雙重底混沌差分演算法應用於資料分群

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

高翊凱
電子工程系
國立高雄應用科技大學
kaikai8053@hotmail.com

莊麗月*
化學工程系
義守大學
chuang@isu.edu.tw

摘要

資料探勘一直常被應用於各種領域，資料分群則是資料探勘中常被使用的一種方法，資料分群主要是在一群資料中，依據資料的相似度高則分類成為同一群，再經由分群結果尋找群集與群集之間的差異是否存在意義。本研究提出雙重底混沌差分演算法應用於資料分群問題，將混沌理論應用於差分演算法可以提升向量的最佳化搜尋方向以及有效避免區域最佳解之特性。本研究使用 UCI (University of California, Irvine) DATA 驗證雙重底混沌差分演算法的實用性，並且驗證六種移動策略中使用混論理論前後差異，驗證方式使用比較群集內距離總和以及錯誤率，實驗結果顯示雙重底混沌差分演算法可以有效的優於其他演算法，並且有效的提升差分演算法的搜尋能力。

1 前言

資料分群是資料探勘中一種重要的分析方法 [1]，資料分群已經被大量應用於許多領域，像是生物科技、股市分析、市場管理以及生產控制等等 [2, 3]，其技術主要方法是依據所有資料中的欄位屬性，將資料分成多個群集，相同的群集中資料的相似度會較高，而不同群集中的資料則差異度會較高 [4]。

資料分群技術中比較常見的有階層式分群法、切割式分群法以及密度式分群法，階層式分群法中又分為凝聚法以及分裂法，凝聚法為將所有資料皆當作一個群集，依據群集間的距離選擇較近的凝聚成一個群集，直到凝聚為一個大群集，之後可將凝聚的順序畫成一個樹狀圖，呈現為階層式分群法的答案；分裂法則反之，將所有資料當作一個群集，依據尋找群集間資料距離最遠的資料則分裂為另一群集，直到分裂至所有資料皆為一個群集後，則完成分群結果。

切割式演算法需先指定分群的群集數量後再進行切割，切割後子群集會再尋找新的群集中心後再次重新切割，此方法代表為 K-means 演算法。Kmeans 演算法 [5]，在分群中是一種常被應用的技術，主要原因是容易實現，並且有著高效率的優點，但其缺點是容易落入解空間中的局部最佳解，也容易受到初始化選擇的中心影響最終結果，因此難以找到全域最佳解。

密度式分群法是為了避免大多數分群法為使用歐幾里得距離為基礎，而密度式可以避免掉入特殊形狀分布的資料集合，密度式分群法概念為，選定某群集持續成長直到群集間的門檻值內找不到資料為止。代表的方法有 DBSCAN[6]、OPTICS[7]等。

近年來已有許多演化式演算法應用於資料分群的應用上，其中包含粒子族群最佳化演算法 (Particle Swarm Optimization, PSO) [8] 及差分演算法 (Differential Evolution, DE) [9]。粒子族群最佳化演算法主要概念來自於鳥群的覓食方式，每隻鳥會依據自己的覓食經驗，以及鳥群中的最好的覓食經驗，藉由移動時的速度以及個體所知的最佳經驗以及群體所知的最佳經驗，得到下次鳥群的位置。

差分演算法則是應用向量概念，藉由三種主要的模組，突變、交配和選擇三個步驟，來得到最佳解的位置，差分演算法本身有些應用到基因演算法交配的概念，並且較粒子族群最佳化演算法相比下，有著參數設定較少的優勢 [10]，近年來不斷有學者應用差分演算法於最佳化問題中，並且有不錯的改善。

目前的最佳化演算法還有著容易陷入區域最佳解、不容易跳出解空間等問題，因此我們應用了雙重底混沌演算法結合差分演算法來有效的提升向量搜尋最佳解的能力，強化了跳脫區域最佳解的可能，並且有效地尋找最佳解。本研究應用了 6 個 UCI 大學所提供的真實資料集進行測試演算法，並且比較分群結果群集內集合總距離以及錯誤率，來驗證演算法的好壞。

