APSO-TVAC 演算法應用於資料分群問題

楊正宏
稻江科技暨管理學院 網路系統學系
國立高雄應用科技大學 電子工程系
chyang@cc.kuas.edu.tw

蕭智仁
國立高雄應用科技大學 電子工程系
1097305142@cc.kuas.edu.tw

莊麗月
義守大學 化學工程系
chuang@isu.edu.tw

***摘要*—資料之分群可協助使用者從龐大的資料中，辨別資料間的結構與簡化資料複雜性，並從中粹取出有意義的資訊，分群結果之優劣，將直接影響使用者之分析結論。本研究提出粒子族群最佳化演算法(Particle Swarm Optimization, PSO)搭配動態調整學習因子之概念(Time-Varying Acceleration Coefficients, TVAC)，以增強搜索全域最佳解之能力，並結合加速策略(Acceleration strategy)以加快 PSO 之收斂速度，使其在解空間中之搜尋過程能擁有更強健之能力及穩定性，稱為 APSO-TVAC。本研究使用六個真實資料集來對所提出之方法進行實驗，並與相關文獻已提出之分群演算法進行群集內距離總和、錯誤率與計算適應函數總數的比較。實驗結果顯示，與文獻之多種分群法相較，本研究方法能有效提高分群問題之效能及效率。**

***關鍵詞*—資料分群、K-means、粒子族群最佳化、動態調適。**

一、前言

非監督式學習(Unsupervised learning)技術在資料探勘(Data mining)中扮演著非常重要的角色，其中以群集分析最具代表性。群集分析(Clusters analysis)主要用於辨別各種不同類別的資料，並整理相同屬性的資料形成不同集合，這些集合即稱為群集(Clusters) [1]。通常特定問題會以多維空間(Multi-dimensional space)的向量來描述物件屬性，群集分析是一種將物件歸類到相似物件之類別的過程，不同於監督式學習(Supervised learning)依照已定義的分類屬性將資料歸類。群集分析是將性質相似的資料進行分群，而把資料歸類成新類別[6]。分群則是透過特定數學函數來找尋空間物件之間的相似性，分析的最終目的是將資料進行分類。分群演算法主要區分為切割式與非切割式群集演算法等兩大類，切割式群集演算法是先指定群集個數，再將資料分割至類似之小組裡以創造分群的集合，主要之方法為 K-means[13]與 K-medoid [10]等演算法；非切割式群集演算法則是將資料組織到大群集裡，大群集包含更小之群集，可分成凝聚法(Agglomerative)及分散法(Divise)兩種方式，主要有 BIRCH [22]、CURE[5]與 Chameleon[9]等演算法。

K-means 演算法[13] 是一種最普遍也最廣泛被應用的分群技術，其概念是由 K 個群集中心點開始進行分群，並將欲分群的集合切割成 K 個子集合，其優點在於容易實現及有著高效率的線性時間複雜度(linear-time complexity) [3]。然而，由於 K-means 之目標函數(Objective function)不是凸面(Convex)函數，故可能包含許多區域最小值(Local minima)，使演算法在執行最小化目標函數的過程時，所獲得的解極可能落入區域最小值而無法尋獲全域最佳解[19]。因此，K-means 演算法的結果與初始群集中心的選擇，兩者之間存在相當重要的關聯性。若資料欲分為 K 個群集，K 值為已知假定，則 d 維空間的 n 個物件將分成 K 群，屬於同一群的物件其相似度較高，反之屬於不同群集的物件則相似度較低[8]。

近年來，已有許多進化式演算法(Evolutionary algorithms)應用於分群領域，如 Murthy 等學者提出以基因演算法(Genetic Algorithms, GA) [14]求解分群問題及 Bandyopadhyay 等學者提出 K-means 混合 GA 之分群法[2]。其中，GA 是以隨機方式初始母體之染色體，透過選擇、交配與突變等機制，使染色體經過多次迭代後，獲得最

