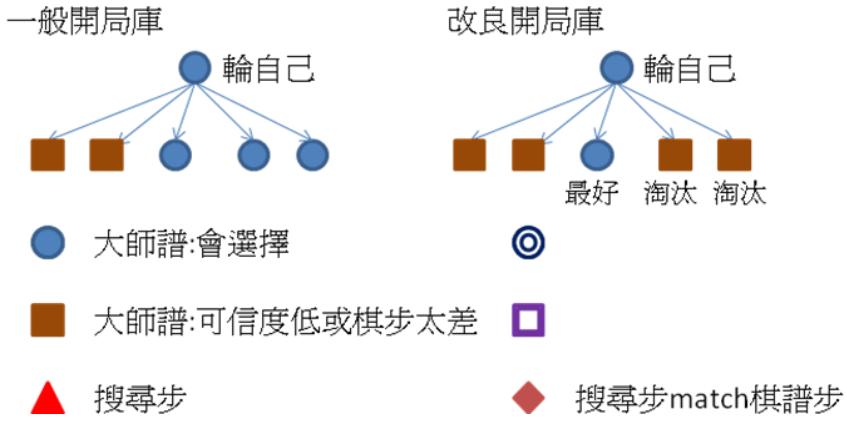


圖 2-22 開局庫-我方選擇

根據輪我方選擇和輪對方選擇的不同策略，開局庫的樹長的如圖 2-23 所示，我們把不會走到的分支去掉，實際上的樹長的如圖 2-24 所示。因為輪我方和對方的不同策略，

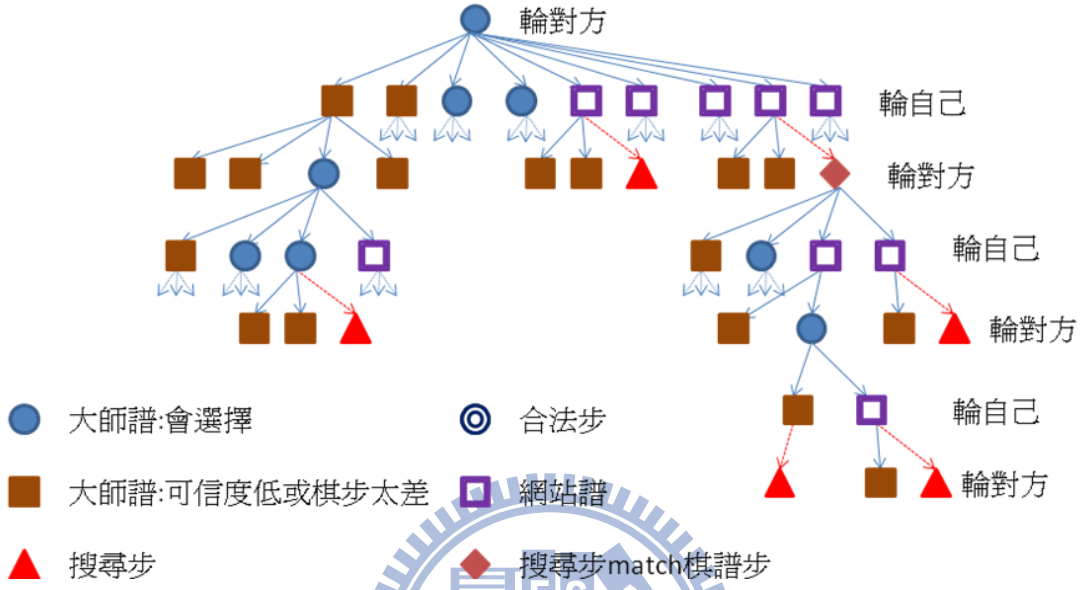


圖 2-23 開局庫-改良後開局庫 1

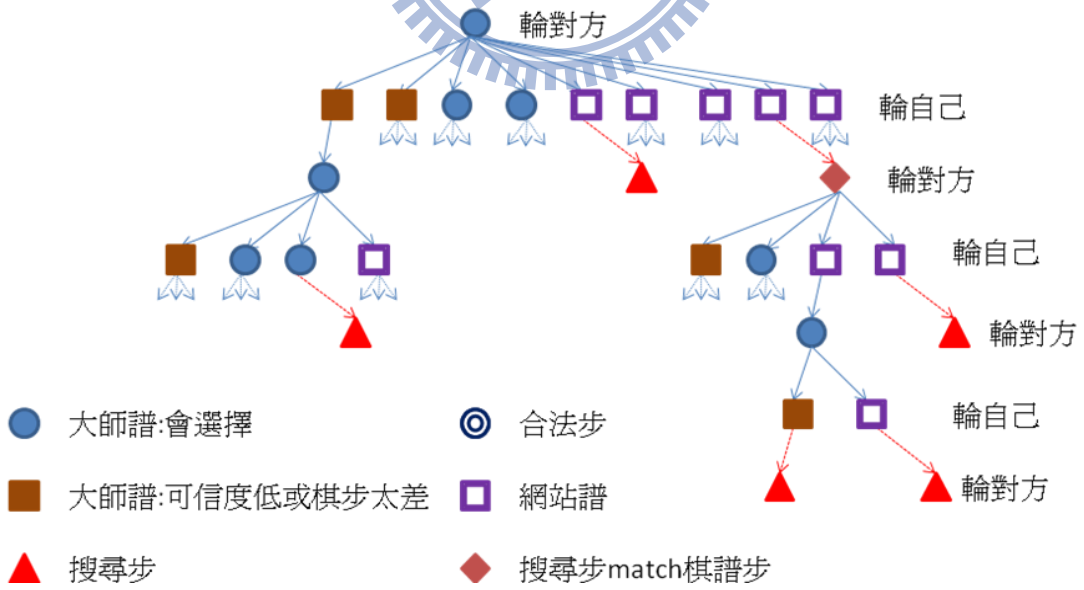


圖 2-24 開局庫-改良後開局庫 2

新開局庫會分成紅黑兩個，持紅持黑時看不同的開局庫。新開局庫的一個好處是不易脫譜，因為輪對方時有各種可能的棋步，且新開局庫會比原開局庫多了至少一層搜尋步，且該搜尋步會比實戰花更多的時間思考;另一個好處是新開局庫也不會大很多，因為雖然有 2 個開局庫，且輪對方時的分支增加，但輪到自己的分支只有一條，大大降低了整個樹的複雜度，而且新開局庫的所有搜尋步都可能在實戰中用到，而不會白白展開搜尋而永遠走不到，浪費時間去搜尋。最後為了使開局有點變化，我們讓整棵樹的前三層允許在幾條好的棋步中隨機選一條，而不是只選最好的。

另外補充一點，就是新開局庫的搜尋程式版本和實戰中的搜尋程式版本最好是同一個，否則可能會出現審局不同的衝突。我們舉一個實戰例子，棋謀(red)對 ELP(black)的一場，開局某盤面如圖 2-25，輪到紅方要走子，紅方如圖 2-26 走了俥九平四，此時還在開局庫中，用的是新開局庫中的搜尋步，當時用的是棋謀 73 版的程式搜尋，黑走了如圖 2-27 的棋步後，紅方脫譜，現場思考出俥四平六這步，實戰用的搜尋程式是 85 版，這紅方俥的行為就是很明顯兩版本的程式對於局勢的判斷不一致，導致使用開局庫和程式搜尋沒有做很好的銜接，車來回移動浪費步數，最後紅也輸棋了。

