互補式最佳化

楊正宏
國立高雄應用科技大學
電子工程系
chyang@cc.kuas.edu.tw

蔡昇偉
國立高雄應用科技大學
電子工程系
1096305108@cc.kuas.edu.tw

莊麗月
義守大學
化學工程系
chuang@isu.edu.tw

摘要

在可接受的時間裡，找到問題的最佳解或近似最佳解是最佳化演算法存在的價值及目的。本文提出一種新的最佳化演算法，稱為互補式演算法，此演算法是以生態平衡為基礎，根據自然界的循環現象所建立的演算模式，模擬生態系統各司其職的法則來求解各種排列組合的最佳化問題。目前所有進化式的最佳化演算法皆以達爾文物競天擇的理論為基礎求最佳解，由於強調適者生存，因此有容易落入區域最佳解的缺點。有鑑於此，互補式演算法採取預防勝於治療的作法，改善傳統進化式演算法容易落入區域最佳解的問題。在本研究中，我們以標準的測試函數分別對進化式演算法與互補式演算法進行測試。實驗結果顯示，互補式演算法之尋優效率有明顯的優勢。

關鍵詞：互補式最佳化、物種交流、生態平衡。

Abstract

To find the global optimal or the approximate global optimal in acceptable time is the value and purpose of the optimization algorithm. This paper propose a new optimization algorithm that called complementary optimization (CO), it is based on the ecological balance, according to the cycle of nature established by the calculation model and simulation of the rules of the ecosystem perform their duties to solve many permutations and combinations problems. At present, all of the evolutionary algorithms are based on Darwinism, because stressed survival of the fittest, it is easy to fall into the local optimal. In view of this, CO use prevention is better than cure practices to improve the evolutionary algorithms easily fall into the local optimal problems. In this study, we respectively test CO and PSO by seven benchmark test functions. The experimental results show that CO had an obvious advantage.

Keywords: Complementary optimization; Exchange; Ecological balance.

1. 前言

最佳化的研究已被廣泛的應用在許多的工程問題甚至藝術設計領域中，例如電力控制、機械工程與時裝設計[1][4]等；而目前最常被廣泛應用的進化式最佳化演算法莫過於基因演算法和粒子族群最佳化演算法。然而從許多研究得知最佳化演算法往往會有落入區域最佳解的問題[7] [10] [12] [13]，如何避免在求解過程中落入區域最佳解，一直都是最佳化演算法研究上的重點。近年來，有許多改良的最佳化演算法陸續被發展出來，例如改良式基因演算法 (MA)[13] 、改良式粒子族群最佳化(IPSO)[7]、混合式基因演算法(HGA)[10]和禁忌基因演算法(Tabu-GA)[12]等。

本文的研究目的在於提出一種新的最佳化演算法，稱為互補式演算法，其求解機制是模擬生態系統間互相依存的生物與當地的環境所組成的平衡關係。生態系統是由英國生態學者 A.G. Tansley 在西元 1935 年所提出的概念[11]，生態系統為有機體(生物組成)和物理環境(非生物的組成部分)所組成的一個單一系統，藉由不同物理環境的生物互相交流使其系統平衡[11]。然而，生物種是生態系統中的功能單位，在互補式演算法中，物理環境代表欲求解的問題，而多種生物組成的群體則代表一個解，某一最佳組合能使該環境達到生態平衡的狀態，此一最佳組合也就是問題的最佳解。

一個健康的生態系統內，各種生物之間以及環境之間是存在一種平衡的關係，其中若有物種消失，則會導致系統失衡，嚴重的話，可能會帶來毀滅性的災難[9]。有鑑於此，互補式演算法在求解過程中會保留初始的所有生物種，並藉由物種交流的方式來產生多個生物種的排列組合使其生態系統能趨近平衡之狀態。

生態平衡是指生物種類的組成和數量比例與非生物環境保持相對穩定，而滿足生態平衡的條件在於生態系統中的生物個體數目、物種數目以及生態環境皆為恆定或以某一個值做小幅度的振動[11]。

本文的研究架構分為五部份：第一部份為互補式演算法之背景及簡介；第二部份為最佳化之相關研究探討；第三部份會詳細說明互補式演算法的流程及設計；第四部份為實驗結果與討論；第五部份為結論。

2.相關研究

目前常被廣泛應用的最佳化演算法，例如基因演算法、粒子族群最佳化演算法和蟻群演算法等，其靈感皆來自對自然現象的觀察；其他最佳化演算法還包括有禁忌搜尋法和貪婪演算法等，其主要目的皆是在可接受的時間內找出問題的最佳解或近似最佳解，上述方法介紹如下。

