搭配擾動策略之差分演化演算法

李維平
中原大學資管所
助理教授
wplee@cycu.edu.tw

江長育
中原大學資管所
研究生
g9794012@cycu.edu.tw

摘要

1996 年首次提出差分演化演算法(Differential Evolution；DE)因其具備結構簡單、高效率、高精確度及所需設定參數較少等特性而受到重視，並被廣泛地應用於許多領域中，與以往常見之演化式演算法不同的是，其更為夠展現出優良的求解成效。從許多相關研究發現，差分演算法本身雖具備強大的求解能力，但對於最佳化問題的求解上，至今仍還存在部分疑慮尚待解決，如收斂情形的不穩定以及一般演化式演算法所常見的通病“易陷入區域最佳解” 之問題，導致差分演算法之效能有所限制。故本研究提出擾動策略之差分演算法，期望透過擾動策略的導入，能夠獲得全面性的穩定成效。

關鍵詞：差分演算法;演化式演算法;擾動策略運算子;最佳化

Abstract

In 1996, differential evolution algorithm was proposed by Storn and Price. The DE has simple structure, high efficiency, high accuracy and required less set the parameters and it has been widely used in many fields. From a number of related research found that although the differential algorithm itself has good ability to solve problems, there are still some doubts to be resolved, such as the convergence situation of instability and the general problem "easy to fall into the local optimal problem", so these shortcomings leading to restrictions on the performance differential algorithm. Therefore, this paper proposes a perturb strategy, and expect the comprehensive stability of effectiveness.

Keywords: Differential Evolution; Evolution Algorithm; Perturb Strategy Operator; Optimization

1. 前言

從古自今，生物界就是在現實社會中各種工程技術、科技原理及突破性發展的創新來源，而人類在探索自然界奧妙的同時也不斷地模仿生物界的運作方式，並且將生物的行動模式、功能、組織與系統等方面的研究，包括生物群體、身體或是器官結構、覓食行為、以及系統功能等，藉由研究、觀察進而認識了解生物體活動的核心本質及各式各樣的本領，將這些原理應用於人類科學技術以及社會發展等層面並提供新的設計構想或是系統架構[1]。

模仿生物的概念早在幾個世紀以前就已經出現，像是研究鳥類的飛行跟海豚快速游動的方式等，在 20 世紀的中期仿生學就此誕生，其廣義的解釋是以自然界的生物做為仿效對象，美國空軍軍官 Jack E. Steele 為仿生學定義為：仿生學的發展主要是根據模仿生物系統原理來建造出技術系統，或是讓人造系統具有類似於生命系統特徵的科學，經過短短這數十年蓬勃發展，演變到現在其相關研究及應用相當廣泛，研究貢獻也非常可觀[1][2][3]。

此外，研究學者也從不同的角度去獲得研究靈感，如透過生物演化的機制，因而提出的演化式計算，其概念是從蟻群、

魚群、鳥群等生物具有一種群體智慧之特性，進而創造許多相關的演化式演算法，如基因演算法、螞蟻演算法到近期較新穎且熱門的粒子群演算法[1]、差分演算法、蜜蜂演算法等。

演化式演算法 (Evolutionary Algorithms; EAs) 是一種採取隨機搜尋的方法模擬自然界的選擇與生物的演化機制，演化式演算法不同於傳統的求解最佳化的方法如模擬退火法、攀登演算法[5]，是因在求解問題的時候，演化式演算法相較於其他的演算法，其是以一個群體維持一定數量的解決方案來進行的，而不單僅以一個解決方案，而這些不同類型的演算法被廣泛地用以解決現實當中許多複雜且困難的組合最佳化問題或是相關領域如

(1) 路徑規劃[6][7]
(2) 背包問題[8][9]
(3) 圖像分群[10][11]
(4) 網路最佳化[12][13]
(5) 決策支援[14]
(6) 工程設計[15][16]

當遇到求解問題的複雜度提高時，傳統的數學計算模型(如:線性規劃)，會造成求解效率的低落或是運算過程冗長，進而降低在所需的時限內所要求的求解品質。研究學者從過去的相關文獻發現，演化式計算具有求解能力的優越性，其在相關設計過程中，所包含的共通特性如下所列[17]：

(1) 關於搜尋的環境，幾乎沒有先前的知識經驗。
(2) 在有限的搜尋空間中，具備卓越的搜索能力。
(3) 避免落入區域最佳解的能力。
(4) 處理較高維度的能力。
(5) 針對廣泛的各種最佳化問題，較具有穩健性。
(6) 提供多組良好的求解方案。
(7) 找出座落於解空間全域最佳解的能力。

差分演化演算法是在 1995 年，由 Storn 及 Price 所提出[5][18][19]的，該演算法是近年來熱門的最佳化演算法之一，其繼承演化式計算的演算法之求解特性。在過去的文獻中發現，其皆能表現出不錯的成效，並廣泛地被應用於各項領域中，其演算概念類似於演化式計算，除了擁有隨機搜索的能力外，其架構還包含了突變、重組及選擇運算子等機制，以及參數設定更為簡潔。不過，雖其擁有這麼多優勢，但是目前的研究方向大多是偏向於應用問題居多，針對單純改良差分演算法架構或是流程之研究相對較少，因此，本研究認為這是可以進行深入研究的地方，特別是在改善各種最佳化問題的求解能力，以及思考如何提出切入改良的重點，進而達到提升求解的精準性及穩定性。

雖然過去研究中可以發現到演化式計算的優越求解成效，使其受到研究學者的關注，並投入大量的精力對此議題進行研究探討，但是演化式計算還是會出現常見的問題(如:易落入區域最佳解、參數設定繁多、運算時間過長、實作不易等)，使其無法滿足現實中各種問題之需求，這些缺陷也造成後續的研究學者，為了研究演化式演算法之相關應用時，就必須花費更多的時間在了解其流程架構，或是等待其處理計算的時間過長等問題。

差分演算法，與其他較主流的演化式演算法相較下，其具備的優勢包含參數設定數少、強健性、實作容易、高準確性以及收斂快速[5][20]等特性。從過去相關研究的實驗數據，可以發現差分演算法的優越性優勢，以及對於未經過改良的差分演

化演算法會優於其他改良過的演化式演算法之求解效能，雖然差分演算法有上述所提及的優點，但還是會遇到演化式計算的關鍵缺陷，如易陷入區域最佳解，或是跳脫出解空間等問題。

