第三章　遺傳演算法

在本章節將介紹遺傳演算法或稱基因演算法(Genetic Algorithms,簡稱為 GAs 或 GA）的基本原理及應用。分五節進行簡介，第一節：遺傳演算法的起源；第二節：遺傳演算法的特點；第三節：遺傳演算法的基本原理；第四節：遺傳演算法的相關技術；第五節：遺傳演算法的應用；第六節：第三章總結。

3.1　遺傳演算法的起源

演化論是生物學最基本的理論之一[21,22]。所謂「演化」或「進化」(Evolution）是指生物在變異、遺傳與自然選擇作用下的演變發展，物種淘汰和物種產生過程。演化論最早是由查爾斯·達爾文（Charles Darwin）於 1859 年提出的，在其名著「物種起源」（The Origin of Species）有詳細的論述，亦為首次提出自然選擇或稱天擇（Natural Selection）是演化的一個機制。

而遺傳演算法（Genetic Algorithm）或稱基因演算法，主要就是依據達爾文所提出的「物競天擇，適者生存，不適者淘汰」的生物演化法則而來。遺傳演算法經由美國密西根大學（University of Michigan）的約翰·賀蘭（John Holland）教授及其學生利用生物模擬技術的啟發，創造出一種基於生物遺傳和進化機制的適合於複雜系統優化的自適應機率優化技術－遺傳演算法。於 1967 年，Holland 的學生 Bagley 在其博士論文中提出了「遺傳演算法」一詞，發展了選種(Selection)、複製（Reproduction）、交配（Crossover）、突變（Mutation）等遺傳算子。1975 年出版了第一本系統論述遺傳演算法與人工自適應系統的著作「Adaptation in Natural and Artificial Systems」，後來在其子弟兵 D. E. Goldberg、De Jong 等努力之下，遺傳演算法開拓出一片新的領域並蓬勃發展。

3.2 遺傳演算法的特點

上一節大略介紹遺傳演算法的起源後，根據達爾文的演化理論，生物演化表現是一代比一代更適應該物種的生活環境，而不是朝著一個已知的最佳解移動。遺傳演算法則是把生物界演化的機制抽象出來，應用在學習適應、乃至於搜尋最佳解的問題上面，讓整個系統朝著更佳的方向自我演化。遺傳演算法是一通用性的學習和最佳解找尋工具，除學術界的研究，亦廣泛的運用於各領域上。

遺傳演算法利用生物界的自然選擇（Natural Selection）和自然遺傳機制的隨機搜尋（Random Searching Algorithms）。其與傳統的算法不同，大多數古典的最佳化算法是基於一個單一的度量函數（Measure Function）或稱評估函數的梯度或較高次統計，以產生一個確定性的試驗解序列。在求解最佳化的問題，有各種不同的演算方法被使用。例如：Newton-Kantorovich Method, Modified Gradient-Inversion Algorithms, Conjugate-Gradient Method....等。這些方法主要是利用數值迭代的技巧來加以求解。這些方法的缺點為當初始猜測值與精確值相差甚遠時，其所得之解為局部最佳解（Local Optimal Solution），而非全域最佳解（Global Optimal Solution）。而遺傳演算法不依賴於梯度（Gradient）的訊息，而是通過模擬自然進化過程來搜尋最佳解（Optimal Solution），採用多點搜尋的方式，將所有搜尋的區域編碼，選擇數個初始猜測值，根據每個猜測值對最佳化目標的適應性值，進行選種、複製、交配、突變的運算以達到逃離局部最佳解的目標。然而，多點搜尋所伴隨而來的問題，就是計算量的增加。因此，如何增進搜尋的效率，以節省運算時間，是十分重要的問題。

遺傳演算法在解決最佳化問題過程中有如下特點[7,10]：

1.遺傳演算法在適應函數（Fitness Function），亦稱代價函數（Cost Function），在選擇不當的情況下，僅可能收斂於局部最佳，而不能達到全域最佳。

2.對於動態數據，用遺傳演算法求最優解比較困難，因染色體種群可能

過早收斂，而對以後變化的數據不再產生變化。針對這個問題，學者提出一些方法增加基因的多樣性，防止過早的收斂。其中一種是觸髮式超級突變，就是當染色體群體的質量下降、彼此的區別減少時增加突變機率；另一種叫隨機外來染色體，是偶爾加入一些全新的隨機生成的染色體個體，增加染色體多樣性。

3.選擇過程很重要，但交配和突變的重要性存在爭議。一種觀點認為交配比突變更重要，因為突變僅僅是保證不丟失某些可能的解；而另一種觀點則認為交配過程的作用只不過是在種群中推廣突變過程所造成的更新，對於初期的種群來說，交配幾乎等效於一個非常大的突變率，而這麼大的突變很可能影響進化過程。

4.遺傳演算法即使處於複雜的解空間中，亦可快速找到良好的解。

5.遺傳演算法並不一定總是最好的優化策略，優化問題要具體情況具體分析。故在使用遺傳演算法的同時，可嘗試其他演算法。

6.遺傳演算法不能解決「大海撈針」的問題。所謂「大海撈針」問題就是沒有一個確切的適應度函數，以便表達個體好壞的問題，遺傳演算法對這類問題無法找到收斂的方向。

7.對於具體的優化問題，調節遺傳演算法的參數可能會有利於更快的收斂，這些參數包括個體數目、交配律和突變律。例如太大的突變律會導致丟失最優解，而過小的突變律會導致演算法過早的收斂於局部最佳點。

8.適應度函數對於演算法的速度和效果也很重要。

3.2.1 遺傳演算法的優點

遺傳演算法具有下列的優點，故常被用來作為最佳化問題（Optimal Solution）解決採用的演算法，其優點如下：

1.對可行解表示的廣泛性

遺傳演算法的處理對象不是參數本身，而是針對那些通過參數及進行編碼得到的基因個體。此編碼操作使得遺傳演算法可以直接對結構對象進行操作。所謂結構對象，泛指集合、序列、矩陣、圖、鏈和表等各種一維或二維甚至多維結構形式的對象。這一個優點使得遺傳演算法具有廣泛的應用領域。