佳染色體以解決最佳化問題。較新穎的應用則為結合粒子族群最佳化(Particle Swarm Optimization, PSO)於分群技術上[17]，PSO 是一種以族群為基礎的最佳化演算法[11]，已被成功應用於解決許多領域之各種最佳化問題，例如函數最佳化[20]、參數最佳化[16]、人工神經網路[12]等。雖然進化式演算法最終能找到較理想的結果，但實際應用於複雜的最佳化問題時仍有高計算量與收斂速度(Convergence rate)慢的缺點[8]。

PSO 在多維空間中有很優秀的表現，然而在解分群問題時其收斂速度卻比 K-means 演算法慢，且其搜尋全域最佳解之能力仍有改進之空間。因此本研究利用動態調整學習因子增強 PSO 於解空間中搜尋全域最佳解之能力，並於初始族群中執行一種與 K-means 演算法相似之加速策略，以改善 PSO 收斂速度較慢之缺點。

二、相關研究

(一) 粒子族群最佳化(Particle Swarm Optimization, PSO)

PSO 是一種以族群智慧為基礎的最佳化演算法，於 1995 年由 Kennedy 與 Eberhart 所提出[11]，其概念為模擬鳥群飛行與魚群活動的族群特性以及族群智慧所發展出來之啟發式演算法。在一個社會化的族群中，每一個個體的行為不但會受其過去經驗和認知的影響，也同時受到整體社會行為影響。在 PSO 中每一個體在搜尋空間中擁有各自的方向和速度，並根據自我過去經驗與族群行為，進行機率式的搜尋策略調整。粒子族群是由 N 個粒子所構成的，並且在 d 維度的搜尋空間中進行移動。首先，藉由隨機方式初始多個可能的解，其中第 i 個粒子的位置和速度可分別表示為 $x_i = (x_{i1}, x_{i2}, \cdots, x_{id})$和 $v_i = (v_{i1}, v_{i2}, \cdots, v_{id})$，且每個粒子的位置與速度皆限制在[Xmin, Xmax]^d 及[Vmin, Vmax]^d 的範圍，其中 Xmin 與 Vmin 分別為粒子的位置與速度之下限，Xmax 與 Vmax 分別為粒子的位置與速度之上限。此外，所有粒子皆有各自的搜尋區域，並會記錄本身的搜尋經驗。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 $pbest_i$，可以表示為 $p_i = (p_{i1}, p_{i2}, \cdots, p_{id})$；然而在粒子群中，最好的個體最佳適應值則稱為族群最佳適應值 $gbest$，其表示為 $g = (g_1, g_2, \cdots, g_d)$。PSO 的位置和速度之更新公式如下：

$$v_{id}^{new} = w \times v_{id}^{old} + c_1 \times r_1 \times \left(pbest_{id} - x_{id}^{old}\right) + c_2 \times r_2 \times \left(gbest_d - x_{id}^{old}\right) \quad (1)$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new} \quad (2)$$

在上述公式中，w 為慣性權重值，c_1、c_2 分別為 $pbest_i$ 與 $gbest$ 的學習因子；r_1 和 r_2 為隨機產生 0 到 1 之間的亂數，x_{id}^{old}、v_{id}^{old}、x_{id}^{new}和 v_{id}^{new} 分別表示粒子更新前與更新後的位置及速度。在本文中，w 如公式(3)所示[4, 7]，圖 1 為 PSO 之流程圖。

$$w = 0.5 + (\frac{rand}{2.0}) \quad (3)$$

(二) 動態調整學習因子(Time-Varying Acceleration Coefficients, TVAC)

所謂動態調整學習因子是指在 PSO 演算法中，學習因子 c_1 和學習因子 c_2 為非固定的值，且會隨著迭代次數動態調變。c_1 和 c_2 代表加速度常數(Acceleration constants)，又可稱為學習因子(Learning factors)，其具有能調控粒子被拉往 $pbest_i$ 或是 $gbest$ 位置之影響力。為避免粒子發生過度發散或提早收斂的情況，因此一般建議將加速因子固定為 2[11]，以維持單一性，目前已被廣泛應用在其他研究。但依據後續之研究發現[18, 21]，當演算法採用 TVAC 時，演算法效能會有較佳的表現，依據 Ratnaweera 和 Halgamuge 等學者之研究指出，當學習個體最佳經驗的因子 c_1 由 2.5 遞減至 0.5 以及學習群體最佳經驗的因子 c_2 由 0.5 遞增至 2.5，且 c_1 與 c_2 同步進行時，一開始演算法會因 c_1 的值較大且 c_2 的值較小，