過去的相關研究中，差分演算法常見的單純改良方式有控制參數改良[21][22]、修改演算法架構流程[23]和搭配其他的演算法[20][24]，其改良最主要的目的就是讓探索能力與開發能力達到一個平衡，探索即是在解空間搜尋到全域最佳解的能力，當探索能力不佳時，就易陷入區域最佳解，而開發能力意味著在全域最佳解附近挖掘出更佳的解，當開發不夠完善即有可能造成收斂不穩定，運算時間拉長以及降低解決方案的品質。

故本研究藉由導入機制於差分演算法之中，並透過擾動策略對收斂情形不穩定及陷入區域最佳解等兩大問題進行改良，實證出本研究所改良之差分演算法能在探索與開發兩者取得一個平衡。

2. 文獻探討

2.1 演化式計算

演化式計算(Evolution Computation; EC)在近十幾年頗為盛行，是屬於新穎的搜尋技術，也大大改變現行的工程計算結構，演化式計算係利用計算機模型的進化過程與選擇，來模擬達爾文進化論”適者生存，不適者淘汰”為基礎的自然界生物的演化過程，其理論不僅在社會層面、宗教信仰產生巨大的激盪效應，在科學技術的發展上也逐漸展現其影響性。其所應用的領域涵蓋了生物工程、資訊、電機等等，而計算機模型即是所謂的演化式演算法並被利用求解各種許多最佳化組合問題在許多工程與科學的相關領域[17]。

演化式計算的理論基礎因從仿生學的概念衍生而成，與生物演化的過程非常相似，根據環境的變動，生物為了生存須適應自然環境，演化後的結果會產生多樣性且高度適應性新品種，每一個新品種也都可以持續地適應不斷變遷的環境。

其演化式計算的研究人員根據初始階段建立出複雜的適應模型，其適應模型參考了許多學科理論，例如：演化理論、遺傳學與細胞生物學，所以在針對一個最佳化的問題上所產生的一個候選人(Candidate)解決方案被稱為個體(Individual)或是染色體，而當前的解決方案(Solution)的集合被叫做群體(Population)，其實際的個體表示方式稱為染色體(Chromosome)，染色體是由序列的多個基因(Gene)所組成，即描述個體的屬性，這也可能就是求解問題的答案，而數個基因結合一起所形成的特性稱做對偶基因(Allele)，當個體解決方案經由修改產生新的候選人解決方案，稱為後代(Offspring)或是子代，當在評估新的子代時，會有一個評估準則分數，稱做適應值(Fitness)，表明了在解決某一問題時的品質，當目前的群體被其後代所替換，新的群體即稱為新的一代，最後，尋找最適合的解決方案的整個過程即稱為演化，也就是一個世代(Generation)[17][4]。

演化式演算法(Evolution Algorithms; EAs)是以群體為基礎的搜尋式演算法去模擬個體的演化架構的相互關聯性的選擇、重組、變異的過程，已經有許多相關演化式演算法研究被提出和證明，基本上都同意這一理念的基本假設，不同的是演化策略，被使用在不同演化式演算法的實做中[17]。

從工程學的角度來看，演化式計算可以解讀成是一個搜尋和優化過程中，群體

會經歷解決方案的逐漸變化過程，這最佳化的過程取決於個體的解決方案在已定義目標函數環境底下(Objective function)的適應值[5][17]。

根據演化式演算法的觀念所發展的規範包括以下步驟[17]:

(1) 初始階段：隨機產生初始的群體。
(2) 評估階段：評估群體中的每一個體的適應值，若符合終止條件即終止演化，反之繼續下列步驟。
(3) 選擇階段：
   a、 從群體中挑選個體當作父代。
   b、 經過不同的遺傳運算子從父代中產生出新的子代。
   c、 評估子代的適應值。
(4) 產生階段：決定部分或全部取代當前的個體成為下一代的群體，再回到步驟 2。

其中最主要也是最早發展的三個演化式演算法為演化式策略(Evolution Strategy; ES)[17][25]、演化式規劃(Evolution Planning; EP) [17][26]、基因演算法(Genetic Algorithm; GA)[17][27]，並可以把上述所提稱為演化式演算法的三種主流架構或基本的理論模式[4][17]。

在本節，本研究提到了演化式演算法相關背景，也因為具有不錯的求解效率如本研究所實驗的實數型最佳化問題上，使得後續也持續提出許多演算法像粒子群演算法、細菌演算法、蜜蜂演算法等，並還可以擴展其他問題的領域，其中包含了非線性限制式最佳化問題、離散型最佳化問題、多限制式最佳化問題等。

2.2 差分演算法

1995 年，Storn 跟 Price 提出一個新穎的差分演化演算法[28][29]，差分演化是以浮點數編碼方式的演化式演算法針對全域最佳化連續型的解空間，也可以以離散的編碼方式進行[30]。差分演化演算法與前一章所提基因演算法相似的地方是都是利用由個體所組成的群體來搜尋最佳解，而其中最主要不同的特性是突變運算子，在基因演算法中，只有很小的機率執行突變機制，因此，也可被稱為背景運算子。另一方面，突變在差分演算法中是利用算數公式來組合個體在每一代中，也就是說，突變再差分演化演算法並非是採取預先定義好的機率值的方式進行，演化過程中的初始階段，差分演化演算法的突變將扮演探索者，並隨著演化的過程收斂到一定的代數時，其群體內的個體會越來越相似，其差異的向量也會自動的變小，突變運算子則是成為具有開發能力的角色[4][31]。

差分演化演算法是個只需少數參數設定、易實作與快速收斂等特性，然而，再處理雜訊的相關問題時，也因具有類似貪婪快速收斂性質，可能會影響其效能[4][32]，差分演化創造一個新的候選解決方案藉由母代與其他幾個個體從相同的母體組合出來，當候選的解決方案取代其母體是只有候選的解決方案能比母體有更佳的適應值。差分演化有三個控制參數: 差分向量的放大係數(F)，重組控制參數(CR)與群體大小(NP)，以下將針對這三個控制參數來做說明[31][33][34]：

- 群體數量(Number of population；NP)：

初始化的群體要盡可能地散布在測試函數上得解空間中，當群體數量大時，會增加群體的多樣性，但運算時間也隨之拉長，相反地，當群體數量越小時，雖然運算時間縮短，不過在群體的多樣性方面可能效果會比較不顯著，在許多文獻指出，

群體數量通常設為 10*維度(Dimension)是不錯的設定方式[34]。

- 差異向量的放大係數(Amplification factor of the difference vector；F)：

F 值的設定會關係到收斂的速度，當 $F>1$ 時，能夠解決許多問題，而 $F<1$ 時，通常更能夠兼具快速與有效性，不過，群體內個體的最佳適應值不宜收斂太快，否則會停滯在區域最佳解而無法跳脫出來，因此，F 值通常超過一個臨界值來避免過早收斂到區域最佳解的問題，若 F 值過大，則找到最佳解的評估適應值次數會快速增加，造成收斂速度緩慢，絕大部分的 F 值範圍都會在[0,2]來做取捨，而傳統差分演化演算法的作者認為介於[0.5,1]之間是不錯的選擇[33]。