2.群體搜尋特性

許多傳統的搜尋方法都是單點搜尋，這種點對點的搜尋方法，對於多峰分佈的搜尋空間常常會陷入局部的某個單峰的極值點，而這些峰值往往都是局部最佳解（Local Optimal Solution），而非全域最佳解（Global Optimal Solution）。相反地，遺傳演算法採用的是同時處理群體中多個個體的方法及同時對搜尋空間中的多個解進行評估。這一個優點使遺傳演算法具有較好的全域搜索性能，也使得遺傳演算法易於並行化。

3.不需要補償信息

遺傳演算法僅用適應函數的數值來評估基因個體，並在此基礎上進行遺傳操作。更重要的是，遺傳演算法的適應函數不僅不受連續可微的約束，且其定義域可以任意設定。適應函數唯一的要求，就是編碼必須與可行解空間對應。由於限制條件的縮小，使得遺傳演算法的應用範圍大大擴展。

4.內在啟發式隨機搜尋特性

遺傳演算法不是採用確定性規則，而是採用概率的變遷規則來指導它的搜尋方向。概率僅僅是作為一種工具來引導其搜尋過程朝著搜尋空間的更優化的解區域移動。表面上雖為一種盲目搜尋方法，實際上有其明確的搜尋方向，具有內在的並行搜尋機制。

5.遺傳演算法在搜尋過程中部容易陷入局部最佳

即使在所定義的適應函數是不連續的、非規則的或是有雜訊的情況下，也能以很大的概率找到全域最佳解。

6.遺傳演算法採用自然進化機制來表現複雜的現象，能夠快速可靠地解決求解非常困難的問題。

7.遺傳演算法具有固定的並行性與並行計算的能力。

8.遺傳演算法具有可擴展性，易於與別的技術混和一起使用。

3.2.2 遺傳演算法的缺點

遺傳演算法亦存在下列的缺點：

1.編碼不規範及編碼存在表示的不準確性。

2.單一的遺傳演算法編碼不能全面性將最佳化問題的約束表示出來。考慮約束的一個方法就是對不可行解採用閾值（Threshold），這樣將會使計算的時間增加。

3.遺傳演算法通常的效率比其他傳統的最佳化方法低。

4.遺傳演算法容易出現過早收斂。

5.遺傳演算法對算法的精確度、可信度、計算複雜性等方面，尚無有效的定量分析方法。

3.3 遺傳演算法的基本原理

遺傳演算法的基本流程圖（Flowchart）如圖 3-1 所示。其內容與步驟進行說明如下：

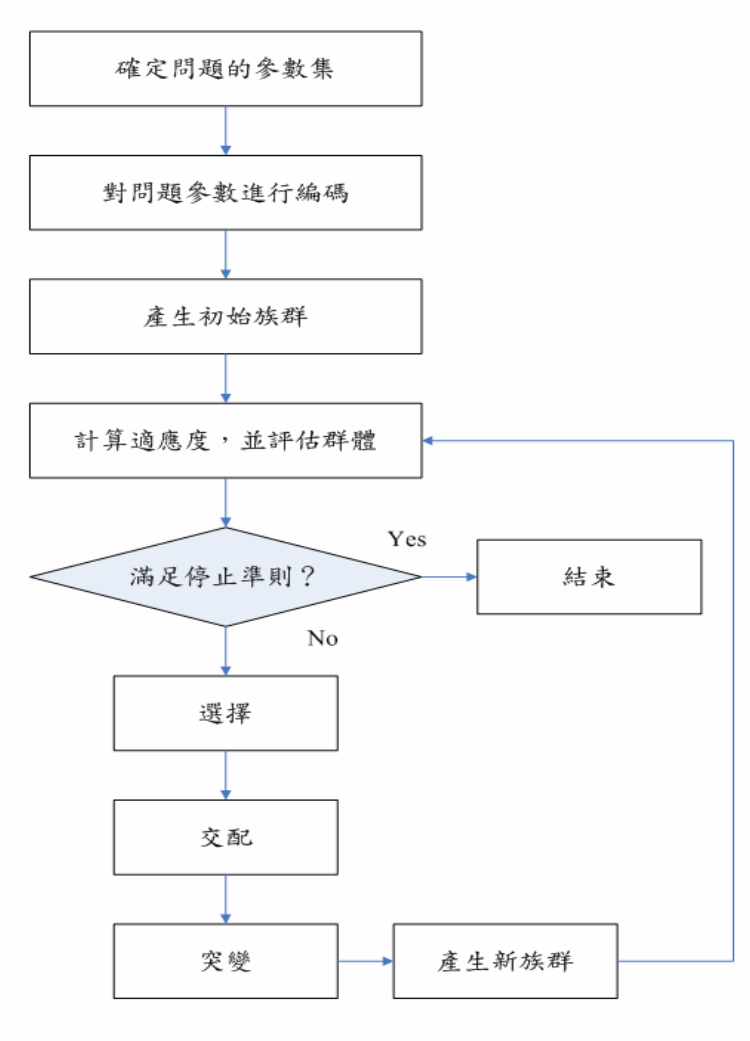

圖 3-1 遺傳演算法的基本流程圖

透過流程圖的作法即為完成基本遺傳演算法，亦稱為標準遺傳演算法或簡單演算法（Sample Genetic Algorithm，縮寫為 SGA），是遺傳演算法裡面最基礎也是最簡單的。

3.3.1 確定問題的參數

首先先定義問題所需的參數（Parameters），若無確定此問題的參數，將無法進行編碼；如果把所有參數都編碼，將會耗費很多計算時間與浪費記憶體空間，而且有些參數是不需進行編碼步驟的，所以第一步就是要先確定「所需的參數」。

