2009 TOPCO 崇越論文大賞

論文題目：

以雙演化策略為基礎的差分演化演算法

報名編號：GB020045

以雙演化策略為基礎的差分演化演算法

摘要

差分演化演算法(Differential Evolution；DE)是近年來演化式計算的熱門演算法之一，它擁有隨機搜尋的方法且具有優越的效能，因此常被應用於各種領域，如：資料探勘、電子工程、決策支援等，差分演算法也存在容易陷入區域最佳解、收斂不穩定等缺點。

故本研究提出以差分演算法和粒子群最佳化演算法(Particle Swarm Optimization；PSO)為基礎的 DEPSO 演算法(Differential Evolution Particle Swarm Optimization；DEPSO)進行改良。DEPSO 透過雙演化策略(Dual Evolution Strategy；EDS)改善差分演算法的缺點，並由實驗結果證明 DEPSO 的改良效果及在收斂上的穩定性。

關鍵字：差分演化式演算法(Differential Evolution；DE)、粒子群演算法(Particle Swarm Optimization)、DEPSO 演算法(Differential Evolution Particle Swarm Optimization；DEPSO)、雙演化策略(Dual Evolution Strategy；EDS)

壹、 緒論

一、 研究背景與動機

演化式計算是模擬大自然演化過程而建立的一種計算模式，透過群居式動物本身以及群體的行為啟發而來，很多學者針對這樣的行為進行數學模型之建立，並提出相關演算法，藉此來求解傳統演算法難以解決的組合最佳化問題，找出問題的最佳解，包含了近年來在最佳化問題求解中較為新穎且熱門的粒子群演算法及差分演算法，這些演算法也被廣泛的應用在各項領域，包括資料探勘、網路最佳化、工作排程、路徑規劃、決策支援等。

Kennedy 與 Eberhart (1995)曾提到粒子群演算法是源自於對鳥類的覓食行為，具有收斂快速、參數設定簡潔、流程容易實現及記憶性等優點，且在很多領域中被廣泛運用，但粒子群演算法也有容易過早收斂及陷入區域最佳解的問題存在。

差分演算法是近年來一種新穎的最佳化演算法之一，擁有隨機搜尋的能力、參數設定更為簡潔、效能佳並適用於大規模的最佳化問題，但差分演算法卻有收斂不夠穩定、演化後期收斂速度不佳、同時也有容易陷入區域最佳解的缺點存在。

二、 研究目的與問題

差分演算法及粒子群演算法，與較早的最佳化方法相比較，各有優勢存在，且在複雜的函數中，更能突顯其優越效能，故本研究要以差分演算法及粒子群演算法為基礎，建立一個雙演化之協同演算法架構，透過彼此訊息交流及儲存的機制，保留兩演算法之優勢且互相補足兩演算法之不足，達到提高求解效能的目的。

綜合以上所述，本研究主要的研究問題是提出雙演化策略改良差分演算法架構且增加其求解效能，希望提出之架構能提高未來在應用領域上的求解效率，且在實驗結果中，利用測試函數與近年來的文獻數據進行比較，證明其效能及有效性。

貳、 文獻探討

一、 粒子群演算法

粒子群最佳化演算法於 1995 年由 Kennedy 和 Eberhart 所提出，是屬於演化式計算中的一支。其核心構想源自於模仿自然界中鳥群或魚群所展現之群體智能(Swarm Intelligence；SI)行為。當粒子群在一個 n 維的空間中移動時，每一個階段中粒子的位置就代表一組可能的解，當粒子經過迭代後會獲得在自身經驗中所求取之最佳解，我們可以稱之為區域最佳解。並藉由粒子間區域最佳解的比較，我們可以找出一個最合適的解成為全域最佳解也就是所處理之問題解，同時其餘粒子將往全域最佳解的方向進行收斂。

粒子的移動向量通常會受到三個因素的影響：(1)粒子原有的慣性向量、(2)粒子本身的區域最佳解(3)整群粒子所找的最佳解，也就是全域最佳解的部分。

粒子群演算法的基本公式如下：

$$V_{id}(t+1)=w*V_{id}(t)+C_1*rand()*(P_{id}- X_{id})+ C_2*rand()*(P_{gd}-X_{id}) \quad (1)$$

$$X_{id}(t+1)= X_{id}(t)+ V_{id}(t+1) \quad (2)$$

公式(1)和(2)中，ω 為稱為慣性權重[15]，$V_{id}(t)$為粒子原本的速度，$V_{id}(t+1)$為粒子新的速度，X_{id}是粒子新的位置。C_1及 C_2為學習因子，而 *rand()*為介於 0 與 1 之間的隨機亂數值。