- 重組率(Crossover rate；CR)：

在傳統差分演化演算法上，CR 值的設定扮演相當重要的角色，因為在針對最佳化問題的特性與複雜度是更加敏感的，並能夠進一步增加群體的多樣性，通常 CR 值範圍會在[0,1]之間，當 CR 越小甚至到 0 時，影響突變運算子的效果也將會慢慢縮小，因試驗向量將直接繼承突變過後的向量，另外當 CR 值越高，則新產生出來的試驗向量將與目標向量有很高的相似度，Storn 認為 CR 值在[0.8,1]能幫助提升解的成效[34]。

差分演化演算法已被成功地解決廣泛的各種最佳化問題，如分群問題(Clustering)、非監督式圖像分類(Unsupervised image classification)、數位濾波器設計(Digital filter design)、非線性函數最佳化問題(Optimization of non-linear functions)、非線性化工流程的全域最佳化(Global optimization of non-linear chemical engineering processes)、多目標最佳化問題(Multi-objective optimization)等，總而言之，差分演化演算法現在被普遍認為是個準確性、可靠性、強健性的快速搜尋的演化式演算法[4]。

不同於其他的演化演算法，差分演算法不利用一些機率分布函數來引入不同的群體，相反地，差分演算法使用不同隨機選擇的差異向量(個體)成為第三個突變向量(Mutant vector)產生出無規則的變化的來源，視為突變流程，而被產生的詴驗向量(Trail vector)藉由已加入的權重差分向量與目標向量(Target vector)產生[5]，重組(交配)流程的步驟是被應用於產生出新的後代，當新的後代要被接受時，是跟父代的個體做比較看其適應值是否有改善，圖 1 為傳統差分演化演算法的整體流程概念圖：

本節將逐一詳盡介紹差分演化演算法之演化概念步驟：

初始化(Initialization)：設定控制參數值、隨機產生初始個體。

突變(Mutation)：隨機挑選出三個不同的個體向量 X_{r1}、X_{r2}、X_{r3}，先運算 $X_{r2}(t)-X_{r3}(t)$ 的差異向量，再與 X_{r1} 透過計算成為突變向量(Mutant vector)，運算公式如下：

$$v_i(t)=X_{r1}(t)+F*(X_{r2}(t)-X_{r3}(t))$$

重組(Recombination)：差分演化演算法採取離散型的重組方法，也就是利用二項式的交配機制，從目標向量 $X_i(t)$ 中的元素與突變向量 $v_i(t)$ 裡的元素去產生出試驗向量 $u_i(t)$，其公式如下：

$$u_{ij}(t)=\begin{cases} v_{ij}(t) & if\ rand<CR\ or\ j=r \\ x_{ij}(t) & otherwise \end{cases}$$

r 指的是說在個體向量維度範圍內的隨機亂數，目的是至少試驗向量會有一個元素會採取突變向量的元素，確保當 CR 值等於 0 時試驗向量與目標向量的不同，以圖

2 輔助說明重組的概念。

選擇(Selection)：差分演化演算法採取一種相對簡單容易實做的方法，當子代$u_i(t)$要取代父代$X_i(t)$時，只有當其適應值優於父代$X_i(t)$時，子代才會作取代的動作，反之，則繼續保留父代到下一迭代。

Storn 與 Price 在 2005 年提出 10 種不同版本用在個體間差異的突變機制，其中比較著名的如表 1 所示：

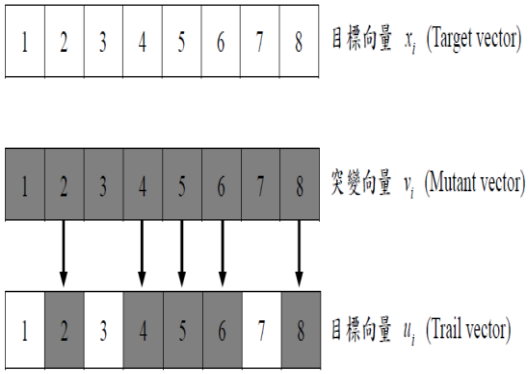


圖 2 差分演算法重組示意圖[21]

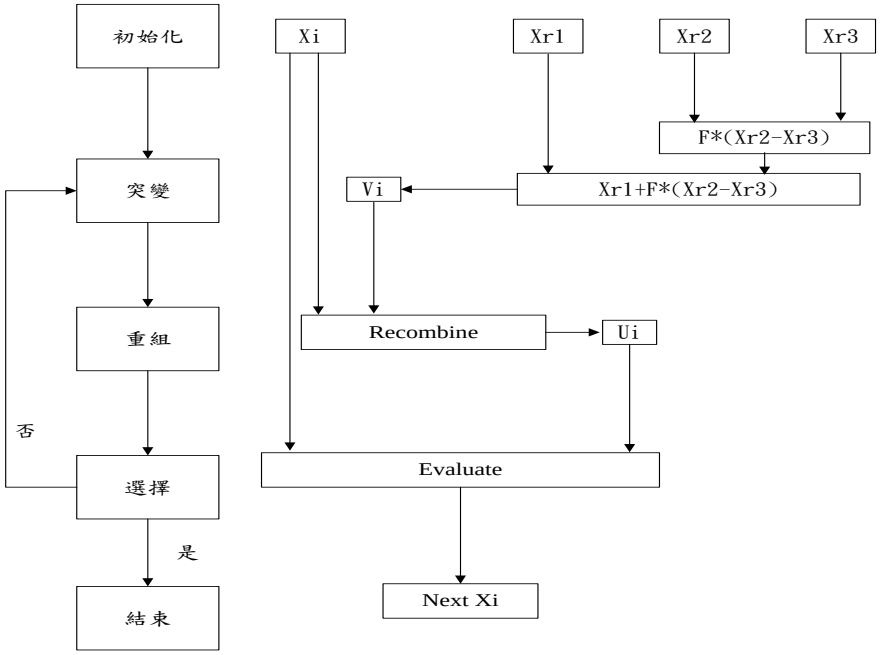


圖 1 差分演算法流程圖

表 1 常見的差分演化演算法突變機制[5][34]

