崑 山 科 技 大 學

電腦與通訊系 四技部

專 題 製 作 報 告

利用基因演算法消除光學鏡頭色差

學生： 陳盈傑（4940D052）
劉邦彥（4940D023）
劉家洋（4940D016）

指導老師： 蔡政穆

中 華 民 國 九 十 八 年 一 月

專題製作報告授權同意書

本授權書所授權之報告為本組在崑山科技大學電腦與通訊系 10 組 97 學年度第 1 學期修習專題製作課程之報告。

報告名稱：利用基因演算法消除光學鏡頭色差

本組就具有著作財產權之報告全文資料，同意提供本校圖書館典藏，並同意圖書館因典藏之目的就該資料進行必要之數位化重製，且依圖書館法、著作權法規定，提供讀者利用。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。

請勾選授權公開年限及範圍：(請勾選一項)

- ☑ 立即公開
- ☐ 五年後公開
- ☐ 三年後公開
- ☐ 校園內公開
- ☐ 館內典藏

指導老師姓名：蔡政彥

學生簽名：劉邦彥 學號：4940D023

陳盈傑 4940D052

劉家洋 4940D016

（親筆正楷） （務必填寫）

日期：民國 98 年 1 月 日

崑 山 科 技 大 學

電腦與通訊系 專題製作

利用基因演算法消除光學鏡頭色差

學生： 陳盈傑
劉家洋
劉邦彥

經考試合格特此證明

指導老師：蔡政翔

系 主 任：陳志林

中 華 民 國 九 十 八 年 一 月

摘　要

本次專題題目,乃是應用基因演算法來求出光學消色差鏡頭之最佳消色差公式解，光學鏡頭的基本組成為透鏡，改善其屈光率與阿比係數，並將數值帶入基因演算中計算並求出最佳近似解，以達到消色差的目的，而消色差的程度更影響了鏡頭解析度與像素，因此光學消色差的動作更是重要的光學透鏡製作流程。基因演算法是一種模仿大自然界中，物競天擇和基因交配，泛指以達爾文進化論”適者生存，不適者淘汰”為基礎，來模擬自然界演化過程所建立的計算模式，而這些計算模式又被稱為演化式演算法。將消色差公式帶入基因演算法中，並設定變數範圍、循環次數，藉由基因演算法三大流程：生存-競爭、交配、突變，以人工模擬自然界演化環境,透過以上三大流程來求出消色差公式最佳近似解，達成透鏡消除色差的目的。

誌　謝

此次專題能順利完成首先必須感謝的是指導教授-蔡政穆教授，由於在專題製作的初期對於程式本身的不熟悉以及對幾何光學的無知，一直阻礙著專題進度的前進，若非蔡教授給予時間與鼓勵並發揮極大的耐心，以及不斷的撥空指導，否則本專題不可能在一學年之內準時結束，另外還有其他組員們的同心協力，分工合作，加諸以上因素，才使本專題順利、準時完成，經過這次專題的洗禮，進而了解到光學產品在市場上的發展性以及程式設計的重要性，若沒有程式的輔助模擬而直接使用硬體，則專題的研究時間將更為耗時，然而模擬的結果雖然並不準確，但也提供了以硬體製作的參考方向，最後靠著本專題，各位組員都有所成長，不論是在程式、理論基礎、小組合作、報告製作皆有收穫，評分時眾位教授的指導與批評更是我們在將來的求職或升學之路產生更重要的影響，最後再一次感謝指導教授-蔡政穆教授的指導與包容，而本專題所帶來的收穫將使未來的路走得更平順。

目錄

圖 目 錄

表 目 錄

第一章

色差原理

1-1 何謂色像差

當一道光線射入透鏡時，如果該光線是複色光，那麼因為介質(透鏡)對不同波長有不同的折射反應，所以會有另外一種像差產生，稱為色像差(chromatic aberration)[1]，簡寫 C.A.。除了反射鏡外，其他任何元件均會產生色像差，即使是在高斯光學下的成像也都具有色像差的成像缺陷。

1-2 如何消除色像差

若要消除色像差，通常是針對系統所使用的波長及需求來設計，能矯正二個波長的色差並同時也矯正球差的系統，稱為消色差透鏡(achromat)[1][2]。而對三個波長矯正色像差，且對兩個波長光的球差加以矯正的系統，稱為複合消色差透鏡(apochromat)[1][3]。一般來說，最簡單的消色像差方法，是利用二種不同的材料做成的膠合系統，其中一個透鏡的正色像差和另一透鏡的負色像差抵消，使得兩特定波長的成像重合在一起。首先，膠合透鏡的屈光率 K 可寫成：

$$K = K1 + K2 \qquad \text{式 1-1}$$

式 1-1 中 K1、K2 分別是膠合系統兩透鏡的屈光率，故

$$K1=(n1-1)(1/r11-1/r12) \quad \text{式 1-2}$$

$$K2=(n2-1)(1/r21-1/r22) \quad \text{式 1-3}$$

其中 n1、n2 為兩透鏡的折射率，r11、r12、r21、r22 分別是透鏡的兩曲率半徑。射屈光率因折射率而產生的變化量為△K，則

$$\begin{aligned} \triangle K &= \triangle K1 + \triangle K2 \\ &= \triangle n1(1/r11-1/r12) + \triangle n2(1/r21-1/r22) \\ &= (\triangle n1/n1-1)K1 + (\triangle n2/n2-2)K2 \\ &= K1/V1 + K2/V2 \end{aligned} \quad \text{式 1-4}$$

式 1-4 的計算，如果針對修正紅光及藍光的色像差而設計，則 V1、V2 分別是兩透鏡的 Abbe 值。在要求無色像差的條件下，我們可得

$$K1/V1 + K2/V2 = 0 \quad \text{式 1-5}$$

式 1-5 正是我們要讓基因演算法演算的光學消色差公式，但在本專題中我們使用了三組透鏡，所以公式會變成

$$K = K1 + K2 + K3 \quad \text{式 1-6}$$

$$K1/V1 + K2/V2 + K3/V3 = 0 \quad \text{式 1-7}$$

第 二 章

基 因 演 算 法 原 理

2-1 基因演算法簡介

基因演算法(Genetic Algorithm)其實是取法大自然的一種演算法，藉著對於演化現象的觀察，John H. Holland[4][5]認為可以透過把問題轉為基因型態(Genotype)，利用競爭-生存以及基因交換-突變，尋求出問題的正確解答。透過競爭-生存，擁有較好基因的品種會有較高的機會生存到下一代，而與生存較無關的基因或品種較低的則會隨著時間逐漸被淘汰。

在理想的狀態下，競爭-生存會迫使不具優勢的品種逐漸消失。然而單單擁有一種好基因是不夠的；透過基因的交配，不同的個體可以把它們的好基因組合起來，變成更具優勢的下一代；而若是組合出來的後代不理想，也只會加速被淘汰。但交換基因也不能改變現有的基因狀態，還要再配合突變，才能夠產生革命性的改變，進而對族群進步有突破性的發展。因此，"生存-競爭"、"交配"、"突變"就是整個基因演算法的中心思想。

基因演算法是所有 EA(Evolutionary Programming)[4]模式中最為著名也是使用最多的演化式演算法。Holland 認為自然界的演化是發生在生物染色體的基因中，每一種生物的特徵係來自於該物種上一代的基因排列，演化是指每一代基因所發生的變化情形。所謂適者生存是指這一代的基因排列優於上一代的基因排列，而產生比上一代更能適應環境生存的世代 (Holland, 1975)。因此，基因演算法是強調基因型的轉變，將欲求解問題的參數經過編碼成為基因格式，利用遺傳運算進行演化來找到問題的最佳解。這些遺傳運算是模擬自然界的演化程序，包括有競爭-生存(competition)、交配(crossover)與突變 (mutation)等等。因此 基因演算法具有下列幾點特性：

(1) 基因演算法是將參數編碼進行演化運算，而不是使用參數本身做搜尋。

(2) 基因演算法只有使用適應函數做為本體的知識，沒有其他繁瑣的數學計算。

(3) 基因演算法使用“機率規則”而不是使用“明確規則”來導引搜尋方向，因此可適用於不同類型的問題上。

2-2染色體、基因組、適配值解釋

本專題所應用的基因演算法，乃為了求出光學消色差公式的最佳解，因此主要的應用部分是放在求“函數最大(小)化問題[6]”，其中以公式所求出的解作為適配值，這也是主要的終止條件之一(另外一個終止條件為達到總循環次數)，而基因組的構成則是以公式中的未知數來取代，以式2-1為例：

MAX 31P-P^2=A (2-1) 式2-1

式2-1所指的意思乃為求出“ A ”的最大解，而未知數的範圍介於在0~31之間，在此“ A ”為此基因演算法的適配值, “ P ”則為基因，將“ P ”之值由0至31依序代入，可得圖2-1如下：

