運用多群協同改良式差分演化演算法

李維平　　簡宛柔
中原大學資管所　　中原大學資管所
wplee@cycu.edu.tw　wanlorgina@yahoo.com.tw

摘要

差分演化演算法(Differential evolution)為近年來新穎且正迅速發展的演化式計算，該演算法擁有結構簡單、容易使用、及快速收斂之特性。而不同於傳統演算法的複雜流程，差分演化演算法除了流程簡單外，並能有效地運用於廣大的複雜型問題之中，也因此成為近年來學者熱門研究對象之一。

但在求解過程當中，差分演算法同樣有收斂不穩定及陷入區域最佳解之問題存在。因此，現今許多學者針對此一缺點去進行改良，除了調整參數及改良演化機制之外，本研究嘗試結合多群的協同演化(Co-evolutionary)的架構及重置因子(reset factor)，提出一CO-DE演算法。

協同演化能預防演算法的過早收斂，同時維持粒子多樣性，因此，本研究嘗試結合多群協同兼顧粒子搜尋過程的精度和效率，同時利用重置因子更新機制防止粒子陷入停滯。

關鍵詞：差分進化演算法(Differential Evolution, DE)、演化式計算(Evolutionary Computation)、協同演化(Co-evolutionary)、最佳化(Global optimization)

Abstract

Differential evolution, termed DE, is a novel and rapidly developed evolution computation in recent year. There are some advantages of DE, including simple structure, easy use and rapid convergence speed. Besides, DE can be also applied on complex optimization problem. However, there are some problems, such as premature convergence and stagnation, remaining in DE algorithm. To overcome those disadvantages, a different method was proposed, named CO-DE, by combining with a simple co-evolutionary model and reset mechanism. Thus, CO-DE can maintain appropriate swarm diversity and reduce the premature convergence. On the other hand, a reset mechanism was set to avoid the particle stagnates, which can further improve the performance of differential evolution. The proposed model can be now successfully applied with some well-known benchmark functions.

Keywords: Differential Evolution、Evolutionary Computation、Co-evolutionary、Global optimization

1. 前言

演化式計算(evolutionary computation)演化至今，近二十餘年來已極為盛行，並吸引全世界無數年輕學者投入這個研究的行列[14]。其主要概念為仿造自然界中生物群體智慧之行為模式而建立的一種計算模式。利用達爾文進化論 ”適者生存，不適者淘汰” (survival of the fittest) 的道理，做為在廣大空間的搜尋機制，藉此可求解各種困難的組合最佳化問題，找到問題的最佳解[16]。

最早出現的演化式計算為演化式規劃、演化策略以及基因演算法，此三種基本模式均以”天擇”概念來進行運作，早期演化式演算法除了擁有本身容易落入區域最佳解之問題存在，還有流程複雜及求解能力較差等等缺陷。而許多相關研究領域之學者為改善此一缺點，紛紛依循此一行為提出更多相關模式之演算法，其中包含 Holland 於 1975 年提出的基因演算法(Genetic Algorithm；GA)、Marco 於 1992 年所提出的螞蟻演算法(Ant Colony Optimization；ACO)[3]、Kennedy 和 Eberhart 於 1995 年提出的粒子群演算法(Particle Swarm optimization；PSO)[4,9]及本文所研究的差分演化演算法(Differential evolution；DE)[9]等，其目前發展至今，演化式計算已被廣泛地運用在各種領域，特別是組合最佳化問題之中，包含工作排程、路徑規劃、網路問題、資料探勘、人工智慧等，甚至機械、工程的設計，及化學工程領域中，也同樣有演化式計算的蹤跡。

差分演化演算法 (Differential evolution)[11,13]由 Stron 及 Price 於 1995 年左右所提出，該演算法同樣以演化式計算為基礎，所延伸出的新穎演算法之一，擁有結構簡單、容易使用、及快速收斂之特性。以效能、精確度、及流程難易度來說，透過實驗測試比較，

差分演化演算法均優於以往的演化式計算，也因此成為本研究做為主要研究之首選，而雖然差分演化演算法已在 1996 年的國際研討會之中，透過多篇文獻證實其求解之優越性，也被公認為最佳的演化式演算法[9,12-14]，其本身仍有舊時傳統演化式計算的缺點，諸如陷入區域最佳解及收斂不穩定性等問題。

基於以上所述，本研究嘗試以共演化為基礎概念，提出一多群協同的架構[10]結合差分演化演算法，有別於以往雙向母體進行交流之共演化機制，本研究將利用四個群體的進行演化，並利用新的突變機制期望透過此種模式能在廣大的搜尋空間中作更精確之搜尋，同時保有粒子多樣性，並為避免過早收斂於區域最佳解，將設定一條件式，若不符合條件即重新重置一新的維度值，提昇此演算法求解之精確度及效率。

2. 文獻探討

2.1 差分演化演算法(Differential Evolution; DE)

差分演化演算法於 1995 年由 Stone 及 Price 兩位學者所提出[13,14]，其基礎概念源自於學者為求解切比雪夫多項式擬合問題所生成，由於此演化式計算已於 1996 年演化式計算的國際型研討會中，由 Stone 及 Price 本人親自以反覆實驗進行，證明其優越的求解效能[12,14]，因此成為近年來熱門的演算法之一。