PSO 演算法的基本流程如下：

Step 1：初始化，包含參數設定及隨機初始化粒子的速度及位置。

Step 2：計算粒子的適應值。

Step 3：計算每個粒子新位置的適應值；對各個粒子，若粒子的適應值優於原來的區域最佳解，則將新的適應值替換原有的區域最解。

Step 4：根據每個粒子的區域最佳解做比較以找全域最佳解。

Step 5：以公式(1)和(2)更新粒子的速度及位置。

Step 6：如未能達到停止條件則回到 Step 2，若達滿足條件則輸出最佳解。

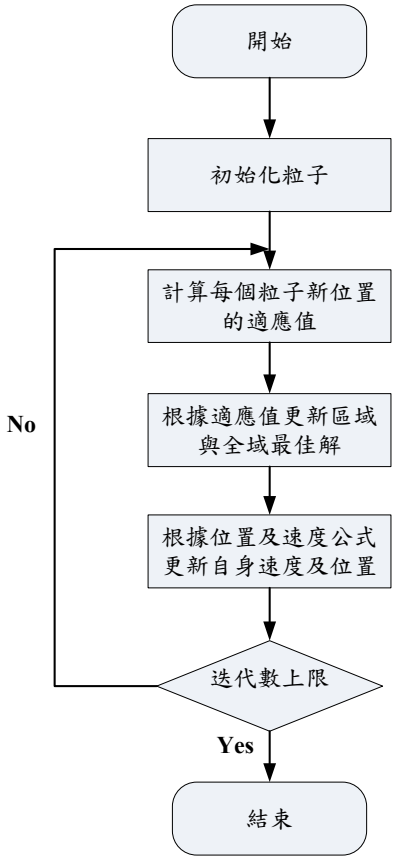

圖 1　PSO 流程

二、　差分演化演算法

(一) 發展背景

差分演算法是 1995 年由 Storn 及 Price 所提出，並在 1996 年第一屆國際演化式計算研討會中，Storn 等(1996)與 Price (1996)證明其求解的優越性，成為近年來熱門的最佳化演算法之一。差分演算法是以母體的差異性為基礎，Storn 等(1996)與 Price (1996)曾提出最早是用來求解柴比雪夫多項式適切性問題，透過一種隨機在解空間搜尋的演化式計算，並利用突變、重組、選擇等機制，不斷透過迭代的運算，依據每個個體的適應值去進行適者生存、不適者淘汰的機制，來達到問題的求解。

Hao 等(2007)曾提出差分演算法主要使用差異性的資訊來引導搜尋能力，但也因此造成在結果上呈現較不穩定的成效。差分演算法是演化式計算的一種，所以有其結構存在，但不同的地方在於演算法本身存在新的候

選解。

表 1　差分演算法及粒子群演算法優缺點

	差分演化演算法	粒子群演算法
優點	•維持母體的多樣性 •利用差異性的計算加強區域探索能力	•快速收斂 •具有記憶性
缺點	•收斂上的不穩定性 •易陷入區域最佳解	•母體多樣性較為不足 •過早收斂 •易陷入區域最佳解

早期的演化式計算以較早被提出的基因演算法被廣泛應用及探討，後來粒子群演算法及差分演算法陸續被 Kennedy 等(1995)、Price (1996)、Storn 等(1995, 1996)與 Shi(1998)提出，經由 Mayer 等(2005)研究證明了兩演算法的卓越效能，然而兩演算法間除了有演化式計算的共同優缺點外，它們彼此間各自也存有優勢及劣勢，整理如表 1 所示，從表 1 可以發現粒子群演算法及差分演算法能夠互相補足彼此的缺點，因此本研究希望透過迭代交換的雙演化機制，互相補足兩演算法之不足，達到提高求解效能的目的，因此以粒子群演算法及差分演算法為基礎提出雙演化策略。

(二) 差分演算法概念及流程

Mayer 等(2005)曾提出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. 初始化(Initialization)：設定差分演算法之參數值並隨機初始化其目標向量。

   $$X_{j,i,0} = rand_j(0,1) * (b_{j,U} - b_{j,L}) + b_{j,L} \quad (3)$$

2. 突變(Mutation)：突變有助於擴展搜尋空間，隨機選取三個變數向量 $X_{r1,G}$、$X_{r2,G}$、$X_{r3,G}$，並透過突變權重因子(Mutation weighting factor；F) ，將 $X_{r1,G}$ 和 $F(X_{r2,G} - X_{r3,G})$相加合成後，獲

得合成向量(Donor Vector)。

$$V_{i,G+1} = X_{r1,G} + F(X_{r2,G} - X_{r3,G}) \qquad (4)$$

3. 重組(Recombination)：重組是把前一代較好的解加以混合，透過突變後產生的合成向量，將與群體中所挑選出來的目標向量 $X_{i,G}$ 以交換率進行重組，並在重組之後產生試驗向量(Trial Vector)。