2 方法

差分演算法 (Differential Evolution, DE) 於 1995 年由 Storn 及 Price 兩位學者所提出，主要概念是由向量為基礎，藉由向量與向量間的組合進而在解空間中移動得到新的向量，差分演算法主要使用突變、交配和選擇三個步驟來進行多次迭代後，找出最佳解。差分演算法開始會先產生 N 個向量，然後會對每一個向量進行適應值計算，之後先對第一個向量做突變，F 是加權因子通常設定為 0 至 2 之間，經過突變策略之後會得到新到 V_i。之後會再進行交配，先定義交配率 CR 數值，之後將對每一個維度進行一次交配行為，交配完畢後進行選擇步驟，計算新產生的向量其適應值，若比原本 X_i 的適

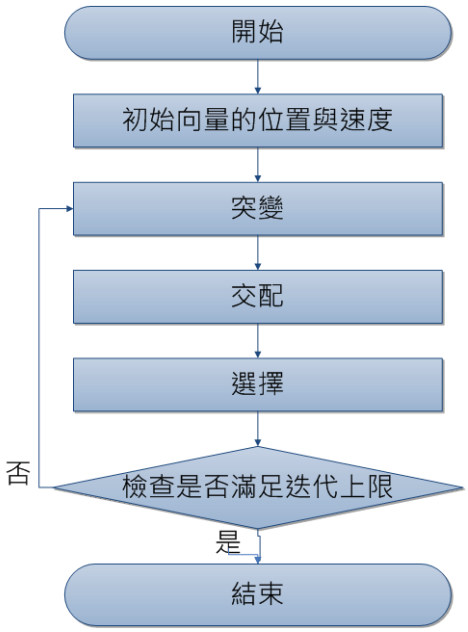


圖 1. 差分演算法流程圖.

應值好則替換，若沒有則保留 X_i，差分演算法的流程圖如圖 1 所示。

雙重底混沌差分演算法

- 初始化

產生 N 個向量，每個向量空間會被定義在解空間之中隨機亂數，如公式 (1) 所示。

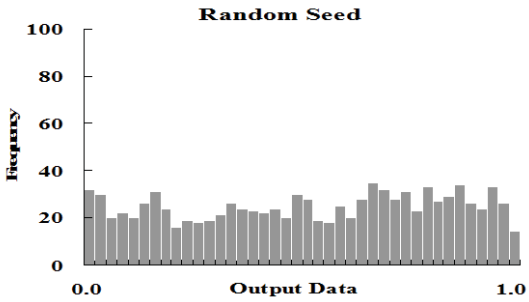
$$N = K \times D \tag{1}$$

- 突變權重因子 F

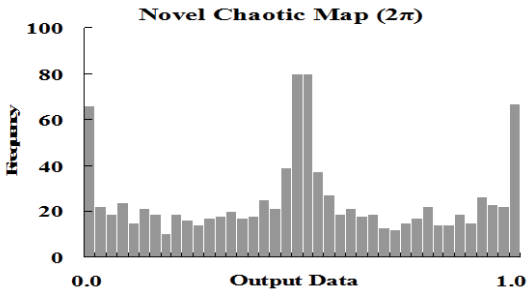
突變權重因子 F 會直接的影響到突變策略中，向量所移動的大小，所以一個差分演算法的好壞突變權重因子佔了很大的比重，F 通常設定比較大時，可以更容易的搜尋到全域最佳解，而當 F 設定較小時，可以較有效的收斂至最佳解，所以突變權重因子的設定可以很直接影響結果的好壞，通常突變權重因子會被設定介於 0 至 2 之間做選擇 [11, 12]，而也有差分演算法的作者認為突變權重因子介於 0.5 至 1 之間是較好的選擇。