在遺傳演算法裡，最佳化問題的解，即稱為個體（Individual），表示為一個參數列表，叫做染色體（Chromosome）或基因串（Gene String）。染色體一般被表達為簡單的字元串（Bit String）或數字串（Number String）。

3.3.2 對問題參數進行編碼

第二步即對所選定的參數進行編碼（Encoding），而編碼方式又有很多種，如果選擇編碼成二進位數（Binary），則稱為二進位遺傳演算法（Binary Genetic Algorithms，簡稱 BGA），如果基因參數直接用浮點數（Floating Point）所組成的遺傳演算法，則稱之為實數遺傳演算法（Real Genetic Algorithms，簡稱為 RGA），也可以稱為浮點遺傳演算法（Floating Point Genetic Algorithms，簡稱為 FPGA）。一般的實數轉換成二進位，採用直接二進位轉換（Straight Binary），例如 x 由 0 至 10，用位元長度（Bit Length）為 8 的二進位編碼，則十進位（Decimal）與二進位的關係如下：

$$0 \Rightarrow 0000\ 0000 = 0$$
$$10 \Rightarrow 1111\ 1111 = 255$$

那麼 5 大約是 fix(5/10×255)=128，亦即1000 0000。而實際上，1000 0000所代表的實數為10×(128/255)=5.0196，而非 5。因二進位有 256 個，故對應該實數也只有 256 個，而不是無限多個。此稱為實數被量化（Quantization）。位元長度越大則越能代表的實數個越數越多，也表示越精確。相對的缺點就是編碼轉換時間越長，更增加程式的執行時間。一般而言，位元長度通常使用到 20，並可依照實際問題的解空間範圍來增加長度，但不超過 50 位元為原則，因為過高亦無法改善解析度（Resolution）。上述的編碼即為染色體（Chromosome）上的一小段基因碼。轉換公式如下：

$$X = U_1 + (\sum_{i=1}^{k} b_i \cdot 2^{i-1}) \cdot \frac{U_2 - U_1}{2^k - 1} \qquad (3.1)$$

其中 $[U_1, U_2]$ 分別表示解空間的下界（Lower Bound）與上界（Upper Bound），

k 表示編碼的位元長度，$\sum_{i=1}^{k} b_i \cdot 2^{i-1}$ 為二進位轉換成十進位，b_i 表示第 i 個位元的位元值，X 則是轉換後的基因實數值。

例 3-1

設 x 的範圍 $[1,4]$，現在用 6 位元長度的二進制對 x 進行編碼，可得 $2^6 = 64$ 個二進位字串如下：

$$\begin{array}{c} 000000,\ 000001,\ \cdots,\ 000111 \\ 001000,\ 001001,\ \cdots,\ 001111 \\ \vdots \qquad \ddots \qquad \vdots \\ 111000,\ 111001,\ \cdots,\ 111111 \end{array}$$

對於任何一個字串，只要代入式（3.1），即可得對應的解碼（Decoding）。例如現有一個 $x=100101$，它對應到二進位轉十進位的值 $\sum_{i=1}^{k} b_i \cdot 2^{i-1} = \sum_{i=1}^{6} 1+0\times 2+1\times 2^2+0\times 2^3+0\times 2^4+1\times 2^5 = 37$，則對應的基因實數值 $X = 1+37\times\frac{4-1}{2^6-1} = 2.7619$。

採用二進位編碼的遺傳演算法進行數值最佳化時，可以通過改變編碼長度，協調搜尋精確度（Precision）與效率（Efficiency）之間的關係。如 Schraudolph 和 Below 提出參數動態編碼（Dynamic Parameter Encoding，簡稱 DPE）的策略，類似搜尋空間尺寸變換的方法。當穩定的基因位數超過某個數值後，利用變焦操作提高編碼精確度。

另二進位編碼還有一些變種，如格雷碼（Gray Code），二倍體（Diploid）編碼等，對某些問題類型的 GA 求解以有人做了大量試驗，並提出一些針對性的參數編碼原則。表 3-1 為四個位元長度的二進位編碼、格雷碼與十進位的對應關係。二進位編碼與格雷碼各有其優點，無法明確說明選用哪種編碼方式較佳，其比較如下：

1.二進位編碼的優點：

（1）編碼、解碼操作簡單容易。

（2）交配、突變等遺傳操作便於實現。

（3）符合最小字符（Alphabet）集編碼原則。

（4）便於利用模式定理（Pattern Theorem）對演算法進行理論分析，因為模式定理是以二進位編碼為基礎的。

2.格雷碼編碼的優點：

（1）便於提高遺傳演算法的局部搜尋能力。

（2）交配、突變等遺傳操作便於實現。

（3）符合最小字符集編碼原則。

（4）便於利用模式定理對演算法進行理論分析。

表 3-1 二進位編碼與格雷碼和十進位對應關係

十進位數	二進位碼	格雷碼	十進位數	二進位碼	格雷碼
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

3.3.3 產生初始族群

初始群體（Initial Population）中的個體一般而演是隨機（Random）生成一定數量的個體，有時候操作者也可以對這個隨機產生過程進行干預，播下

已經部分優化的種子。在不具有關於問題解空間知識的情況下，很難判定最佳解的數量及其在可行解空間中的分佈狀況；通常問題解空間均勻取樣，隨機生成一定數目的個體（通常是群體規模的 2 倍，即 $2n$），然後從中挑出較好的個體構成初始群體。對於二進位編碼，染色體位於字串上的每一個基因在 $\{0,1\}$ 隨機均勻選擇，假設編碼長度為 k，總共有 n 個變數需要編碼，所以群體初始化至少需要 $k \times n$ 次隨機取值，一共會有 $2^{k \times n}$ 個物種數。