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if rand} \le CR \\ x_{j,i,G} & \text{if rand} > CR \end{cases} \qquad (5)$$

4. 選擇 (Selection)：在完成突變及重組之機制後，利用適應值計算目標向量 $X_{i,G}$ 及試驗向量，適應值較佳者會被當成下一個迭代的目標向量。

$$X_{i,G+1} = \begin{cases} u_{i,G+1} & if\ F(u_{i,G+1}) \leqq F(X_{i,G}) \\ X_{i,G} & otherwise \end{cases} \qquad (6)$$

差分演算法的基本流程如下：

Step 1：初始化參數。

Step 2：依據公式(3)隨機初始化目標向量 $X_{i,G}$。

Step 3：計算適應值。

Step 4：進行突變之運算，隨機選取數個變數向量，並以公式(4)或(5)進行突變，產生合成向量(Donor vector)。

Step 5：利用公式(5)進行重組之運算，以合成向量及目標向量 $X_{i,G}$ 進行重組，並在重組之後產生試驗向量(Trial Vector)。

Step 6：將目標向量 $X_{i,G}$ 和試驗向量比較，並以適應值計算，適應值較佳的一個會被當成下一個迭代的目標向量。

Step 7：未能達到停止條件則回到 Step2，若達滿足條件則輸出最佳解。

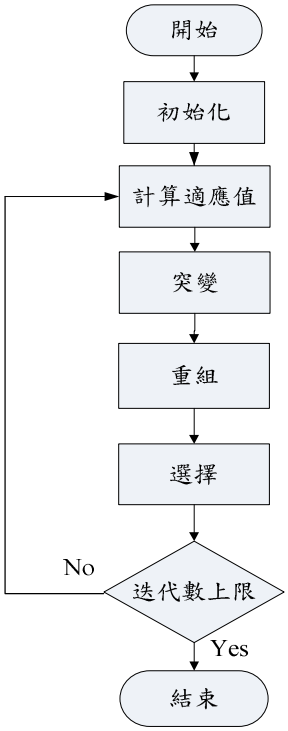

圖 2　DE 流程

(三) 差分演算法之常見策略

Qin 等(2003)與 Noman 等(2008)曾提出差分演算法常見的策略，主要是針對突變向量的公式去產生變化，常見的策略策整理如表 2 所示，可以看到策略標記的格式為 DE/x/y，x 表示變數向量的型態，y 則表示差異向量之個數，本研究在以差分演算法進行運算時，會以最常見的 DE/rand/1 策略進行實驗研究，利用隨機的方式達到較佳的成效，常見兩種的策略如表 2 所示。

表 2　差分演算法常見策略

策略標記	突變向量策略公式	概述
DE/rand/1	$V_{i,G+1} = X_{r1,G} + F(X_{r2,G} - X_{r3,G})$	隨機選取三個變數向量 $X_{r1,G}$、$X_{r2,G}$、$X_{r3,G}$，並透過突變權重因子(Mutation weighting factor；F) ，將向量 $X_{r1,G}$ 和 $F(X_{r2,G} - X_{r3,G})$ 相加合成後，獲得合成向量(Donor Vector)。此策略也是最常見的差分演算法策略。
DE/best/1	$V_{i,G+1} = X_{best,G} + F(X_{r2,G} -$	以 DE/rand/1 為比較基礎，該策略是將隨機

	$X_{r3,G}$)	選取的變數向量 $X_{r1,G}$ 由目前迭代的最佳粒子 $X_{best,G}$ 所取代。
DE/rand/2	$V_{i,G+1} = X_{r1,G} + F(X_{r2,G} - X_{r3,G} + X_{r4,G} - X_{r5,G})$	以 DE/rand/1 為比較基礎，該策略是多選取一組變數向量 $X_{r4,G} - X_{r5,G}$，擷取其中的差異，藉由更多向量取得差異來增加求解的多樣性。
DE/best/2	$V_{i,G+1} = X_{best,G} + F(X_{r2,G} - X_{r3,G} - X_{r4,G} - X_{r5,G})$	以 DE/rand/2 為比較基礎，該策略是將隨機選取的變數向量 $X_{r1,G}$ 由目前迭代的最佳粒子 $X_{best,G}$ 所取代。
DE/current -to-best	$V_{i,G+1} = X_{i,G} + F(X_{best,G} - X_{i,G}) + F(X_{r1,G} - X_{r2,G}))$	透過目前迭代中的最佳解、欲突變的粒子間的差異與一組隨機粒子的差異相加組合而成。

三、 雙演化策略

以粒子群演算法及差分演算法的雙演化的方式來進行差分演算法的效能改良，最早是由 Xu 等(2003)所提出，透過以差分演算法為主，粒子群演算法為輔的演算方式，並命名為 DE-SI 架構，DE-SI 和 Price 及 Storn 在 1995 年所提出的差分演算法的演算方式類似，主要是利用了合併演算法的構想，並改良粒子群演算法的速度更新公式，希望能夠得到一個快速收斂的效果。