- 雙重底混沌演算法

本研究將雙重底混沌演算法應用於突變權重因子之中，雙重底混沌演算法於 2012 年由楊等人所提出 [13]，雙重底混沌演算法具有產生的亂數較容易落於 0、0.5、1 等數值的附近，藉由此特性來提升差分演算法搜尋最佳解的能力，圖 2 為使用一般亂數以及雙重底混沌 map 所產生的亂數，並紀錄其一千次的統計，統計 0 到 1 之間所出現的次數與區間，雙重底混沌演算法公式如公式 (2) 所示。



(a) 一般亂數產生的頻率



(b) 雙重底混沌 map 產生的頻率

圖 2. 1000 次隨機亂數之頻率統計圖

$$DBMr_{t+1} = [\sin(2n DBMr_t) + 1] \div 2, \quad n \in \mathbf{N} \tag{2}$$

- 突變

從所有向量中挑選出不重複的向量後，並且透過突變權重因子 F 得到合成向量 V_G，突變中公式內的 X_r 為隨機挑選出的向量，X_{best} 為目前所得最好的解，X_i 為向量本身，G 為目前的世代數，本研究使用了常被應用的六種突變策略 [14]。

DE/rand/1

為 DE 中最基礎的突變策略，會選擇三個不重複的向量合成出新的向量，如公式 (3) 所示。

$$\mathbf{v}_{i,G+1} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}) \tag{3}$$

DE/rand/2

由 DE/rand/1 改良而來，策略會選擇五個不重複的向量合成出新的向量，由於增加了兩個選擇的向量，希望能更廣泛的搜尋可能，如公式 (4) 所示。

$$\mathbf{v}_{i,G+1} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}) + F \times (X_{r4,G} - X_{r5,G}) \tag{4}$$

DE/best/1

與 DE/rand/1 差異並不大，將 DE/rand/1 中選擇的 X_{r1} 改變成 X_{best}，X_{best} 為目前迭代中群體中最好的個體向量，如公式 (5) 所示。

$$\mathbf{V}_{i,G+1} = X_{best,G} + F \times (X_{r1,G} - X_{r2,G}) \tag{5}$$

DE/best/2

將 DE/best/1 改良而來，策略會選擇四個不重複的向量，並且搭配群體最佳解 X_{best} 中合成得到新向量，如公式 (6) 所示。

$$V_{i,G+1} = X_{best,G} + F \times (X_{r1,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G}) \quad (6)$$

DE/rand-to-best/1

基於 *rand* 與 best 的概念改良而來，藉由群體最佳解 X_{best}、向量個體本身及不重複的兩個向量，組合而成一種突變策略，如公式 (7) 所示。

$$V_{i,G+1} = X_{i,G} + F \times (X_{best,G} - X_{i,G}) + F \times (X_{r1,G} - X_{r2,G}) \quad (7)$$

DE/current-to-best/1

藉由向量個體本身，以及三個不重複的向量組合而成的突變策略，如公式(8)所示。

$$V_{i,G+1} = X_{i,G} + F \times (X_{r1,G} - X_{i,G}) + F \times (X_{r2,G} - X_{r3,G}) \quad (8)$$

- 交換

當產生出 V_G 向量後，經由 *CR* 交配率來選擇當下維度所選擇的向量屬性，如公式所示。交配率 *CR* 的設定也是相當重要，*CR* 值會設定在 0 至 1 之間，當 *CR* 值設定越小時，可能會造成產生出來的 u_G 向量與原本的 X_G 向量越相似，當設定的 *CR* 值過小時，可能會造成向量改變的幅度過小，則需要長時間的迭代才能收斂至較佳解；而當 *CR* 值設定越高時，可以令產生出來的 u_G 向量可以較為擁有經由突變產生出來的 V_G 向量屬性，在本文中 *CR* 值設定為 0.9。

$$u_{j,i,G+1} = \begin{cases} V_{j,i,G} & ,if\ rand \le CR \\ X_{j,i,G} & ,if\ rand > CR \end{cases} \quad (9)$$