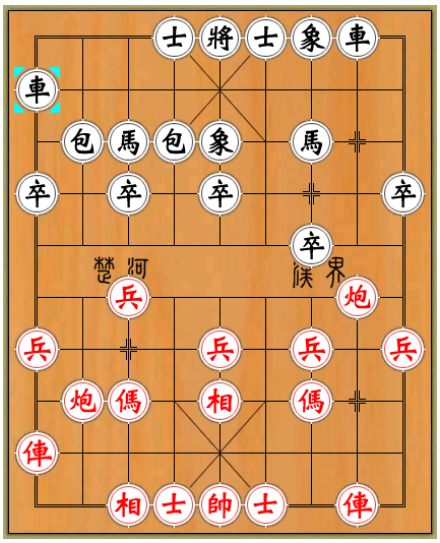


圖 2-25 開局庫補充 1

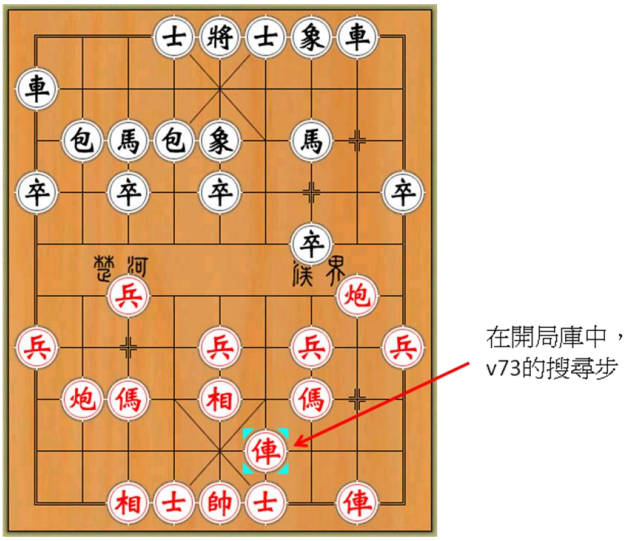


圖 2-26 開局庫補充 2

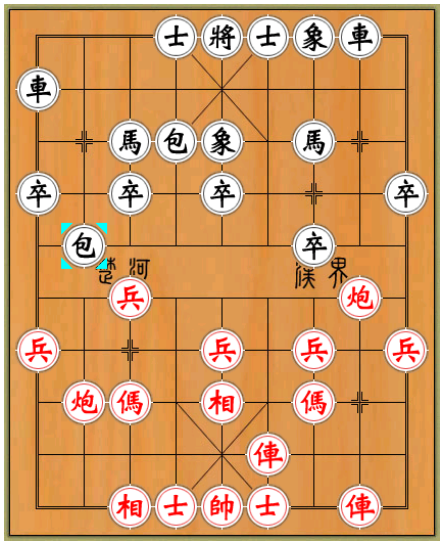


圖 2-27 開局庫補充 3

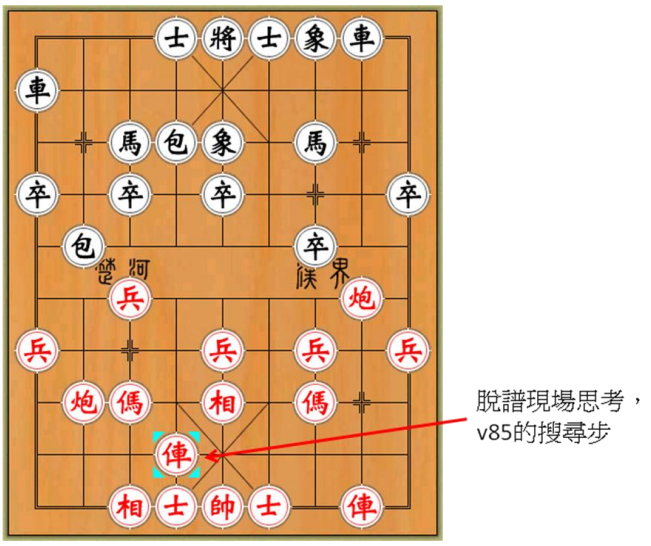


圖 2-28 開局庫補充 4

2.3.4 產生測資

之前在 2.2.3 有提到我們有三個測資，那三個測資也是三種不同的方式產生的:

1. 電子測資檔，如「象棋路邊攤」軟體，這種電子檔很少有人會釋放出來，所以難以收集，並不容易找到很多。
2. 人工看棋譜產生的，如「棄子攻殺法」棋譜，象棋的棋譜非常地多，資料龐大，然而大多是棋書，必須人工自行翻閱，並從棋譜中找尋可用的測資盤面，製作成電子檔，所以雖然資料多，但還是需要象棋專家來幫忙選擇測資，以及人工地輸入成電子測資檔，費時費工。
3. 實戰譜統計產生，我們使用的是象棋大師實戰譜，實戰譜數量龐大，我們希望利用大量棋譜，自動產生電子測資檔，而不要依靠人力製作。

實戰譜統計產生測資的方式，首先要先將實戰譜製作成開局庫，以便得到各盤面和棋步的資訊，然後對所有盤面，分析出必走步，記錄成測資。判斷必走步的方法是利用勝敗機率差，也就是該棋步的勝率減敗率，將某盤面的所有棋步勝敗機率差做排序，當第一名和第二名的勝敗機率差，相差 X%機率以上，代表第一名棋步比別步好很多，

該棋步視為必走步。以圖 2-29 所示，X 設定為 30，例子 1 的前兩名相差 50%，第一名就判斷為必走步，例子 2 的前兩名僅相差 8%，所以沒有必走步。

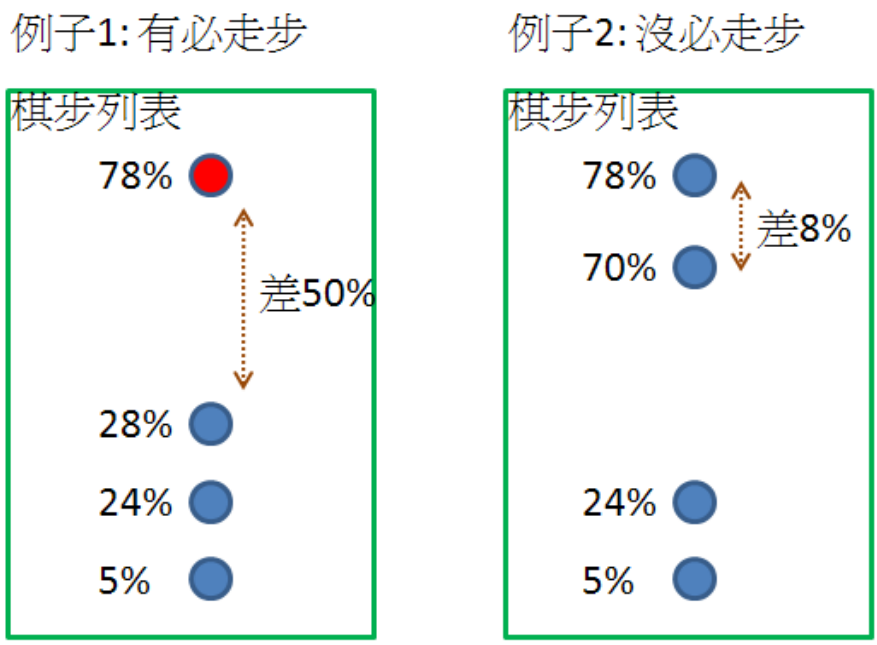


圖 2-29 判斷必走步

先前的判斷可以判斷某步比其他步好很多，不過並不能保證該步足夠好，為了讓該步是很好且比別步好很多，我們必須增加必走步的選取範圍。經過統計象棋大師實戰譜，紅方勝率 38.88%、和率 34.29%、黑方勝率 26.83%，紅黑勝率差了約 12%，也就是紅方平均勝敗機率差是 12%，黑方則是-12%。我們的選取範圍，紅方的勝敗機率差要在(X+12)%以上才選擇，黑方的勝敗機率差要在(X-12)%以上才選擇，先前提到 X 是 30，以圖 2-30 所示，紅方就是選勝敗機率差 42%以上，黑方選勝敗機率差 18%以上，若是棋譜中只有一步，並沒有第二名，則該步同樣在紅方 42%以上或黑方 18%以上就選擇，判斷它為足夠好。

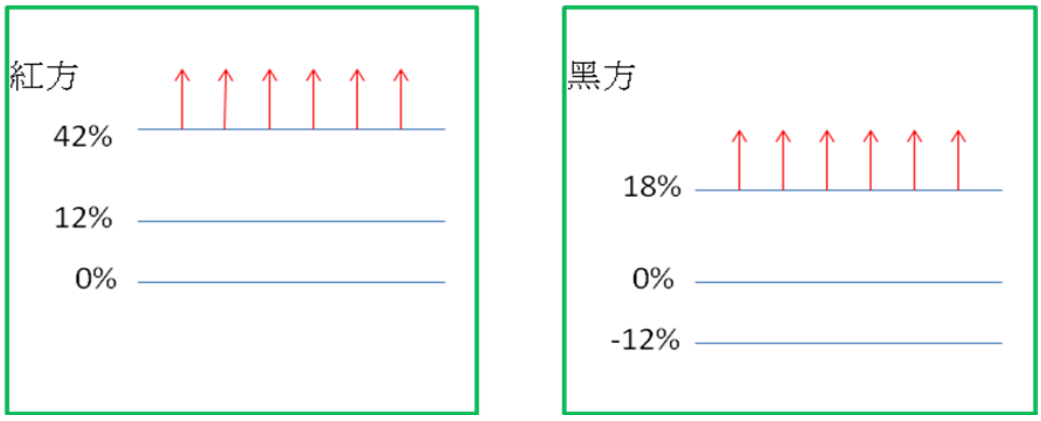


圖 2-30 必走步範圍

2.3.5 產生程式對戰起始開局盤面

雖然我們在基因演算法系統的過濾模組，並沒有使用程式對戰，但演化結果的實驗驗證還是會使用程式對戰，程式對戰需要隨機選取幾個起始開局盤面，而那起始開局盤面，我們使用程式自動挑選產生。