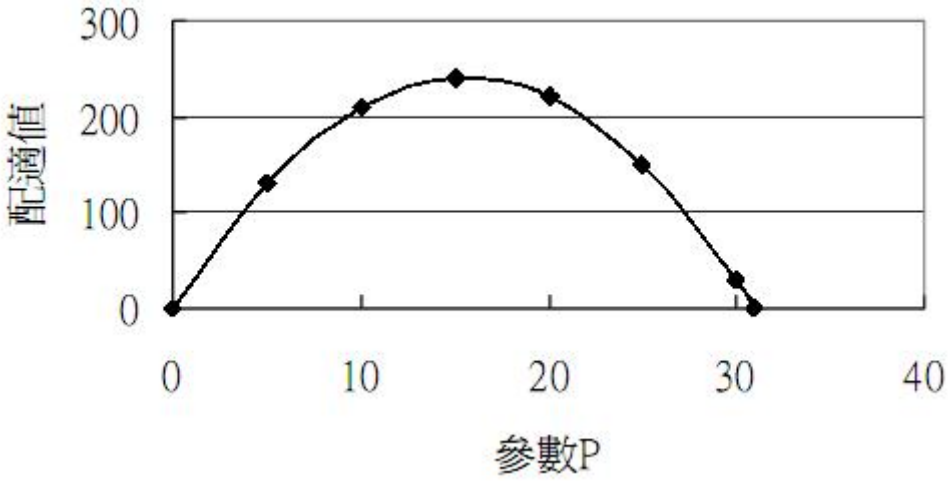

圖2-1 適配值落點圖

附註:此公式雖然可一眼看出其“ P ”值該代何數方能使“ A ”為最大值,但此例主要用來說明基因演算法公式中,適配值與基因組之關係。

有了適配值這個終止條件後，便可以進行基因組編碼；此例子所使用的編碼乃是用二進位制編碼，而本專題所使用的，則是直接編碼，，編碼的方式並不會影響數值，只會影響數值呈現的方式以及交配的模式，在此從0~31範圍中**隨機**取出2數進行編碼：

(1)取出數字 10 二進制編碼 01010

(2)取出數字 20 二進制編碼 10100

此為顯示5個位元的二進制編碼，每一個位元代表一個染色體，5個染色體組合就成為了一個基因，而基因之間的交配是以機率的方式讓產生出來的新基因選取父親或是母親的基因，讓我們以表2-1來表示：

基因	染色體1	染色體2	染色體3	染色體4	染色體5
基因-父	0	1	0	1	0
基因-母	1	0	1	0	0
隨機數	0.4	0.7	0.2	0.6	0.1
基因-子	1	1	1	1	0

表2-1 母代基因表示法

這邊的範例是以當隨機數(範圍0.1~1)小於0.5時則選擇母親，大於0.5時則選擇父親，而藉由交配，可以繁衍出擁有父母優異染色體的子代基因，並在透過循環不斷交配，進而找出最佳解。

2.3基因演算法動作流程

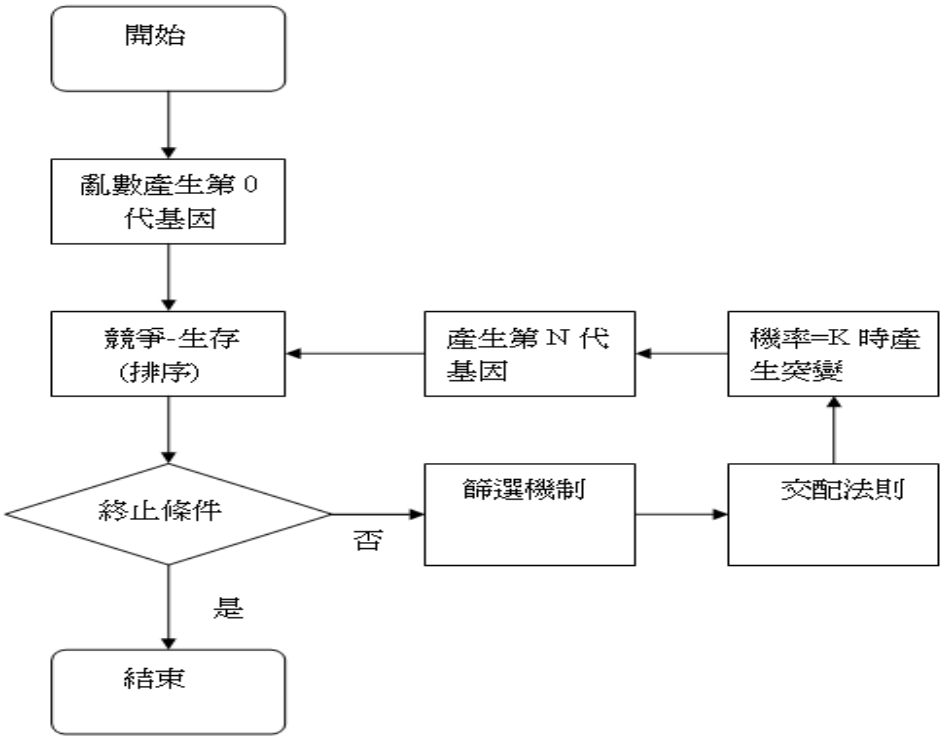

圖2-2 基因演算法流程圖

2-4基因演算法主要工作原理

由前面所提到的基因演算法三大主軸：競爭-生存、交配、突變[4]，以下便開始來說明這三者於基因演算法當中所扮演的角色與定位為何。

2-4-1競爭-生存(排序)

競爭-生存，簡單來說就是透過排序相互比較的方式來決定基因品種的優劣，越接近適應值的基因，在排列上的名次會相對的高存留下來的機會也越高，而名次越後面的基因，則會在演化的循環中被淘汰，每經

過一次循環的交配與突變機制之後，會在經過“競爭-生存”這樣的步驟來檢視繁衍出來的下一代品種的優劣，當整個GA流程全部結束後，於排列上名次最高的基因組，便是該次循環的最佳近似解。

本次專題在於排序的部分使用了heapsort累堆排序法搭配tablesort表格排序法相互應用，由於基因演算法循環次數動輒百萬次起跳，為了有效節省運算時間，因此使用了上述兩種排序法來減低時間複雜度與空間複雜度。

2-4-2篩選機制(輪盤法則)

挑選機制[4]其實就是探討如何從群體(樣本空間)挑選出個體(樣本）的取樣方式，被挑選的個體就是親代，可經由遺傳運算來產生子代。基本的取樣機制有三種[5]：

(1) 隨機取樣 (stochastic sampling)

(2) 明確取樣 (deterministic sampling)

(3) 混合取樣 (mixed sampling) (Gen and Cheng, 1997;Goldberg, 1989)。

早期的 GA 大都使用隨機取樣，做法是先計算出基因體的適配值，再根據基因體規模將適配值轉換成個體數目。最著名的隨機取樣就是 Holland 提出的輪盤式(roulette wheel selection)[6]選擇。輪盤式

選擇是將每一個體視為是輪盤上的一個區塊，區塊面積的大小與該個體的適配值成正比，適配值越大則區塊面積也越大，被挑選的機率也就越高。假如將輪盤看成是射飛鏢的遊戲，明顯的可看出面積越大就越容易射中。

基因體在排序完成後，較好的適配值會排名較前面，而**輪餅比例圖**也根據該排序結果所制定，就會造成較好基因的區塊較大，較差基因的區塊較小，在依照機率篩選，以確保我們所挑選出來的都是良好基因。

在此說明本次專題所使用的適配值並不是上述例子所求的最大值“ MAX ”，而是最小值“ min ”。

以求最大值為例，並排序完成之圓餅比例圖分配如下：

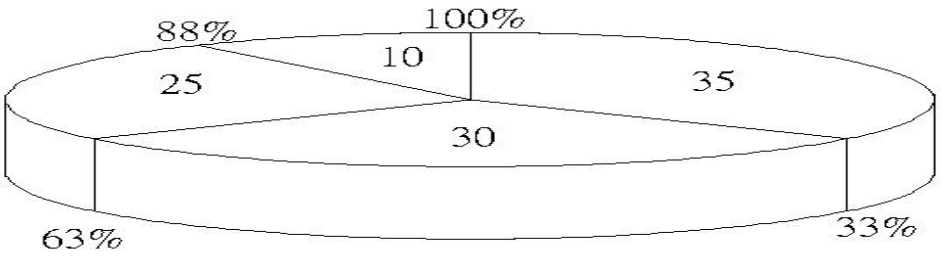

圖2-3(a) 求最大值範例圓餅比例圖

基因組-1 適配值 35　　基因組-3 適配值 25

基因組-2 適配值 30　　基因組-4 適配值 10

從圓餅圖的比例中可以發現，越大的適配值所佔的比例越高，這將有利我們的篩選，篩選機制也是以隨機產生亂數(範圍0.01~1)的方式來決定取哪一塊等份的基因組，若亂數落到了“ 0.41 ”則取百分比33%至百分比63%之間的基因組，而該結果也符合上述所說，越好的適配值占越大的比例，也較容易被篩選到。

以上是有關於求最大值的圓餅比例分配法，然而若是要求最小值則不能直接以適配值大小這樣形式呈現，必須做點改造。

以求最小值為例，並排序完成之圓餅比例圖分配如下：

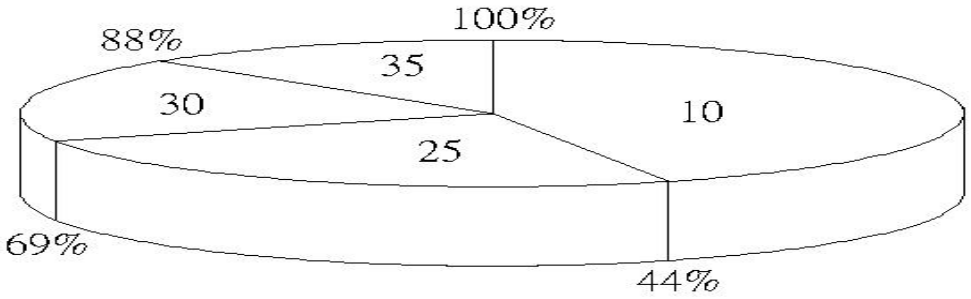

圖2-3(b) 求最小值範例圓餅比例圖

註：圓餅圖內之數字表示適配值，並非比例!!!