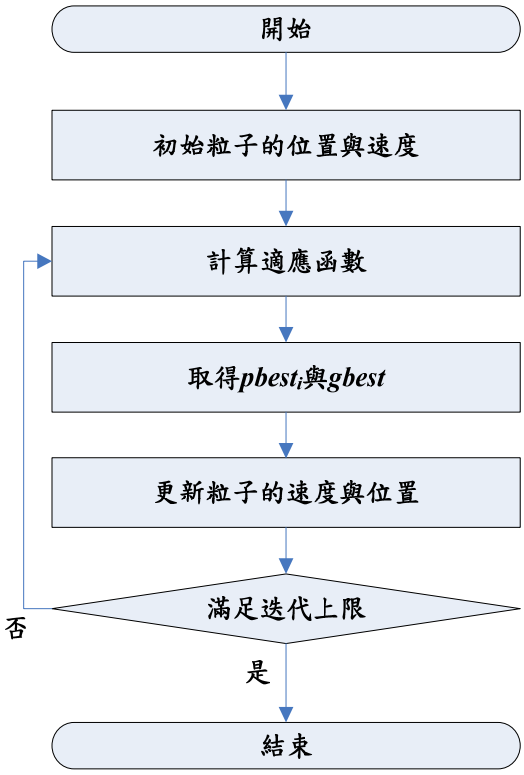

圖 1. PSO 之流程圖

而較著重於參考個體最佳以搜尋全域最佳解，反之隨著迭代增加，c_2 的值較大而 c_1 的值較小，會比較偏向於參考群體最佳以搜尋區域最佳解，此時演算法會具有較大之效能[18]，c_1 與 c_2 之線性遞減如公式(4)與公式(5)所示。此外針對粒子搜尋過程中，可能發生過早收斂的情況，文獻中亦有針對其求解標準差進行比較分析。

$$c_1 = (c_{1f} - c_{1i}) \times \left(\frac{iter}{\max iter} \right) + c_{1i} \tag{4}$$

$$c_2 = (c_{2f} - c_{2i}) \times \left(\frac{iter}{\max iter} \right) + c_{2i} \tag{5}$$

其中 c_{1i} 與 c_{1f} 分別表示學習因子 c_1 之起始值與結束值，c_{2i} 與 c_{2f} 分別表示學習因子 c_2 之起始值與結束值，$iter$ 為目前迭代數，max$iter$ 則表示最大之迭代數。

三、研究方法

基於 PSO 收斂速度較為緩慢且跳脫區域最佳解的能力仍不夠強健，因此，本文提出動態調整學習因子(TVAC)於 PSO 中以增強區域及全域搜尋之能力，並結合一加速策略，稱之為加速 PSO-TVAC (Accelerated PSO-TVAC, APSO-TVAC)。藉由與 K-means 演算法相似之加速策略，以改善 PSO 於解分群問題時搜尋最佳解之能力。在本節中我們將對 APSO-TVAC 作詳細之介紹。

(一) 粒子編碼

每個粒子皆包含所有群集數之中心點，且每個粒子的總維度為資料向量維度與群集數相乘之結果。例如，若要在二維搜尋空間中進行群集數為四的分群時，則 PSO 初始族群裡每個粒子皆包含四個群集中心點之位置，其編碼方式如圖 2 所示，其中二十個藍色菱形表示資料點，四個黑色圓圈代表四個群集的中心點，其座標位置分別為(1.1, 3.6), (3.0, 1.2), (7.0, 2.0)與(8.0, 4.6)，此例之粒子編碼即為(1.1, 3.6, 3.0, 1.2, 7.0, 2.0, 8.0, 4.6)。