- 選擇

之後評估 $u_{i,j}$ 向量計算的適應函數，$u_{i,j}$ 向量如果數值較好，則將 $u_{i,j}$ 取代掉 X_i，如果沒有則維持原樣，如公式 (10) 所示。

$$X_{i,j,G+1} = \begin{cases} u_{i,j,G} & ,if\ F(u_{i,j,G}) \ge F(X_{i,j,G}) \\ X_{i,j,G} & ,otherwise \end{cases} \quad (10)$$

- 向量編碼

每一個向量都為一種解的可能，而每一個解都定義成群集內的中心位置，所以將向量內的維度大小則為群集的數量乘上分群的總數量，如下所示。例如在三維空間分成兩個群集，群集之間會產生一個中心點，中心點位置分別為 (2, 5, 8)、(3, 7, 4)，則向量的編碼則為 (2, 5, 8, 3, 7, 4)。

- 適應函數

本研究適應函數為計算各群集中，中心點到群集內的資料點的距離總和作為適應值，適應值越小則代表取得的中心點位置越佳。適應函數公式如 (11) 所示。*Fit* 為適應值，$\underline{S_{pi}}$ 為中心點，X_{ij} 為屬於 S_{pi} 中心點群集內的資料。

$$Fit(S_p \bullet X_j) = \sqrt{\sum_{i=1}^{d} (S_{pi} - X_{ij})^2} \quad (11)$$

- 資料集

本研究使用 UCI 測試資料，資料來源於美國加州 Irvine 大學資訊與電腦科學系 (ftp://ftp.ics.uci.edu/pub/machine-learning-data-bases/)，分別為 Iris Plants、Contraceptive Method Choice (CMC)、Breast Cancer、Wine、Indian Telugu vowel sounds 與 Crude Oil，資料集詳細內容如下所示：

表 1 資料庫相關特性

資料集	群集數	特徵數	資料數量
Iris	3	4	150(50,50,50)
CMC	3	9	1473(629,333,511)
Cancer	2	9	683(444,239)
Wine	3	13	178(59,71,48)
Vowel	6	3	871(72,89,172,151,207,180)
Crude Oil	3	5	56(7,11,38)

鳶尾花 (Iris Plants): 內有三種鳶尾花 Iris Setosa、Versicolour、Virginica，每種鳶尾花各有 50 筆資料共 150 筆資料，每筆鳶尾花資料含有四種屬性萼片長度、萼片寬度、花瓣長度與花瓣寬度。

避孕方法的選擇 (Contraceptive Method Choice):CMC 有三種主要分群方式分別為不使用 (No-use) 共 629 筆、長期使用 (Long-term) 共 333 筆、短期使用 (Short-term) 共 511 筆，而每筆資料內含有 9 種屬性皆為妻子與丈夫的相關資料。

乳癌資料 (Breast Cancer):乳癌資料主要分為兩群，良性細胞 (Benign) 共 444 筆與惡性細胞 (Malignant) 共 239 筆，乳癌資料庫中有 699 筆資料，但由於有 16 筆資料中屬性有遺失，所以只採用其中 683 筆資料進行測試，其中內含九種屬性。

葡萄酒 (Wine):葡萄酒資料庫來源於義大利同一區所產的三種類型的葡萄酒，分別為 class1 共 59 筆、class2 共 71 筆與 class3 共 48 筆，葡萄酒資料庫中含有 13 種屬性包含葡萄酒的酒

精濃度、蘋果酸、鎂成分含量等等。

印度泰盧固語聲音 (Vowel):聲音資料庫含有六種母音分群為δ 共 72 筆、a 共 89 筆、i 共 172 筆、u 共 151 筆、e 共 207 筆、o 共 180 筆，其中含有三種音頻屬性。