基因組-1 適配值 10　　基因組-3 適配值 30

基因組-2 適配值 25　　基因組-4 適配值 35

此時圖2-3(b)顯示出和圖2-3(a)截然不同的比例分配，那是因為在

畫最小值的圓餅圖時須對適配值進行比例轉換。

比例轉換方式如下：

陣列編號：	A1	A2	A3	A4
原基因組陣列(由小到大排序)：	10	25	30	35

比例轉換公式：(A1+A4)-An(An=自身適配值)

轉換後：

適配值10 (10 + 35) - 10 = 35　適配值25 (10 + 35) - 25 = 20

適配值30 (10 + 35) - 30 = 15　適配值35 (10 + 35) - 35 = 10

再將轉換後的新比例累加(=85)，讓新比例去除以累加數就可以得到如上圖般的比例分配表，而透過如此比例轉換的方式可以讓最小值的適配值擁有較大的比例，而篩選的機制也一樣採取隨機產生亂數的方式，來篩選出欲交配的基因組(母代)。

2-4-3交配與突變法則

透過上述所說篩選機制,可以令品種良好的基因被篩選出來並做為繁衍下一代基因組(母代),但也並不一定每一個繁衍出來的基因會遺傳到父母的優良基因,所以縱使是兩種不好的基因交配也會產生接近適應值的良好基因，然而在交配的過程中，更因為有突變的機率產生，使得

基因組有的突破性、革命性的發展。

首先，為何要有突變的存在，如流程圖所述，一開始的基因組皆為亂數產生，透過競爭、篩選與繁衍，從最初亂數的基因組演化出其優良的基因，但整體基因架構已然無法跳脫，透過基因組裡面染色體的突變，可以讓基因演算法進入未曾尋找過的基因架構，進而開創新的一輪的基因組與圓餅圖，但是突變次數過多將會導致基因演算法變成隨機演算法，會任意破壞原來的架構，造成繁衍出來的基因與母代基因喪失若干相似的特徵，反而使前面的演化付諸流水。

本專題所使用的交配定律與前述(染色體、基因組、適配值解釋)一樣採用隨機亂數產生方式選取父親或母親之染色體，然而唯一不同的是，本專題採用直接編碼(意謂染色體以最初設定的方式呈現而不再另行編碼)，另外本專題所制定的公式有6個未知數，不是像先前所描述的範例這般簡單，在變數值上有更多種變化，因此適合採用基因演算法中的求“最佳解”的應用。

關於突變法則，本專題還是採用隨機亂數方式，只不過在交配過程中多加入了突變機率與突變方式，詳細解釋在表2-2(a)、2-2(b)：

基因	染色體1	染色體2	**染色體3**	染色體4	染色體5
基因-父	0	1	0	1	0
基因-母	1	0	1	0	0
隨機數	0.4	0.7	**0.3**	0.6	0.1
基因-子	1	1	1	1	0

表2-2(a) 突變基因步驟表示

表 2-2(a)所示，是在“ 染色體、基因組、適配值解釋 ”中所述說之例，在此依然舉該例講述突變法則，請注意到隨機數的部分，隨機數既是應用在交配法則也應用在突變機率，也就是兩者所使用的隨機數是相同的，倘若突變機率為“當隨機數=0.3 或 0.8 時則該數進行圖變”而突變方式採取該位元反向，則該染色體突變後會變成“ 0 ”，也就是 11110。

而本專題在與上述唯一的差別在於突變方式的不同，由於使用直接編碼的關係，基因所呈現的方式如下(因版面有限，只顯示出5個染色體)：(接下頁)

基因	染色體1	染色體2	**染色體3**	染色體4	染色體5
基因-父	79	34	58	66	21
基因-母	64	15	56	81	42
隨機數	0.4	0.7	**0.3**	0.6	0.1
基因-子	64	34	**56**	66	42

表2-2(b) 突變基因步驟表示

這時我們採用的突變方式為該數加上“正或負0.5所乘上隨機數(範圍1~100)”，倘若隨機數為10，則突變後的染色體3為“ 61 ”。

2-4-4 循環

當基因演算法從：

(1) 制定公式

(2) 亂數產生基因組

(3) 競爭-生存(排序)

(4) 篩選(圓餅法則)

(5) 交配與突變機制當演算至交配與突變機制完畢後，此時演算法便會回到第(3)直到循環次數結束或是適配值範圍符合我們先前所設定的條件時才會停止動作。

第 三 章

基 因 演 算 法 程 式 流 程

本專題所撰寫的基因演算法程式主要包含下列設計重點：

(1) 排序-累堆排序法、表格排序法

(2) 欲求解之公式制定與變數範圍

(3) 篩選機制-使用圓餅法則

(4) 交配與突變機制

(5) 整體循環

以下便開始說明基因演算法程式的設計步驟及使用的語法、邏輯

3-1 排序-累堆排序法(heapsort)

使用累堆排序[8][9]法主要原因來自於其時間複雜度(O(n*log2*n))較短，一般基因演算法所求的解之精度與演化(循環)次數和變數範圍的大小有關，越多次的循環可以確保所求的解越接近最佳解，也因此需要時間複雜度較短的排序法，然而這並不代表不能使用如氣泡排序法般時間複雜度較長(O(n^2))的排序法，只不過在使用這些排序法的同時，會使得程式在等待其排序結果完成所花費的時間加長，若又在加大演化次

數與變數範圍將使程式運行時間過長並提高硬體需求，因此程式撰寫以累堆排序法為優先。

累堆排序法之原理主要是將要被排序的陣列化為二元樹的方式，藉由調整將陣列(二元樹)化成最大累堆(MAX heap)，第一次調整，化為最大累堆並找出該陣列的最大數，而每當一個最大數被調整出來後，該數就會脫離被排序的陣列轉移到排序完成的陣列，之後的第二次調整、第三次調整將數值由大至小依序向左排列，詳細動作圖如下：

未排序完成之陣列：

陣列	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
資料	16	27	23	43	10	22	88	33	*27