2.1 基因演算法

基因演算法是模擬進化論中物競天擇、適者生存的最佳化演算法，經由隨機初始多個可能的解，並將其編碼成固定長度的染色體，透過隨機輪盤法或優生學的方式，從中選取兩個染色體進行交配與突變的機制進而產生兩個新染色體，保留適應值較佳的染色體，淘汰適應值較差的染色體，以此持續選擇、交配與突變的流程來進行演化，追求最佳解[5]；然而在經過多次迭代後，染色體的排列組合會越來越相似，導致之後儘管再經過多次迭代，結果只會得到一組相同的最佳解。

2.2 粒子族群最佳化演算法

粒子族群最佳化演算法是科學家藉由觀察鳥類飛行與魚類活動的族群特性所發展出來的進化式演算法[8]，是藉由隨機方式產生多個可能的解，以粒子本身的最佳經驗和群體的最佳經驗做運算，其中所有粒子皆各自有收尋的區域，並把本身的收尋經驗記錄下來。對單一粒子而言，自己所有記錄中最好的適應值稱為個體最佳適應值；然而在粒子群中最好的個體最佳適應值則稱為群體最佳適應值。利用個體最佳適應值、群體最佳適應值與粒子移動之速度所構成的向量決定下次位置，粒子移動一次為迭代一次，以此持續的更新粒子位置，追求最佳解；然而粒子群在更新位置過程中，一但群體最佳適應值落入區域最佳解，便很難有機會再跳出。

2.3 蟻群演算法

蟻群演算法是最佳化演算法的一種，模擬螞蟻利用費洛蒙在巢穴與目的地間之間尋找一條最短路徑[3]，由於自然界的螞蟻無法依賴視覺尋找食物，因此會在所走過的路徑上留下費洛蒙，當其他螞蟻經過時，能藉由前一隻螞蟻的費洛蒙得知食物之所在。在初始情況下，每隻螞蟻找到同一食物的路徑皆不相同，在同一時間裡，距離較短的路徑相對的會有較多螞蟻走過，由於螞蟻會選擇費洛蒙濃度較高的路徑，因此隨著時間增加，其他螞蟻會有較高的機率選擇較短之路徑。

2.4 禁忌搜尋法

禁忌搜尋法所搜尋的方式不同於一般區域搜尋法，由於禁忌搜尋法可接受較差的解，並利用記憶結構紀錄已搜尋過的區域之歷史資訊，因此可以使其不易落入區域最佳解，其主要概念包括避免走重複路徑和導入不同的記憶結構與策略[12]。

2.5 貪婪演算法

貪婪演算法在求解最佳化問題中，是經過一步步的單向選擇來求解，每一步的選擇皆以最大利益為考量，但由於短視近利而往往失去了找到最佳解的優勢[2]。

3.研究方法

本研究所研發的互補式最佳化演算法，主要是藉由不同環境的生物群互相交流來尋找最佳的生態平衡關係。在此，為以文獻[6]的測試函數評估互補式最佳化的效能，其整體研究方法之架構分為以下部分說明：第一部份是說明演算法的編碼方式；第二部份說明生物群的交流機制；第三會說明演算法虛擬碼及其執行步驟流程；第四部份會說明平衡值之設計方法；最後的第五部份則會舉例說明在多維度中之互補演算法流程。

3.1 編碼方式

不同的環境(問題)有不同的生態系統(解答)，由於生態系統是由許多生物及其環境組成的，且滿足生態平衡的要件在於恆定的生物個體數(出生率與死亡率相等)、恆定的物種數及恆定的生態環境[11]，因此我們將生物群編碼

成二進碼型式，並根據環境的條件來設計目標函數，在本研究中稱之為生態平衡函數，生態平衡值可判斷該環境的生態系統是否健康。在此，每種生物皆以單一位元表示，多種生物所組成的生物群為一個二進碼型式的一維陣列，而每一生物群為單一生態，其中編碼為 1 的，表示該生物生存在此生物群裡；反之編碼為 0 的，則表示該生物沒有生存在此生物群裡，至於生物群編碼的長度 L 與生態的維度 D 則需要依據問題的本質及執行效率而定，其生物群的編碼與生態維度之示意圖如圖 1 所示。