石油 (Crude Oil):石油來自於三種不同砂岩地區所採集而來，分別為 7 筆 Wilhelm、11 筆 Sub-Mulnia、38 筆 Upper，共有 56 筆資料，石油內含 5 種屬性分別為釩 (Vanadium) 、鐵 (Iron) 、鈹 (Beryllium) 、飽和碳氫化合物 (Saturated hydrocarbons) 芳香族碳氫化合物 (Aromatic hydrocarbons)。

3 結果與討論

本研究將雙重底混沌差分演算法與其他五種演算法進行比較 (K-means、PSO、NM-PSO、K-PSO、K-NM-PSO) 優缺點，為了與其他演算法[15]比較優劣，實驗相關的參數皆設定相同，演算法的迭代次數設定為 10N，N 的算法如公式 (1) 所示，族群數量則皆設定為 3N，藉由比較資料集中群集內距離總和及錯誤率，評估方式細節如下所表示：

群集內距離總和

群集內距離總和的如公式 (11) 所示，群集內距離總和數值越小，代表分群的結果越好。

錯誤率

錯誤率的公式如 (12) 所示，其中 n 代表資料的數量，*Sum_error* 表示將分群正確的資料加總所得結果，最後除上資料總數則為錯誤率。

$$ER = Sum\_error \div n \times 100 \qquad (12)$$

表 2 及表 3 為使用了六種演算法 (K-means、PSO、NM-PSO、K-PSO、K-NM-PSO、DBMDE) 的實驗結果，本研究為了驗證演算法的好壞，計算了 20 次的實驗結果，之後取得其平均值、標準差 (Standard deviations) 以及最佳值；平均值是為了驗證演算法的好壞，而最佳值是為了驗證演算法是否能收斂到最佳解，計算得到最小的群集距離總和，而標準差是為了驗證 20 次的計算中，群集內距離的離散程度，如果標準差值越大，代表演算法的穩定度越差，如果其標準差值越小，代表演算法的穩定度越高。

從表 2 中可以得知，DBMDE 在六個資料集中都有較好的群集內總距離，雖然實驗結果在 Vowel 資料集時，K-NM-PSO 較 DBMDE 優良，K-NM-PSO 數值為 149141.4 而 DBMDE 為 149565.91，但是 DBMDE 在其他資料集中，都較 K-NM-PSO 演算法佳，並且在 CMC 以及 Cancer 資料集中的標準差都為 0，代表可以很穩定的尋找到最佳解。其中 NM-PSO 與 K-PSO 為結合 Nelder-Mead 單體法 (Nelder-Mead simplex searchmethod) [14]以及 K-means 分群演算法，可以有效的幫助 PSO 提升搜尋能力，NM 演算法可以有效地提升區域最佳解搜尋能力，而 K-means 可以幫助 PSO 一開始就在較為優良的位置幫助 PSO 進行收斂，在 K-NM-PSO 可以看到結合 NM 可有效的對區域最佳解有效地進行