我們希望選取的開局盤面，是越常走到的越好，所以我們選擇開局庫中棋譜數量較多的盤面當作起始開局盤面，我們先假設這裡提到的棋譜數量是 Y，令 Y 是 500，如圖 2-31 所示，圓形和雙圓的結點形成的樹就是棋譜數 500 以上的樹，由於一個盤面的 parent 很有可能會走向該盤面，如果取整個樹當起始開局盤面，會有很多重複走到同一盤面的情況，所以我們只取樹的 leaf nodes 當起始開局盤面，圖中雙圓的節點就是起始開局盤面。我們的棋譜採用象棋大師實戰譜，假設上面提到的 Y 設為 100，則能產生 64 個開局盤面，實際上 Y 設定各值，產生的起始開局盤面數如表 2-32。

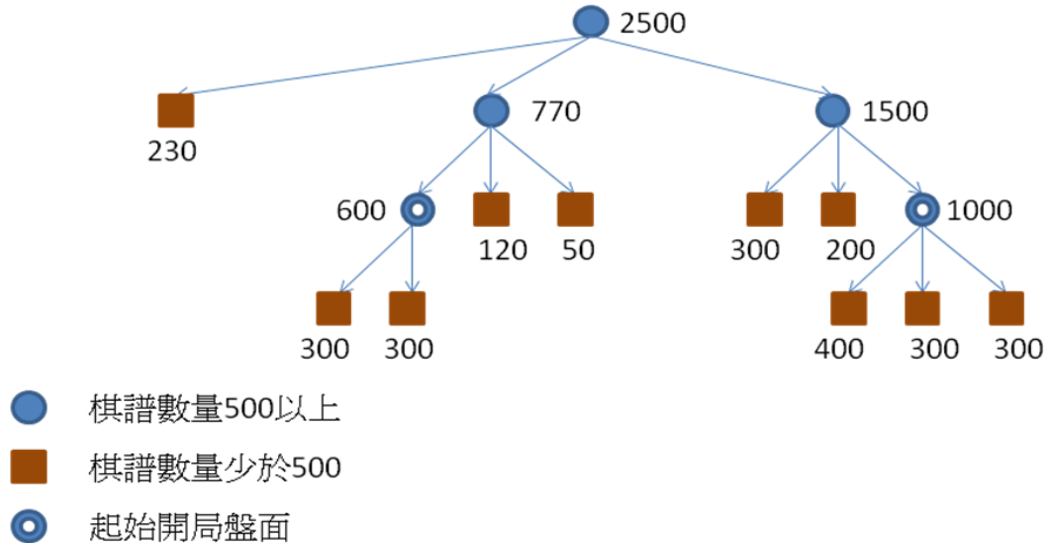


圖 2-31 必走步範圍

棋譜數量 Y	產生盤面數
5	1171
30	206
50	129
70	90
100	64
150	42
200	29
500	12
1000	5

表 2-32 產生盤面數

第三章、研究結果和實驗

本章第一節介紹實驗環境。第二節介紹各種演化參數的比較，說明演化參數該如何設定可以得到較好結果。第三節介紹實戰比較，說明參數組合，經過基因演算法系統得參數優化後，電腦象棋棋力是否增強。第四節介紹棋謀程式各個版本的演化結果。

3.1 實驗環境

本實驗所用的電腦是使用 Intel Core2 Quad CPU Q6600 2.40GHz、4GB RAM、1TB 硬碟，作業系統是 Microsoft Windows Server 2003 Standard Edition Service Pack 2。

本章 3.1 和 3.2 的實驗，所用的程式引擎是本實驗室所開發的電腦象棋程式「棋謀」第 85 版。棋謀 85 版的參數組合 5793 個參數，我們要自動調整這些參數，使程式的審局更準，進而讓程式棋力變強。

使用程式時，不使用開局庫、不使用背景思考，以免影響程式的思考，時間控制是採用限制節點數，1 秒的節點數是 1 百萬個節點，所以實驗中的限時 5 秒代表限制 5 百萬個節點，限制節點的好處是，在各台機器跑或不同時候跑，所搜尋的節點數還是一樣。

3.2 演化參數比較

基因演算法中也有許多的參數，我們對某些參數做調整，比較這些參數設定對演化效果的影響。3.2.1 做參數量和變化量的比較，3.2.2 做 n_g 值的比較，3.2.3 做 n_g 值、參數量和變化量的比較，3.2.4 做 n_b 值的比較，3.2.5 做交配率和突變率的比較，3.2.6 做子力位置表選擇方式的比較。

3.2.1 參數量和變化量的比較

我們使用棋謀 85，演化參數中的 $n_b = 4$ 、 $n_g = 104$，交配率 70%、突變率 30%，測試測資的時間是每個盤面思考 5 秒，我們總共讓它演化了 10 代，演化時間約 6 天整。

我們修改參數量和變化量來比較，我們有 4 個演化實驗，分別是 SPSV、SPLV、LPSV、LPLV 的演化。

比較結果如圖 3-3 所示，縱軸是測試測資的錯誤數，橫軸是演化代數，LPSV 是正方形、LPLV 是三角形、SPSV 是圓形、SPLV 是菱形。由實驗結果看來，演化參數設定的好壞是，LPSV > SPLV > LPLV ≈ SPSV，參數組合一次改太多或太少，演化的速度都不快。

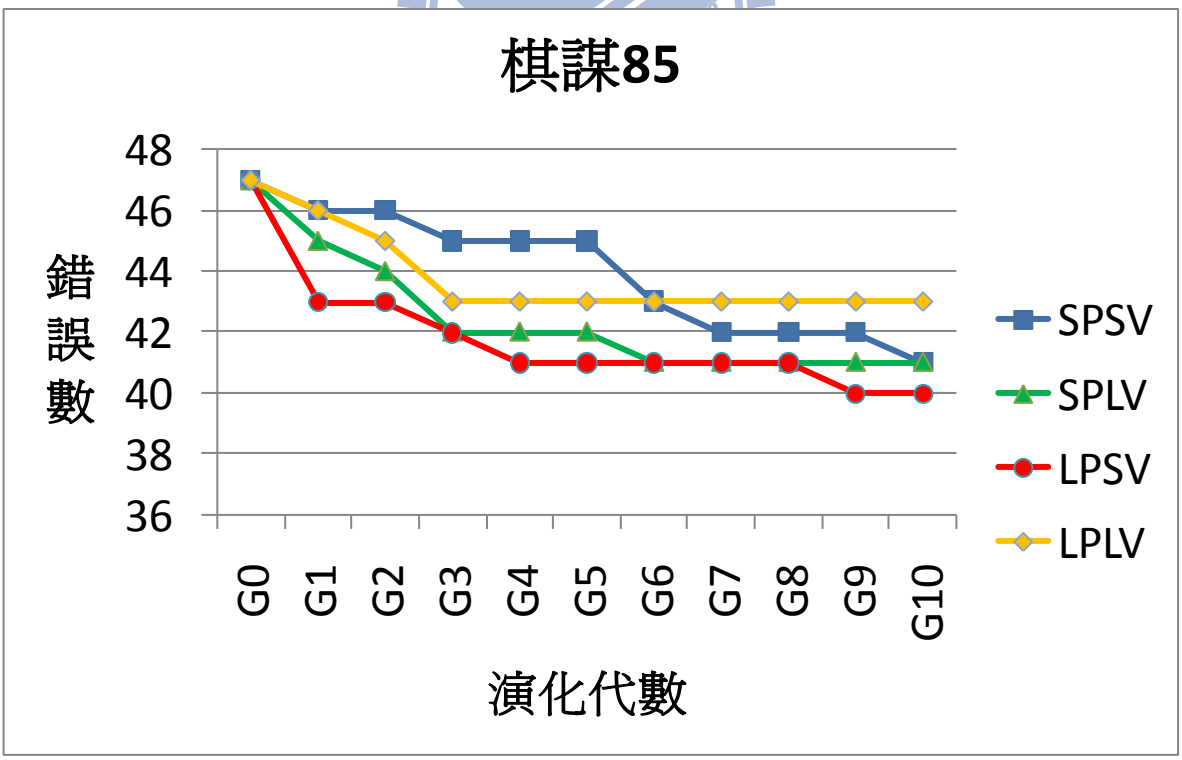


圖 3-1 參數量和變化量的比較

3.2.2 n_g 值的比較

跟 3.2.1 的實驗類似，我們使用 LPSV，修改 n_g 值，我們有 4 個演化實驗，分別是 n_g = 29（新產生 25 個參數組合）、 n_g = 34（新產生 30 個參數組合）、n_g = 54（新產生 55 個參數組合）、n_g = 104（新產生 100 個參數組合）。由於 n_g 數的不同，會影響每一代演化所需要的時間，所以我們把演化代數標準化成時間， n_g = 29 標準化成 1T 的時間單位， n_g = 34、54、104 分別標準成 1.2T、2T、4T 的時間，來看看同樣的時間之內，n_g 數值改變，對演化帶來的影響。