表 3-1 陣列元素表示

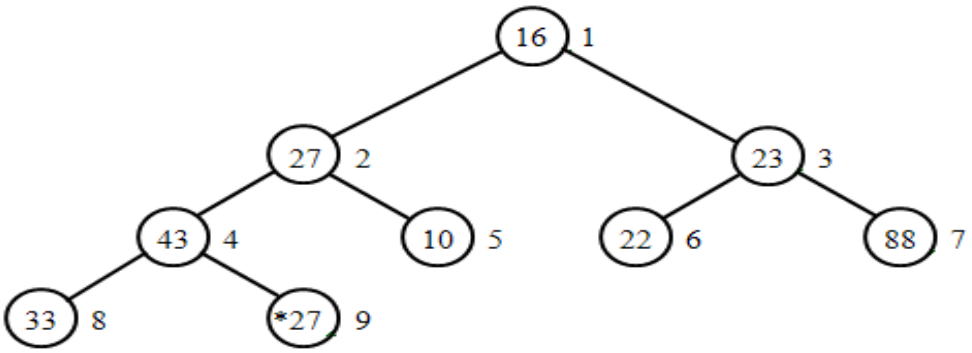

圖 3-1(a) 化為二元樹之圖

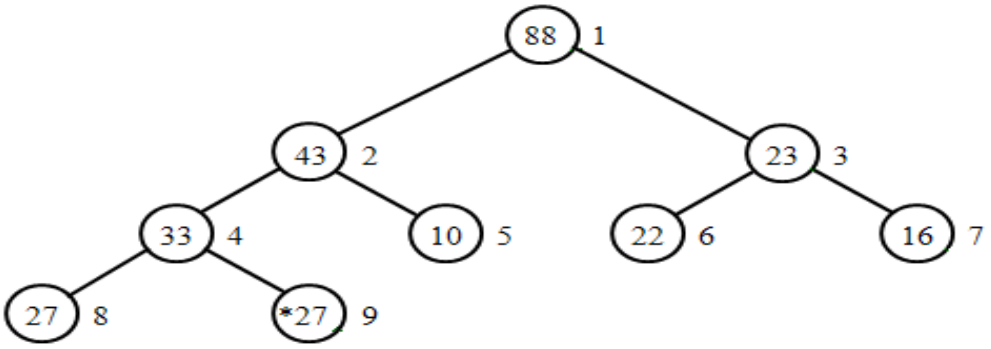

圖 3-1(b) 第一次調整化成最大累堆，並找出最大數“ 88 ”。

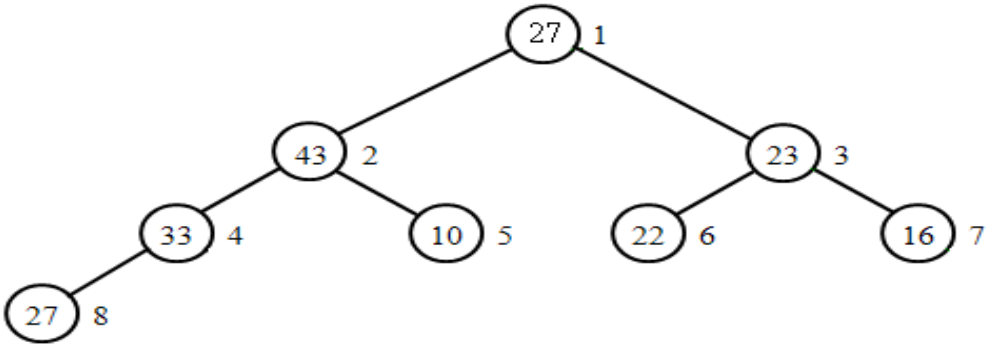

圖 3-1(c) 將最大數移除欲排序的陣列外，將最末端的數移置頂層並再次調整。

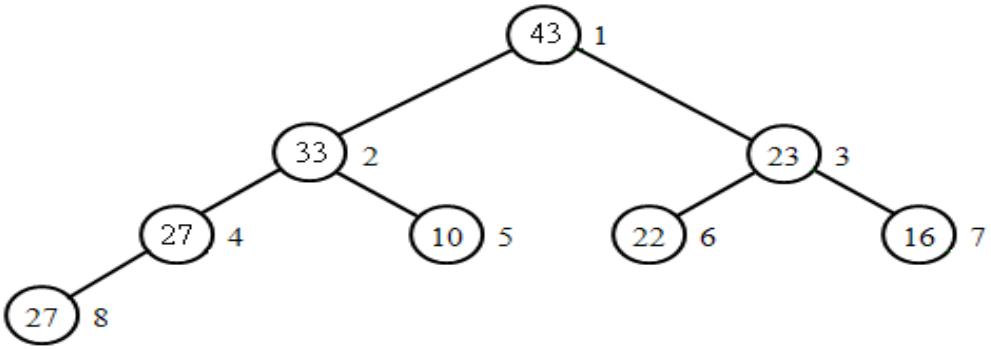

圖 3-1(d) 再次調整後，找出該累堆最大數“ 43 ”。

在調整的過程中，並找出最大數的同時，二元樹內部的數會不斷的減少(**圖 3-1(a)至 3-1(d)**)，當二元樹中只剩下一個數時該排序就宣告完成。

3-2 排序-表格排序法(tablesort)

表格排序法(tablesort)[8][10]在本專題的應用是用來輔助累堆排序法(heapsort)，由於排序是針對適配值來執行，而非基因組，也就是當適配值在經過排序後，基因組卻還是紋風不動，無法對應到該基因組所得到的適配值，此時，就需要表格排序法來改善這一問題；表格排序法主要的動作流程便是索引，在適配值尚未排序之前便先賦予其編號如下表：

基因組		適配值	編號
基因組 1	=	23	(1)
基因組 2	=	10	(2)
基因組 3	=	91	(3)
基因組 4	=	46	(4)
基因組 5	=	71	(5)

表 3-2(a)表格排序法示意圖(1)

當排序完成後，基因組與適配值的關係將會變成：

基因組		適配值	編號
基因組 1	=	10	(2)
基因組 2	=	23	(1)
基因組 3	=	46	(4)
基因組 4	=	71	(5)
基因組 5	=	91	(3)

表 3-2(b)表格排序法示意圖(2)

現在我們可以看到適配值的排列順序變了，但基因組的位置並沒有跟著改變，原因如同前面說的，排序的目的是根據適配值，基因組本身並不具有參考點，但是若基因組無法去對應適配值，那之後的篩選與交配也就無法進行，因為不知道哪一個基因組所求出來的適配值是哪個，也就無從斷定該基因的品種優劣，這時，我們先前賦予適配值的編號便起了作用，編號的用意便是註明適配值在未排序前是在陣列中的哪個位置，如同標籤一樣註明商品是從哪裡生產出來的，縱使適配值被排序，但編號卻不會進入排序，從表 3-2(b)的例子來看，目前適配值最小的，是由第二組基因組所排列出來的。

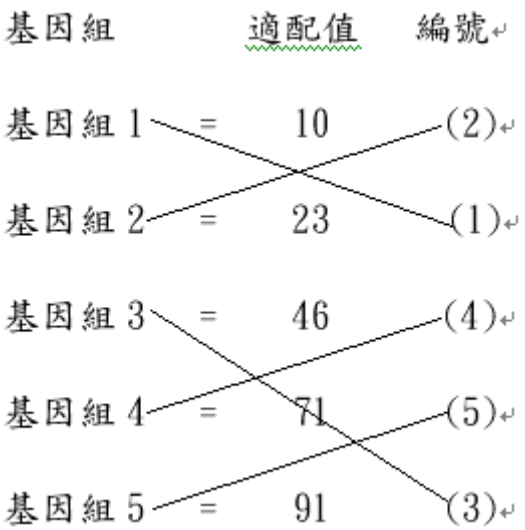

表 3-2(c) 表格排序法示意圖(3)

有了編號這樣的索引標籤，便可以很容易的將基因組移動到相對應的適配值上，而本專題也應用累堆排序法和表格排序法的輔助完成了基因演算法中“ 競爭-生存 ”這個重要環節之一。

3-3 欲求解之公式制定與變數範圍

公式制定我們採用定義新結構的方式來實行，透過這樣的方式我們可以更好管理基因組中的變數，也讓後面的程式撰寫不至於混淆。

以下是定義結構的語法：

```
typedef struct {
        double key;   //適配值。
        int index;    //tablesort 編號。
        double k1,k2,k3,v1,v2,v3;   //基因組染色體變數。
        } element; //該結構之命名。
```

element list[MAX_SIZE]； //宣告一 element 型態的變數陣列 list。

請注意結構中定義的 k1，k2，k3，v1，v2，v3，這些就是我們基因組內部的染色體，key 則是基因組所求出的適配值，之後再進行各變數的算式制定，本專題所要加入的算式為：

k1+k2+k3=S　　(k1/v1)+(k2/v2)+(k3/v3)=0

以上的兩行算式，我們希望求出來的近似解是越接近 0 越好，

因此我們在算式上又做了以下修改：

A=|1-S|　　B=(k1/v1)+(k2/v2)+(k3/v3)=0

Key=| A+B |

如此一來便可以將兩個算式所期望的適配值放入到同一個變數裡來接受排序，而詳細程式碼如下：

```
for(t=1;t<=MAX_SIZE-1;t++)
{
        mark[t].k1=1+rand()%1000;
        mark[t].k2=-(1+rand()%1000);
        mark[t].k3=1+rand()%1000;
        a=fabs(1-(mark[t].k1+mark[t].k2+mark[t].k3));
        mark[t].v1=1+rand()%1000;
```

```
        mark[t].v2=1+rand()%1000;
        mark[t].v3=1+rand()%1000;
        b=(mark[t].k1/mark[t].v1)+(mark[t].k2/mark[t].v2)+(m
        ark[t].k3/mark[t].v3);
        list[t].key=fabs(a+b);
}
```

這段程式碼便是基因演算法制定算式的區塊，由結構制定變數與接下來的排序，再交由此區塊將各變數制定範圍與算式，外部的迴圈可以控制整個演化群組的數量，因此這區塊會是影響整個基因演算法精度的所在。

3-4 篩選機制-圓餅法則

由於本專題是以越接近 0 為最佳近似解，也因此在圓餅比例的劃分上會做些改變，以下是詳細程式碼：

```
float list2[MAX_SIZE];
        float all=0,po=0;
        float oa=0,oa2=0;
        double L=list[MAX_SIZE-1].key+list[1].key;
      for(x=1;x<MAX_SIZE;x++){
       list[x].oa=L-list[x].key;
```

```
}
   for(x=1;x<MAX_SIZE;x++){
   list2[x]=(float)list[x].oa;
   all=all+list2[x];
   }
       for(x=1;x<MAX_SIZE;x++){
       oa=oa+list[x].oa;
       oa2=oa/all;
       list[x].noir=oa2; }
```

如同在緒論章節-圓餅法則所提到的，求最大值和球最小值的圓餅圖的不同，在程式碼上也做了同樣的修改動作，以下便是各程式變數之間的相互關係：

L = list[MAX_SIZE-1].key+list[1].key

此為陣列位置第一位與最後一位適配值的總和。

list[x].oa = L-list[x].key;

將總和扣除掉自身的適配值，並將結果放入到 list[x].oa 陣列。

```
list2[x]=(float)list[x].oa;
all=all+list2[x];
```