突變設計方式	說明	
DE/x/y	x 代表做突變的個體擾動的方式，y 指的是幾組個體差異向量用於 x 的擾動方式。	
突變機制	簡述	公式
DE/rand/1	隨機選取出三個個體向量，並只有產生一組差異向量組，此策略是目前最常廣泛成功應用的突變策略。	$X_{r1}+F*(X_{r2}-X_{r3})$
DE/best/1	基本上，DE/best/1 策略大致上與 DE/rand/1 策略相同，除了 Xr1 改成 Xbest，Xbest 是在目前迭代的群體中最佳的個體向量。	$X_{best}+F*(X_{r2}-X_{r3})$

表 1(續)常見的差分演化演算法突變機制[5][34]

DE/best/2	基於 DE/best/1 衍生出來的策略，DE/best/2 使用了兩組差異向量在突變上，根據中央極限定理，其變化被平順地偏移到高斯方向，且常應用到多種函數上。	$X_{best}+F*(X_{r2}-X_{r3})+F*(X_{r4}-X_{r5})$
DE/rand/2	基於 DE/rand/1 所衍生出來的策略，也使用了兩組差異向量在突變上，希望能夠更廣泛搜尋達到群體的多樣性。	$X_{r1}+F*(X_{r2}-X_{r3})+F*(X_{r4}-X_{r5})$
DE/rand-to-best/1	基於 rand 與 best 的概念，能在群體中隨機挑選個體與最佳的個體，希望能達到廣泛的搜尋與較快的收斂速度。	$X_{r1}+F*(X_{best}-X_{r1})+F*(X_{r2}-X_{r3})$
DE/current-to-best/1	透過目前迭代中的最佳的個體與目標個體間的差異還有一組隨機粒子的差異相加組合而成。	$X_{i}+F*(X_{best}-X_{i})+F*(X_{r1}-X_{r2})$

3. PsDE 演算法

本研究所提出之改良差分演化演算法 PsDE 之建立構想針對差分演算法的主要缺點如陷入區域最佳解、收斂情形不穩定等，其主要的改良方法有三種在第一章有提到：

(1) 修改差分演算法運算流程或是架構

(2) 改良控制參數

(3) 搭配其他演算法

當然，還是有許多其他類的改良方法結合兩種以上的方法來進行改良，不外乎就是希望能到求解的精確性、穩定性、以及強健性，而在傳統差分演算法的重組運算上，以離散的二項式方式來選擇目標向量或是突變向量的元素，藉此達到群體的多樣性，因此，利用其個體向量的離散組成方式，透過個體本身的元素交換，來尋求更多樣的解向量。

本研究將改良差分演算法的運算架構，導入擾動策略的機制，首先將針對演算法架構中在進行突變運算子前執行擾動的動作，目的是希望當落入區域最佳解時，能給予個體再次跳脫出區域最佳解的機會，而在能夠繼續演化的同時，也促使差分演算法擁有更為廣泛多元的解空間搜尋能力。

3.1 改良差分演化演算法流程

Step1: 在 D 維度解空間中，定義如何初始解空間與個體上下界、個體個數、迭代數、控制參數值。

Step2: 計算適應值。

Step3: 擾動策略。挑選目前最佳的個體。

Step4: 隨機挑選元素來進行互換。

Step5: 進行突變運算。

Step6: 進行重組運算，將突變向量與目標向量執行二項式的重組方式，以 CR 值來決定試驗向量的元素是以突變向量還是目標向量為主。

Step7: 進行選擇運算，評估目標向量與試驗向量的適應值，挑選較佳的向量並保留至下一代。

Step8: 假如達到終止條件，則輸出最佳的運算結果，反之，回到 Step 2 繼續演化。

3.2 擾動策略的啟發與概念

在演化式演算法的缺點中，落入區域最佳解這一項缺點在過去的文獻常常是研究的重點之一，因此在本研究的實驗過程中，發現到差分演算法當落入區域最佳解時，某幾個維度會不再繼續演化，直到演化結束，如圖 4 所示，並每當落入區域最佳解時，陷入區域最佳解的維度不是固定的，因此啟發了本研究改良的靈感來源。

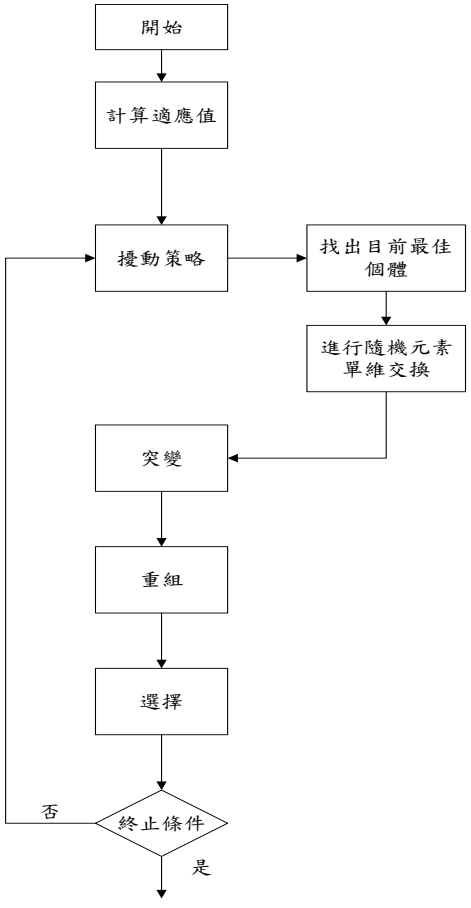


圖 1 雙突變策略差分演算法流程

針對期望能增加本研究所提的改差分演算法在全域探索能力與區域開發能力，首先，針對目前迭代中的最佳個體向量來進行擾動的動作，也就是從最佳的個體向量 Xbest 中，隨機挑選兩個元素來進行單維交換的動作，如圖 5 所示。其主要目的是在一開始的演化中，由於每一個體向量的差異非常大，透過這樣的機制能夠達到跳躍性的全域探索能力，緊接著在演化過程的中後半段，群體中的個體向量會開始慢慢收斂，主要是因為個體間的相似度也會越來越高，透過擾動策略機制，能達到群體多樣性以及提升跳脫出區域最佳解的能力，達到全域探索與區域開發的特質。