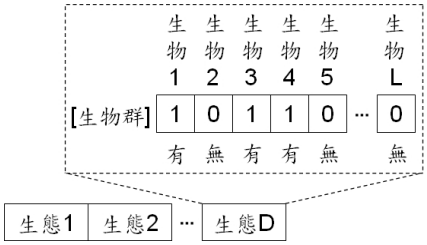

圖 1 生物群編碼與生態維度之示意圖

3.2 交流機制

互補式演算法的交流機制是指生物由某一環境移動至另一環境的過程[11]，其動作原理同於基因演算法的交配機制，如圖 2 所示。

圖 2 交流示意圖

3.3 執行步驟與虛擬碼

以下會詳細說明互補式演算法的整體流程及虛擬碼，其執行步驟依序如下：

步驟一：初始設定互補式演算法的參數，依序為生物群編碼長度 L、生態維度 D、交流次數 N 和迭代次數。

步驟二：隨機初始一個二進碼型式的一維陣列，在此稱為 A_D^1。

步驟三：產生 A_D^1 的 1 的補數陣列 B_D^1。

步驟四：A_D^1 與 B_D^1 透過隨機交流機制，產生 A_D^2 與 B_D^2；A_D^2 與 B_D^2 透過隨機交流機制，產生 A_D^3 與 B_D^3；依此類推直到 A_D^{N-1} 與 B_D^{N-1} 透過隨機交流機制，產生 A_D^N 與 B_D^N 後則停止交流，其中交流次數 N=$\lceil \log_2 L \rceil$。

步驟五：計算所有生物群的生態平衡值。

步驟六：選取生態平衡值最佳的生物群。

步驟七：判斷是否符合終止條件，若符合則終止演算法，反之則執行下個步驟。

步驟八：判斷所選取的最佳平衡值是否優於 A_D^1 的平衡值，若優於 A_D^1 的平衡值則取代 A_D^1 並回步驟三，反之則直接回步驟四。

Pseudo-code for Complementary Optimization

```
01: begin
02:   Randomly initialize a couple of
      complementary ecological A_D^1 and
      ecological B_D^1
03:   while (stopping criterion is not met)
04:     for N=1 to ⌈log2 L⌉
05:       The complementary ecological
          exchange the biota until A_D^{N-1}
          and B_D^{N-1} exchange the biota to
          create A_D^N and B_D^N
06:       Evaluate all of the ecological
          balance value
07:       Choose the best couple, which
          has the best balance value
08:       if the best couple is better than
          original, then original couple is
          replaced by the best couple
09:       end if
10:     next N
11:   next generation until stopping criterion
12: end
```

3.4 平衡值之設計方法

在本研究中，我們藉由標準的測試函數來驗證互補式演算法的效能，以標準的測試函數來模擬平衡函數，並說明如何應用互補式演算法來求解複雜的數學函數問題，由於互補式演算法中的生物群編碼為一個二進碼型式的一維陣列，而欲求解的平衡函數(測試函數)為十進碼型式，因此生物群必需經過適當的轉換與處理才能得到其生態平衡值，其中包含三個主要流程，依序為二進碼轉十進碼、X_i 產生器及平衡函數(測試函數)，以下會各別詳細說明轉換流程的功能及目的，其中算平衡值的設計方法的示意圖如圖 3 所示。

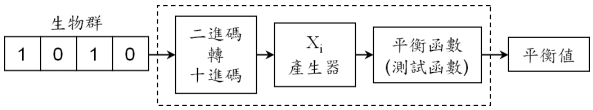

圖 3 平衡值之設計方法示意圖

3.4.1 二進碼轉十進碼

二進碼轉十進碼的目的主要是將二進碼型式的生物群轉為標準的測試函數所能讀取的十進碼型式，使函數能計算該生物群所組成之生態的生態平衡值。舉例來說，二進碼為 1001 的生物群轉為十進碼變成 9，接著輸入至平衡函數 $f=X_i^2$ 中計算，進而產生平衡值為 81 的結果。其中，若要以二進碼表示具有小數位數的十進碼，則需經過範圍放大的修正，其詳細過程請參閱 3.4.2 之 X_i 產生器處理機制。

3.4.2 X_i 產生器之設計

X_i 產生器是二進碼型式之生物群得以正確產生平衡值的關鍵。這裡所謂的 X_i 指的是平衡函數(測試函數)的變數，不同的平衡函數(測試函數)有不同的 X_i 範圍，例如著名的 De Jong 函數 $f=X_i^2$ 的 X_i 範圍為-5.12 到 5.12；函數 $f=100(X_{i+1}-X_i^2)^2+(1-X_i)^2$ 的 X_i 範圍為-2.048 到 2.048 [6]。而 X_i 產生器的主要功能是確保由二進碼轉為十進碼的值不會超過平衡函數(測試函數)所限制的 X_i 範圍。由於二進碼的生物群是與互為 1 的補數的生物群互相交流，例如生物群 0001 與生物群 1110 交流，因此有可能在轉為十進碼後的值會超過平衡函數(測試函數)的 X_i 範圍。舉例來說，X_i 範圍為 0 到 5，在精確度為個位數的情況下則需要以 3 位元來表示，若初始的生物群 A_D^t 為 001，經過轉碼後為 1，符合 X_i 所限制之範圍，但 A_D^t 的 1 的補數生物群 B_D^t 為 110，經過轉碼後為 6，超過 X_i 所限制之範圍，導致生物群 B_D^t 無法產生正確的平衡值。在此，為避免 X_i 超過限制範圍的情況發生，X_i 產生器提供一個妥善的處理機制，其處理機制的數學式如下：