差分演化演算法之概念是以向量為基礎，透過母體間個體之差異性，並以隨機搜尋方式將差異向量加上其中之個體中，並透過突變(Mutation)、重組(Recombination)、選擇(Selection)三個步驟進行演化，而透過迭代不斷地進行運算，並觀察個體是否能在搜尋解空間中找到最適解。差分演化演算法同樣運用了群體智慧，透過生物演化概念，物種與物種間的競爭，具有優勢個體往往能成功存活下來，而不能適應環境者則將遭到淘汰。差分演化演算法在結構上與基因演算法類似，均是以交配、突變為演化機制，但與基因演算法相比，差分演化演算法更有參數設定簡潔，流程設計更為簡單之優點，且不同於基因演算法之演化流程，差分演化演算法為先執行突變策略緊接著進行交配，並於最後一步驟才透過選擇機制將較差解淘汰，也因每一迭代均會執行突變及交配之演化，比起基因演算法，又更有具有解之多樣性，其求解能力更大幅超越傳統基因演算法，以下將詳細說明差分演化演算法如何執行其演化模式。

差分演算法的概念和基因演算法相似，主要的演算流程有初始化、突變、重組、選擇…等[1,2]，整體流程及演算公式如下所示：

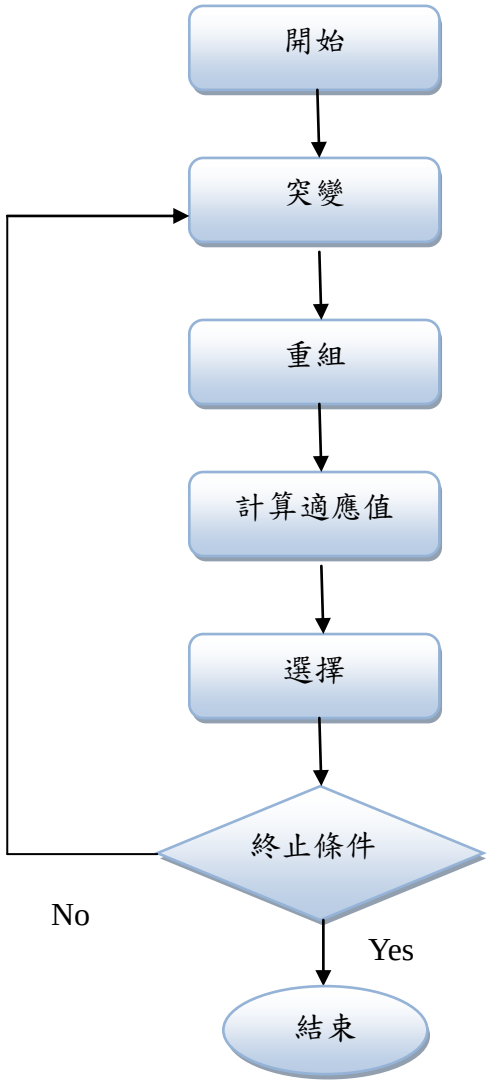


圖 2.1 差分演算法流程圖

初始化(Initialize)：設定參數值及以隨機產生方式初始其目標向量，其公式如下。

$$X(t) = [\chi_{i,1}, \chi_{i,2}, \cdots \chi_{i,d}], (I = 1, 2, \ldots, NP)$$

突變(Mutation)： 隨機由母體中選取三個目標向量，分別為$\chi_{r1,G}$，$\chi_{r2,G}$ 及$\chi_{r3,G}$，並透過突變權重因子(Mutation weighting factor；F)結合而得一合成向量(Donor Vector) $V_{i,G+1}$，其公式如下。

$$V_{i,G+1} = \chi_{r1,G} + F(\chi_{r2,G} - \chi_{r3,G})$$

重組(Recombination)：經過前一突變步驟產生出合成向量(Donor Vector)後，將利用條件公式與挑選中的目標向量$\chi_{i,G}$作一結合重組，並於重組後得到一新的試驗向量(Trial Vector) $u_{i,G+1}$。此公式簡單說明如下，此一機制將設定一參數值 CR 值，接著隨機選取一亂數值，若此亂數值小於初始設定之 CR 值，則新的試驗向量的第一維度將放入$V_{i,G+1}$，反之；若大於 CR 值則由$\chi_{i,G}$取代。

$$U_{i,G+1} = \begin{cases} V_{i,G+1}, & \text{if rand} \le CR \\ \chi_{i,G}, & \text{if } rand > CR \end{cases}$$

選擇(Selection)：經由上述步驟，緊接著進入選擇階段，其主要執行方式為藉由計算適應值(Fitness Function)來評估其該選擇試驗向量(Trial Vector)亦或是原始的目標向量進入下一迭代，成為此代演化過後的子代。

$$\chi_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } F(u_{i,G+1}) < F(\chi_{i,G+1}) \\ \chi_{i,G+1} & \text{otherwise} \end{cases}$$

2.2 共演化模式(Co-Evolutionary Mode)

共演化(Co-Evolutionary)，又稱作協同進化，其模式之概念於 1964 年由 Ehrlich 及 Raven 兩位學者所提出[5]。此兩位學者藉由觀察蝴蝶與其寄生植物間的關係而得到啟發，學者發現蝴蝶所寄主的植物之子代本身會發生某種變化，其子代含有某種有毒性之化合物，而其來自於蝴蝶。而因本身植物屬於固有性植物，無法自行抵抗蟲害，因此產生此種有毒物質來保護自己，而蝴蝶也因此產生抵抗機制，產生了此種交互演化的效果[16]。

而另一學者 Hilli 也於 1991 年提出[7]了一種掠食者與被掠食者的一種概念作為共演化的模型，不同於上述交互演化的概念，Hilli 提出之核心為相互演化的概念，也就是互相競爭的一種演化方式。而 Rosin 及 Belew 也提到共演化是一模擬兩種以上生物演化的一種概念[10]，其演化模式能改善傳統演算法之缺點，高度複雜及過早收斂之問題。協同進化研究的內容相當廣泛，其中包括了競爭物種間的協同進化、掠食者與獵物間的協同進化、寄生物與寄主之間的協同進化以及互利的協同進化等等[17]。