Omran 等(2007) 提出粒子群演算法為主，差分演算法為輔的機制，該機制在粒子群演算法中加入了差分演算法的三個機制(突變，合併，選擇)去進行演算，其目的是為了解決粒子群演算法會有過早收斂的缺點。

綜合以上所述，本研究將過去的相關文獻整理，過去研究一般採用將粒子群演算法及差分演算法合併的一個想法，目的是為了結合兩演算法之優點，來加強其求解效能，並適時的應用在相關領域上面。 而它的類型主要可以分成兩類：

1. PSO 為主體，DE 當其演算機制

2. DE 為主體，PSO 當其演算機制

本研究提出了以迭代交換為基礎的雙演化策略，摒除了過去 Ning 等(2008)、Bipul 與 Ganesh(2006)、Luan 等(2007)、Omran 等(2007)、Xu 等

(2007)、Zhang 與 Xie(2003)、Xu 等(2008)與 Hao 等(2007)採用合併型態來改良差分演算法的方式，透過一定迭代分享彼此的粒子資訊，把好的結果利用迭代交換的方式分享給另一演算法，一方面能可以同時擁有粒子群演算法及差分演算法之優點，另一方面也可以互補彼此的不足。

參、 研究方法及設計

一、 實驗設計

本研究會以粒子群演算法及差分演算法當成 DEPSO 演算法之基礎，在進行差分演算法運算時，會採取 DE/rand/1 策略，而不採用 DE/best/1 求解效果較佳之策略，主要是要利用 DE/rand/1 策略隨機選取粒子的方法，減少落入區域最佳解的情況發生，並利用粒子群演算法的方向性特質避免粒子變動到錯誤位置。