比較結果如圖 3-2 所示，縱軸是測試測資的錯誤數，橫軸是演化時間， n_g= 29 是菱形、 n_g= 34 是正方形、 n_g = 54 是三角形、 n_g =104 是圓形。由實驗結果看來，n_g 數的設定，對演化並沒有什麼影響。

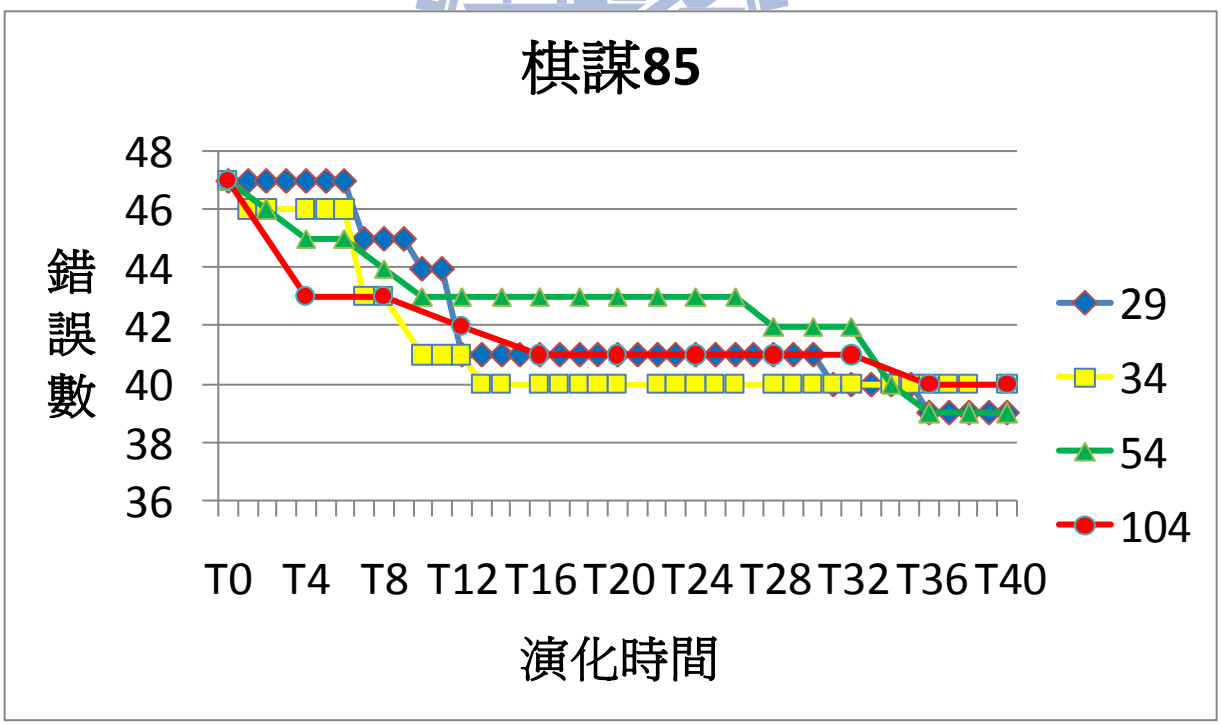


圖 3-2 n_g 值的比較

3.2.3 n_g 值、參數量和變化量的比較

跟 3.2.1 的實驗類似，不過我們再針對 SPLV 和 LPSV 做比較，對這兩種方式，用不同的 n_g 做演化。我們設定 n_g = 34、54、104，每個設定都跑 SPLV 和 LPSV 兩種。

比較結果如圖 3-3、3-4、3-5 所示，縱軸是測試測資的錯誤數，橫軸是演化時間，SPLV 是正方形、LPSV 是三角形，圖 3-3 是 n_g = 34 的演化比較、圖 3-4 是 n_g = 54 的演化比較、圖 3-5 是 n_g = 104 的演化比較。由實驗結果看來，不論 n_g 的設定為何，LPSV 的結果都的確比 SPLV 好。