將 list[x].oa 得值加總起來放入到變數 all。

```
oa=oa+list[x].oa;
oa2=oa/all;
list[x].noir=oa2;
```

將逐項的 list[x].oa 之總和除以全部的 list[x].oa 的總和，並將結果放置到 list[x].noir 變數陣列，完成比例轉換。

以下便是完成圓餅圖後所產生的比例表：

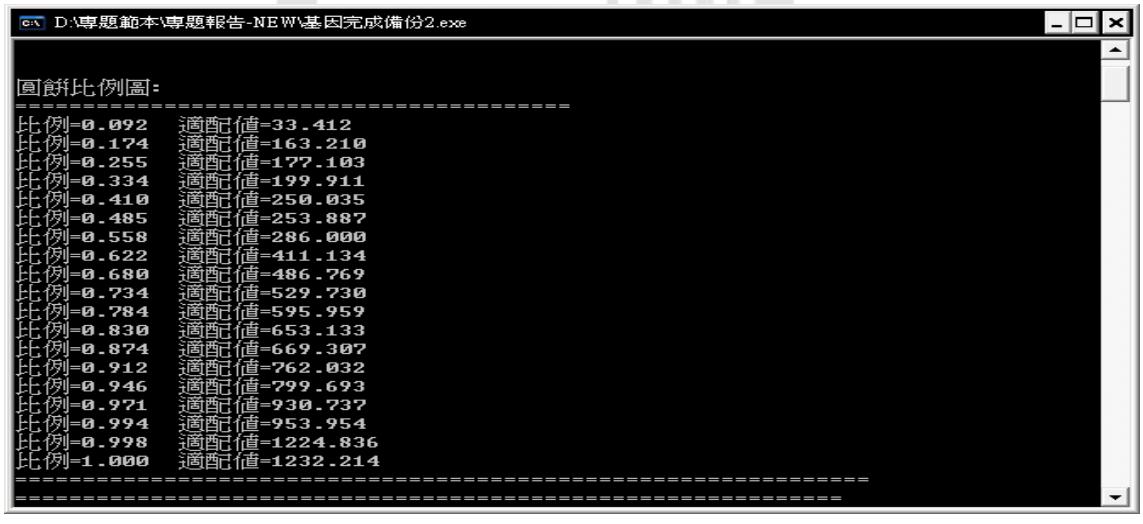

圖 3-2 演化群組 20 組時的圓餅比例分配

從圖 3-2 可知，適配值越小的所加總的比例是越大，而有了圓餅比例後更可使篩選出來的基因組為優良的機率大大提升，以下便來介紹關於篩

選機制的程式碼實例：

```
float parent1[CHILD_SIZE],parent2[CHILD_SIZE];
        int reg1[CHILD_SIZE],reg2[CHILD_SIZE],y,z;
        for(x=1;x<=CHILD_SIZE;x++){
        parent1[x]=(1+rand()%1000)*0.001;
        parent2[x]=(1+rand()%1000)*0.001;
           for(y=1;y<MAX_SIZE;y++){
              if(parent1[x]<list[y].noir){
              reg1[x]=y-1;
                 if(reg1[x]==0){
                 reg1[x]=1;
                 }
                 break;
           }
        }
```

註：以上程式碼只放入 parent1 的篩選，其 parent2 篩選程式碼與 parent1 完全相同，故不在顯示。

parent1[CHILD_SIZE] & parent2[CHILD_SIZE]

此兩個變數陣列表示篩選出來的兩個將要交配基因組，而陣列大小[CHILD_SIZE]則用來控制將要繁衍出多少基因組來取代原先的基因組。

reg1[CHILD_SIZE],reg2[CHILD_SIZE]

此兩個變數陣列用來放置基因組被選取出來所代表的比例。

```
parent1[x]=(1+rand()%1000)*0.001;
```

以隨機方式亂數產生 0.001~1 的數字來決定哪個比例區間將要被選取。

```
for(y=1;y<MAX_SIZE;y++){
                if(parent1[x]<list[y].noir){
                reg1[x]=y-1;
                   if(reg1[x]==0){
                   reg1[x]=1;
                   }
                   break;
            }
```

此段程式的目的在於使隨機數對應到圓盤比例，例如：產生 0.446，則程式便會找尋 0.446 是落在哪一個比例之間的區間，以圖 3-2 為例，

0.446 將會落在 0.410~0.485 之間,此時在程式的設計上會以選取 0.410 為優先，所以也就是適配值為 250.035 的基因組，而若產生出來的隨機數未滿第一個區間比例(如圖 3-2 所示，未滿 0.092)，則會直接選取第一個基因組、適配值為 33.412 來代入。

3-5 交配與突變機制

在前面制定好基因組與圓餅圖之後，接下來開始便是基因演算法中重要的交配階段，交配的方式在前面介紹中提到，是使用隨機數來決定選取哪一個母代基因，也因此在接下來的程式中仍會大量使用 rand()函式,另外基因組的呈現在交配階段時也會有所改變,詳細程式碼如下：

```
 for(x=1;x<=CHILD_SIZE;x++){
g1[x][1]=mark[reg1[x]].k1;g1[x][2]=mark[reg1[x]].k2;g1[x][3]=
mark[reg1[x]].k3;
g1[x][4]=mark[reg1[x]].v1;g1[x][5]=mark[reg1[x]].v2;g1[x][6]=
mark[reg1[x]].v3;
}//g2 也是相同程式碼，故在此只顯示 g1 設定部分。
for(x=1;x<=CHILD_SIZE;x++){
        for(y=1;y<=6;y++){
        rad[y]=(1+rand()%1000)*0.001;
```

```
        if(rad[y]>0.6){
            g3[x][y]=g1[x][y];
            rad2[y]=1+rand()%10;
            if(rad2[y]==2)

g3[x][y]=(1-CHANGE)*g3[x][y]+2*CHANGE*rand()*g3[x][y];
            }
            else{
            g3[x][y]=g2[x][y];
            rad2[y]=1+rand()%10;
            if(rad2[y]==2)
      g3[x][y]=(1-CHANGE)*g3[x][y]+2*CHANGE*rand()*g3[x][y];
            }
        }
        }
```

從交配階段開始，會把先前在公式制定時以結構方式呈現的基因組改成以二維陣列的方式呈現：

```
g1[x][1]=mark[reg1[x]].k1;   g1[x][4]=mark[reg1[x]].v1;
g1[x][2]=mark[reg1[x]].k2;   g1[x][5]=mark[reg1[x]].v2;
```

```
g1[x][3]=mark[reg1[x]].k3;   g1[x][6]=mark[reg1[x]].v3;
```

註:reg[x]為隨機數落在圓餅圖的哪一區間所代表的基因組。

```
rad[y]=(1+rand()%1000)*0.001;
        if(rad[y]>0.6){
           g3[x][y]=g1[x][y];
```

此段程式碼便是交配的主要動作，透過隨機數產生 rad[y]，如果隨機數大於 0.6 則選取第一母代基因組的基因，反之則選第二組母代基因組之基因，整個程序需循環六次。

```
rad2[y]=1+rand()%10;
        if(rad2[y]==2)
    g3[x][y]=(1-CHANGE)*g3[x][y]+2*CHANGE*rand()*g3[x][y];
        }
```

然而在交配之後還須加上突變機制，一樣產生隨機數 rad2[y]，當 rad2[y]=2 時，經過交配後的基因會產生突變，而突變的方式我們是設定(1-CHANGE)*交配後基因+2*CHANGE*rand()*交配後基因;其中的 CHANGE 我們將其制定為突變範圍定義在開頭部分,本專題的突變範圍設定在 0.5。

3-6 取代

在完成交配與突變後，必須將這些經過交配與突變後的基因重新放入到主要排序陣列，再讓完成排序的基因組再次執行圓餅圖、交配與突變這些階段的循環，設定取代的詳細程式碼如下：

```
for(x=1;x<=CHILD_SIZE;x++){
        mark[x+PARENT_SIZE].k1=g3[x][1];
        mark[x+PARENT_SIZE].k2=g3[x][2];
        mark[x+PARENT_SIZE].k3=g3[x][3];
        mark[x+PARENT_SIZE].v1=g3[x][4];
        mark[x+PARENT_SIZE].v2=g3[x][5];
        mark[x+PARENT_SIZE].v3=g3[x][6];

a=fabs(1-(mark[x+PARENT_SIZE].k1+mark[x+PARENT_SIZE].k2+mark[
x+PARENT_SIZE].k3));

b=(mark[x+PARENT_SIZE].k1/mark[x+PARENT_SIZE].v1)+(mark[x+PAR
ENT_SIZE].k2/mark[x+PARENT_SIZE].v2)+(mark[x+PARENT_SIZE].k3/
mark[x+PARENT_SIZE].v3);
        list[x+PARENT_SIZE].key=fabs(a+b);
```