令 $X_L=X_i$ 的下限值，$X_H=X_i$ 的上限值，X_i 範圍=$[X_L \sim X_H]$，s=解析度；d=放大率= $(X_H-X_L)*s$

$$X_i \text{ 的範圍} = [X_L \sim X_H]$$
$$= [(0 \sim (X_H-X_L)) + X_L]$$
$$= [\boxed{(d*0 \sim d*(X_H-X_L))} + d*X_L] / d$$

$\because$ 2^{N-1} 的範圍 $< \boxed{(d*0 \sim d*(X_H-X_L))} \le 2^N$ 的範圍; 取 N (N 位元)
$\therefore$ 2^N (N 位元)的範圍= $(0 \sim 2^N-1) \ge \boxed{(d*0 \sim d*(X_H-X_L))}$

$\because$ $\boxed{(d*0 \sim d*(X_H-X_L))} = \boxed{[(0 \sim 2^N-1)*(d*(X_H-X_L) / (2^N-1))]}$
$\therefore X_i$ 的範圍= $\{\boxed{[(0 \sim 2^N-1)*(d*(X_H-X_L) / (2^N-1))]} + d*X_L\} / d$
$= [X_L \sim X_H]$

在此我們以著名的 De Jong 函數 $f=X_i^2$ 的 X_i 範圍[-5.12~5.12]來說明 X_i 產生器的處理機制，詳如下：

其 X_L=-5.12，X_H=5.12，X_i 範圍=[-5.12 ~ 5.12]，s=1000(表示精確度到達小數 4 位)；d= (5.12-(-5.12))*1000=10240

$$X_i \text{ 的範圍} = [-5.12 \sim 5.12]$$
$$= [(0 \sim 10.24) + (-5.12)]$$
$$= [(10240*0 \sim 10240*10.24) + 10240*(-5.12)] / 10240$$
$$= [\boxed{(0 \sim 104857.6)} - 52428.8] / 10240$$

$\because$ 2^{17-1} 的範圍 $< \boxed{(0 \sim 104857.6)} \le 2^{17}$ 的範圍; 取 17 (17 位元)
$\therefore$ 2^{17} (17 位元)的範圍= $(0 \sim 2^{17}-1) \ge \boxed{(0 \sim 104857.6)}$

$\because$ $\boxed{(0 \sim 104857.6)} = \boxed{[(0 \sim 2^{17}-1)*(104857.6 / (2^{17}-1))]}$
$\therefore X_i$ 的範圍= $\{\boxed{[(0 \sim 2^{17}-1)*(104857.6 / (2^{17}-1))]} - 52428.8\} / 10240$
$= [-5.12 \sim 5.12]$

由此可知，若欲產生正確的生態平衡值，首先必須確定其平衡函數(測試函數) 的 X_i 範圍並設定 X_i 產生器的參數，其參數包含 X_i 的下限值 X_L，X_i 的上限值 X_H、解析度 s 與放大率 d。由上式結果得知，De Jong 函數 $f=X_i^2$ 的 X_i 範圍(-5.12 到 5.12)經過 X_i 產生器的處理機制後，決定以 17 位元作為生物群的編碼長度。在此，為驗證 17 位元的生物群透過 X_i 產生器所產生的 X_i 不會超過限制範圍，我們分別以編碼為 00000000000000000 的生物群(十進碼為 0)、編碼為 00000000000000001 的生物群(十進碼為 1)、編碼為 00000000000000010 的生物群(十進碼為 2)、編碼為 11111111111111110 的生物群(十進碼為 131070) 以及編碼為 11111111111111111 的生物群(十進碼為 131071) 代入 X_i 產生器$(0 \sim 2^{17}-1)$中驗算，依序代入 0 得到 X_i=-5.12、代入 1 得到 X_i=-5.119921874404、代入 2 得到 X_i=-5.119843748808、代入 131070 得到 X_i= 5.119921874404、代入 131071 則得到 X_i=5.12，驗証所產生的 X_i 並沒有超過限制之範圍，因此我們可以確定，由任意編碼長度為 17 位元的生物群透過二進碼轉十進碼後，且藉由 X_i 產生器產生的 X_i 範圍皆在-5.12 到 5.12 的範圍內，且精確度到達小數 4 位。

3.4.3 平衡函數

在本研究中我們以標準的最佳化演算法測試函數作為平衡函數。舉例來說，將二進碼為 10101010101010101 的生物群轉為十進碼後變為 87381，接著輸入至 De Jong 函數 $f=X_i^2$ 的 X_i 產生器中，產生的 X_i=1.706692708532，最後再將 1.706692708532 輸入至平衡函數 $f=X_i^2$ 中計算，進而產生平衡值為 2.9128 的結果，如