DEPSO 演算法的基本流程如下：

Step 1：初始化參數。

Step 2：差分演算法及粒子群演算法分別進行演算，並比較兩演算法的適應值，選擇適應值較佳者開始進行演化。

Step 3：在奇數代中將演算法結果中排名前 20%的粒子進行分享，藉由較佳的粒子加快求解速度及成效。

Step 4：在偶數代中進行差分演算法及粒子群演算法間的競爭，選出較好的演算法來進行分享。

Step 5：未能達到停止條件前，回到 Step2，若達滿足條件則輸出最佳解。

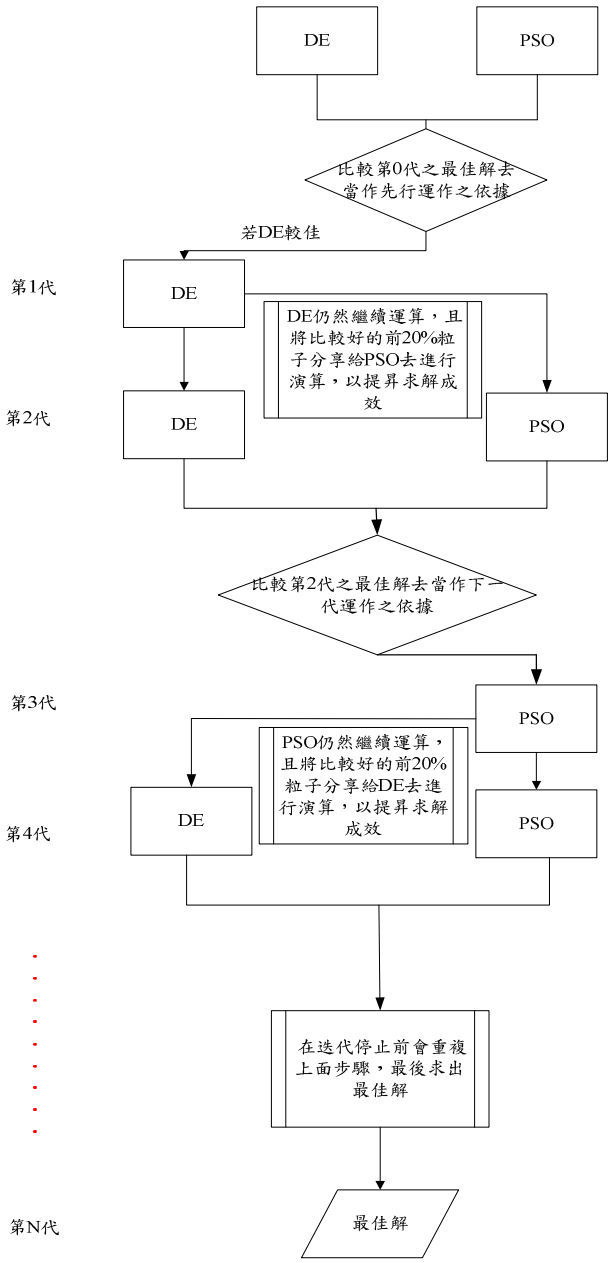

圖 3　DEPSO 運作流程

以下針對 Step 3 及 Step 4 進行說明，這兩個步驟是以奇數代分享，偶數代競爭的師徒制概念進行，如下所述：

(一) **分享**：

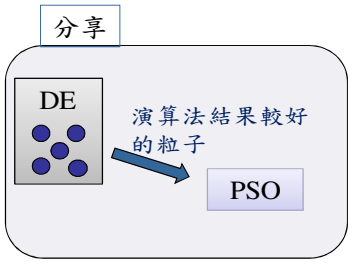

圖 4　師徒制策略－分享

在奇數迭代中，如同師傅帶領徒弟，將自身的知識給予徒弟進行學習。在本研究中，由目前結果較佳的演算法扮演傳承、分享粒子資訊的角色，希望可以使另一演算法能夠透過學習，進而產生較佳的結果。

(二) **競爭**：

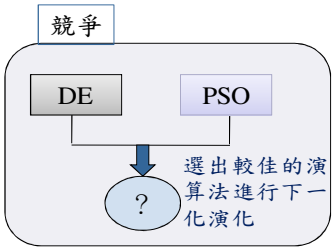

圖 5　師徒制策略－競爭

在偶數迭代中，師傅在將本身知識給予徒弟後，在未來會進行競爭。在本研究中，由差分演算法及粒子群演算法進行互相競爭的運作，選出較佳之演算法。

二、　參數設定

本研究在高維度實驗階段所使用之參數設定如表 3 所示，另外與 DEPSO 期刊比較之參數設定如表 4 所示，若要與其他文獻之實驗結果進行效能評估及比較時，則參照該研究之實驗參數設定。

表 3　高維度實驗階段參數設定

參數	值
粒子數 (*NP*)	50
精英解分享率 (elite)	0.2
維度 (dim)	30, 60
突變率 (*F*)	0.5
交換率 (*CR*)	0.9
最大迭代數 (iter)	1000,2000
解空間上界 (now_up)	100
解空間下界 (now_down)	-100
C_1（c1）	1.4
C_2（c2）	1.4
搜尋速度 (search *v*)	20
慣性權重 (*w*)	0.9-0.4
執行次數 (test_times)	200, 100

表 4　與 DEPSO 期刊參數設定

參數	值
粒子數 (*NP*)	40
精英解分享率 (elite)	0.2
維度 (dim)	10
突變率 (*F*)	0.8
交換率 (*CR*)	0.5
最大迭代數 (iter)	1000
解空間上界 (now_up)	100, 30, 5.12, 600
解空間下界 (now_down)	-100, -30, -5.12, -600
C_1（c1）	2
C_2（c2）	2
搜尋速度 (search *v*)	解空間/2
慣性權重 (*w*)	0.9-0.4
執行次數 (test_times)	20

三、　評估與測試

本研究主要是進行差分演算法求解效能之改善，因此會以四個最經典之測試函數為基礎，分別是單峰函數 *f*1(Sphere)、*f*2(Rosenbrock)及多峰函數 *f*3(Rastrigin)、*f*4(Griewank)來做為評估之準則，四個測試函數的最小值均為 0，如表 5 所示　：

表 5　測試函數

函數名稱	多峰函數	單峰函數	公式
Sphere function *f1*	否	是	$f_1(x)=\sum_{i=1}^{n}x^2$
Rosenbrock function *f2*	否	是	$f_2(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$
Rastrigin function *f3*	是	否	$f_3(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$
Griewank function *f4*	是	否	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$

肆、 實驗結果(一) - 高維度實驗

在本研究當中，首次提出以雙演化架構改良差分演算法，並在 30 維及 60 維的高維度解空間中進行實驗，透過與 DE/rand/1、DE/best/1 及 PSO 三種演算法分別在四個常見的測試函數上進行求解效能上的比較，可以看出 DEPSO 不論在單峰或多峰函數上均獲得相當程度的提升，並透過該架構的引入強化其收斂性以達到穩定收斂的效果。

一、 **測試函數**：Sphere (f1)

在 Sphere 函數的實驗結果中，可以看到在 30 維及 60 維中的效能較其他演算法獲得相當顯著的提昇，且透過圖 6 及圖 7 收斂圖所示，DEPSO 也呈現了穩定的收斂結果，整體來說在 Sphere 測試函數獲得了顯著的改善。

二、 **測試函數**：Rosenbrock (f2)

Rosenbrock 是屬於單峰中較為複雜的函數，所以可以看出四個演算法的實驗結果均差異不大，且由圖 7 可以看出 Rosenbrock 很容易陷入區域最佳解，但 DEPSO 整體來說仍然呈現出較穩定之收斂及較佳的求解成效。