由於先前我們把基因組轉換成二維陣列的方式呈現，在回歸主要排序陣列時是無法通用的，因此必須再將之轉回結構方式呈現，並且同時代換掉原先的基因組，將交配後的基因組放入到主要排序陣列，以mark[x+PARENT_SIZE].k1=g3[x][1];來解釋，g3 代表交配後的 k1 基因，而我們的設定是讓交配後的基因從後面開始取代，所以mark[x+PARENT_SIZE]中的 PARENT_SIZE 代表從哪一個位置開始取代，若 PARENT_SIZE 設定為 500，那交配後的基因會從主要排序陣列第 501 組的地方開始取代，而 PARENT_SIZE 真正的設定是：全部基因組(MAX_SIZE)-要保留的基因組(CHILD_SIZE)，這三個陣列數也是設定再開頭部分直接以定義方式設定。

3-7 循環

循環是整個基因演算法中代表演化的部分，循環次數的多寡會影響到數值的精確度，本專題是使用 for 迴圈，將程式從排序至取代完整包括起來，迴圈執行的次數便是演化的次數，如此一來整個使用基因演算法所做的運算程式到此大致完成。

第 四 章

基 因 演 算 法 收 斂 實 驗 結 果

4-1 平均適配值走向

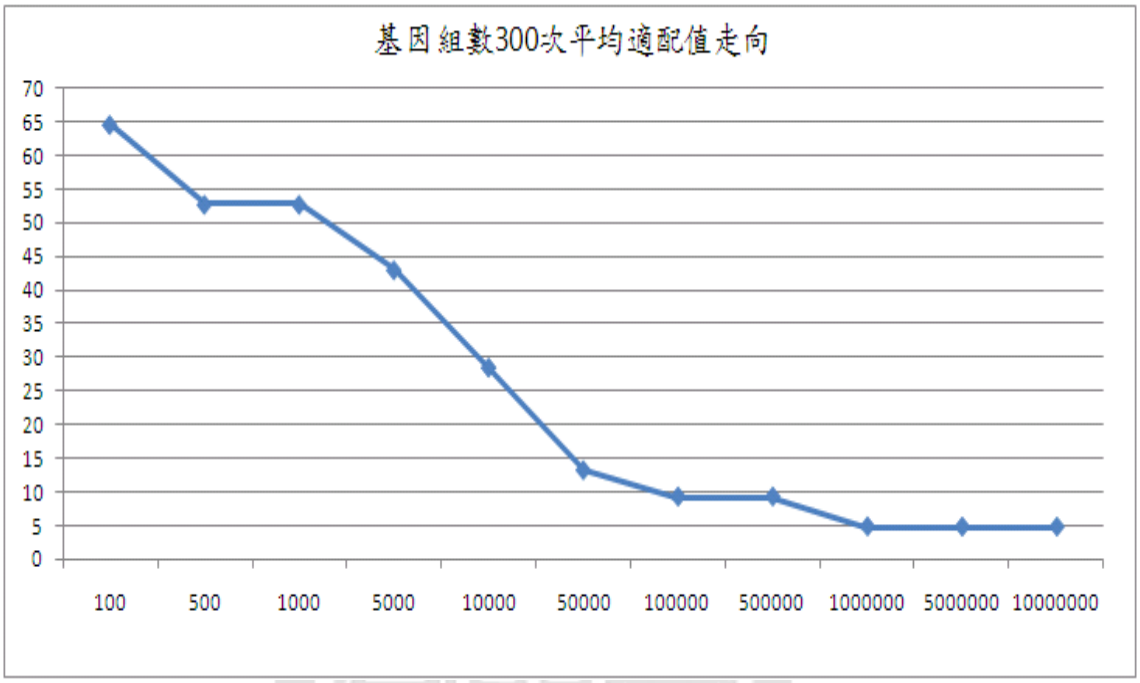

圖 4-1 基因組 300 組時在各循環次數的平均適配值

圖 4-1 是在各循環次數時所求出來的適配值的平均，詳細數值為：初代基因組數 300，保留基因組數 150，突變指數 0.2，隨機數範圍 10000，從圖 4-1 可以看出在循環第 100 次時收斂一次，而在第 1000 次時在收斂一次，而至第 5 萬次之後，在前面在排行前 5 組基因組中已經看不出有被取代的現象，至此已經求出該區域的最佳解。

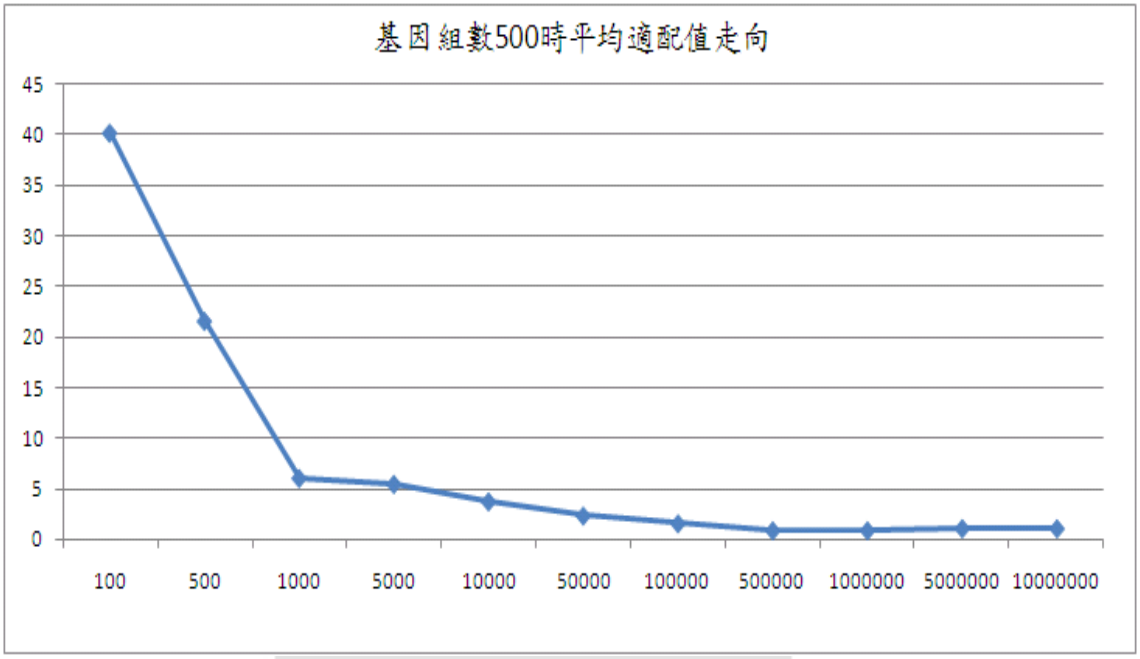

圖 4-2 基因組 500 組時在各循環次數的平均適配值

然而並不只循環次數會造成適配值的改變，在初代基因組數與保留基因組數的不同也會影響適配值，而越大的初代基因組數(如圖 4-3)，會在循環次數不大時就可能已經逼近於最佳解。