3.3.4 計算適應度，並評估群體

在每一代（Generation）中，完成初始族群取選後，須針對所有被選取的個體，計算其適應能力（Fitness），做為接下來選種的依據。每一個個體都被評估（Evaluate）得到一個適應能力，相當於系統的性能指標（Performance Index，簡稱為 PI）。族群中的個體被按照適應度排序（Rank），適應度高的在前面。這裡的高是相對於初始的種群的低適應度來說的。如果用代價函數（Cost Function）來當評估標準，則需看是要實際問題是要求最大值（Maximum）還是最小值（Minimum）的問題，求最大值時，代價函數值越大越好；然而在求最小值時，代價函數則是越小越好。在進行評估時，須先清楚評估問題是否為代價函數，選擇錯誤的話，那將會產生與預期相反的結果。若是有一個明確的適應函數，例如：$\mathrm{f}(x) = x^3 + 4x^2 + 5x + 6$，$x \in [-100,100]$，要求最小值，那 x 就是所要進行編碼的參數，$\mathrm{f}(x)$ 就是適應函數，在本題中 $\mathrm{f}(x)$ 的值是越小越好，且不可以小於零，所以適應函數可以表示為式（3.2）：

$$\text{fitness} = \frac{1}{|\mathrm{f}(x)|} \tag{3.2}$$

3.3.5 停止準則

停止準則（Terminative Rule），亦稱之為終止條件（Terminative Condition）。一般的停止準則有以下幾種：

1.進化次數限制。

2.計算耗費的資源限制（例如計算時間、計算佔用的記憶體等）。

3.一個個體已經滿足最小值的條件，即最佳解（最佳的意義，視問題的要求而定，可以是該函數的極值、方程式的根等）已經找到。

4.適應度已經達到飽和，繼續進化不會造成適應度更好的個體。

5.人為干預。

6.以及以上兩種或更多種的組合。

只要滿足上列的任何一項標準，遺傳演算法的迴圈就會停止運算，如果沒有達到任何一項，則繼續下一個步驟。通常初始族群無法一開始就符合終止條件，必須經過幾個世代的迭代之後才有機會達成。

3.3.6 選擇

當沒有滿足上一個步驟的終止條件時，就會進入這個階段。選擇（Selection）也可以稱為複製（Reproduction），是在群體中選擇生命力強的個體產生新群體的過程，這些下一代的新個體稱為子代（Offspring）。遺傳演算法使用選擇運算子（Selection Operator）來對群體中的個體進行優勝劣汰操作：根據 3.3.4 節的適應函數評估出每一個個體的適應度值大小選擇，適應度較高的個體會被挑選出來複製到下一代群體中的機率（Probability）比較大，適應度低的自然被選出來的機率就比較小。這樣可以使得群體中個體的適應度不斷的往最佳解的方向移動，而不是盲目搜尋。選擇操作的主要目的是避免有遺傳訊息的遺失，並提高全局收斂性（Global Astringency）和計算效率。選擇運算子若不適當，將會造成群體中相似度（Similarity）值相近的

個體增加，使得子代與父代（Paternal Generation）個體相近，導致演化停滯不前；或使得適應度偏大的個體誤導群體的發展方向，使得遺傳失去多樣性，產生早熟問題。表 3-2 為一些常見的選擇運算子[7,9,10,14]。

表 3-2 選擇運算子

編號	名稱	特點	備註
1	輪盤法	選擇誤差較大	GA 成員
2	隨機競爭法	比輪盤法好	
3	最佳保留法	保證迭代終止結果， 為歷代最高適應度個體	
4	確定式法	選擇誤差更小，操作容易	
5	無回放式 餘數隨機法	誤差最小	應用較廣
6	均勻排序法	適應度大小差異程度與正負無關	
7	最優保存策略	全局收斂，提高搜尋效率， 不宜於非線性強的問題	
8	排擠法	提高群體的多樣性	
9	聯賽競爭法	比隨機競爭法更大規模的選擇方式	

資料來源：本研究整理

另說明一些常用的選擇運算子。

1.輪盤法

輪盤法（Roulette Wheel Selection）是一種回放式隨機取樣法。這方法類似賭博的賭輪盤，把一個輪盤分成若干扇形，面積越大的編號，越容易中獎，因此獎金會比較低。若以適應函數的觀點來看，適應度越大者所佔面積就會比較大。而輪盤法就是由此衍生出來的法則，這也是選擇運算子裡最基本也是最簡單的方法。此方法首先要計算每一個體的適應度或可直接從 3.3.4 節步驟取得，然後計算出此適應度在群

體適應度總和中所佔的比例，表示該個體在選擇過程中會被選中的機率。選擇過程就如同生物進化過程中的「適者生存，不適者淘汰」的觀念，並且保證優良基因會遺傳給下一代的個體。

對於給定的規模為 n 的群體 $P=\{a_1,a_2,\cdots,a_n\}$，個體 $a_j \in P$ 的適應度為 $f(a_j)$，其選擇機率為

$$p_s(a_j)=\frac{f(a_j)}{\sum_{i=1}^{n} f(a_j)}, \quad j \in 1,2,\cdots,n \qquad (3.3)$$

式（3.3）決定後代族群中個體的機率分佈。經過選擇操作生成用於繁殖的交配池（Mating Pool）的緩衝區中，等待進一步的繁衍。父代族群中個體生存的期望數目為：

$$P(a_j)=n \cdot p_s(a_j), \quad j \in 1,2,\cdots,n \qquad (3.4)$$

當群體中個體適應度的差異非常大時，最佳個體與最差個體被選擇的機率之比也將按指數（Exponent）增長。最佳個體在下一代的生存機會將會顯著增加，而最差個體的生存機會將會被剝奪。當前群體中的最佳化個體將迅速的充滿整個群體，導致群體的多樣性迅速降低，遺傳演算法也就過早地喪失了進化能力，這就是早熟現象（Premature Appearance）。這是適應度比例選擇容易出現的問題。

例 3-2

設有一函數 $f(x)=x(15-x)+1$，x 的範圍 $[0,15]$，求 $f(x)$ 的最大值，現在用 4 位元長度的二進制對 x 進行編碼，可以得到 $2^4=16$ 個二進位字串如下表 3-3，本題所有 x 值帶入函數皆為正數，直接用 $f(x)$ 的值當作適應度：