綜合以上所述之整理，標準的共演化模式大約可被分為兩類，以圖 2.2 及 2.3 為例：

1. 合作協同型共演化模式(Cooperative co-ecolution)
2. 競爭協同型的共演化模式(Competitive co-ecolution)

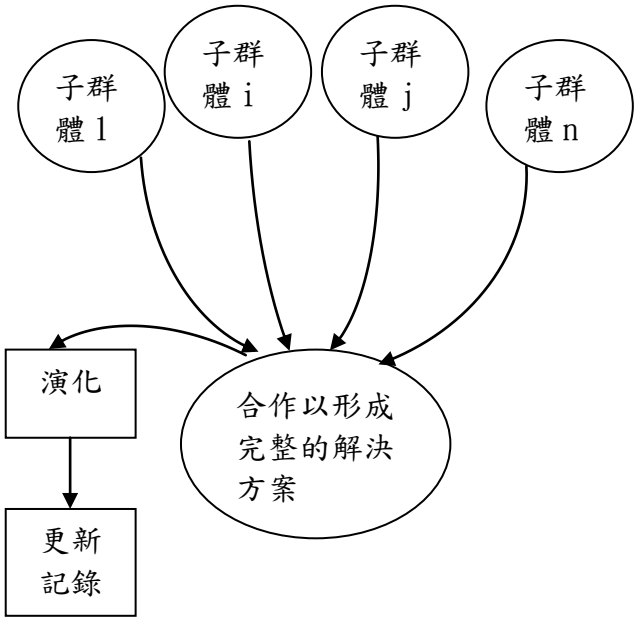


圖 2.2 合作協同型共演化模式(Cooperative co-ecolution)[6]

而在共同演化的基礎概念下，有學者認為，除了兩種生物共同生存演化外，共演化同時也包含了”兩種以上”的生物演化，而類似的概念首先被使用於基因演算法中，Husband 於 1991 年提出了多種族群合作型的協同進化遺傳演算法模式[8]，而將此種多群體協同進化概念運用於粒子群演算法則是由 Bergh 及 Engelbrecht[6]兩位學者所提出。合作協同型的共演化模式則是在進化過程中依靠子群體間相互的交換資訊來達到最佳化，反之；對於競爭型共演化模式來說，子群體間必須藉由不斷地演化競爭來優於其他群體，然而，無論是哪種方式，均須考慮到如何分解問題、參數設定之間的相互影響及分層的任務如何指派，此些行為均會影響演化的進行[2]。

依據 Bergh 及 Engelbrecht 所提出之多群架構，在此以圖 2.2 為例，圖 2.2 中，問題將被分解成數個子群體，每個子群體均分別以亂數模式進行初始化，而被分成數個子群體之後，將會利用粒子群演算法模式各自演化，並在每一群中選出一代表性的個體，結合進主群體中，在

進行二度演化、更新機制。

而 Bergh 及 Engelbrecht 所提出另一種架構為競爭型的共演化模式(Competitive co-evolution)，在理想的狀態下，為了要達成最佳化，所有的

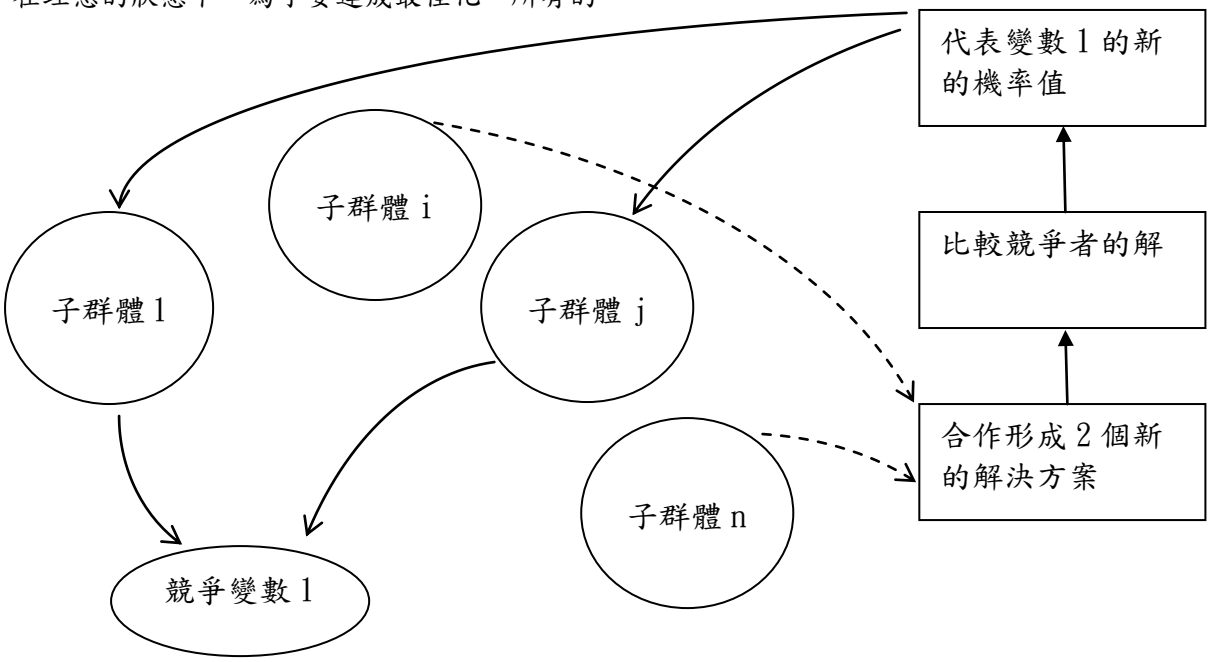


圖 2.3　競爭型共演化模式(Competitive co-ecolution)[6]