表 2 各種演算法的群集內距離總和比較表

資料集	評估	K-means [15]	PSO [15]	NM-PSO [15]	K-PSO [15]	K-NM-PSO [15]	DBMDE with DE/current-to-best/1
Iris	平均值	106.05	103.51	100.72	96.76	96.67	**96.66**
	(標準差)	(14.11)	(9.69)	(5.82)	(0.07)	(0.008)	(0.02)
	最佳值	97.33	96.66	96.66	96.66	96.66	96.66
CMC	平均值	5693.6	5734.2	5563.4	5532.9	5532.7	**5532.18**
	(標準差)	(473.14)	(289.00)	(30.27)	(0.09)	(0.23)	(0.00)
	最佳值	5542.2	5538.5	5537.3	5532.88	5532.4	5532.18
Cancer	平均值	2988.3	3334.6	2977.7	2965.8	2964.7	**2964.39**
	(標準差)	(0.46)	(357.66)	(13.73)	(1.63)	(0.15)	(0.00)
	最佳值	2987	2976.3	2965.59	2964.5	2964.5	2964.39
Wine	平均值	18061	16311	16303	16294	16293	**16292.77**
	(標準差)	(793.21)	(22.98)	(4.28)	(1.70)	(0.46)	(0.62)
	最佳值	16555.68	16294	16292	16292	16292	16292.18
Vowel	平均值	159242.87	168477	151983.91	149375.7	**149141.4**	149565.91
	(標準差)	(916)	(3715.73)	(4386.43)	(155.56)	(120.38)	(944.52)
	最佳值	149422.26	163882	149240.02	149206.1	149005	148967.24
Crude Oil	平均值	287.36	285.51	277.59	277.77	277.29	**277.25**
	(標準差)	(25.41)	(10.31)	(0.37)	(0.33)	(0.095)	(0.04)
	最佳值	279.2	279.07	277.19	277.45	277.15	277.21

註:粗體為各種演算法執行相同資料集中，群集內距離的平均值與最佳值之方法

表 3 各種演算法的錯誤率比較表

資料集	評估	K-means	PSO	NM-PSO	K-PSO	K-NM-PSO	DBMDE with DE/current-to-best/1
Iris	平均值	17.8	12.53	11.13	10.2	10.07	**10.03**
	(標準差)	(10.72)	(5.38)	(3.02)	(0.32)	(0.21)	(0.15)
	最佳值	10.67	10	8	10	10	10
CMC	平均值	54.49	54.41	54.47	54.38	**54.31**	54.38
	(標準差)	(0.04)	(0.13)	(0.06)	(0.00)	(0.054)	(0.00)
	最佳值	54.45	54.24	54.38	54.38	54.31	54.38
Cancer	平均值	4.08	5.11	4.28	3.66	3.66	**3.51**
	(標準差)	(0.46)	(1.32)	(1.10)	(0.00)	(0.00)	(0.00)
	最佳值	3.95	3.66	3.66	3.66	3.66	3.51
Wine	平均值	31.12	28.71	28.48	28.48	**28.37**	28.6
	(標準差)	(0.71)	(0.27)	(0.27)	(0.40)	(0.27)	(0.35)
	最佳值	29.78	28.09	28.09	28.09	28.09	28.09
Vowel	平均值	44.26	44.65	41.96	42.24	**41.94**	42.23
	(標準差)	(2.15)	(2.55)	(0.98)	(0.95)	(0.95)	(1.96)
	最佳值	42.02	41.45	40.07	40.64	40.64	37.31
Crude Oil	平均值	24.46	24.64	24.29	24.29	**23.93**	26.16
	(標準差)	(1.21)	(1.73)	(0.75)	(0.92)	(0.72)	(0.85)
	最佳值	23.21	23.21	23.21	23.21	23.21	25