三、 **測試函數**：Rastrigrin (f3)

在 Rastrigrin 函數的實驗結果中，透過圖 8 可以看出 DEPSO 在 30 維時

存在穩定的收斂性，且不像 DE/best/1 及 PSO 會有陷入區域最佳解的情況發生，而 DEPSO 在求解成效上也較其他三個演算法較佳。而在 60 維時 DEPSO 的實驗結果較 DE/best/1 略差，但較 DE/best/1 穩定收斂。

四、 測試函數：Griewank (f4)

在 Griewank 函數的實驗結果中，PSO 及 DE/best/1 會落入區域最佳解，而 DEPSO 及 DE/rand/1 則在收斂上呈現較佳的穩定性，也都不錯的求解成效，但整體來說，DEPSO 的成效仍然在四個演算法中表現為優越。

表 6　Sphere (f1)之實驗結果

f1 維度		DE/rand/1	DE/best/1
Avg	30	4.879650e-010	4.236681e-016
	60	2.218372e-002	1.932865e-019
Best	30	2.045238e-010	5.664503e-017
	60	1.259653e-004	5.920183e-020
	維度	PSO	DEPSO
Avg	30	5.376984e-007	2.130045e-026
	60	1.843294e-011	8.930297e-027
Best	30	1.283394e-011	1.216424e-035
	60	7.135705e-012	8.666767e-65

表 7　Rosenbrock (f2)之實驗結果

f2 維度		DE/rand/1	DE/best/1
Avg	30	1.290789e+002	6.460700e+001
	60	1.195137e+002	1.286586e+002
Best	30	4.479539e+001	2.831109e+000
	60	6.195290e+001	4.604165e+001
	維度	PSO	DEPSO
Avg	30	6.765685e+001	2.633769e+001
	60	5.703899e+002	5.714797e+001
Best	30	8.236947e+000	2.522003e+001
	60	2.129353e+002	5.521501e+01

表 8　Rastrigrin (f3)之實驗結果

f3 維度		DE/rand/1	DE/best/1
Avg	30	4.499533e+000	5.372812e-001
	60	1.035306e+002	3.872560e+001
Best	30	4.960861e-001	2.644430e-008
	60	8.537815e+001	2.665797e+000
	維度	PSO	DEPSO
Avg	30	6.539890e+001	1.744925e-001
	60	2.413862e+002	9.095539e+001
Best	30	2.805312e+001	0
	60	1.337377e+002	6.997933e+01

表 9　Griewank (f4)之實驗結果

f4 維度		DE/rand/1	DE/best/1
Avg	30	8.943127e-010	8.499917e-004
	60	1.161345e-012	4.190023e-004
Best	30	2.645439e-011	2.705168e-002
	60	2.832179e-013	0
	維度	PSO	DEPSO
Avg	30	1.384572e-002	1.155985e-011
	60	1.289793e-002	9.547918e-016
Best	30	3.339440e-012	0
	60	9.445777e-013	0