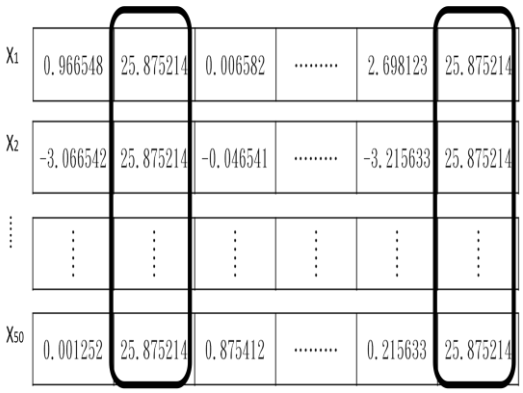


圖 2 個體落入區域最佳解時示意圖

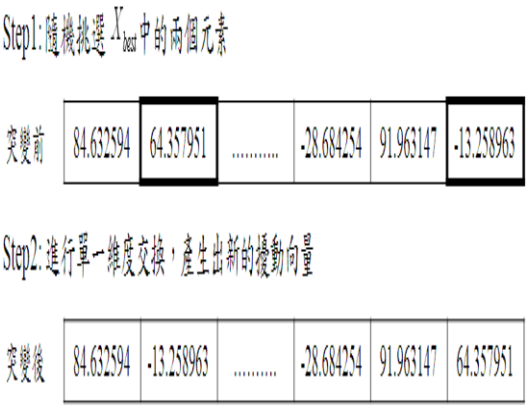


圖 3 擾動策略示意圖

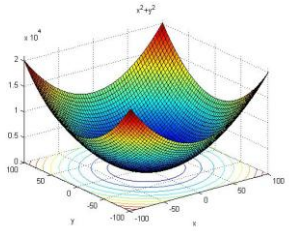
4. 實驗評估

4.1 測試函數

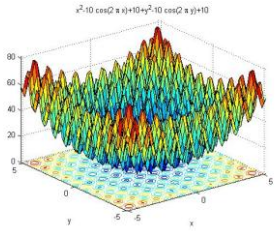
本節將列出一些測試函數用來比較本研究所提的改良差分演算法與傳統差分演算法與其他演算法的效能，主要可分為無區域最佳解的單峰函數以及有區域最佳解的多峰函數，而多峰函數又可分為少量的區域最佳解以及大量區域最佳解兩種類型，其評估的標準將利用求解答案的平均值與標準差來進行精確度與穩定性的效能評估。

表 2 測試函數

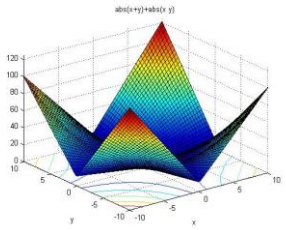
測試函數 (Benchmark functions)	函數類型	公式	最佳解
Sphere	單峰	$f_1(x)=\sum_{i=1}^{n}x_i^2$	0
Schwefel's problem 2.22	單峰	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	0
Step	單峰	$f_3(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	0
Rotated hyper-ellispoid	單峰	$f_5(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	0
Rastrigin	多峰	$f_7(x)=\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	0
Ackley's	多峰	$f_8(x)=-20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{30}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	0
Griewank	多峰	$f_9(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	0
Six-hump Camel-back	多峰	$f_{10}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	-1.0316285



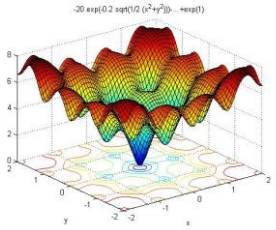
Sphere



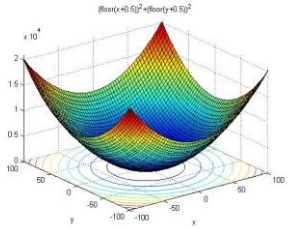
Rastrigin



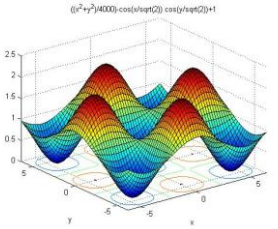
Schwefel's problem 2.22



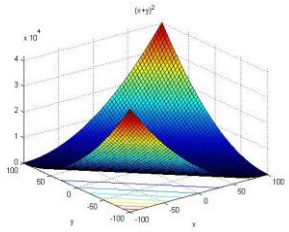
Ackley's



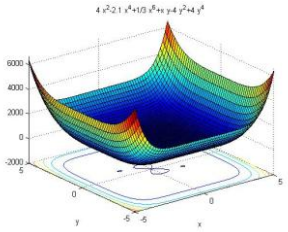
Step



Griewank



Rotated hyper-ellispoid



Six-hump Camel-back

圖 4 單峰函數 2 維圖型

圖 5 多峰函數 2 維圖型

4.2 參數設定

本研究會進行高維度實驗與傳統 DE 的策略 DE/rand/1 與 DE/best/2 比較之實驗，與傳統差分演算法進行比較時，本研究的 F 值採取變動的方式，利用平均數為 0.4，標準差 0.15 的常態分配方式產生，讓每個個體都會有自己 F 值，其餘參數設定以表 4 之設定方式。而每個測試函數的搜尋空間參考文獻[5]，同時獨立反覆執行測試 30 次並求取其平均值、標準差、最佳值與最差值，再將其測試結果進行分析比較，利用高維度環境下的實驗來以驗證本研究的改良策略是否確實具備其有效性與穩定性。

表 3 相關實驗參數設定

F(差異向量權重)	0.5
CR(重組率)	0.9
Generation(迭代數上限)	1000
NP(群體數量)	50
Dimension(維度)	30,60

4.3 實驗結果

以 Sphere(f_1)測試函數來說，是相對簡單的非線性單峰函數，大部分的演算法都具有不錯的求解成效，在 30 維與 60 維中，PsDE 與 DE/best/2 能比 DE/rand/1 更能具有不錯的效率，而從圖 8 與圖 9 可以發現 PsDE 與 DE/Best/2 具有快速收斂的特性。

Schwefel's problem 2.22 (f_2)是一個不可分離且較複雜的單峰測試函數，在 30 維中，PsDE 縮短了與 DE/best/2 的差距，從圖 9 得知而 PsDE 在 60 維中比 DE/best/2 更有優勢存在。

以 Step(f_3) 測試函數來說，是一個不連續型的單峰函數，雖然跟 Sphere 函數圖型相似度很高，但其表面是由很多平順的陡坡構成，在 30 維時，都可達到不錯的效能，不過當維度提高時， PsDE 更能夠具有較佳的穩定性與精確性。

Rotated hyper-ellispoid (f_4)測試函數跟 Schwefel's problem 2.22 (f_2)一樣，是屬於不可分離類型的單峰函數，不過從圖 8、圖 9 發現 PsDE 與 DE/rand/1、DE/best/2 都有快速收斂的效果，不管在 30 維與 60 維，其精確性上不會相差太多，都有良好的成效。