(二) 族群初始化

依設定的族群數 $3N$ 及編碼的長度 N，以隨機方式在資料集的向量空間中產生每一粒子之位置及速度，參數 N 如公式(6)所示，其中 K 表示群集個數，d 為維度，參數 N 會依不同資料集而進行更動。每個粒子均表示一種解，每種解皆代表一種可能之分群結果。

$$N = K \times d \tag{6}$$

(三) 適應函數(Fitness function)

適應函數是將各群集的中心點與歸屬在同一群集的資料點之最短距離加總，當總和越小時

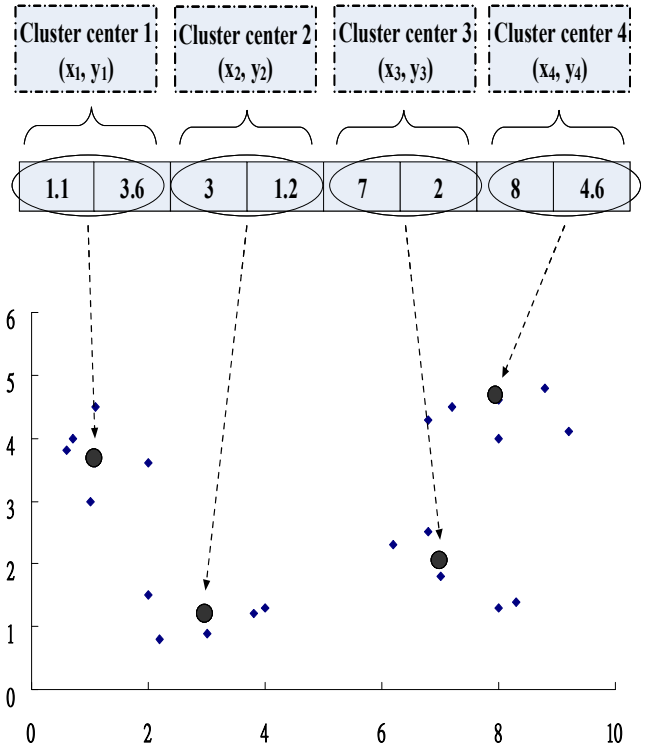

圖 2. 粒子編碼示意圖

代表分群結果越好。一般而言，距離之總和對於錯誤率也具有相當大的影響程度。公式(7)為分群評估函數，F 為群集內距離總和，Z 代表 K 個群集中心點，X 則表示 n 個資料點。

$$F = \sum \left\| X_j - Z_i \right\|,\ i = 1, ..., K,\ j = 1, ..., n \tag{7}$$

(四) 加速策略(Acceleration strategy)

將初始族群前三分之一粒子進行群集中心點位置的重置，以加速 PSO-TVAC 演算法之收斂速度。先對所有資料物件，利用歐基里德公式計算與群集中心點之距離，以距離最近者相似度最高之原則，逐一找出與其最相似的群集中心，再將該物件歸屬到最近似的群集，歐基里德距離如公式(8)所示。接下來利用公式(9)計算出算術平均中心點並取代原本之群集中心點成為新的粒子。

$$D(x_p \cdot z_j) = \sqrt{\sum_{i=1}^{d} (x_{pi} - z_{ji})^2} \tag{8}$$

$$z_j = \frac{1}{n_j} \sum_{\forall x_p \in c_j} x_p \tag{9}$$

其中 i 表示維度，j 表示群集，x_p 表示第 p 個資料向量，z_j 為群集 j 的中心點，d 為每個中心點的維度，n_j 為群集 j 的資料向量個數，C_j 為群集 j 的資料向量。