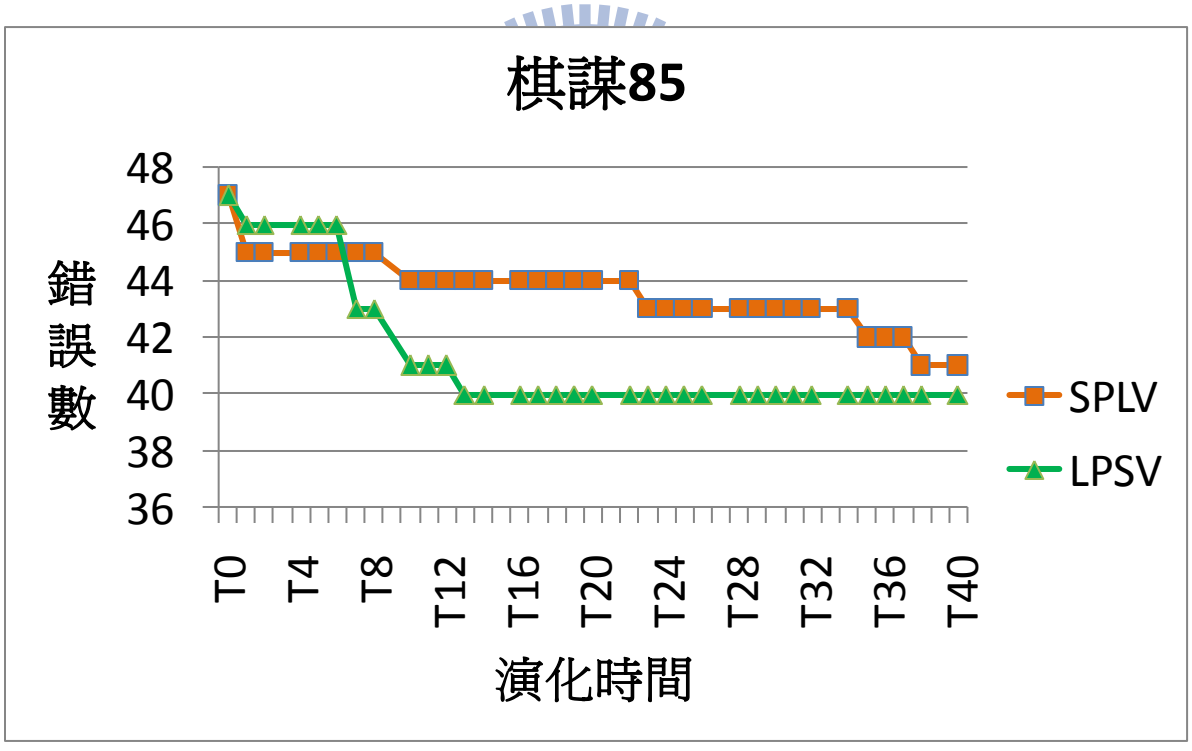


圖 3-3 n_g =34，SPLV 和 LPSV 的比較

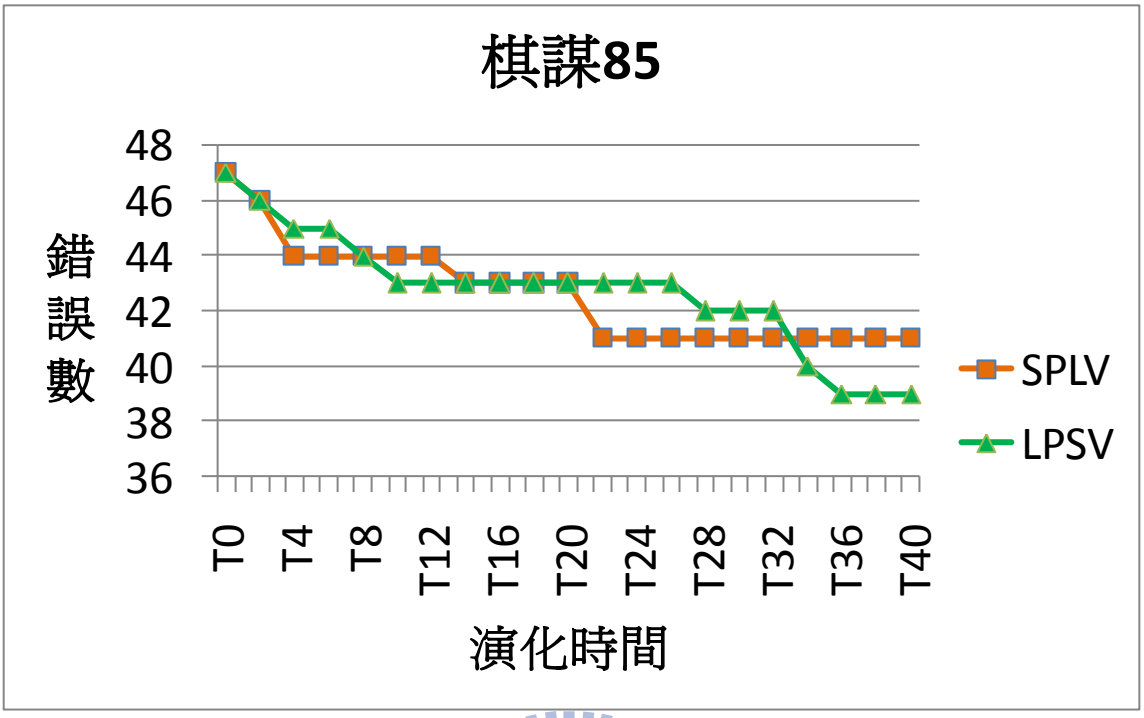


圖 3-4 n_g =54，SPLV 和 LPSV 的比較

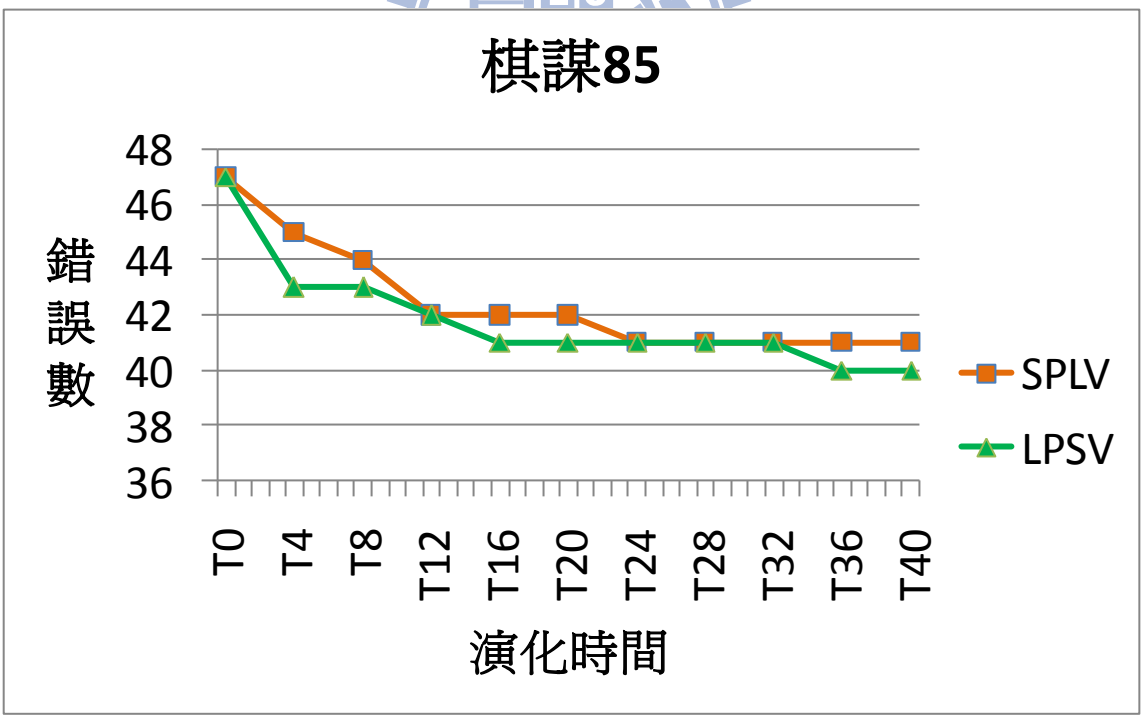


圖 3-5 n_g =104，SPLV 和 LPSV 的比較

3.2.4 n_b 值的比較

跟 3.2.1 的實驗類似，我們使用 LPSV，修改 n_b 值，我們有 4 個演化實驗，n_b = 2、 n_b = 4、n_b = 10、 n_b = 20，來看看同樣的時間之內，n_b 數值改變，對演化帶來的影響。

比較結果如圖 3-6 所示，縱軸是測試測資的錯誤數，橫軸是演化代數，n_b = 2 是菱形、 n_b = 4 是正方形、 n_b = 10 是三角形、n_b = 20 是圓形。由實驗結果看來，n_b = 4 比較實驗結果較好，n_b = 2 略差， n_b 在 10 以上則越來越差。

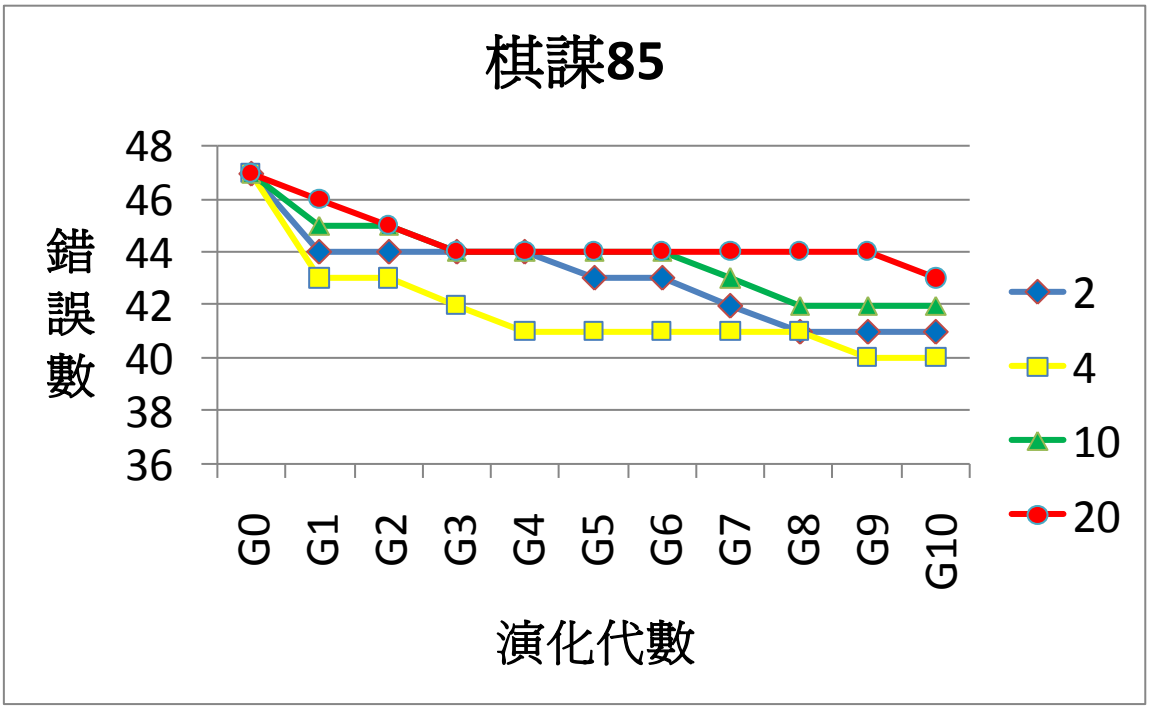


圖 3-6 n_b 值的比較

3.2.5 交配率和突變率的比較

我們觀察 3.2.1 的實驗中使用 LPSV 的那個演化，演化了 10 代，每代會留下 4 個冠軍參數組合，觀察這共 40 個冠軍參數組合的演化來源，發現有 31 個是交配出來的參數組合，有 2 個是突變出來的參

數組合，有 7 個是前一代沒改變的原始參數組合，而先前實驗，我們的交配率設定 70%，突變率設定 30%，交配產生出來的參數組合成功變成冠軍參數組合的成功率，和突變產生出來的成功率，是 $\frac{31}{70\%}:\frac{2}{30\%}=87\%:13\%$ ，也就是說一個冠軍參數組合，從交配產生的機率和從突變產生的機率的比例是 87:13。

跟 3.2.1 的實驗類似，我們使用 LPSV，修改交配率和突變率，我們有 3 個演化實驗，交配率 70%和突變率 30%、交配率 87%和突變率 13%、交配率 10%和突變率 90%的演化。

比較結果如圖 3-7 所示，縱軸是測試測資的錯誤數，橫軸是演化代數，交配率 70%和突變率 30%是菱形、交配率 87%和突變率 13%是三角形、交配率 10%和突變率 90%是正方形。由實驗結果看來，交配率改成 87%和突變率改成 13%，演化效果比較好一些，而交配率 10%和突變率 90%也真的演化效果較差。