圖 4 所示。

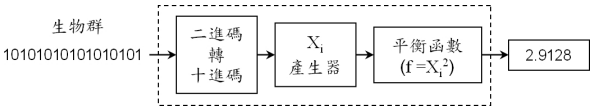

圖 4 產生平衡值之示意圖

3.5 多維度之互補演算法流程詳情

為求解多維的測試函數，互補式演算法須設定生態之維度。舉例來說，假設生態的維度 D=10、單一生態的生物群編碼之長度 L=4，所以交流次數 N=$\lceil \log_2 L \rceil$=2，如圖 5 所示。

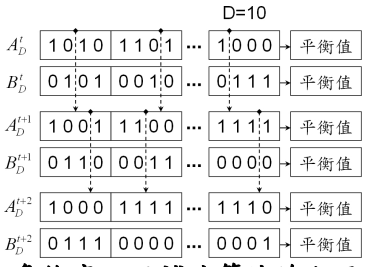

圖 5 多維度之互補演算法流程示意圖

4. 實驗結果與討論

為證明本研究所提出之演算法效能，本文使用 5 個常見的測試函數來來進行實驗[6]，其測試函數所受理的 X_i 範圍及已知最佳解之規格詳情，如表 1 所示。在此，我們會分別呈現應用互補式演算法與粒子族群最佳化演算法在標準測試函數上尋找最佳解之結果，並進行效能之比較及探討。

4.1 實驗結果

表 2 為互補式演算法與粒子族群最佳化演算法之計算效能比較表，此實驗中我們分別以互補式演算法和粒子族群最佳化演算法針對每項測試函數進行 10 次獨立的試驗，並將 10 次內之最佳結果、最差結果與 10 次之平均結果記錄起來，以便於進行效能之比較。實驗結果顯示，互補式演算法在測試函數 F1、F2、F3 和 F5 找尋的最佳解之結果皆優於粒子族群最佳化演算法，然而在 F4 中也能找到近似全域最佳解，與粒子族群最佳化演算法找到的近似全域最佳解之誤差皆在十萬分位內。本實驗中，互補式演算法之生物交流次數 N=$\lceil \log_2 L \rceil$，L 為單一維度 D 之生物群編碼的長度，其中 F1 之 L=17、N=5；F2 之 L=26、N=5；F3 之 L=17、N=5；F4 之 L=17、N=5；F5 之 L=14、N=4，而粒子族群最佳化演算法所設定的粒子數 p=30、慣性權重值 ω=0.2、學習因子 C1= C2=2，其中所有測試函數的維度 D 皆設為 10，迭代次數皆設為 100。以上兩種演算法在 5 種測試函數上 10 次的平均收斂效率，如圖 10 所示。

4.2 討論

互補式演算法與進化式演算法的存在目的及價值皆是為了在短時間內，找到問題的最佳解或近似最佳解。然而，目前所有的進化式最佳化演算法在求解過程中，除非找到的解滿足演算法的終止條件，否則會一直持續淘汰適應值較差的解，保留適應值較佳的解。由於進化式演算法無法保證所淘汰之適應值較差的解，在未來沒有產生最佳解的可能，因此有可能因淘汰適應值較差的解(此適應值較差的解可能是找到最佳解的關鍵)，導致最後無法找到全域最佳解。適者生存法則與貪婪演算法有相似的缺點，在此我們以圖 6 所示的旅行推銷員問題來說明適者生存法則與貪婪演算的相似之處。假設一個推銷員要從城市 A 出發，且必需走遍其他城市，再回到城市 A，求其最短路徑。我們知道貪婪演算法所選擇的城市路徑為 ACBDA，由於短視近利，每次都選擇離本身最近的城市，淘汰較遠的城市，雖然剛開始有佔到便宜，但就整體而言卻是吃虧的，因此始終無法找到最佳解 ABDCA 的結果。

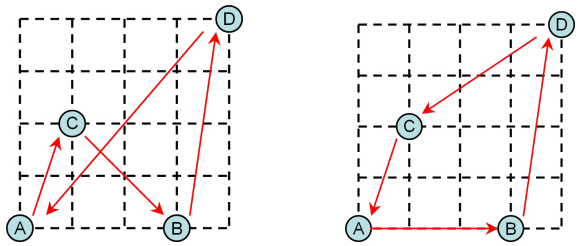

圖 6 旅行推銷員路徑示意圖

本文所提出的互補式演算法其主要精神在於由兩組互為 1 的補數之生物群進行生物的交流，並沒有傳統進化式演算法的突變及淘汰機制。由於互為 1 的補數之生物群 A_D^t 與 B_D^t 透過交流機制所產生的生物群 A_D^{t+1} 與 B_D^{t+1} 也互為 1 的補數，因此我們可以知道 A_D^t 有 B_D^t 所沒有的生物種，B_D^t 有 A_D^t 所沒有的生物種；同樣的 A_D^{t+1} 有 B_D^{t+1} 所沒有的生物種，B_D^{t+1} 有 A_D^{t+1} 所沒有的生物種。在兩個互為 1 的補數之生物群互相交流部分物種後，其中有一個生物群的生態平衡值會變好；而另一個生物群生態平衡值則