註:粗體為各種演算法執行相同資料集中，群集內距離的平均值與最佳值之方法

表 4 差分演算法與雙重底差分演算法於六種突變策略比較表

資料集	突變策略	DE/rand/1		DE/rand/2		DE/best/1		DE/best/2		DE/rand-to-best/1		DE/current-to-best/1	
	F設定值	F=0.5	F=DBM	F=0.5	F=DBM	F=0.5	F=DBM	F=0.5	F=DBM	F=0.5	F=DBM	F=0.5	F=DBM
Iris	平均值	**96.96**	97.18	**114.33**	113.48	102.04	**101.5**	**97.86**	**97.86**	**97.58**	96.79	98.03	**96.66***
	(標準差)	(0.72)	(0.95)	(7.06)	(9.03)	(7.25)	(8.63)	(5.25)	(5.25)	(1.17)	(0.23)	(1.28)	(0.02)
	最佳值	96.66	96.67	100.08	100.98	96.69	96.66	96.66	96.66	96.66	96.66	96.68	96.66
CMC	平均值	**5534.38**	5539.57	5942	**5852.92**	5728.57	**5568.44**	**5532.19**	5532.21	5600.18	**5532.18***	5612.12	**5532.18***
	(標準差)	(5.13)	(5.21)	(58.97)	(122.52)	(86.74)	(36.03)	(0.03)	(0.02)	(18.27)	(0.00)	(27.29)	(0.00)
	最佳值	5532.32	5533.7	5783.28	5601.65	5610.47	5535.34	5532.18	5532.19	5567.96	5532.18	5545.74	5532.18
Cancer	平均值	**2964.7**	2974.29	**2980.57**	2984.32	3269.4	**3020.5**	2964.49	**2964.39**	3111.59	**2964.39***	3141.21	**2964.39***
	(標準差)	(0.79)	(40.75)	(25.72)	(7.02)	(130.28)	(95.41)	(0.42)	(0.00)	(122.69)	(0.00)	(101.28)	(0.00)
	最佳值	2964.4	2964.6	2970.53	2976.73	3018.3	2964.46	2964.39	2964.39	2976.01	2964.39	3004	2964.39
Wine	平均值	**16292.5***	16293.53	16398.08	**16366.38**	16329.16	**16301.2**	16292.82	**16292.55**	**16299.79**	16292.8	16296.85	**16292.77**
	(標準差)	(0.48)	(0.54)	(39.53)	(34.91)	(21.7)	(7.21)	(0.68)	(0.45)	(3.56)	(0.61)	(2.05)	(0.62)
	最佳值	16292.23	16292.91	16308.68	16305.94	16307.13	16293.32	16292.18	16292.2	16294.5	16292.18	16293.55	16292.18
Vowel	平均值	**165205.3**	169686.2	**196892.8**	199479.7	160471.3	**156169.3**	150056.1	**149231.5***	151594.1	**149570.6**	152604.7	**149565.9**
	(標準差)	(8951.46)	(6166.46)	(4377.3)	(4364.39)	(8334.69)	(5486.52)	(2620.79)	(395.28)	(3050.59)	(919.75)	(3753.41)	(944.52)
	最佳值	149375.9	158045.7	188848.7	190515.9	150271.2	149453.1	148967.3	148972.9	149024.4	148967.2	149228.9	148967.2
Crude Oil	平均值	**277.32**	277.55	293.93	**292.1**	280.91	**278.43**	**277.24**	**277.24***	278.08	**277.24***	278.54	**277.25**
	(標準差)	(0.17)	(0.4)	(8.47)	(8.97)	(2.13)	(3.85)	(0.04)	(0.04)	(0.72)	(0.07)	(1.06)	(0.04)
	最佳值	277.22	277.24	279.68	280.39	277.92	277.21	277.21	277.21	277.3	277.21	277.27	277.21

註:粗體為各策略中使用不同突變權重因子所得較好的群集內距離的平均值與最佳值之方法

收斂，但是 PSO 本身就擁有不錯的區域搜尋最佳解的能力，較需提升的是搜尋全域最佳解的能力，由此表可以看出 DMBDE 可以有效的尋找到最佳解。

表 3 則為演算法在六個資料集中的錯誤率評估，在錯誤率評估中可以看出，在六個資料集中，DBMDE 在錯誤率中雖然略輸於 K-NM-PSO，只有在 Iris 以及 Cancer 資料集中有較為優良的錯誤率，但是通常群集內距離總和和錯誤率之間並沒有絕對的關係，由於使用

的六個資料集，其資料屬性與其正確的群集並不一定會呈現規則性分布，因此可能會造成顯示群集內距離總和距離較佳，但是錯誤率的數值卻是較差的可能，因此看到雖然在群集內資料集總和中有五個資料集有較 K-NM-PSO 演算法佳，但是其五個資料集在錯誤率的實驗結果，只有兩個資料集較 K-NM-PSO 佳。