以 Rastrigin (f_5)測試函數來說，是一個利用 cos 函數來產生大量的區域最佳解的複雜多峰測試經典函數，因此非常容易陷入區域最佳解，由表 4 與表 6 皆能看出 PsDE 不管在 30 維與 60 維的 Rastrigin (f_5) 測試函上皆比 DE/rand/1 有更好的求解最佳解能力，而 DE/best/2 因快速收斂的特性，而導致在演化後期落入區域最佳解，不再繼續演化，如圖 8、圖 9 所示。

Ackley's (f_6)為連續、不可分離的多峰測試函數，由 3 維圖形可以知道該函數的上半部比較平坦，中間的區域有一個全域最佳解的谷峰，主要是因為 cos 函數來造成大量的區域最佳解，PsDE 同樣得在此多峰函數上，都能夠比其他兩種傳統 DE 突變方式有更好的精確性與穩定性，在圖 8 可以發現 DE/best/2 策略同樣會有陷入區域最佳解的情形發生。

以 Griewank (f_7)測試函數來說，與 Rastrigin 函數相當類似，不過值得注意的是，當 Griewank 測試函數維度越來越高時，區域最佳解的範圍也隨之變窄，因而就會相對容易達到全域最佳解，可以發現 PsDE 不管在 30 維與 60 維還是能比 DE/rand/1 與 DE/best/2 找到更佳的解。

Six-hump Camel-back (f_8)是一個只有 2 維的測試函數，是屬於少量區域最佳解的多峰函數，要注意的是最小值是 -1.0316285，而不是 0，可以探討出當最佳解不是 0 時，是否還具有不錯的求解能力，由表 4 與表 5 可以得知 PsDE 與不管哪種突變策略皆能夠達到理論的最佳值。

5. 結論

本研究發現差分演算法在演化的過程中，當落入區域最佳解時，某些維度並不會繼續演化，因此，透過擾動的機制，使得個體會有跳脫出區域最佳解的可能性，並增加了群體的多樣性，實驗採用常見的測試函數並設定在 30 維及 60 維高維度的情況下來進行評估，其中 DE/rand/1 是被廣泛應用在各種應用問題中，

表 4 執行 30 次的平均值與標準差實驗結果(30 維)

	PsDE		Rand1		Best2	
	Mean	Std	Mean	Std	Mean	Std
$f_1(x)$	7.3322e-020	8.7141e-020	1.8423e-011	1.1765e-011	**5.4096e-021**	**1.0245e-020**
$f_2(x)$	**7.5575e-012**	**4.8765e-012**	9.7385e-007	4.3412e-007	9.3775e-012	1.5366e-011
$f_3(x)$	**0**	**0**	**0**	**0**	**0**	**0**
$f_4(x)$	**1.0928e-021**	3.7311e-021	2.7175e-019	9.5833e-019	1.2394e-021	**1.7878e-021**
$f_5(x)$	**4.5821e+001**	**1.0919e+001**	1.5174e+002	2.3728e+001	2.09797e+002	2.01917e+001
$f_6(x)$	**6.2077e-011**	**4.0294e-011**	1.4273e-006	8.1667e-007	1.0793e+000	8.2395e-001
$f_7(x)$	**0**	**0**	1.5623e-003	4.2637e-003	1.2773e-002	1.3928e-002
$f_8(x)$	**-1.031628e+0**	**4.5168e-016**	**-1.031628e+0**	**4.5168e-016**	**-1.031628e+0**	**4.5168e-016**

表 5 執行 30 次的最佳解與最差解實驗結果(30 維)

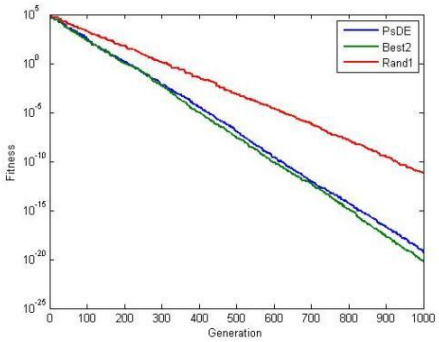
	PsDE		Rand1		Best2	
	Best	Worst	Best	Worst	Best	Worst
$f_1(x)$	3.4798e-021	3.1880e-019	2.0574e-012	4.7446e-011	**6.4768e-023**	**4.4812e-020**
$f_2(x)$	1.1709e-012	**2.2295e-011**	4.1836e-007	2.3245e-006	**9.9985e-013**	8.3632e-011
$f_3(x)$	**0**	**0**	**0**	**0**	**0**	**0**
$f_4(x)$	**0**	2.0299e-020	3.3036-024	5.1178e-018	**0**	**5.4265e-021**
$f_5(x)$	**2.4698e+001**	**7.2672e+001**	8.6174e+001	1.8902e+002	1.7509e+002	2.5490e+002
$f_6(x)$	**1.5746e-011**	**1.8440e-010**	5.1283e-007	3.5806e-006	5.7065e-011	2.4958e+000
$f_7(x)$	**0**	**0**	4.5380e-012	1.7276e-002	**0**	5.3760e-002
$f_8(x)$	**-1.031628e+0**	**-1.031628e+0**	**-1.031628e+0**	**-1.031628e+0**	**-1.031628e+0**	**-1.031628e+0**

表 6 執行 30 次的平均值與標準差實驗結果(60 維)

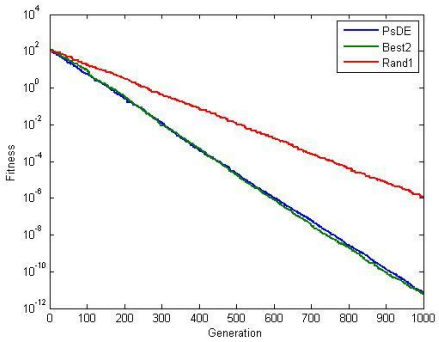
	PsDE		Rand1		Best2	
	Mean	Std	Mean	Std	Mean	Std
$f_1(x)$	3.0180e-006	1.6382e-005	4.3950e-003	7.2754e-003	**6.9721e-006**	**1.0251e-005**
$f_2(x)$	**4.3906e-006**	**1.8141e-006**	1.3032e-002	2.7741e-002	5.4335e-004	8.6463e-004
$f_3(x)$	**0**	**0**	8.8333e+000	1.5596e+001	8.6800e+001	8.6652e+001
$f_4(x)$	**2.2764e-005**	3.0325e-005	1.1033e-004	2.2265e-004	4.5101e-005	**7.1933e-005**
$f_5(x)$	**2.0933e+002**	**3.1598e+001**	4.4733e+002	3.6791e+001	5.5937e+002	4.4832e+001
$f_6(x)$	**2.3579e-005**	**1.2550e-005**	1.8359e-001	3.7564e-001	3.8813e+000	9.4639e-001
$f_7(x)$	**6.3137e-003**	**9.8065e-003**	9.9574e-003	1.2338e-002	1.6704e-002	2.5414e-002