表 3-3 所有的基因編碼與適應度

x值	編碼	適應度	百分比	x值	編碼	適應度	百分比
0	0000	1	0.18%	8	1000	57	9.90%
1	0001	15	2.60%	9	1001	55	9.55%
2	0010	27	4.69%	10	1010	51	8.85%
3	0011	37	6.42%	11	1011	46	7.81%
4	0100	45	7.81%	12	1100	37	6.42%
5	0101	51	8.85%	13	1101	27	4.69%
6	0110	55	9.55%	14	1110	15	2.60%
7	0111	57	9.90%	15	1111	1	0.18%

假設今天隨機產生出四組基因，0100，0111，1101，1111，可得下表 3-4，以及圖 3-2(a)與(b)輪盤法的圖形。

表 3-4 隨機產生的基因編碼與適應度

x值	編碼	適應度	百分比
4	0100	45	34.62%
7	0111	57	43.85%
13	1101	27	20.77%
15	1111	1	0.76%
總和		130	100%

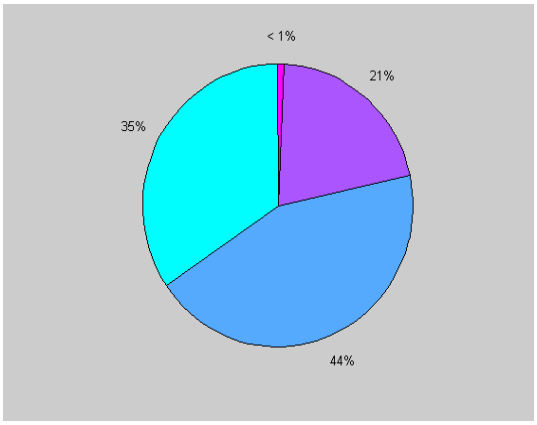

圖 3-2 (a)輪盤法的適應度百分比圖

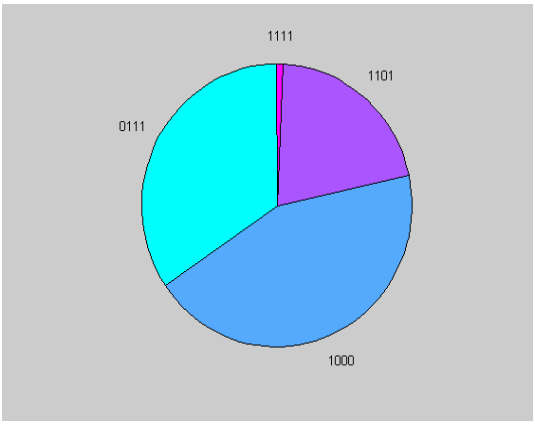

圖 3-2 (b)輪盤法的適應度對應編碼圖

從圖 3-2(a)和(b)中看出，適應度所佔比例以及其相對應的編碼。可產生一個亂數，如果其值介於[0,1]，那就會選中其對應的編碼1111，以此類推的範圍[1,22]會對應到1101，[22,66]會對應到0111，[66,100]會對應到0100。x值為 8 的這組佔有的面積最大，用數學分析函式 $f(x)=x(15-x)+1$的一次微分值為 $f'(x)=15-2x=0$之處有極大值出現，得到$x=7.5$。因此題因為編碼不夠精細故得不到 7.5 這個數值，但是根據遺傳演算法的特性，最後這個族群將會被0111和1000兩組基因給填滿整個族群空間。

2.隨機競爭法

隨機競爭法（Stochastic Tournament）與輪盤法的基本原理是一樣。隨機競爭法中，是按照輪盤法上面的機率為依據，隨機取得兩個亂數，立刻加以比較，把適應函數比較好的基因保留至新族群，較差的就淘汰掉，這個程序（Procedure）會一直持續到交配池被填滿為止。因為用亂數產生難免有一定的機率成分在，如果挑中之後立刻競爭，這樣就比較能篩選出較好的基因。

3.最優保留法

最優保留法（Elitism），原理與輪盤法一樣，但多了將當前群體中適應度最高的個體直接完整的複製到下一代的群體中。主要優點是能保證遺傳演算法終止時得到的最後結果是歷代出現過的最高適應度的個體。

4.無回放餘數隨機法

確保適應度比平均值高一些的個體能夠被遺傳到下一代群體中，所以它的選擇誤差比較小。可避免陷入族群過早就被一些目前適應度較高的個體給填滿，出現早熟現象。

3.3.7 交配

在生物界的自然進化過程中，兩個同源染色體（Chromosome）通過交配（Crossover）重組，形成了新的染色體，從而產生出新的個體或物種。交配是生物遺傳與進化過程中的一個重要環節。模仿這個環節，遺傳演算法則是使用交配運算子（Crossover Operator）來產生新的個體。交配又有人稱為重組（Recombination），依照所設定的一個機率值稱為交配率（Crossover Rate），通常設定在 0.6～1，不可以設為 1，若設為 1 即等於每一組都完全交配。交配率太高或太低都不好，太高會造成每一組挑選出來的基因都進行交配動作，而沒辦法保有原本族群一些基因的特性。太低則又造成過多的舊有族群裡的個體在新的世代中，造成演化緩慢不前。設定交配率亦可使用動態交配率（Dynamic Crossover Rate）。最常用的交配方式有四種，單點交配（Single-point Crossover）、兩點交配（Double-point Crossover）、多點交配（Multi-point Crossover）以及均勻交配（Uniform Crossover）。