本研究主要是為了研究差分演算法應用於雙重底混沌演算法是否有較優良的改善，因此我們將同樣六個資料集，也有進行六種突變策略的改善如表 4 所示，表 4 將六個資料集分別對於差分演算法的六種策略進行突變因子的調控，一種為固定設為 0.5 [16]，另一種為使用雙重底混沌演算法，最後可以得到 36 個比較結果，其中將 *F* 固定為 0.5 的話得到較好的結果有 12 個，將 *F* 使用雙重底混沌演算法的話則有 22 個較好的結果，剩餘 2 個結果則為平手，可以看出使用雙重底混沌演算法可以較好的計算群集內距離總和。表 4 在使用 DE/rand/1 的突變策略時，將 *F* 值設定為 0.5 都較設定為雙重底混沌演算法來的優良，但在使用 DE/best/1 策略以及 DE/current-to-best/1 策略時，*F* 值使用雙重底混沌演算法的六個群集都較固定為 0.5 來的優良，而在其他策略中則是互有輸贏；本研究將使用不同策略時所得到的最優解，旁邊有加上米字號(*)，而只有使用 DE/current-to-best/1 時得到的米字號為最多的，且六個資料集中，就有五個資料集為使用雙重底混沌演算法，從表可以看出，使用不同的策略應用於不同題目，*F* 值的設定會有相當的影響。

4 結論

本研究利用六筆真實 UCI 的資料來驗證本研究提出的雙重底混沌差分演算法是否能有效的解決分群問題，在與文獻中的五種分群方法 K-means、PSO、NM-PSO、K-PSO、K-NM-PSO 進行比較，來驗證雙重底混沌差分演算法的優劣。雙重底混沌差分演算法，主要是將雙重底混沌方法的概念應用在差分演算法中的突變權重因子上，利用雙重底混沌方法的特性，令向量可以更好的提升搜尋全域搜尋解，並且往最佳解快速收斂。實驗結果與其他方法進行比較，雙重底混沌差分演算法在群集內距離總和及錯誤率都較其他演算法優異，顯示此方法可以有效的尋找出最佳解。

References

[1] M. R. Anderberg, "Cluster analysis for applications," DTIC Document1973.

[2] L. Harris, "Stock price clustering and discreteness," *Review of financial studies,* vol. 4, pp. 389-415, 1991.

[3] M. J. Berry and G. S. Linoff, *Data mining techniques: for marketing, sales, and customer relationship management*: John Wiley & Sons, 2004.

[4] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*: Morgan Kaufmann, 2005.

[5] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, pp. 281-297.

[6] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering,* vol. 60, pp. 208-221, 2007.

[7] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *ACM Sigmod Record*, 1999, pp. 49-60.

[8] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, ed: Springer, 2010, pp. 760-766.

[9] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization,* vol. 11, pp. 341-359, 1997.

[10] M. M. Ali and A. Torn, "Population set-based global optimization algorithms: some modifications and numerical studies," *Computers & Operations Research,* vol. 31, pp. 1703-1725, Sep 2004.

[11] K. V. Price, "Differential evolution: a fast and simple numerical optimizer," in *Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American*, 1996, pp. 524-527.

[12] R. Storn, "On the usage of differential evolution for function optimization," in *Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American*, 1996, pp. 519-523.

[13] C.-H. Yang, Y.-D. Lin, L.-Y. Chuang, and H.-W. Chang, "Double-bottom chaotic map particle swarm optimization based on chi-square test to determine gene-gene interactions," *BioMed research international,* vol. 2014, 2014.

[14] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing,* vol. 11, pp. 1679-1696, Mar 2011.

[15] Y.-T. Kao, E. Zahara, and I.-W. Kao, "A hybridized approach to data clustering," *Expert Systems with Applications,* vol. 34, pp. 1754-1762, 2008.

[16] K. V. Price, "An introduction to differential evolution," in *New ideas in optimization*, 1999, pp. 79-108.