相對的變差。在這當中包含兩種可能：第一種情況就是原本平衡值較佳的生物群經由交流機制後平衡值變差了，而原本平衡值較差的生物群經由交流機制後平衡值卻變好了；第二種情況就是原本平衡值較佳的生物群經由交流機制後平衡值變更好，而原本平衡值較差的生物群經由交流機制後平衡值變更差，如圖 7 所示(假設平衡值為生物群的十進碼，求最大)。

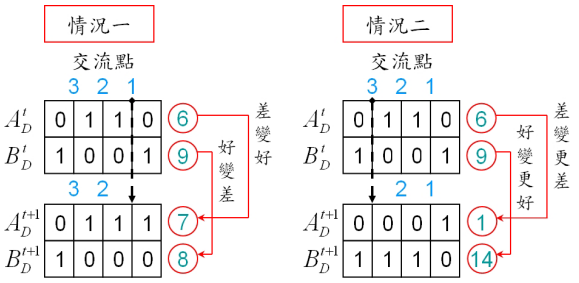

圖 7 補數交流的情況

然而適者生存法則會驅使進化式演算法在圖 7 的情況一中，淘汰結果較差的 A_D^t 與 A_D^{t+1}，並保留結果較好的 B_D^t 與 B_D^{t+1} 來進行下次的演化，由於所保留的 B_D^t 與 B_D^{t+1} 編碼相似，因此導致進化式演算法容易落入區域最佳解。本文所提出的互補式演算法由於沒有淘汰結果較差的 A_D^t 或 A_D^{t+1}，乃是使互為 1 的補數之生物群互相交流，因此不易落入區域最佳解，互補式演算法不論是在情況一或是情況二中，皆能找到最佳解，如圖 8 所示。

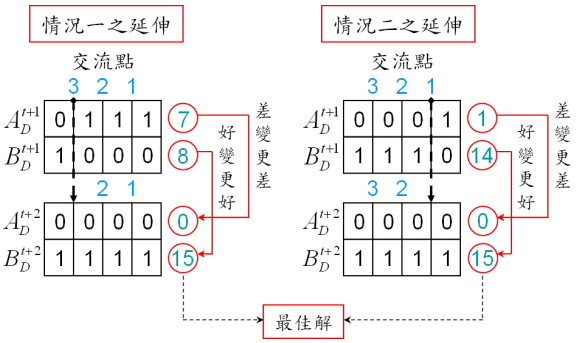

圖 8 補數交流的情況之延伸

在這當中我們希望互為 1 的補數之生物群透過交流機制後，能使原本平衡值較佳的生物群變得更好，如此才有機會達到最佳化的目的。在此，平衡值較差的生物群扮演著使原本平衡值較佳的生物群變得更好的重要關鍵角色。在互補式演算法中，平衡值較佳的生物群之誕生往往伴隨著平衡值比自己差的生物群。本研究中我們所關心的是，藉由哪幾種生物所組成的生物群，其生態平衡值是最佳的；換句話說，最佳的生態平衡值就在某種生物群的排列組合中，然而實驗結果顯示，由互為 1 的補數之生物群經過互相交流後，能有效的產生我們所期望的排列組合。在此，我們分別以解析度、交流機制和交流次數等三部分來探討其對互補式演算法執行效率之影響。

4.2.1 解析度

這裡所謂的解析度，指的是變數 X_i 的精確度，解析度越高，生物群編碼的長度就越長，理論上編碼較長的生物群，在有限的交流次數中，得到較佳解之機會則相對的變少；但若能找到最佳解，其最佳解的精確度一定比編碼長度短的生物群所找到的更高。目前為止，解析度的值並沒有明確的定論，通常要看問題的本質及執行效率或時間因素，並經過多次實驗後才決定。本研究中所設定的解析度為 1000(表示所產生的 X_i 之精確度到達小數 4 位)就能求得不錯的近似最佳解甚至全域最佳解。

4.2.2 交流機制

由於交流點為隨機選擇，因此互補式演算法在進行交流機制時，有可能會有連續相同的交流點之情形發生，此時會浪費生物群交流的次數，如圖 9 所示。然而到目前為止，由於此情況發生機率小，因此對互補式演算法的影響並不大，往後我們希望能夠在交流機制中加入禁忌收尋，藉此提高交流的效率。