粒子群體必須相互競爭，然而，這類的架構模式將造成演算法運算時間過長，因此實際上來說是不可行的[6]，因此有學者提出將群體分成兩群，分別是主群體及競爭群體，如圖 2.3 所示。

如前所述，若將群體細分成數個小群體進行競爭模式演化，將浪費大量之運算時間，對演算法整體效能來說是不利的，因此作者雖同樣利用多群架構，但先將群體分成數個小群體後，利用類似分組的想法，首先，指定子群體 1 為第一組，接著，利用輪盤法之選擇方式，將第二個競爭者選出，也就是圖中的子群體 j，如此一來，子群體 1 及子群體 j 同為競爭者 1，而剩餘的子群體 i 及 n 同為競爭者 2，以此概念之下，將數個群體又重新分為主群體與競爭群體，不同於以往共演化模式，在接下來的每個迭代中，任兩群體都有可能成為同一組，而此模式又被作者稱為競爭型的共演化模式[6]。

2.3 合作型共演化模式相關研究

2.3.1 多粒子群協同優化算法(PSCO)

多粒子群協同優化算法(PSCO)於 2004 年由李愛國先生所提出[17]。其基本概念是利用 S (S > 1)的獨立的粒子群進行最佳化演算法之進行。而前 S-1 個粒子群體均為獨立運作，以群內所有粒子之最佳解為該群之 gbest。而主群體 S 則是以迄今所蒐尋過的最佳解當作搜尋之參考。此種多群協同之模式，將達成利用小群體進行擴大在解空間中的搜尋，並利用主群體 S 來達到收斂之成效，另外，在此 PSCO 演算法中，作者為避免落入區域最佳解，另行加入一擾動因子之策略，若是此演算法於搜尋過程中，連續 u 代均無變化，則置入擾動策略—重置粒子的速度，讓其演算法再度活化，搜尋更精確之解。本研究將採用此一類似多群協同合作模式，提出一 CO-DE 演算法，並將其架構運用於差分演化演算法中，期望能將此一架構也適用於部分改良式差分演算法，達到更精確之求解。

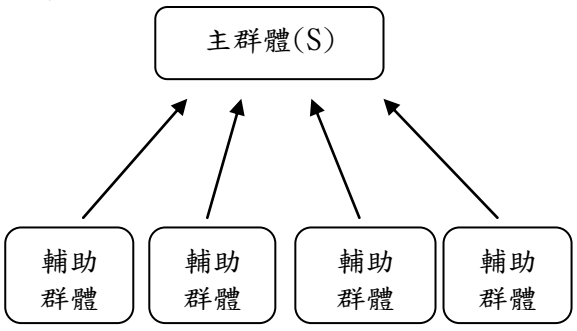


圖 2.4 多粒子群優化算法

3. 研究方法設計

本研究發現，將多群架構運用於演化式計算之研究大部分均為粒子群演算法為主，而針對差分演化演算法來說，均是以雙演化為主，顯少有多群架構之研究，因此本研究將分成兩階段，第一階段將利用多群協同之架構為基礎，同時將此架構運用於傳統差分演化演算法及對此方法作一探討，第二階段將運用新的突變方式改良此一多群協同架構，提出一 CO-DE 演算法。

本研究在此將多群協同之架構運用於差分演化演算法之中，其差分演化演算法之主要演化架構均無變動，將會以傳統的方式進行，其中包括初始化、突變、重組、選擇，而因舊有的傳統差分演化演算法中，突變是以隨機選取三個個體，並利用個體間差異去進行突變，此一方式並不適用於多群協同，分群的結果會造成在隨機選取三個個體時，選擇之機會減少，因此，本研究將修正此一缺點，採用新的突變模式，並再根據修正後之多群協同架構進行進一步的改良，以預防整體粒子過早進入區域最佳解。而在多群協同模式上將會採用類似於多粒子群協同優化算法所提出的 PSCO[25]來做一參考。其流程如圖 3-1 所示。

CO-DE 演算法的基本流程如下：

Step1：初始化參數，以隨機亂數方式分別進行母體的初始化。

Step2：計算適應值。

Step3：更新全域最佳解及區域最佳解。

Step4：利用全域最佳解、區域最佳解及隨機選取一變數向量，進行突變運算，並產生新的合成向量(Donor Vector)。

$$V_{i,G+1} = \chi_{r1,G} + F(g_{best} - \rho_{best})$$

Step5：利用新產生之合成向量(Donor Vector)及目標向量$\chi_{i,G}$以機率的型式進行交換並產生一新的試驗向量(Trial Vector)。

Step6：進行選擇機制，判斷合成向量及試驗向量何者較佳，較佳者將被保留下來。

Step7：進行迭代數判斷。若連續五個迭代最佳解均相同，則利用重置因子將演算法再度活化；反之，直接回到 Step2 進行下一迭代演化。

Step8：加入一重置因子。若連續五個迭代最佳解均相同，則利用演算法中每一迭代均會記錄之全域最佳解及區域最佳解，取其平均值，並隨機挑選非全域最佳解之個體，以隨機挑選位置的方式將解置換進個體的維度中。