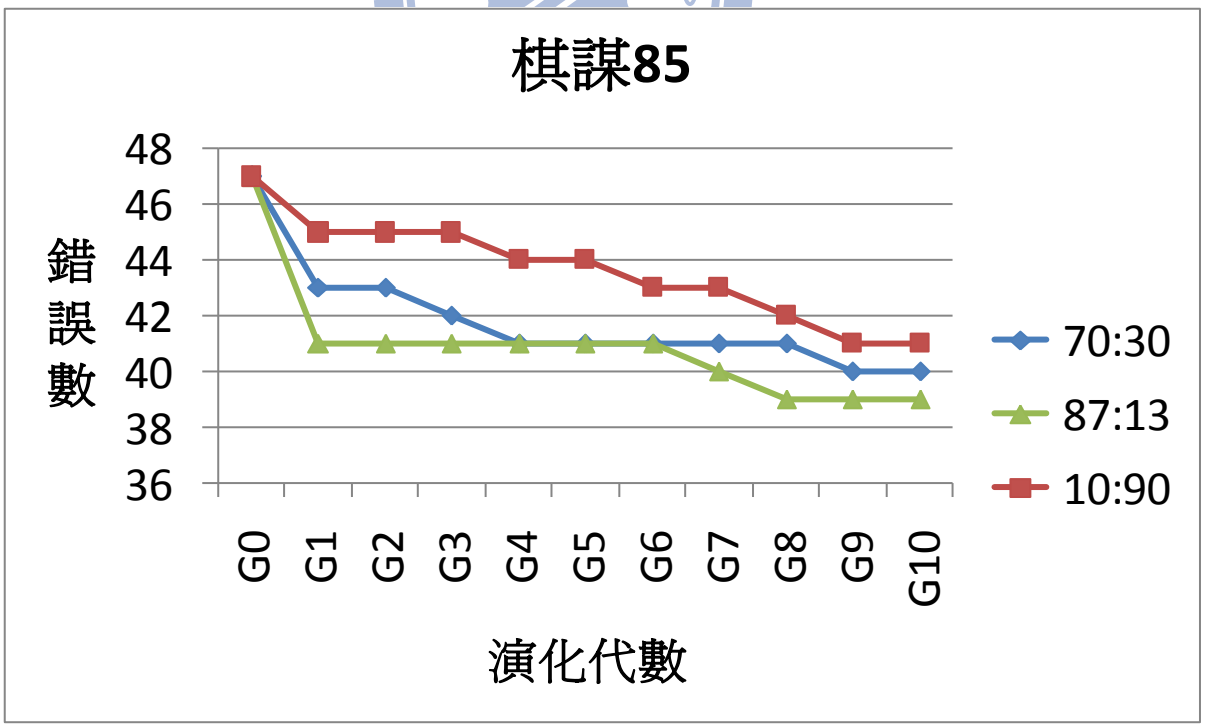


圖 3-7 交配率和突變率的比較

3.2.6 子力位置表選擇方式的比較

跟 3.2.1 的實驗類似，我們使用 LPSV，修改子力位置表選擇方式，我們有 3 個演化實驗，分別是原本的選擇區塊方式、隨機選 5 點方式、隨機選 10 點方式，藉由不同選取子力位置表的方式的比較，來看選取區塊的方式是否有效果。為了減少其他因素的影響，我們這次所演化的參數組合只演化子力位置表，不演化其他參數，我們將 n_g 設為 54(新產生 50 個)，只改子力位置表。

比較結果如圖 3-8 所示，縱軸是測試測資的錯誤數，橫軸是演化代數，原本的選擇區塊方式是菱形、隨機選 5 點是三角形、隨機選 10 點是正方形。由實驗結果看來，選擇區塊的方式的確比較好。

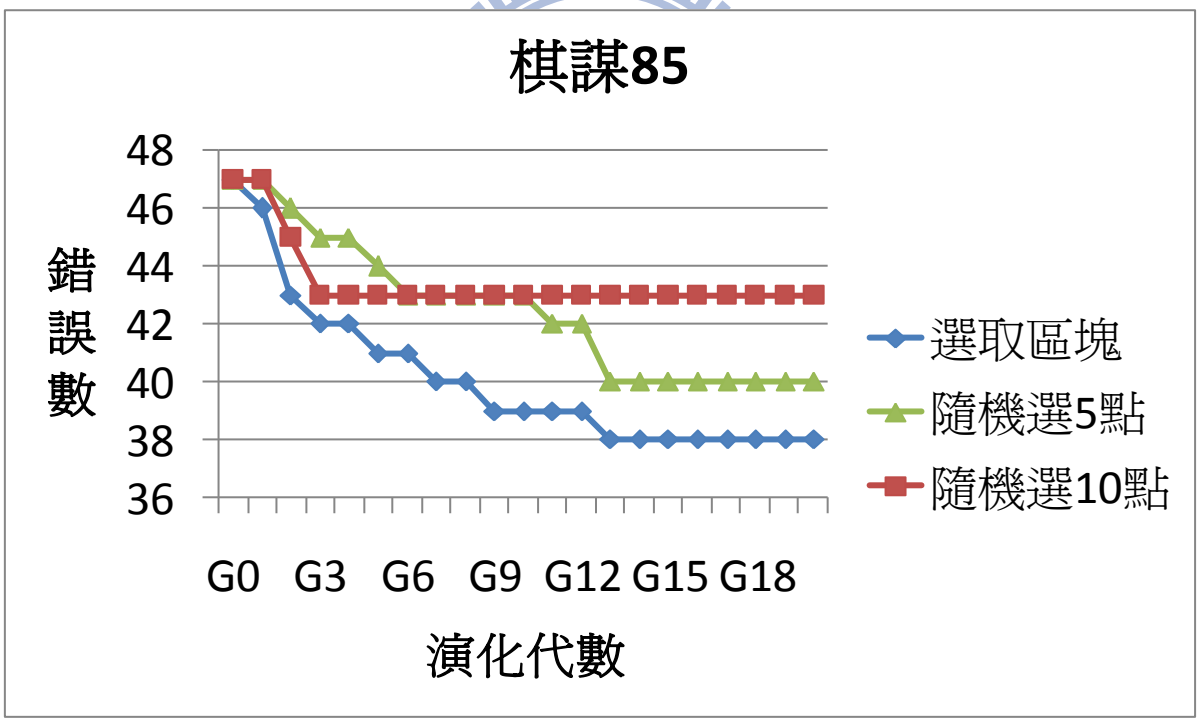


圖 3-8 子力位置表選擇方式的比較

3.3 實戰比較

3.3.1 介紹我們要實戰比較的是使用哪個演化結果，3.3.2 介紹演化程式和原始程式對戰的結果，3.3.3 介紹演化程式和玩家對戰的結果，3.3.4 介紹各演化代數的實戰比較。

3.3.1 演化結果

在我們做 3.2 的演化參數比較時，我們同時來做實戰的驗證，我們使用棋謀 v85，演化參數中的 $n_b = 4$ 、 $n_g = 104$ ，使用 LPSV，交配率 70%、突變率 30%，這個演化參數設定的演化後版本來對戰。

演化的結果如圖 3-9 所示，縱軸是測試測資的錯誤數，橫軸是演化代數，由圖可知，原本的 AI 還有 47 個錯誤，經由演化，錯誤數逐漸降低，演化 10 代後剩 40 個錯誤。

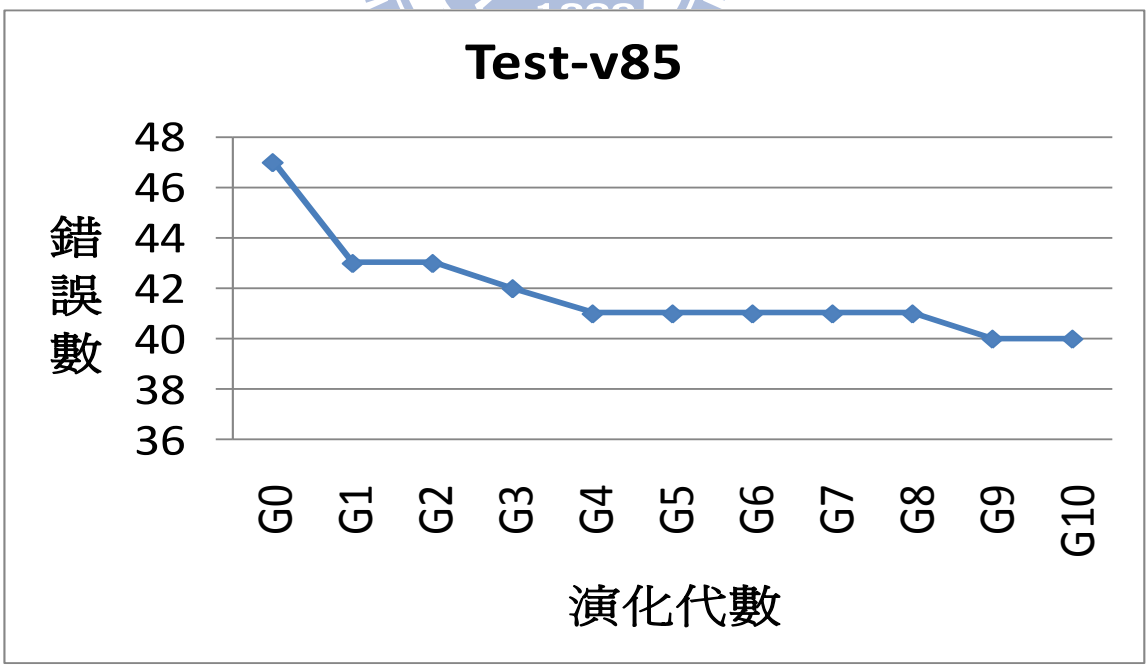


圖 3-9 演化結果

3.3.2 演化程式和原始程式對戰

由於測試測資的結果好壞，並不同等於程式棋力，我們必須再做些實戰驗證。我們令 AI_0 是原始的參數組合的棋謀 AI，AI_10 是演化 10 代後的參數組合的棋謀程式，我們要讓 AI_0 和 AI_10 相互對戰，我們隨機選不同的開局盤面，每個盤面雙方紅黑各持一場對戰，時間的設定是每步 30 秒，若雙方合計走了 200 步後不分勝負，判定和棋。