1.單點交配

簡稱為簡單交配，指在個體編碼字串中，只隨機設置一個交配點，依設定的點上互相交換兩個配對個體的部分基因編碼。單點交配的實際執行過程如下：

（1）對個體進行兩兩隨機配對，若群體大小為N，則共有[N/2]對相互配對的個體組。

（2）對每一對相互配對的個體，隨機設置某一個位置為交配點，若染色體長度為L，則有L−1個可能的交配點位置。

（3）對每一對相互交配的個體，依設定的交配率在其交配點處相互交換兩個個體部分基因編碼，從而產生出兩個新的個體。

圖 3-3 所示為單點交配運算的示意圖。

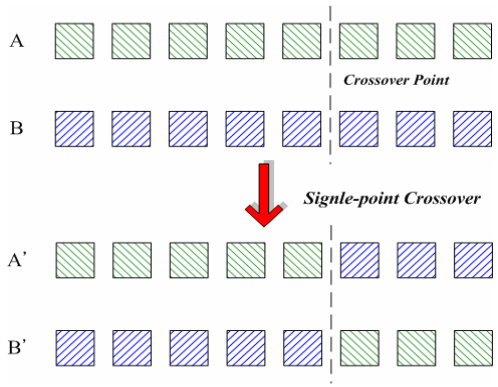

圖 3-3 單點交配運算示意圖

2.兩點交配與多點交配

即在個體編碼字串中，只隨機設置兩個個交配點，依設定的點上互相進行部分基因交換。操作過程如下：

（1）在相互交配的兩個個體編碼串中，隨機設置兩個交配點。交配點位置不重複，重複就和單點交配意思一樣，這樣做兩點交配就失去意義。長度設為L，則有L(L−1)/2種可能的交配點配置方式。

（2） 交換兩個個體在所定的兩個交配點間，做部分基因編碼交換。圖 3-4 所示為兩點交配的示意圖形。

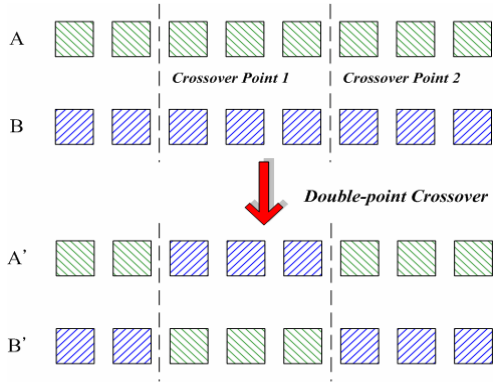

圖 3-4 兩點交配運算示意圖

將單點交配與兩點交配的概念加以推廣，可得到多點交配的概念。多點交配是指在個體編碼字串中隨機設置多個交配點後，再進行基因交換動作。多點交配也稱為廣義交配。一般情況下是不會使用多點交配的，因其可能破壞一些好的個體。事實上，隨著交配點數的增加，個體的結構被破壞的可能性也逐漸增大，如此一來即很難有效的保存較好的個體，將影響到遺傳演算法的性能。

3.均勻交配

指兩個配對個體的每個基因都已相同的交配率進行交換。均勻交配實際上可屬於多點交配的範圍，具體運算可以通過設置一個與基因長度一樣的二進位亂數做遮蔽（Mask）之用。均勻交配的主要操作過程與示意圖如下：

（1） 隨機產生一個與個體編碼長度相等的遮蔽字串（Mask String）W ， W=$w_1 w_2 \cdots w_i \cdots w_L$ ，L 為個體編碼長度。

（2） 若 $w_i = 0$ ，則相對應的父代基因保留；若 $w_i = 1$ ，則相對應的父代基因上下交換。

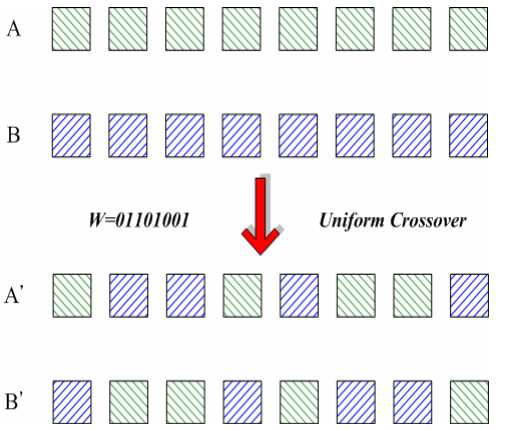

圖 3-5 (a) 均勻交配示意圖 w=01101001

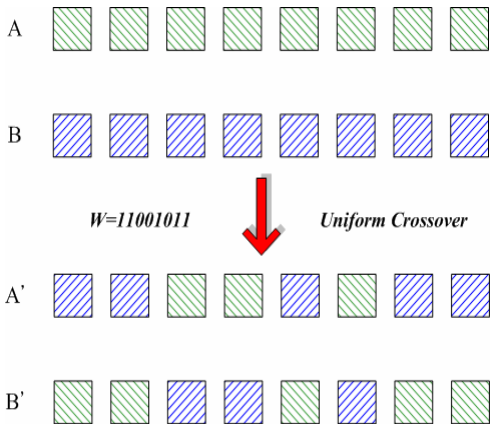

圖 3-5 (b) 均勻交配示意圖 w=11001011

3.3.8 突變

在生物界的遺傳與自然進化過程中，細胞分裂複製環節可能會因為某些偶然因素的影響而產生一些錯誤，這樣會導致生物的某些基因因為突變（Mutation），而產生出新的物種。這些突變的物種在生物界中大多數是不好的，也是適應性比較差的個體，但是也有少數個體是遠比現有的物種適應能力強的。故遺傳演算法的突變就是來自這個環節。一般遺傳演算法都有一個固定的突變率（Mutation Rate），通常是 0.01 至 0.08 或者更小，這代表突變發生的機率。根據這個機率，新個體的染色體隨機的突變，通常就是改變染色體的一個位元組（0 變到 1，或者 1 變到 0）。突變本身是一種隨機算法，但是與選擇及交配運算結合後，能夠避免由於選擇與交配運算造成的某些信息遺失，保證遺傳演算法的有效性。圖 3-6 為突變的示意圖。