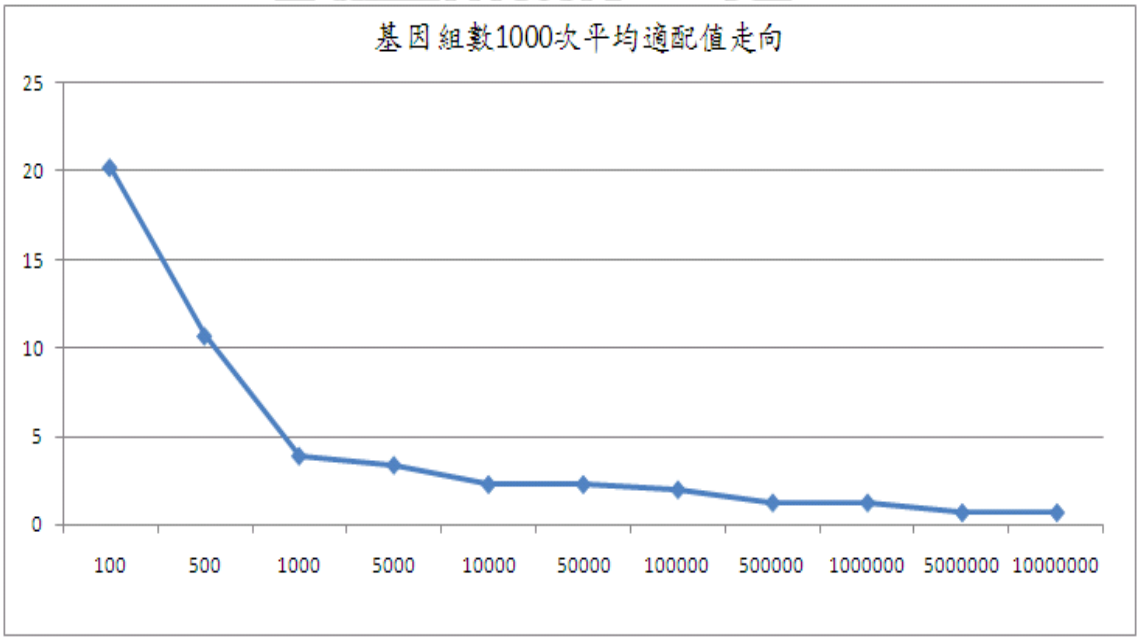

圖 4-3 基因組 1000 組時在各循環次數的平均適配值

4-2 最佳適配值走向

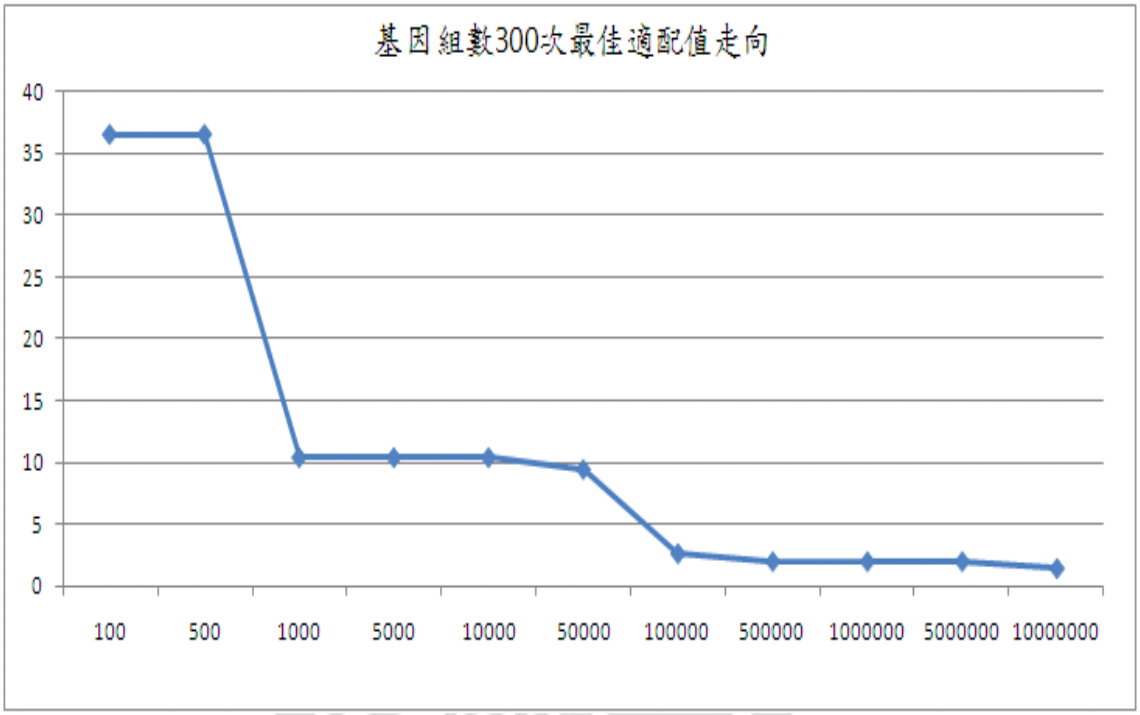

圖 4-4 基因組數 300 時最佳適配值走向

圖 4-4 是在基因組 300 組時的平均適配值走向，分別統計在循環次數 100、500、1000、5000、1 萬、5 萬、10 萬、50 萬、100 萬、500 萬和 1000 萬時的最佳適配值，而在 500 次開始第一次收斂，又在第 5 萬次時進行第二次收斂，而之後適配值就已呈現穩定，並不再出現最佳值來取代，我們也可以稱之為該區域最佳解已找出。

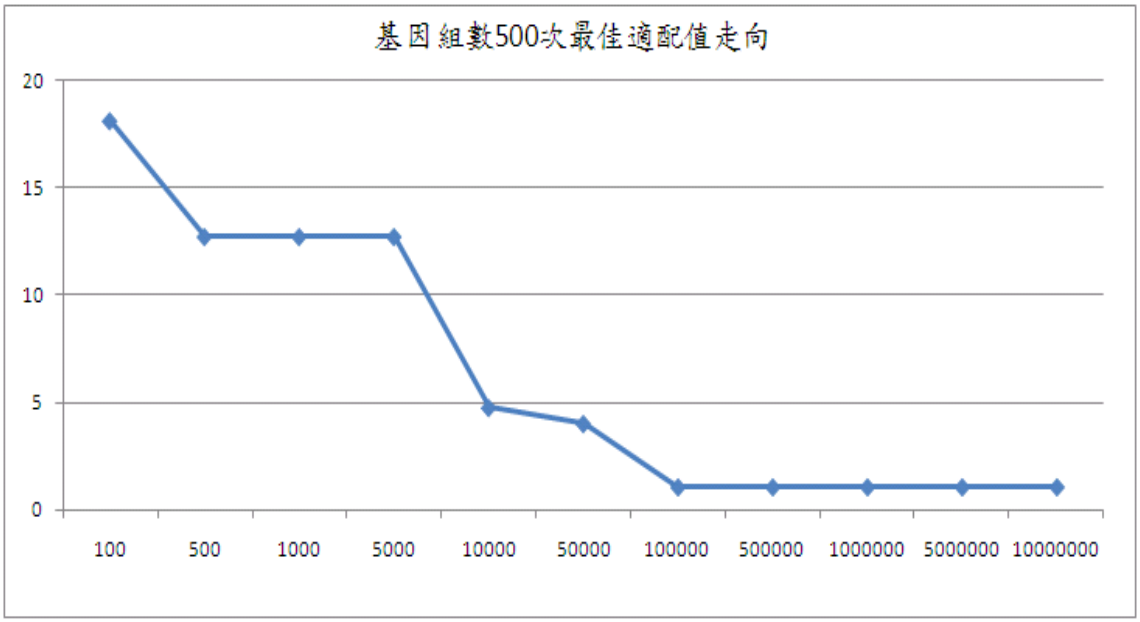

圖 4-5 基因組數 500 時最佳適配值走向

而圖 4-5 則是以基因組 500 組去做統計，從圖表可以看出在越大的基因組數在較少循環次數時所呈現的適配值已經較為接近我們的理想值 0，而在循環第 10 萬次之後，整個適配值呈現穩定。

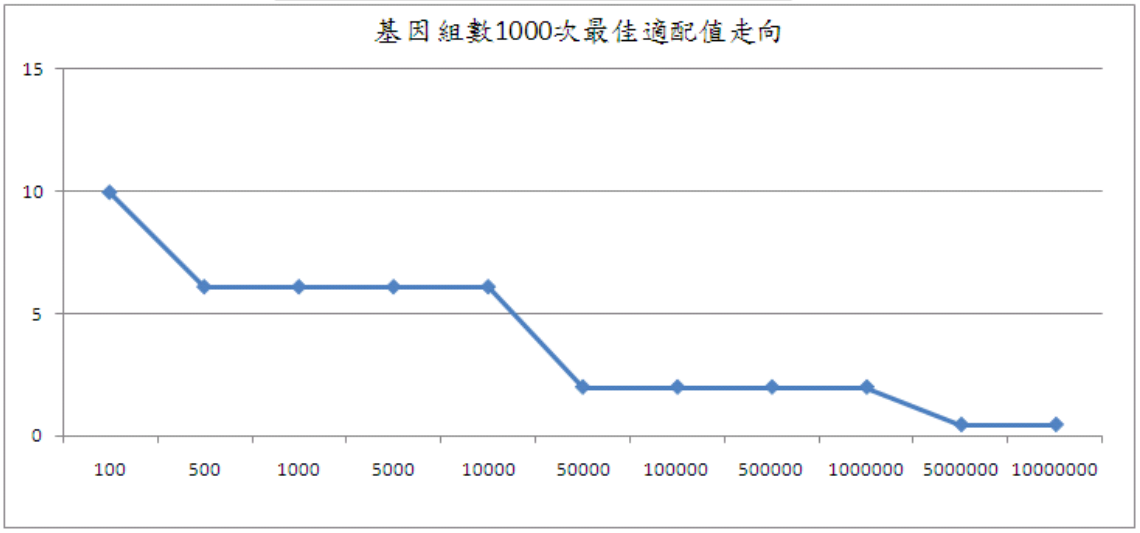

圖 4-6 基因組數 1000 時最佳適配值走向

圖 4-6 是基因組數 1000 時的最佳適配值走向，從圖可以看出，在開始 100 次循環時，最佳適配值的表現比起圖 4-4 的 300 組、4-5 的 500 組來得更佳，而在收斂的程度上卻不如圖 4-4、圖 4-5 來得好，那是因為，本專題運算程式的撰寫，在每一次重新執行時會產生新的基因組，而這些新基因組可以稱之為我們的區域，由於區域的不同所找出來的該區域最佳解也不同，加上突變的機率，會導致適配值的收斂程度有所變化，而保留的基因組數目的多寡也會影響適配值得走向。