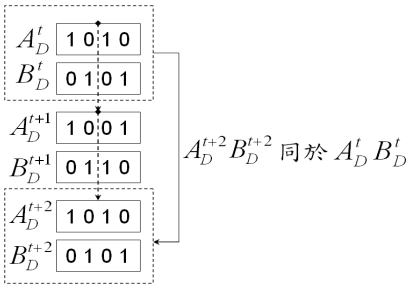

圖 9 浪費生物群交流次數之示意圖

4.2.3 交流次數

交流次數越多，理論上找到較佳解之機會越大，反之則不然。本研究中為降低演算法執行的時間，所設定的交流次數 $N=\lceil \log_2 L \rceil$，其中 L 為單一維度 D 之生物群編碼的長度。

5. 結論

本研究所提出的互補式演算法，是模擬生態系統各司其職的法則來求解各種排列組合

的最佳化問題，以互為 1 的補數之生物群經過互相交流後，能有效的產生我們所期望的排列組合，並能避免如同進化式演算法容易落入區域最佳解的情形。整體實驗結果顯示，互補式演算法在 5 種測試函數的收斂速度和所求的最佳解結果，除了測試函數 4 不相上下外，其餘皆優於粒子族群最佳化演算法。因此，往後我們可以使用互補式演算法來求解許多最佳化問題，相信可以提高尋找最佳解的效率。

參考文獻

[1] CHO, S. B., "Towards Creative evolutionary Systems with Interactive Genetic Algorithm," ***Applied Intelligence,*** Vol. 16, No. 2, pp. 129-138, 2002.

[2] Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms," ***MIT Press and McGraw-Hill***, 2001.

[3] Dorigo, M and L. M. Gambardella, "Ant colony system: A Cooperative Learning Approach to the Traveling Salesman Problem," ***IEEE Transaction on Evolutionary Computation*****,** Vol. 1, No. 1, pp. 53-66, 1997.

[4] Elbeltagi, E., T. Hegazy and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," ***Advanced Engineering Informatics,*** Vol. 19, No. 1, pp. 43-53, 2005.

[5] Holland, J. H., "Adaptation in Natural and Artificial System", ***The University of Michigan Press***, Ann Arbor, MI, 1975.

[6] Ho, S. Y., L. S. Shu and J. H. Chen, "Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems," ***IEEE Transactions on Evolutionary Computation,*** Vol. 8, No. 6, 2004.

[7] Jiang, Y., T. Hu, C. C. Huang and X. Wu"An improved particle swarm optimization algorithm," ***Applied Mathematics and Computation,*** Vol.193, pp. 231-239, 2007.

[8] Kennedy, J., R. C. Eberhart, "Particle swarm optimization", ***Proceedings of IEEE International Conference on Neural Networks***, Piscataway, NJ, pp. 1942-1948, 1995.

[9] Kim, S. J., H. Guo, P. Gu, "Environmental History and Origin of Ecological Crisis," ***Nanjing Forestry University***, Vol. 5, No. 3, pp. 5-10, 2005.

[10] Oh, I. S., J. S. Lee and B. R. Moon, "Hybrid Genetic Algorithms for Feature Selection," ***IEEE Trans. Pattern Analysis and Machine Intelligence***, Vol. 26, No. 11, 2004.

[11] Tansley, A. G., "The Use and Abuse of Vegetational Concepts and Terms," ***Ecology***, Vol. 1, No. 3, pp. 284-307, 1935.

[12] Vilcot, G., J. C. Billaut, "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem," ***European Journal of Operational esearch,*** Vol. In Press, Corrected Proof, 2007.

[13] Zhu, Z., Y. S. Ong and M. Dash, "Wrapper-Filter Feature Selection Algorithm Using a Memetic Framework," ***IEEE Trans Systems, Man, and Cybernetics, Part B***, Vol. 37, No. 1, pp. 1083-4419, 2007.

表 1 測試函數

測試函數	x_i 範圍	已知最佳解
$f_1 = -\sum_{i=1}^{D}\left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right)\right]$	[3, 13]	1.21598D (max)
$f_2 = \sum_{i=1}^{D}\lfloor x_i + 0.5\rfloor^2$	[-100, 100]	0 (min)
$f_3 = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	[-5.12, 5.12]	0 (min)
$f_4 = \sum_{i=1}^{D} x_i^2$	[-5.12, 5.12]	0 (min)
$f_5 = \sum_{i=1}^{D}\left(x_i \sin(10\pi x_i)\right)$	[-1.0, 2.0]	1.85D (max)

表 2 互補式演算法與粒子族群最佳化演算法之計算效能比較表(D=10)