結果 AI_10 對 AI_0 的結果是，AI_10 得到 130 勝 107 敗 202 和，也就是 AI_10 的勝率 29.6%、敗率 24.4%、和率 46.0%，如表 3-10 所示，由對戰結果看來，演化後的版本的確有比原始版本強。

	勝	敗	和
AI_10 對 AI_0	130 (29.6%)	107 (24.4%)	202 (46.0%)

表 3-10 演化程式和原始程式對戰結果

3.3.3 演化程式和玩家對戰

然而對戰只看自己程式和自己程式對戰，也不完全準確，因為自己程式的行為是固定的，變化性太少，所以我們再讓程式和人類玩家對戰看看。同樣地，AI_0 是始的參數組合的棋謀 AI，AI_10 是演化 10 代後的參數組合的棋謀程式，我們把 AI_0 和 AI_10 都放到「群想」網站上，讓這兩個程式和人類玩家對戰。

由於群想網站沒有分別勝和，只有記錄敗，所以我們勝和放一起。放上去 4 天的結果，AI_0 跟玩家比了 258 場，有 27 場敗，敗率 10.47%，AI_10 跟玩家比了 194 場，有 16 場敗，敗率 8.25%，如表 3-11 所示，由程式和玩家的對戰結果看來，演化後的版本的確有比

原始版本強。

	勝和	敗	全部
AI_0 對玩家	231 (89.53%)	27 (10.47%)	258 (100%)
AI_10 對玩家	178 (91.75%)	16 (8.25%)	194 (100%)

表 3-11 演化程式和玩家對戰結果

3.3.4 各演化代數對戰

我們想知道 3.3.1 實驗中，棋力是否隨著演化代數逐漸加強，我們取演化 2 代後、5 代後、8 代後的冠軍參數組合，各和起始的參數組合做程式對戰，來比較其對戰結果。

圖 3-12 是演化各代的程式和起始程式對戰的結果，縱軸是勝敗機率差，橫軸是演化代數，G0 一定是 0%，由圖可知，演化 10 代後的程式的確是最強的，然而演化過程中間的代數卻沒有原始版好，並不如我們所預期的，逐代加強棋力。這個原因，我們認為是測試測資結果和實際程式強度的關係不夠強，導致測試結果雖然越來越好，但棋力不一定變好。例如我們的測資中有「棄子攻殺法」，測試這個冊資的結果變好，我們的參數組合就比較偏向於棄子強攻，子力基本價值降低，可能導致棋力反而不好。

要解決這方面的問題，我們需要找尋更好的測資，或是為不同好壞的測資設定不同的權重分數，使測試測資的結果和實際棋力的關係更加強烈，等找到更好的測資後，再用本基因演化系統演化，找尋更好的參數組合。

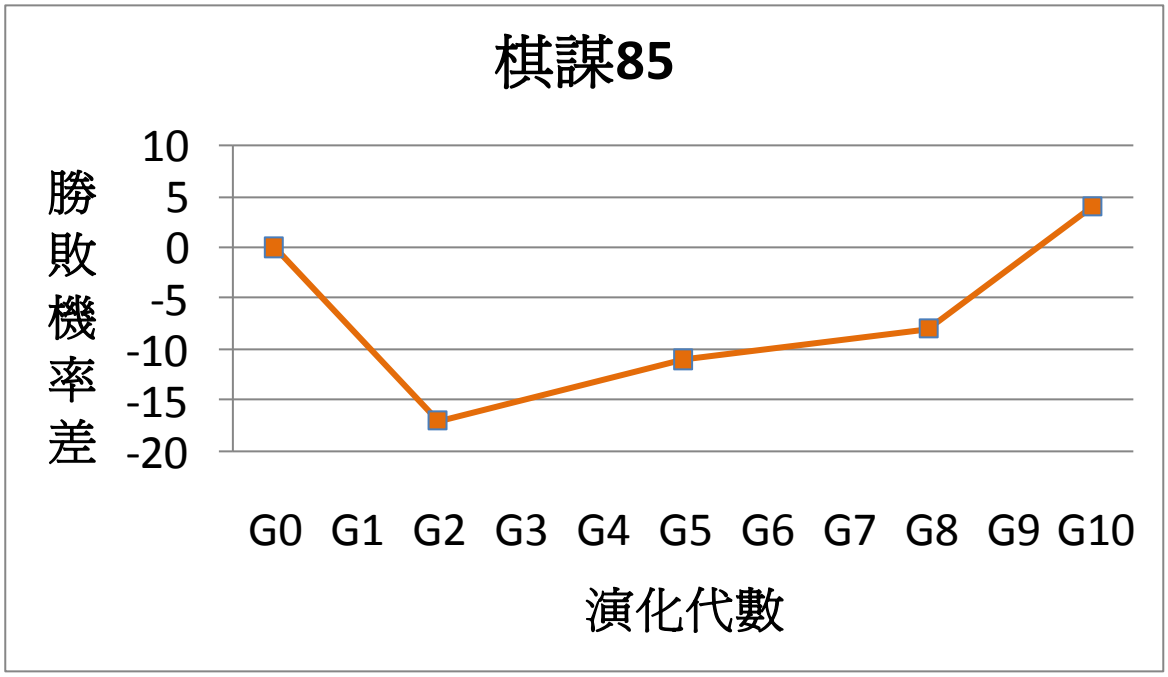


圖 3-12 各演化代數對戰結果

3.4 棋謀程式各個版本的演化結果

先前的實驗都是以棋謀程式的 85 版做實驗，不過不只 85 版，本基因演化系統在各個版本都可以使用，且都能找到更好的參數組合，圖 3-13 是其中 4 個版本的演化結果，所以當程式版本更新時，只要再經由我們演化系統調整參數組合，就可以再增強一些棋力。

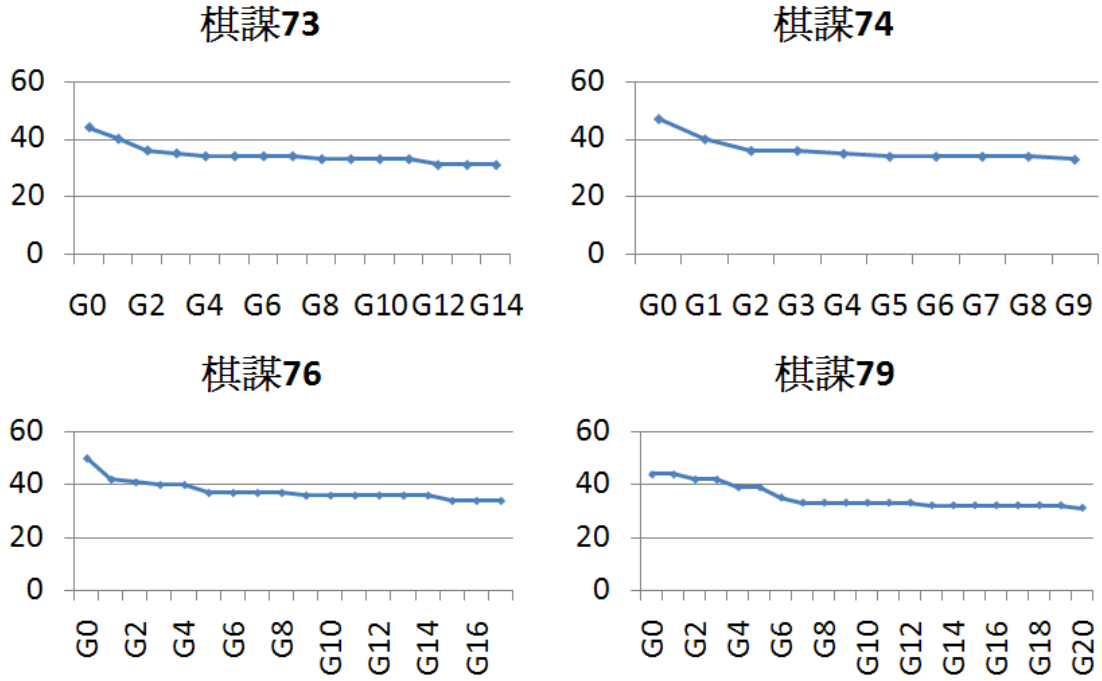


圖 3-13 其中幾個程式版本的演化結果

第四章、結論與未來展望

本篇論文的主要貢獻是，藉由基因演算法和統計的方法，自動的調整審局的參數值，不再依靠人工緩慢地不斷調整，對於棋局不同階段或是各個版本的程式，都可以快速地自動找到更好的參數設定，提升電腦象棋程式的棋力。