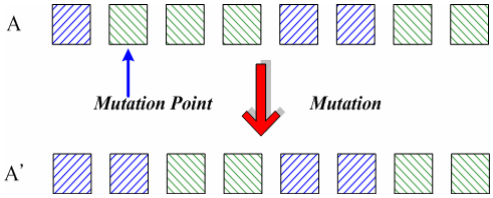

圖 3-6 突變示意圖

在遺傳演算法中使用突變主要有兩個目的：

（1）改善遺傳演算法的局部搜尋能力。遺傳演算法使用交配運算已從全局的角度出發找到一些較好的個體編碼結構，有助於求得接近問題的最佳解，但僅使用交配運算無法對搜尋空間的細節進行局部搜尋（Local Search）。這時若再使用突變來調整個體編碼串中的部分基因值，即可從局部的角度出發使個體更加逼近最佳解，進而提高遺傳演算法的局部搜尋能力。

（2）維持群體的多樣性，防止出現早熟現象。突變運算使得遺傳演算法在接近最佳解附近時能加速向最佳解收斂。

另介紹兩種突變的操作方法，適合於二進制編碼的個體和浮點數編碼的個體。

1.基本位突變

基本位突變（Simple Mutation）操作是指對個體編碼字串中以突變機率、隨機指定的某一位或是某幾位基因座上面做突變運算，其操作過程如下：

（1）對個體的每一個基因座，以突變機率指定為其突變點。

（2）對每一個指定的突變點，對其基因值做反運算（Inverse）或其他等位基因值來代替，從而產生出新的個體。

2.均勻突變

均勻突變（Uniform Mutation）操作是指分別用符合某一範圍內均勻分佈的隨機數，以某一較小的機率來替換個體編碼字串中各個體基因座上原有的基因值。其操作過程：

（1）依次指定個體編碼字串中的每個基因座為突變點。

（2）對每一個突變點，以突變機率從對應基因的選值範圍內取一隨機數來替代原有值。

均勻突變操作特別適合於遺傳演算法的初級運算階段，將使得搜尋點可在整個搜尋空間內自由地移動，增加群體的多樣性，使演算法處理更多的模式。

3.3.9 產生新世代

經過選擇、交配和突變一系列的過程，產生的新一代（New Generation）個體，不同於初始的族群（Initial Population），最好的個體總是有更多的機率被選擇去產生下一代，而適應度低的個體逐漸被淘汰掉，故整體的適應度將朝更佳的方向發展。本步驟完成後，再回到 3.3.4 的步驟：選擇的個體被評價，計算出適應度，如果沒有滿足終止條件，就繼續複製、交配，然後突變，產生新的世代。這樣的過程不斷的重複，直到終止條件滿足為止。

3.4 遺傳演算法的相關技術

本節將介紹遺傳程序、遺傳編程、互動式遺傳演算法、模擬退火演算法及禁忌搜索法相關的說明及技術[7,10,21-23]。

3.4.1 遺傳程序

遺傳程序（Genetic Procedure）是約翰・科拉（John Koza）與遺傳演算法相關的一個技術，在遺傳程序中，並不是參數優化，而是電腦程序優化。遺傳程序一般採用樹型結構（Tree-structures）表示電腦程序用於進化，而不是遺傳演算法中的列表或者數組。一般來說，遺傳程序比遺傳演算法慢，但同時亦可解決一些遺傳演算法解決不了的問題。

3.4.2 遺傳編程

遺傳編程（Genetic Programming，簡稱 GP），是一種從生物演化過程得到靈感的自動化生成和選擇電腦程序來完成用戶定義的任務的技術。從理論上講，人類用遺傳編程只需要告訴電腦「需要完成什麼」，而不用告訴它「如何去完成」，最終可能實現真正意義上的人工智慧：自動化的發明機器。

遺傳編程是一種特殊的利用進化演算法的機器學習技術，開始於一群由隨機生成的千百萬個電腦程序組成的「人群」，然後根據一個程序完成給定的任務的能力來確定某個程序的適合度，應用達爾文的自然選擇確定勝出的程序，電腦程序間也可模擬兩性組合、變異、基因複製及基因刪除等進化，直到達到預先確定的某個終止條件為止。

遺傳編程的首批試驗由史蒂芬·史密斯（Stephen F. Smith, 1980）和尼可·克拉姆（Nichael L. Cramer, 1985）發表。約翰·科拉於 1992 完成一本著名的書「遺傳編程：用自然選擇讓電腦編程（Genetic Programming：On the Programming of Computers by Means of Natural Selection）」介紹遺傳編程。

使用遺傳編程的電腦程序可以用很多種程式語言來寫成。早期的 GP 實現中，程序的指令和數據的值使用樹狀結構組織方式，故本來就提供樹狀組織形式的程式語言最適合與 GP 結合，例如約翰·科拉使用的 Lisp 語言。其他形式的 GP 也被提倡和實現，相對簡單的適合傳統程式語言的線性遺傳編程，如 Fortran、BASIC 與 C。結合商業化的 GP 軟體把線性遺傳編程和彙編語言結合來獲得更好的性能。

遺傳編程所需的計算量非常之大，故在 90 年代的時候只能用來解決一些簡單的問題。近年來，隨著遺傳編程技術自身的發展和中央處理器計算能力的指數級提升，GP 開始產生顯著的結果。例如在 2004 年左右，GP 在多個領域取得近 40 項成果：量子計算，電子設計，遊戲比賽，排序，搜索等等。這些電腦自動生成的程序中有些與 2000 年後人工產生的發明十分類似，甚至

有兩項結果產生了可以申請專利的新發明。GP 的理論取得重大發展，建立確切的 GP 機率模型（Probabilistic Models of GP）和馬爾可夫鏈模型（Markov Chain Models）已成為可能。遺傳編程實際上包含了遺傳演算法，故使用上比遺傳演算法適用的範圍更廣。

3.4.3 互動式遺傳演算法

互動式遺傳演算法（Interactive Genetic Algorithm，簡稱 IGA）是利用人工評價進行操作的遺傳演算法，一般用於適應度函數無法得到的情況。主要應用例如：演變的圖像、音樂、各式各樣的藝術性的設計和形式適合用戶的審美特選的領域，或者對運動員的訓練等。