4-3 保留數不同時的適配值走向

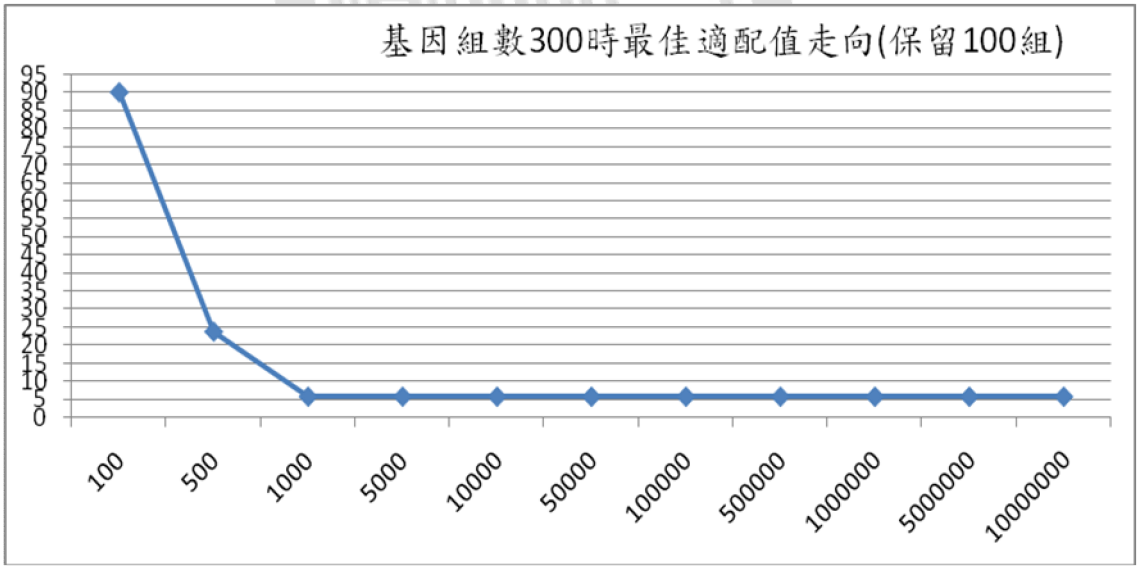

圖 4-7(a)基因組數 300 時(保留 100 組)最佳適配值走向

在前面所舉的例子當中皆是以保留全部基因組的 50%的走向圖，而圖 4-7(a)是只保留 3 分之 1，從圖表可以看出保留的基因組數越小收斂的程度越發急遽。

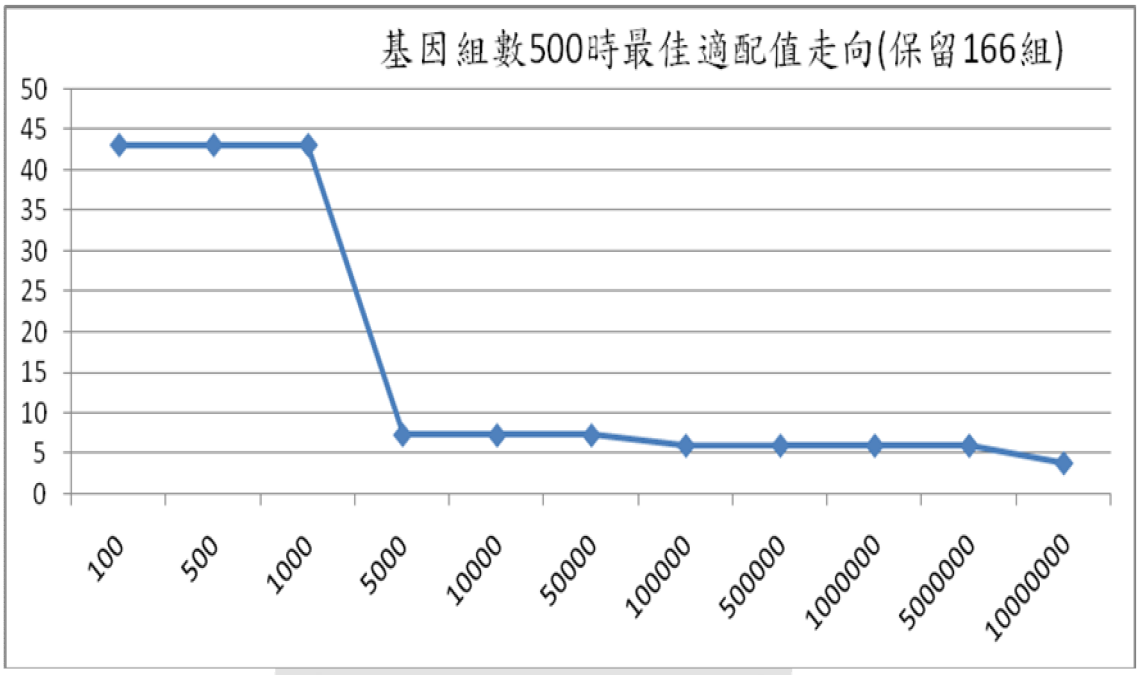

圖 4-7(b)基因組數 500 時(保留 166 組)最佳適配值走向

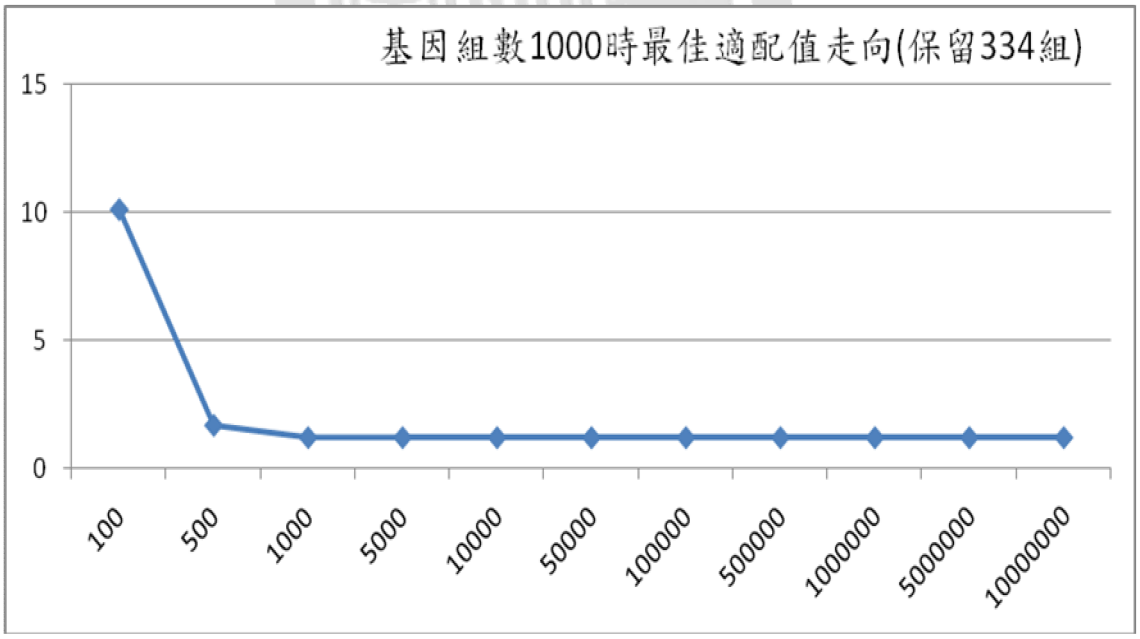

圖 4-7(c)基因組數 1000 時(保留 334 組)最佳適配值走向

從圖 4-7(a)、4-7(b)、4-7(c)得知，在減少保留基因組數目後，不僅在收斂的程度上加大，也讓適配值在循環次數不大時就已經逼近區域最佳解，而產生的基因組數的大小，也決定了初始適配值。

第 五 章

結 論

為使光學鏡片執行消色差動作而定立的演算公式，再套入至基因演算法程式運算後，從實驗結果可看出其最終解不斷收斂直至能有效的鎖定區域中的最佳解，使該演算公式能對消色差動作成立，該演算成事的動作要求，是成功的，然而基因演算法卻不只能運算公式，在其他領域方面也有出色表現，而依此消色差數值所製作出來的透鏡，更可以用來當作攝影鏡頭，並且影響焦距、解析度、以及像素，而透過模擬的結果，也證明該光學鏡組是能有效消散色差，達到做為上述產品的條件。

參考文獻：

[01]幾何光學 耿繼業,何建娃編著　　全華出版

[02]http://zh.wikipedia.org/w/index.php?title=%E6%B6%88%E8%89%B2%E5%B7%AE%E9%80%8F%E9%8F%A1&variant=zh-tw

[03]http://zh.wikipedia.org/wiki/%E8%A4%87%E6%B6%88%E8%89%B2%E5%B7%AE%E9%80%8F%E9%8F%A1

[04]國立成功大學建築研究所博士班　基因演算法之基本概念、方法與國內相關研究概況　黃衍明

[05]文化大學應用數學系 演化式計算下篇 基因演算法以及三種運用實例　林豐澤

[06]http://el.mdu.edu.tw/datacos//09422332004A/Ch4%20%E5%9F%BA%E5%9B%A0%E6%BC%94%E7%AE%97%E6%B3%95.pdf

[07]實踐大學資訊管理系 基因演算法概述　李建國

[08]資料結構使用 C 語言　蘇維雅譯-Ellis Horowitz Sartaj Sahni Susan Anderson-Freed　　松崗出版

[09] http://cc.chit.edu.tw/~ven/mis1/ch09-2.ppt

[10] http://www.csie.ntu.edu.tw/~ds/ppt/ch7/sld001.htm

[11]程式設計藝術第四版　吳國梁譯-DEITEL&DEITEL 全華出版