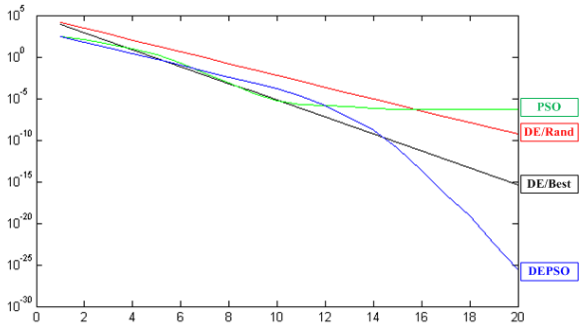

圖 6　Sphere (f1) – 30 維收斂曲線

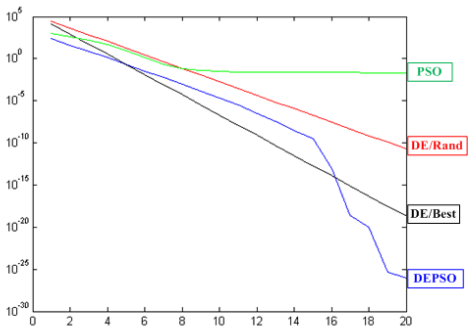

圖 7　Sphere (f1) –60 維收斂曲線

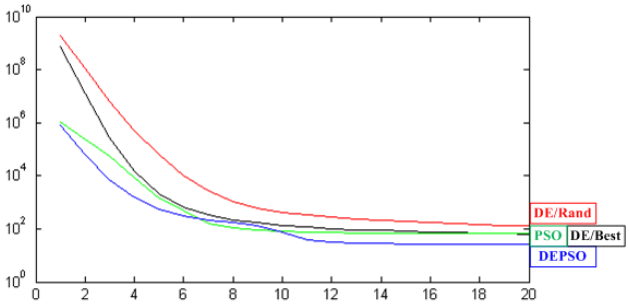

圖 8　Rosenbrock (f2) – 30 維收斂曲線

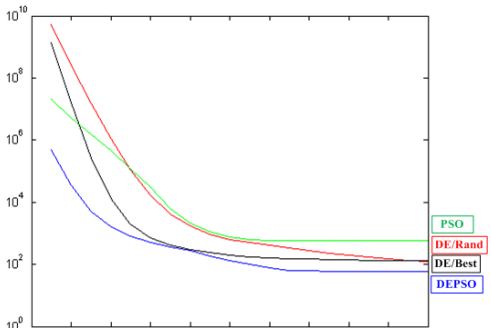

圖 9　Rosenbrock (f2) – 60 維收斂曲線

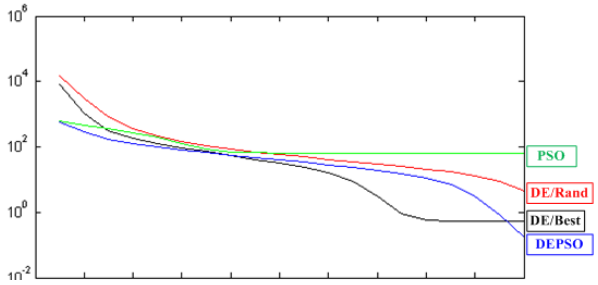

圖 10　Rastrigrin (f3) – 30 維收斂曲線

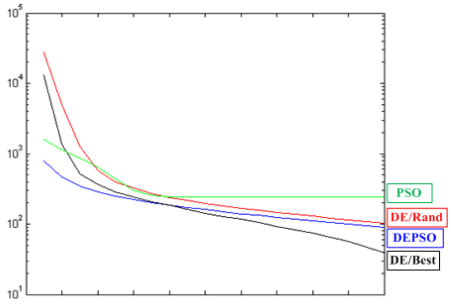

圖 11　Rastrigrin (f3) – 60 維收斂曲線

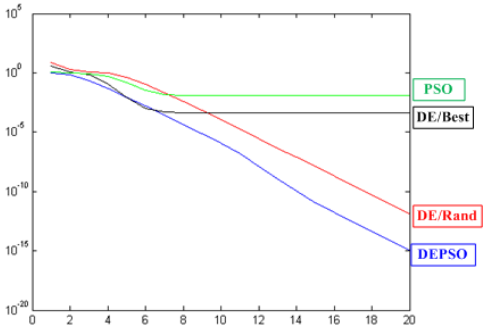

圖 12　Griewank (f4) – 30 維收斂曲線

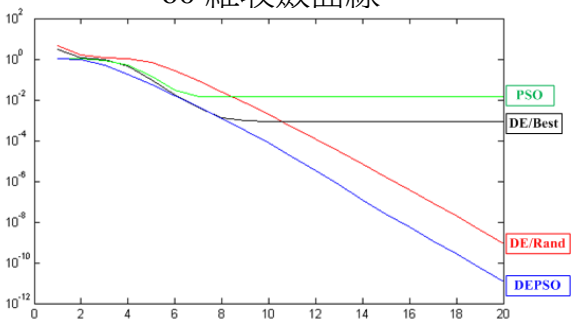

圖 13　Griewank (f4) – 60 維收斂曲線

伍、 實驗結果(二) - 期刊實驗比較

在本研究當中，選擇 Ben 與 Li 所發表的期刊為比較依據，證明本實驗的有效性及改良之效能，此篇研究是所提出的 PSODE 演算法是目前唯一一篇雙演化中收錄在期刊的文獻，並由該研究實驗證明較其他雙演化之文獻卓越，故本研究選擇此篇期刊當作比較之依據。

表 10　實驗數據

Function	Results	DE/rand/1	PSO	DEPSO	PSODE
Sphere	Mean	2.89E-33	6.69E-40	3.809E-66	2.286E-51
Rosenbrock	Mean	2.24E+00	4.13E+00	1.107E+00	8.804E-01
Rastrigrin	Mean	4.89E-10	4.18E+01	6.729E-13	7.960E-01
Griewank	Mean	7.04E-02	7.53E-02	0.000E+00	3.101E-02

一、 **測試函數**：Sphere (f1)

在 Sphere 函數的實驗結果中，可以看到在 10 維的解空間中，本研究所提出的 DEPSO 演算法較其他演算法獲得相當顯著的提昇，其中也包括 PSODE 演算法。

二、 **測試函數**：Rosenbrock (f2)

Rosenbrock 是屬於單峰中較為複雜的函數，所以可以看出四個演算法的實驗結果均差異不大，但 DEPSO 整體來說仍呈現出較較佳的求解成效。

三、 **測試函數**：Rastrigrin (f3)