3.4.4 模擬退火法

模擬退火法（Simulated Annealing，簡稱 SA）是解決全局優化問題的另一個可能選擇，是一種通用機率演算法（Probabilistic Meta-algorithm），用來在一個大的搜尋空間內找尋命題的最佳解。模擬退火是 S. Kirkpatrick, C. D. Gelatt 和 M. P. Vecchi 在 1983 年所發明。而 V. Černý 在 1985 年也獨立發明此演算法。

模擬退火來自冶金學（Metallurgy）的專有名詞退火。退火是將材料加熱後再經特定速率冷卻，目的是增大晶粒的體積，並且減少晶格（Crystals）中的缺陷。材料中的原子原來會停留在使內能有局部最小值的位置，加熱使能量變大，原子會離開原來位置，而隨機在其他位置中移動。退火冷卻時速度較慢，使得原子有較多可能可以找到內能比原先更低的位置。

模擬退火的原理也和金屬退火的原理近似：即通過一個解在搜索空間的隨機變動尋找最優點的方法：如果某一階段的隨機變動增加適應度，則被接受，而降低適應度的隨機變動，依一定的機率下，被有選擇的接受。這個機

率由當時的退火溫度和適應度惡化的程度決定，而退火溫度按一定速度降低。從模擬退火演算法看，最佳化問題的解是通過尋找最小能量點而得，非尋找最佳適應點求得；模擬退火也可運用於標準遺傳演算法裡。

3.4.5 禁忌搜索

禁忌搜索（Tabu Search，簡稱 TS）是一種現代啟發式演算法，由美國科羅拉多大學教授 Fred Glover 在 1977 年左右提出的，是一個用來跳脫局部最佳（Local Optimal）的搜尋方法。其先創立一個初始化的方案；基於此，演算法「移動」到一相鄰的方案。經過許多連續的移動過程，提高方案的品質。

3.5 遺傳演算法的應用

遺傳演算法提供一種求解複雜系統最佳化問題的通用框架，不依賴於問題具體的領域，對問題的種類有很強的強健性（Robustness），故廣泛被應用於許多學科。近十幾年來，遺傳演算法迅速的發展，目前在生物工程、化學工程、計算機科學、動態處理、人工智慧、建模與模擬、醫學工程、商業與金融等，為求解全域最佳化問題的有利工具。部份應用領域如下所述：

1.函數最佳化

函數最佳化（Function Optimization）是遺傳演算法的經典應用領域，亦為遺傳演算法進行性能評估的常用範例。可用各式各樣的函數來驗證遺傳演算法的性能。對於一些非線性、多模型及多目標的函數最佳化問題，亦可得到較好的結果。

2.自動控制領域

在自動控制（Automatic Control）領域中有很多與最佳化相關的問題需要求解，遺傳演算法已經在其中得到初步的應用，並顯示出了良好的效果。

例如，基於遺傳演算法的模糊控制器最佳化設計，用遺傳演算法進行航空控制系的最佳化，使用遺傳演算法設計空間交會控制器等。

3.旅行商問題

旅行商問題（Traveling Salesman Problem，簡稱 TSP）是經典的組合最佳化問題之一，以遠遠超過其本身的含意，成為一種衡量演算法優劣的標準。TSP 問題是採用非標準編碼遺傳演算法求解最成功的一例，用銷售員順序經過的程式名稱表示基因編碼。因使用標準交配產生的後代可能有重複或遺失的機因而成為非可行解，故提出了非標準的交配與突變方法。

4.資料探勘

資料探勘（Data Mining）是指從大型資料庫中提取隱含的、未知的，即具有潛在應用價值的訊息或模式，是資料庫研究中一個很有前瞻性的新領域，基於遺傳演算法的特點，可用於資料探勘中的規則開發。

其他領域的應用如表 3-5 簡略介紹。

表 3-5 遺傳演算法應用領域

應用領域	範例
控制	瓦斯管道控制，防避導彈控制
電路設計	VLSI 佈局，電腦網路的拓撲結構
影像處理	模式識別，圖像恢復
機器人	路徑規劃，機器人學習
組合最佳化	TSP 問題、背包問題
生產調度	生產規劃、任務分配
醫療	蛋白質結構分析
時間表安排	安排不衝突的課程時間表
汽車設計	材料選擇，減輕重量

資料來源：本研究整理

3.6 第三章總結

本章一開始介紹遺傳演算法的起源、基本原理及各式各樣的遺傳運算子，並列舉一些遺傳演算法可以使用的領域。因為遺傳演算法為本研究改良之解模糊關係方程式演算法於非線性最佳化問題之應用中會使用到，所以先於第三章做一個概括的介紹遺傳演算法。這樣於第四章應用時能有基本概念，不至於茫然不清。第四章應用遺傳演算法時就不再做一些基本描述。

從圖 3-2(a)和(b)中看出，適應度所佔比例以及其相對應的編碼。可產生一個亂數，如果其值介於[0,1]，那就會選中其對應的編碼1111，以此類推的範圍[1,22]會對應到1101，[22,66]會對應到0111，[66,100]會對應到0100。x值為 8 的這組佔有的面積最大，用數學分析函式$f(x)=x(15-x)+1$的一次微分值為$f'(x)=15-2x=0$之處有極大值出現，得到$x=7.5$。因此題因為編碼不夠精細故得不到 7.5 這個數值，但是根據遺傳演算法的特性，最後這個族群將會被0111和1000兩組基因給填滿整個族群空間。

2.隨機競爭法

隨機競爭法（Stochastic Tournament）與輪盤法的基本原理是一樣。隨機競爭法中，是按照輪盤法上面的機率為依據，隨機取得兩個亂數，立刻加以比較，把適應函數比較好的基因保留至新族群，較差的就淘汰掉，這個程序（Procedure）會一直持續到交配池被填滿為止。因為用亂數產生難免有一定的機率成分在，如果挑中之後立刻競爭，這樣就比較能篩選出較好的基因。