表 7 執行 30 次的最佳解與最差實驗結果(60 維)

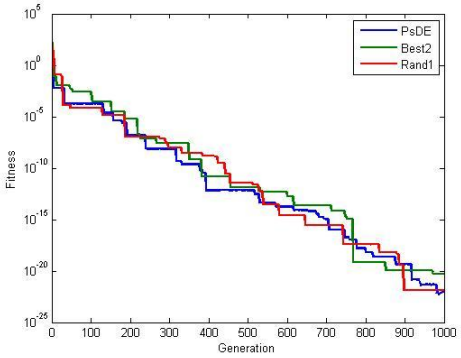
	PsDE		Rand1		Best2	
	Best	Worst	Best	Worst	Best	Worst
$f_1(x)$	**4.5873e-009**	8.9759e-005	7.4636e-004	3.9630e-002	9.2047e-008	**4.1198e-005**
$f_2(x)$	**2.0509e-006**	**9.0651e-006**	2.5479e-003	1.4907e-001	4.0179e-005	3.2338e-003
$f_3(x)$	**0**	**0**	**0**	83	15	495
$f_4(x)$	1.8322e-008	**1.3592e-004**	1.2635e-006	1.1744e-003	**2.0023e-009**	2.7257e-004
$f_5(x)$	**1.4248e+002**	**2.7030e+002**	3.2721e+002	5.0264e+002	4.0559e+002	6.2636e+002
$f_6(x)$	**7.8917e-006**	**6.1175e-005**	5.1183e-003	1.1787e+000	2.1702e+000	5.5910e+000
$f_7(x)$	**1.4662e-008**	**3.4646e-002**	6.0293e-004	4.5013e-002	1.3276e-007	9.6943e-002



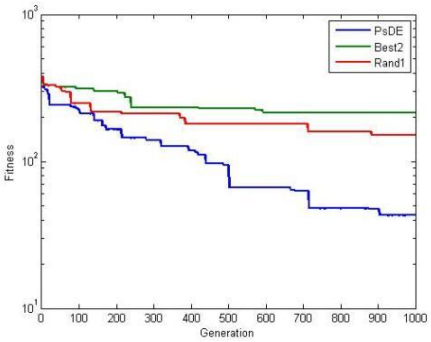
Sphere



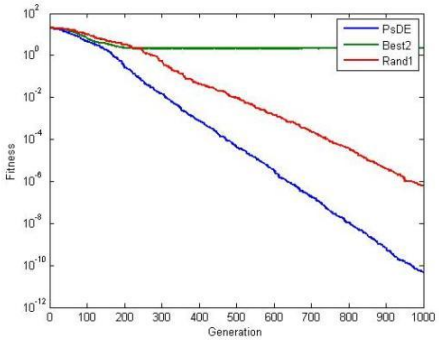
Schwefel's problem 2.22



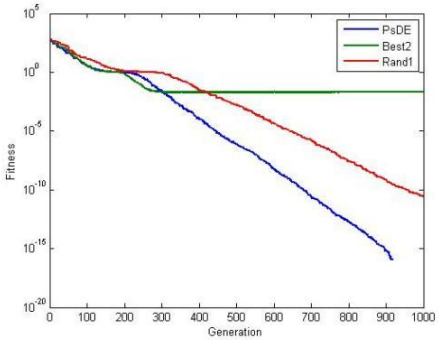
Rotated hyper-ellispoid



Rastrigin

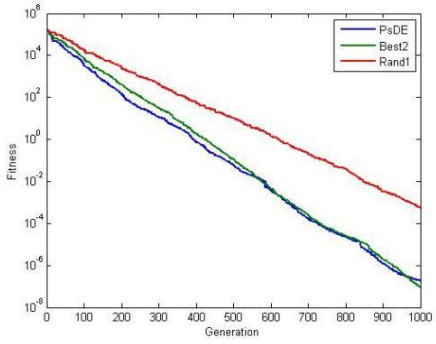


Ackley's

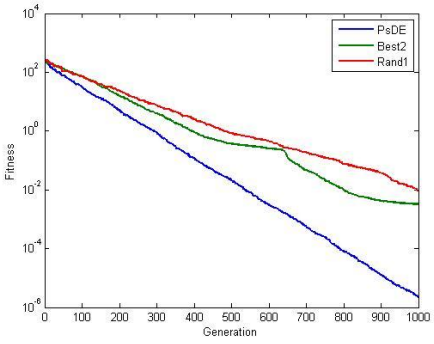


Griewank

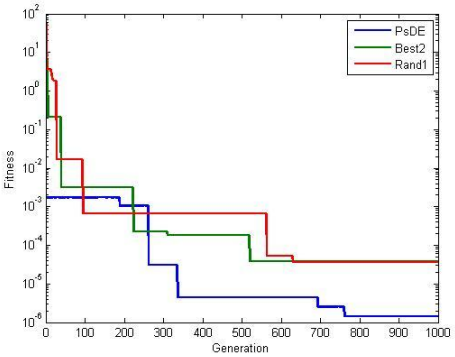
圖 6 PsDE 與傳統差分演算法收斂圖(30 維)



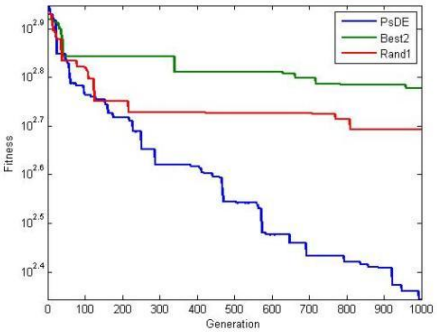
Sphere



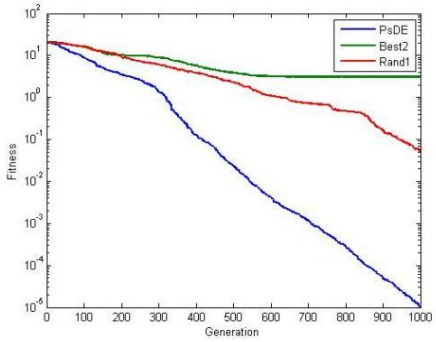
Schwefel's problem 2.22



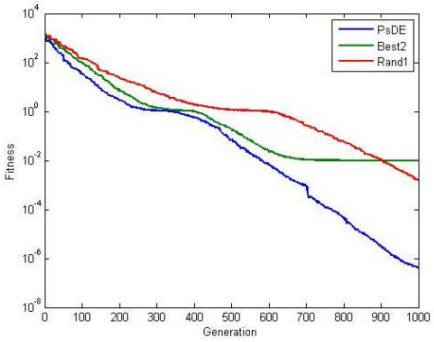
Rotated hyper-ellispoid



Rastrigin



Ackley's



Griewank

圖 7 PsDE 與傳統差分演算法收斂圖(60 維)