根據我們的研究實驗數據來看，較好的演化的參數設定如下:

1. $n_b = 4$，n_g則是設定多少都可以(參見第 3.2.2 節、第 3.2.4 節)。
2. 參數量和變化量方式採取 LPSV(參見第 3.2.1 節、第 3.1.3 節)。
3. 交配率 87%，突變率 13%(參見第 3.2.5 節)。
4. 子力位置表採取選取區塊方式(參見第 3.2.6 節)。

另外根據我們的實驗性對戰分析，也顯示不論程式間相互對戰，或是程式和玩家對戰，演化過後的程式都比較好，代表經過演化，程式棋力確實有提升，我們的實驗性對戰結果如下:

1. 演化過後的程式跟原始程式比，勝率是 29.6%、敗率是 24.4%、和率是 46.0%。(參見第 3.3.2 節)。
2. 放到群想網站上與玩家對戰，演化過的程式敗率 10.47%，原始程式 8.52%(參見第 3.3.3 節)。

我們應用此系統於本實驗室研發的「棋謀」程式上，改進棋謀裡審局部分的參數數值，使該程式棋力增強。棋謀在 2009 年的第十四屆國際電腦奧林匹亞(Computer Olympiad)中，獲得電腦象棋競賽之銅牌。

在未來的發展上，有幾個項目可以進行:

1. 棋譜統計值轉換成電腦程式審局分數的研究
2. 基因演算法的平行化研究。
3. 增加更加優良的測資以及替測資設定權重值。

4. 運用在其他遊戲上的研究。

以象棋作為研究背景，我們利用基因演算法和統計的方法做參數優化，確實曾進了電腦象棋程式的棋力。我們相信在其他棋類遊戲上，也能運用此方法做參數的優化，提升棋力，對人工智慧的研究上有相當大的幫助。

參考文獻

[1] Bo-Nian Chan, Pangfang Liu, Shun-Chin Hsu, Tsan-sheng Hsu(2007). Abstracting Knowledge from Annotated Chinese-Chess Game Records. Computers and Games, CG 2006, H.J. van den Herik et al.(eds.), pp. 100-111, Vol. 4630 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISBN 978-3-.540-75537-1.

[2] Chrilly Donnninger, Ulf Lorenz(2006). Innovative Opening-Book Handling. Advances in Computer Games, ACG 11, H.J. van den Herik et al.(eds.), pp. 1-10, Vol. 4250 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISBN 978-3-.540-78887-3.

[3] Haw-ren Fang, Tsan-sheng Hsu, and Shun-chin Hsu (2001). Construction of Chinese Chess Endgame Databases by Retrograde Analysis. Computers and Games, Second International Conference, CG 2000 (eds. T.A. Marsland and I. Frank), pp. 96-114, Vol. 2063 of Lecture Notes in Computer Search. Springer-Verlag, Berlin. ISBN 3-540-43080-6.

[4] Haw-ren Fang, Tsan-sheng Hsu, and Shun-chin Hsu (2003). Indefinite Sequence of Moves in Chinese Chess Endgames. Lecture Notes in Computer Science: Proceedings of the 3rd International Conference on Computers and Games (eds. J. Schaeffer and M. Müller). Springer-Verlag, New York, N.Y.

[5] Haw-ren Fang (2005). The Nature of Retrograde Analysis for Chinese Chess – Part 1. ICGA J 28 (2): 91-105 SEP 2005.

[6] Haw-ren Fang (2005). The Nature of Retrograde Analysis for Chinese Chess – Part 2. ICGA J 28 (3): 140-152 SEP 2005.

[7] Kuang-che Wu, Shun-Chin Hsu, Tsan-sheng Hsu(2006). The Graph-History Interaction Problem in Chinese Chess. Advances in

Computer Games, H.J. van den Herik et al. (eds.), pp. 165-179, Vol. 4250 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISBN 978-3-.540-48887-3.

[8] Michael Buro(1999). Toward Opening Book Learning. ICGA Journal, Vol.22, No.2, pp.98-102.

[9] R. Wu, D.F. Beal(2002). A memory efficient retrograde algorithm and its application to solve Chinese Chess endgames. More Games of No Chance, MSRI Publications, Vol. 42, pp. 207-228.

[10] Shi-Jim Yen, Jr-Chang Chen, Tai-Ning Yang, Shin-Chin Hsu(2004). Computer Chinese chess. ICGA Journal, Vol.27, No.1, pp.3-18.

[11] Shi-Jim Yen, Tai-Ning Yang, Jr-Chang Chan, Shun-Chin Hsu(2007). Study on Loop Problem in Opening Database. New Trends in Applied Artificial Intelligence, H.G. Okuno and M. Ali(eds.), pp. 1118-1126, Vol. 4570 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISBN 978-3-.540-73322-5.

[12] Thomas R. Lincke(2003). Position-Value Representation in Opening Books. Computers and Games, CG 2002, J. Schaeffer et al.(eds.), pp. 249-263, Vol. 2883 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISBN 978-3-.540-20545-6.

[13] Thomas R. Lincke(2001). Strategies for the Automatic Construction of Opening Books. Computers and Games, CG 2000, T.A. Marsland and I. Frank(eds.), pp. 74-86, Vol. 2063 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISBN 978-3-.540-45579-5.

[14] Wang Jiao, Luo Yan-hong, Qiu Dong-ni, Xu xin-he(2005). Adaptive genetic algorithm's implement on evaluation function in computer Chinese chess. IEEE International Symposium, Vol. 2, pp. 1253-1256. ISBN 0-7803-9538-7.

[15] Wu, R., Beal, D.F. (2001). Solving Chinese Chess Endgames by Database Construction. Information Sciences, Vol. 135, Nos. 3-4,

pp. 207-228. ISSN 0020-0255.

[16] 鄭明政(2006). 電腦象棋程式位置評分表之研究, 成功大學工程科學研究所, 碩士論文.

[17] 王驕, 王濤, 羅艷紅, 徐心和(2005). 中國象棋計算機博弈系統評估函數的自適應遺傳算法實現. 東北大學學報(自然科學版), pp.949-952, Vol.26, No.10.

[18] 許舜欽(1990). 電腦西洋棋和電腦象棋的回顧與前瞻. 電腦學刊, 第二卷, 第二期, 第1-8頁.

[19] 許舜欽(1990). 電腦象棋的設計與製作. 電腦學刊, 第二卷, 第四期, 第1-11頁.

[20] 許舜欽, 林益興(1991). 電腦象棋的盲點解析. 電腦學刊, 第三卷, 第四期, 第1-6頁,.

[21] 蔡旭程(2005). 高效率電腦象棋停著殺程式之設計與製作. 台灣大學資訊工程學研究所, 碩士論文.

[22] 許俊彬(2006). 象棋棋形辨識之研究. 交通大學資訊科學與工程研究所, 碩士論文.

[23] 曾汶傑(2008). 象棋殘局庫之研究. 交通大學資訊科學與工程研究所, 碩士論文.

[24] 陳志昌(2005). 電腦象棋知識庫系統之研製. 台灣大學資訊工程學研究所, 博士論文.

[25] 亞洲象棋聯合會裁判組(2003). 象棋比賽規則.

[26] 許舜欽, 陳志昌, 李天佑(1999). 網路應用於電腦象棋學習系統之設計與製作. 高速計算世界, 第七卷, 第四期, pp.12-16.

[27] 徐心和, 王驕(2006). 中國象棋計算機博弈關鍵技術分析. 小型微型計算機系統, 第二十七卷, 第六期, pp.961-969.

[28] 付強, 陳煥文(2007). 基於RL算法的自學習博弈程序設計及實驗. 長沙理工大學學報, 第四卷, 第四期, pp.73-78.

[29] 馬占欣, 李亞, 陸玉昌(2007). 用遺傳算法解決博弈問題. 河南科學, 第二卷, 第二期, pp. 273-277.

[30] 姜榮富(1985). 棄子攻殺法. 象棋叢書21, 世界文物出版社, 台北市, 初版.