Step9：未能達到停止條件則回到 Step2，若達滿足條件則輸出最佳解。

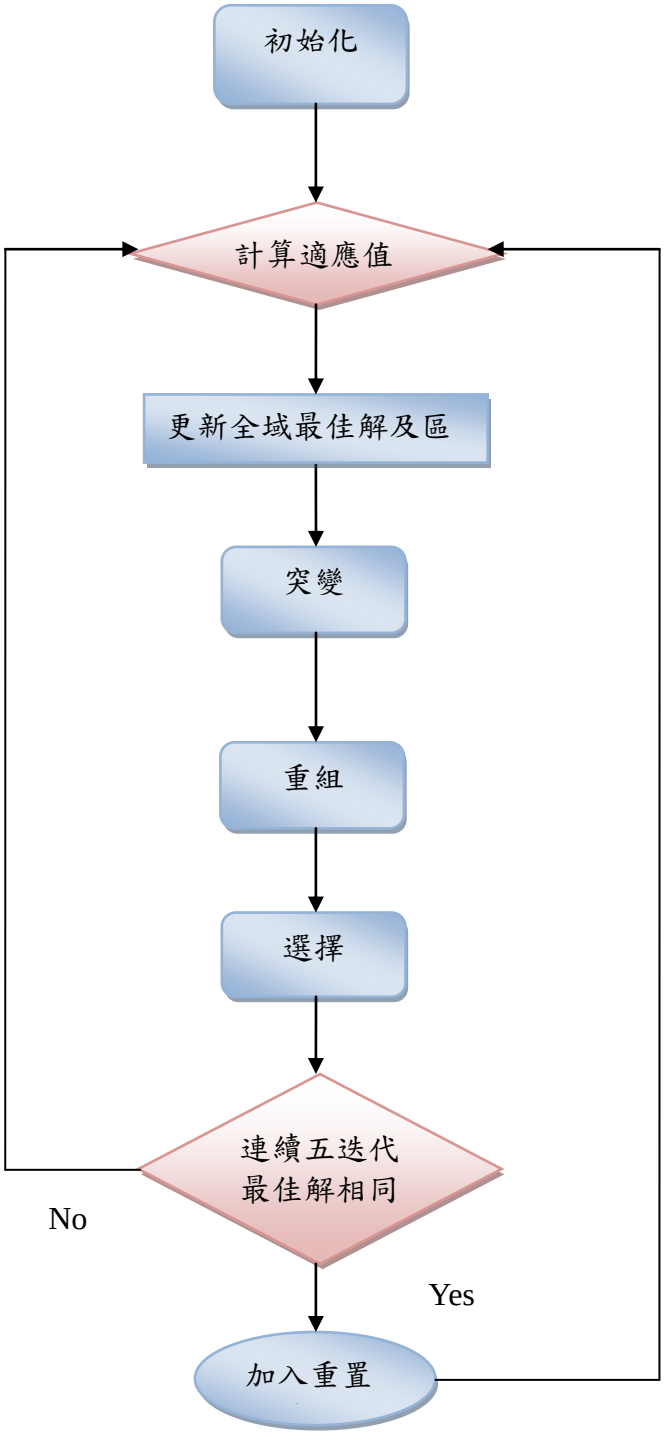


圖 3.1CO-DE 流程

以下將針對 Step4 及 Step8 作一詳細說明，首先在 Step4 中，本研究將以新的突變公式進行突變機制的運算：

Step4 中，將記錄每一迭代的全域最佳解及區域最佳解，並利用新的突變公式讓粒子進行四群的分區搜尋，如下圖所示：

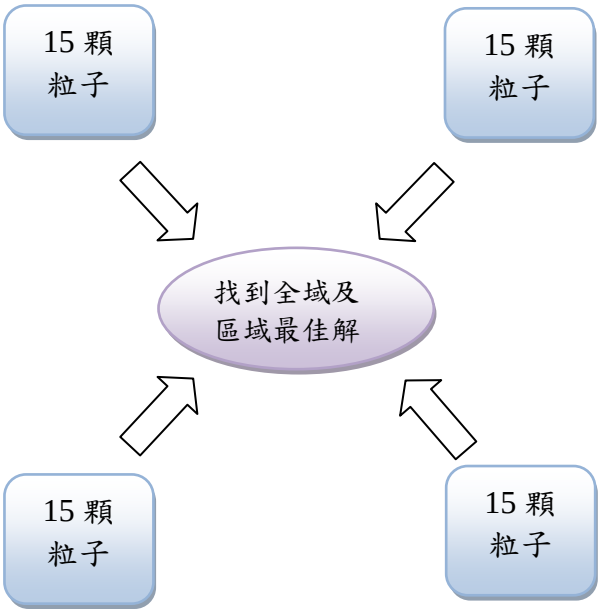


圖 3.2 CO-DE 架構

Step8 中，首先必須先判斷是否連續五個迭代的最佳解均相同，若相同，則進行重置因子的置換，此階段之方法，將利用全域最佳解及區域最佳解，以隨機方式各取一維度，並取其平均值，再以隨機方式選擇個體中其中一維度，並進行置換，如下圖所示：