在 Rastrigrin 函數的實驗結果中，本研究所提出的 DEPSO 演算法在求解成效上也較其他三個演算法較佳。

四、 **測試函數**：Griewank (f4)

在 Griewank 函數的實驗結果中，本研究所提出的 DEPSO 演算法已經收斂至該 Griewank 函數的最佳解，更能證明本研究的成效。

陸、 結論

本研究以差分演算法及粒子群演算法為基礎，透過彼此資訊分享的機制，互補彼此的優缺點，加強探索能力及維持演算法的多樣性，使演算法能以較少迭代數就可以進行收斂，得到最佳解並有效降低了易陷入區域最佳解的機會。且利用常見的四個測試函數進行比較，證明其求解效能、穩定性及有效性。

本研究所提出之 DEPSO 演算法，透過高維度實驗結果中可以看出 DEPSO 在效率上相較於 DE/rand/1、DE/best/1 及 PSO 有顯著的提昇，且與雙演化之期刊實驗進行比較，透過兩者實驗即能證明此演算架構的有效性；且在求解平均值上也呈現出差分演算法透過雙演化架構及師徒制機制，使得差分演算法更具穩定性，減少落入區域最佳解的機會；而在求解效能上從平均值及最佳值可以看出其改良後的優越成效，未來可把 DEPSO 為基礎，在應用及改良上繼續延伸，在應用領域中，可以應用於資管領域，如：資料探勘、排程問題、路徑規劃等，並去探討其成效；而在改良領域中，可以容易的將改良後之差分演算法及粒子群演算法加入，進而再強化其求解效能。

柒、 參考文獻

Ben N. and Li. L. (2008), “A Novel PSO-DE-Based Hybrid Algorithm for Global Optimization”, Computer Science, Vol. 5227, pp. 156-163.

Bipul L. and Ganesh K. V. (2008), “Differential evolution particle swarm optimization for digital filter design”, 2008 IEEE Congress on Evolutionary Computation,pp. 3954-3961.

Hao Z.-F., Guo G.-H., and Huang H. (2007), “A Particle Swarm Optimization Algorithm with Differential Evolution,” 2007 IEEE International Conference on Machine Learning and Cybernetics, Vol. 2, pp. 1031-1035.

Kennedy, J. and Eberhart, R. C. (1995), ”Particle swarm optimization”, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. , pp. 1942-1948.

Luan L.-J., Tan L.-J. and Niu B. (2007), "A Novel Hybrid Global Optimization Algorithm Based on Particle Swarm Optimization and Differential Evolution", Information and Control, Vol. 36, No. 6, pp. 708-714.

Mayer, D. G. and Archer, A. A. (2005), "Differential evolution – an easy and efficient evolutionary algorithm for model optimisation", Agricultural Systems, Vol. 83, pp. 315–328.

Ning, D. W. Zhang and Li, B. (2008), "Differential evolution based particle swarm optimizer for neural network learning", 2008. 7th World Congress on Intelligent Control and Automation, pp. 4444-4447.

Noman N. and Iba H. (2008), "Accelerating Differential Evolution Using an Adaptive Local Search", IEEE Transactions on Evolutionary Computation, Vol. 12, No. 1, pp. 107-125.

Omran M. G. H., Engelbrecht A. P. and Salman A. (2007), "Differential Evolution Based Particle Swarm Optimization", Proc. the 2007 IEEE Swarm Intelligence Symposium, pp. 112–119.

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

Qin K., Huang V. L., and Suganthan P. N. (2003), "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization", IEEE Transactions on Evolutionary Computation , pp. 1-20.

Storn R. and Price K. (1995), "Differential evolution–A simple efficient adaptive scheme for global optimization over continuous spaces", Technical report, California: International Computer Science Institute, Berkeley.

Storn R. and Price, K. (1996) "Minimizing the real functions of the ICEC'96 contest by Differential Evolution", Int. Conf. on Evolutionary Computation, Nagoya, Japan, pp. 842-844.

Shi Y. and Eberhart R. C. (1998), "A modified particle swarm optimizer", Proceedings of IEEE International Confernce on Evolutionary Compution, Piscataway, NJ., pp. 69-73.

Xu R., Venayagamoorthy G. K. and Wunsch D. C. (2007), “Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization”, ESCI on Neural Networks Vol. 20, pp. 917-927.

Xu X., Li Y., Fang S., Wu Y., and Wang F. (2008), “A novel differential evolution scheme combined with particle swarm intelligence”, 2008 IEEE Congress on Evolutionary Computation, pp. 1057-1062.

Zhang W. J. and Xie X. F. (2003), “DEPSO: Hybrid Particle Swarm with Differential Evolution Operator”, Proc. IEEE International Conference on Systems, Man and Cybernetics, pp. 3816–3821.