測試函數	演算法	10 次內之最佳結果	10 次內之最差結果	10 次之平均結果	已知最佳解
F1	**互補**	**12.1598**	**12.1598**	**12.1598**	1.21598D (max)
	PSO	**12.1598**	10.1745	11.0679	
F2	**互補**	**0**	**1**	**0.2**	0 (min)
	PSO	**0**	4	0.7	
F3	**互補**	**0.002156**	**3.984363**	**0.786519903**	0 (min)
	PSO	9.957327	21.02287	13.65559215	
F4	互補	6.13E-07	2.94E-05	8.35141E-06	0 (min)
	PSO	**4.27E-11**	**3.60E-07**	**4.80091E-08**	
F5	**互補**	**18.50273**	**18.45153**	**18.4951395**	1.85D (max)
	PSO	14.10299	10.70246	11.8980277	

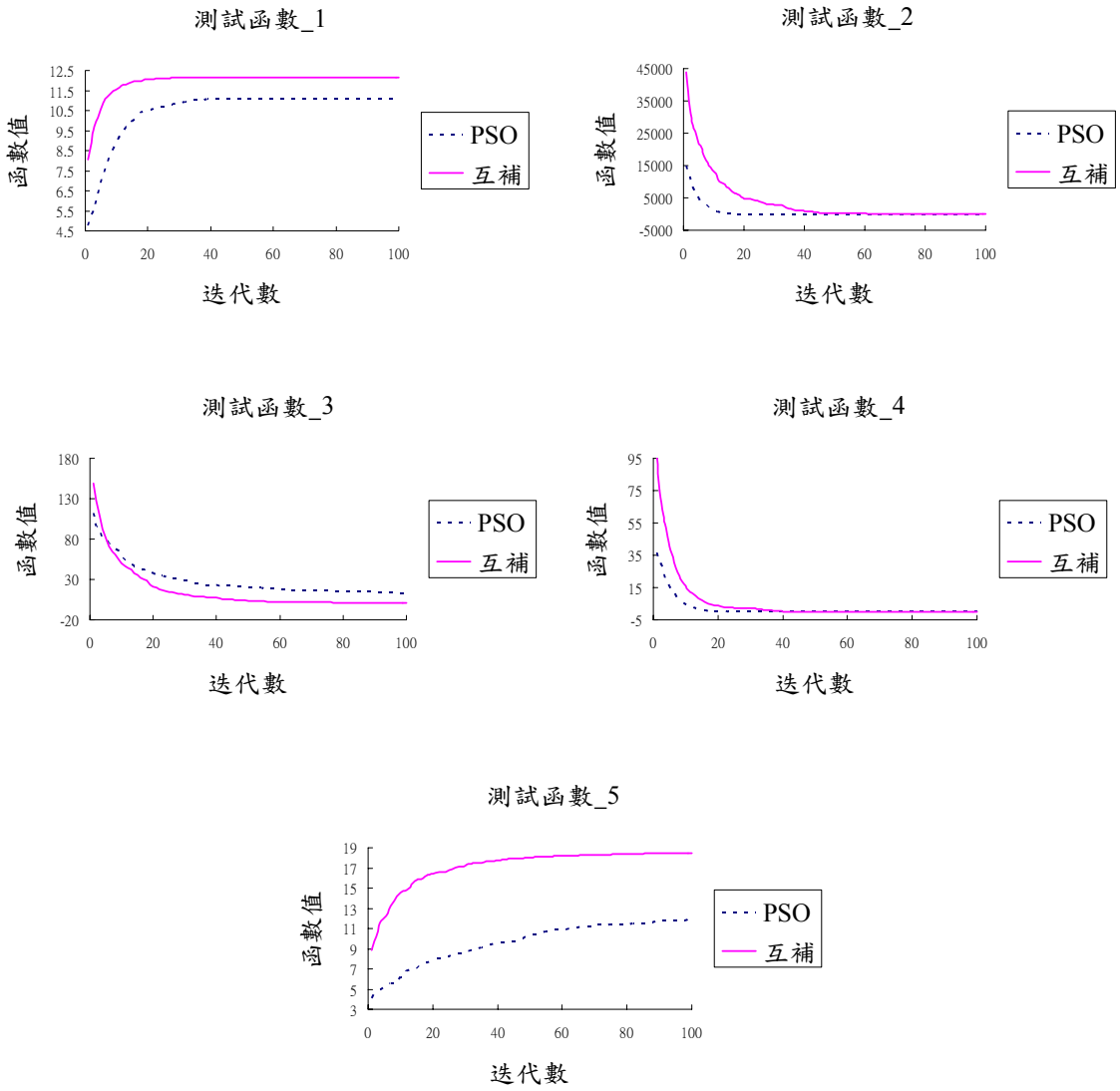

圖 10 互補式演算法與粒子族群最佳化演算法之計算效能比較圖(10 次平均)