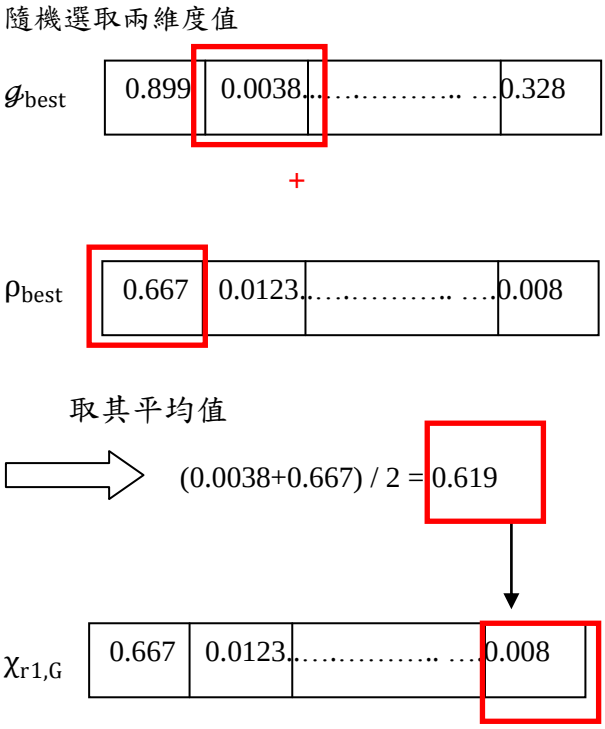


圖 3.3 重置因子流程圖

4. 實驗

本研究將改良傳統差分演化演算法之原始架構，以多群協同概念架構套用之，並改良其突變策略，並與傳統差分演化演算法及粒子群演算法作一比較，接著為避免落入區域最佳解，本研究將於其多群架構中設置一重置因子，增進其效能。

4.1 參數設定

表 1 CO-DE 參數設定表

F(突變權重)	0.7
CR(交換率)	0.7
Iter(迭代數上限)	1000
upper_bound(搜尋上界)	100
low_bound(搜尋下界)	-100
NP(解向量數)	60
Dim(維度)	10/30

表 2 DE/rand1 參數設定表

F(突變權重)	0.5
CR(交換率)	0.9
Iter(迭代數上限)	1000
upper_bound(搜尋上界)	100
low_bound(搜尋下界)	-100
NP(解向量數)	60
Dim(維度)	10/30

本研究詳細參數設定如表 1 及表 2 所示。為力求準確性，本研究於搜尋空間、迭代數、解向量數設定均相同，而其餘參數設定部分均為固定值，而 CO-DE 以及傳統 DE 於 F 值和 CR 值均以當前演算法最佳狀態為主設定之，因此 CO-DE 的 F 值及 CR 值設定為 0.7，而 DE/rand1 的 F 值及 CR 值分別設定為 0.5 及 0.9。

4.2 測試函數

為驗證此一架構下演算法之有效性，將使用學者公認之評估測試函數進行一效能檢測，此四類函數為最經典也最為基本之測試函數，其最佳解均為零，本研究將利用此四個函數進行一個效能上的評估。且為了進行之精確性，本研究所有的測試函數都將被重複執行 30 次並求取其平均值。

函數及其特性如下表所示：

表 3 測試函數

編號	函數名稱	公式
f 1	Sphere	$f_1(x)=\sum_{i=1}^{n}x^2$
f 2	Rosenbrock	$f_2(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right.$
f 3	Rastrigin	$f_3(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+1\right.$
f 4	Griewank	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$

4.3 實驗結果

4.3.1 不同維度下的實驗

在本節當中，將本研究所提出的以多群架構為主的改良式差分演化演算法與傳統的差分演化演算法以及粒子群演算法進行比較。

表 4 不同維度下的實驗

(a)

f1 函數實驗結果

(f1)	Dim.	Avg.	Best
PSO	10	1.45091E-35	6.69989757e-40
	30	5.37698e-07	1.283394e-011
DE/rand1	10	5.28435e-37	1.1687943e-37
	30	2.73768e-09	8.1094707e-10
CO-DE	10	**8.54537e-75**	**8.1076948e-84**
	30	**2.75097e-31**	**9.76357020e-38**

(b)

f2 函數實驗結果

(f1)	Dim.	Avg.	Best
PSO	10	4.13567e+00	2.9865467E+00
	30	6.76568e+01	8.236947e+000
DE/rand1	10	2.89645e+00	4.2679621e+00
	30	3.97248e+01	2.7857743e+01
CO-DE	10	**7.02535e+00**	**1.3542165e+00**
	30	**2.57517e+01**	**1.0420747e+01**

(c)

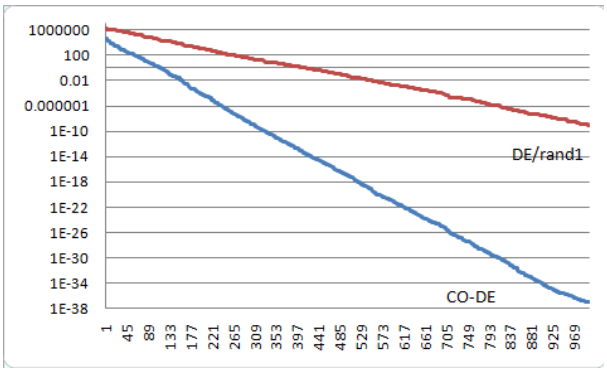
f3 函數實驗結果

(f1)	Dim.	Avg.	Best
PSO	10	4.18998e+01	2.6123662e+01
	30	6.53989e+01	2.805312e+01
DE/rand1	10	1.68466e+01	1.5835218e+01
	30	1.79341e+02	1.6299709e+02
CO-DE	10	**1.63408E-12**	**0**
	30	**8.62321e-01**	**0**