DE/best/2 也是常常採取或是做為改良標的策略之一，因此採取這兩種突變策略做為比較對象。

由實驗結果可以看到在八個測試函數中，PsDE 演算法在求解效能上皆能接近或是超越傳統差分演算法。尤其在多峰函數上如 Rastrigrin 與 Ackley's 60 維的平均值及最佳值，其求解成效都有所提升，因此更能證明 PsDE 演算法在求解上的有效性及穩定性。因此，本研究建議，未來的研究可把 PsDE 作為基礎，以探討相關的應用領域問題，或是持續改良以加強演算法的探索與開發之能力。

參考文獻

[1] 三郎, 杉江, "*仿生學潛說*". [北京]: 科學出版社, 1982.

[2] 岑海堂、陳五一, *"仿生學及其演變"*. 機械設計, 2007. **24**(7).

[3] 高尚、楊靜宇, *“群智能算法及其應用”, 中國水利水電出版社*, 2006.

[4] 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*, 2005. **3**(1).

[5] Salman, A., A.P. Engelbrecht, and M.G.H. Omran, *Empirical analysis of self-adaptive differential evolution.* European Journal of Operational Research, 2007. **183**(2): p. 785-804.

[6] Sauer, J.G. and L. Coelho. *Discrete Differential Evolution with local search to solve the Traveling Salesman Problem: Fundamentals and case studies*. in *Cybernetic Intelligent Systems, 2008. CIS 2008. 7th IEEE International Conference on*. 2008.

[7] Wang, C., M. Zeng, and J. Li. *Solving traveling salesman problems with time windows by genetic particle swarm optimization*. in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*. 2008.

[8] Chen, P., J. Li, and Z. Liu. *Solving 0-1 Knapsack Problems by a Discrete Binary Version of Differential Evolution*. in *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*. 2008.

[9] Yuxiang, S., X. Hongwen, and Y. Weiming. *Solve Zero-One Knapsack Problem by Greedy Genetic Algorithm*. in *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*. 2009.

[10] Das, S., A. Konar, and U.K. Chakraborty. *Automatic Fuzzy Segmentation of Images with Differential Evolution*. in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. 2006.

[11] Omran, M.G.H. and A.P. Engelbrecht. *Self-Adaptive Differential Evolution Methods for Unsupervised Image Classification*. in *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*. 2006.

[12] Ji-Pyng, C., C. Chung-Fu, and S. Ching-Tzong, *Variable scaling hybrid differential evolution for solving network reconfiguration of distribution systems.* Power Systems, IEEE Transactions on, 2005. **20**(2): p. 668-674.

[13] Su, C.T. and C.S. Lee, *Network Reconfiguration of Distribution Systems Using Improved Mixed-Integer Hybrid Differential Evolution.* Power Engineering Review, IEEE, 2002. **22**(12): p. 66-66.

[14] Feng, X., A.C. Sanderson, and R.J. Graves, *Multiobjective Evolutionary Decision Support for Design Supplier Manufacturing Planning.* Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 2009. **39**(2): p. 309-320.

[15] Cai, H.R., C.Y. Chung, and K.P. Wong, *Application of Differential Evolution Algorithm for Transient Stability Constrained Optimal Power Flow.* Power Systems, IEEE Transactions on, 2008. **23**(2): p. 719-728.

[16] Guang Ya, Y., D. Zhao Yang, and W. Kit Po, *A Modified Differential Evolution Algorithm With Fitness Sharing for Power System Planning.* Power Systems, IEEE Transactions on, 2008. **23**(2): p. 514-522.

[17] Kicinger, R., T. Arciszewski, and K.D. Jong, *Evolutionary computation and*

structural design: A survey of the state-of-the-art. Computers & Structures, 2005. **83**(23-24): p. 1943-1978.

[18] Storn, R. and K. Price. *Minimizing the real functions of the ICEC'96 contest by differential evolution*. in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. 1996.

[19] Storn, R.P., K., *Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces.* Technical report, California: International Computer Science Institute, Berkeley., 1995.

[20] Zhi-Feng, H., G. Guang-Han, and H. Han. *A Particle Swarm Optimization Algorithm with Differential Evolution*. in *Machine Learning and Cybernetics, 2007 International Conference on*. 2007.

[21] Abbass, H.A. *The self-adaptive Pareto differential evolution algorithm*. in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*. 2002.

[22] Das, S., A. Konar, and U.K. Chakraborty. *Improved differential evolution algorithms for handling noisy optimization problems*. in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. 2005.

[23] Changshou, D., et al. *New Differential Evolution Algorithm with a Second Enhanced Mutation Operator*. in *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*. 2009.

[24] Luitel, B. and G.K. Venayagamoorthy. *Differential evolution particle swarm optimization for digital filter design*. in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*. 2008.

[25] JH., H., *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press;, 1975.

[26] H-P., S., *Kybernetische Evolution als Strategie der experimentelen Forschung in der Stromungstechnik.* Masters thesis, Hermann Fo¨ ttinger Institute for Hydrodynamics,Technical University of Berlin, 1965.

[27] Fogel LJ, O.A., Walsh MJ., *Artificial Intelligence through simulated evolution.* Chichester, UK: John Wiley, 1966.

[28] Brest, J., et al., *Performance comparison of self-adaptive and adaptive differential evolution algorithms.* Soft Computing - A Fusion of Foundations, Methodologies and Applications, 2007. **11**(7): p. 617-629.

[29] Omran, M.G.H., A.P. Engelbrecht, and A. Salman, *Bare bones differential evolution.* European Journal of Operational Research, 2009. **196**(1): p. 128-139.

[30] Unhong, L. and L. Jouni. *A fuzzy adaptive differential evolution algorithm*. in *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*. 2002.

[31] Gong, W., Z. Cai, and L. Jiang, *Enhancing the performance of differential evolution using orthogonal design method.* Applied Mathematics and Computation, 2008. **206**(1): p. 56-69.

[32] Krink, T., B. Filipic, and G.B. Fogel. *Noisy optimization problems - a particular challenge for differential evolution?* in

Evolutionary Computation, 2004. CEC2004. Congress on. 2004.

[33] Qin, A.K. and P.N. Suganthan. *Self-adaptive differential evolution algorithm for numerical optimization*. in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. 2005.