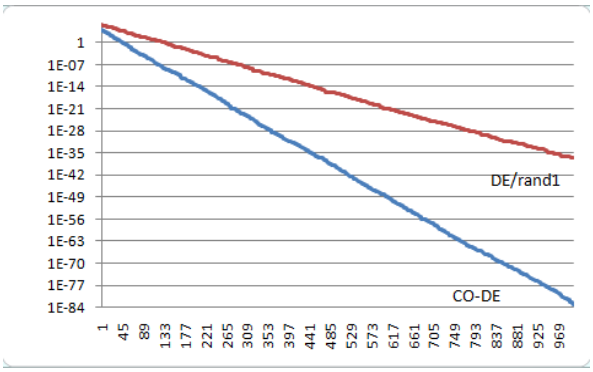
(d)

f4 函數實驗結果

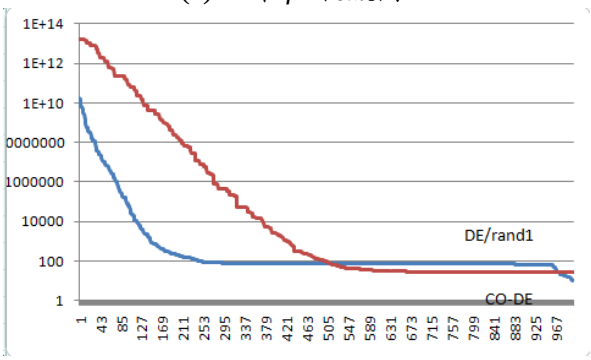
(f1)	Dim.	Avg.	Best
PSO	10	7.5363E-02	3.45908728E-07
	30	1.3845e-02	3.3394409e-012
DE/rand1	10	7.6499e-02	3.46148004e-02
	30	7.3927e-05	6.70393740E-11
CO-DE	10	**0**	**0**
	30	**8.4950E-14**	**0**



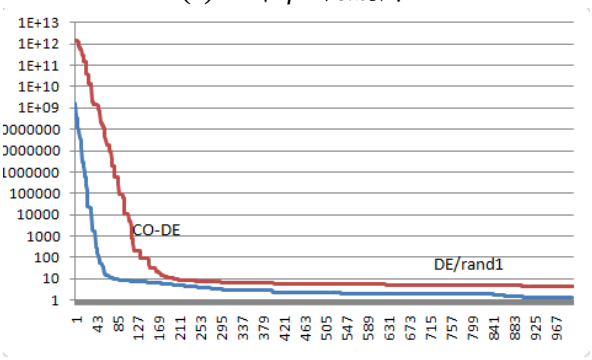
(a)30 維 *f1* 收斂圖



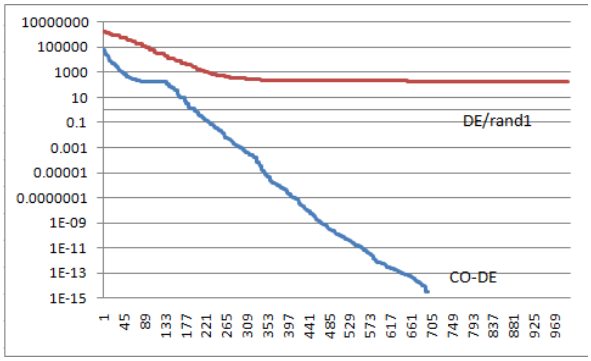
(a)10 維 *f1* 收斂圖



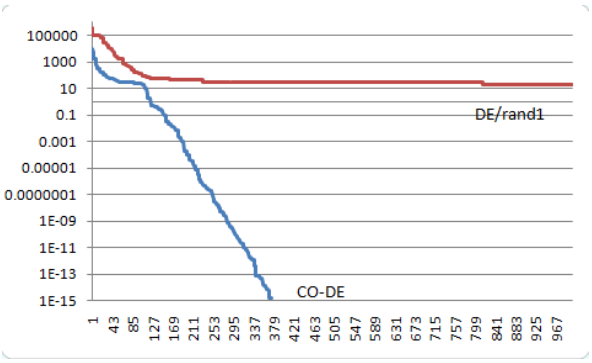
(b)30 維 *f2* 收斂圖



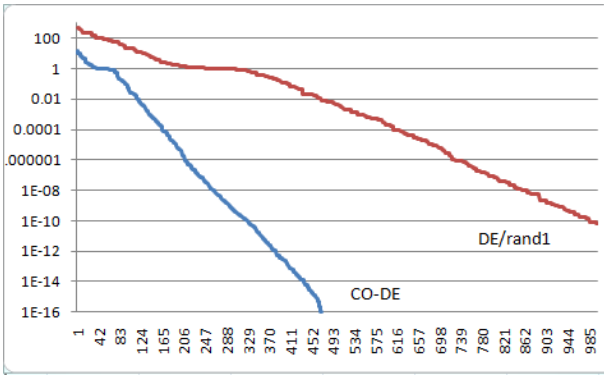
(b)10 維 *f2* 收斂圖



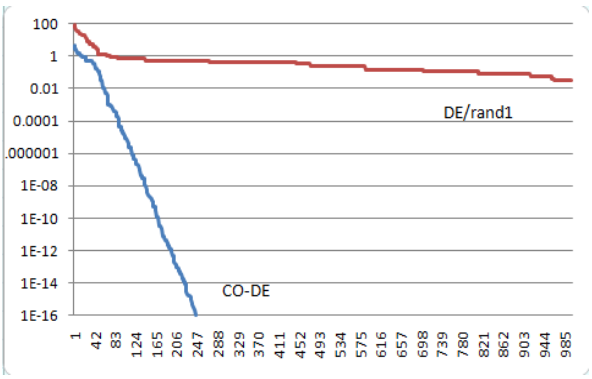
(c)30 維 *f3* 收斂圖



(c)10 維 *f3* 收斂圖



(d)30 維 *f4* 收斂圖



(d)10 維 *f4* 收斂圖

圖 4.1 不同維度下的收斂

表 4 是在四個測試函數中不同維度之表現。本研究利用 10 維及 30 維兩種不同維度的情況下，分別與最原始的差分演算法及粒子群演算法進行比較，證明了 CO-DE 演算法的一個有效性及穩定性，本研究再進行差分演化演算法，採用了 DE/rand1 策略，並設定當前最理想之參數設定與 CO-DE 進行比較，而由本節實驗結果可以看出，其四個測試函數表現均略為突出，而 CO-DE 在低維度實驗中，佔了明顯的優勢，於四個函數中均能求得極佳值。

圖 4.1 為不同維度下的一個收斂情型，可以看出於單峰函數 *f1* 中，CO-DE 在演算中的一個效能，而在 *f2* 的 30 維中，CO-DE 雖於後期陷入一個停滯，但最後也跳出了區域最佳解，繼續往下探索，而這邊可以看見，多群架構在低維度中的一個明顯的優勢，於四個測試函數均能求得一個不錯的解。

5. 結論

本研究提出以粒子群演算法常見的多群架構套用至差分演化演算法中，其目的為了防止差分演算法之過早收斂，並為避免過早落入區域最佳解，本研究利用一重置因子讓粒子有更多機會可以搜尋新空間。以下將針對經由實驗結果所得到的結論：

1. 透過多群概念，可發現在低維度中能更深入去探索該領域，能有效的加強其搜索的精度。
2. 多群概念本身為一個分工合作的機制，透過這樣的機制能減緩粒子落入區域最佳解的機會。
3. 為了求得更精確解且不增加其演算法複雜性，本研究設置一重置因子，以簡單方式來有效幫助粒子跳脫出區域最佳解。

本研究所提出之 CO-DE 演算法經由實驗證實能有效的解決複雜函數問題，其不管是在低維度或較高維度中均能有不錯的成效。也因多群協同本身為一個架構性的改良，未來可透過不同的其餘改良方式來精進此一架構下的演化。目前在差分演算法改良中，除了策略改良外，也有許多相關參數的改良以及調整，本研究提出一個以多群協同架構為基礎的差分演算法，將適用於類似調整參數以及策略改良之研究，未來或許可將此一演算法與其他相關改良演算法作一結合，相信會有不錯的成效。

參考文獻

[1] Amin N., Hong W., "Co-evolutionary Self-Adaptive Differential Evolution with a Uniform-distribution Update Rule", Intemational Symposium on Intelligent Control Munich, Germany, October 4-6,2006.

[2] [13]C.K. Goh, K.C. Tan, D.S. Liu, S.C. Chiam, A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design, European Journal of Operational Research, 2009, p42-45

[3] Dorigo, M. and Maniezzo, V. and Colorni, A., "The ant system: Optimizatoin by a colony of cooperating agents", ***IEEE Transactions on Systems and Cybernetics - Part B***, 1996, Vol 26-1, pp.29-41.

[4] Eberhart, R.C. and Kennedy, J. , "A new optimizer using particle swarm theory, " ***Proc. Sixth International Symposium on Micro Machine and Human Science***, Nagoya, Japan,1996, pp.39-43.

[5] Ehrlich, P.R., P.H. Raven, "ButterDies and plants: a study in coevolution," ***Evolution vol***.1964, 18 pp586–608.

[6] F. van den Bergh, A.P. Engelbrecht, "A cooperative approach to particle swarm optimization " , ***IEEE Transaction on Evolutionary Computation*** , 2004, 225–239.

[7] Hillis, W.D., "Coevolving parasites improve simulated evolution as an optimization procedure," ***In Langton, C.G., Taylor, C., Farmer, J.D., & Rasmussen, S. (Eds), Artificial Life II , Redwood City, CA: Addison Wesley***. 1992, pp.313-324.

[8] Husband P, Mill F. "Simulated co-evolution as the mechanism for emergent planning and scheduling." ***Proceedings of 4th International Conference on Genetic Algorithms, San Mateo,Morgan Kaufmann,*** 1991, 264-270

[9] Kennedy, J. and Eberhart, R.C., "Particle swarm optimization, " Proc. ***IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, pp. IV : 1942-1948.Proceedings of the First IEEE Conference***, 1995, June Vol 2 pp.645-650.

[10] Rosin, C.D. and Belew, R.K., "New methods for Competitive Coevolution."

Evolutionary Computation,1997, 5:1-30.

[11] Storn, R., Price, K., "Minimizing the real functions of the ICEC'96 contest by differential evolution ", ***Evolutionary Computation, 1996., Proceedings of IEEE International Conference on 20-22*** May 1996 Page(s):842 – 844

[12] Price K., "Differential Evolution: A Fast and Simple Numerical Optimizer," ***1996 Biennial Conference of the North American Fuzzy Information Processing Society***, Jun. 1996, pp.524-527.

[13] Storn, R. and Price, K., "Differential Evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces", ***Technical Report TR***-95-012, ICSI

[14] 林豐澤，"演化式計算上篇：演化式演算法的三種理論模式"，***智慧科技與應用統計學報***，第 3 卷，第 1 期，2005 年 6 月，第 1-28 頁。

[15] 李愛國，"多粒子群協同優化算法"，***負旦學報***，第四十三卷，第一期：923-925 頁，2004。

[16] 張宏瑋，動態式準則評估於旅遊行程安排之最優化，***大同大學資訊經營所*** 碩士論文，第 8-11 頁。

[17] 鞏敦衛，孫曉燕"協同進化遺傳算法理論及應用"，***智能科技*** 2009 年 5 月，ISBN:9787030244642。