(五) 演算法流程

APSO-TVAC 演算法之執行步驟如下所述：

步驟一：初始族群中每個粒子的位置與速度，粒子位置即為每個群集的中心點位置。

步驟二：利用公式(8)與公式(9)對族群前 1/3 個粒子進行加速策略。

步驟三：對每個粒子進行分群，即利用歐基里德距離將所有資料向量歸屬給群集中心點。

步驟四：計算每一個粒子的適應函數值。

步驟五：比較每個粒子的個體最佳適應值 *pbest*，若比目前的 *pbest* 佳，則取代 *pbest*。

步驟六：比較每個粒子的族群最佳適應值 *gbest*，若比目前的 *gbest* 佳，則取代 *gbest*。

步驟七：根據公式(1)與公式(2)更新每一個粒子的速度與位置，其中 c_1 與 c_2 如公式(4)與公式(5)所示。

步驟八：重複執行步驟三到步驟七，當達到所設定的迭代次數時則停止。

四、實驗結果與討論

(一) 資料集

本研究採用六組由美國加州 Irvine 大學資

訊與電腦科學系提供的實際資料集作為測試資料 (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/)，相關資訊如表 1 所示。資料集名稱分別為 Iris Plants、Contraceptive Method Choice (CMC)、Breast Cancer、Wine、Vowel 與 Crude Oil，資料集之詳細說明如下所述：

(1) Iris Plants (n = 150, d = 4, k = 3)：鳶尾植物資料庫共計 150 筆資料，其中包含了萼片(Sepal)與花瓣(Petal)的長度(Length)、寬度(Width)等四種屬性；鳶尾植物資料庫是由 Iris Setosa (50 筆)、Versicolour (50 筆)與 Virginica (50 筆)這 3 種鳶尾花的種類所組成。

(2) Contraceptive Method Choice (n = 1473, d = 9, k = 3) ：避孕器資料庫共計 1473 筆資料，其中包含九種屬性，經由分群可將資料歸納為不使用(No-use) 629 筆、長期使用(Long-term) 334 筆以及短期使用(Short-term)510 筆。

(3) Breast Cancer (n = 683, d = 9, k = 2)：乳癌資料庫共計 699 筆資料，扣除遺失資料，共整理出 683 筆資料，其中包含 9 種屬性，是由良性細胞(Benign)以及惡性細胞(Mali-gnant)兩種所組成。

(4) Wine (n = 178, d = 13, k = 3)：葡萄酒資料庫共計 178 筆資料，其中包含 13 種屬性；葡萄酒種類共有 3 種，分別是 class1 (59 筆)、class2 (71 筆)與 class3 (48 筆)。

(5) Vowel (n = 871, d = 3, k = 6)：母音資料庫是由 871 筆印地安語言之母音資料所構成，其中包含 3 種音頻屬性，並將之分類為六種母音，分別為 δ (72 筆)、a (89 筆)、i (172 筆)、u (151 筆)、e (207 筆)與 o (180 筆)。

(6) Crude Oil (n = 56, d = 5, k = 3)：天然石油資料庫共計 56 筆資料，每筆資料中皆含有五種屬性，分別是釩(Vanadium)、鐵(Iron)、鈹(Beryllium)、飽和碳氫化合物(Saturated Hydrocarbons)與芳香族碳氫化合物(Aromatic Hydrocarbons)；天然石油資料庫分成三種，分別為 7 筆 Wilhelm、11 筆 Sub-Mulnia 與 38 筆 Upper。

表 1：資料庫相關特性

資料集	群集數	特徵數	資料數目(括號裡為各類別之資料數目)
Iris	3	4	150 (50, 50, 50)
CMC	3	9	1473 (629, 334, 510)
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)

(二) 實驗結果

在本研究中，我們利用下列兩種評估準則與文獻上多種分群演算法作比較，進而評估 APSO-TVAC 與各種分群演算法的效能：

- **群集內距離總和(intra-cluster distances)**：即群集內的資料向量與群集中心點間之距離，定義如公式(7)。若距離總和越小，表示分群效能越佳。

- **錯誤率(error rate)**：資料集包含之所有資料點中，被歸屬於不正確群集的資料點比率，算法如公式(10)所示。其中 n 代表所有資料點的數目，A_i 與 B_i 分別表示第 i 個資料點正確的群集歸屬和分群後所歸屬之群集，資料點若歸屬群集正確以 0 表示，反之若歸屬群集錯誤則以 1 表示。若錯誤率越低，表示分群效果越佳。以表 2 為例，五個資料點要分為兩群，其中有二個資料點在分群後所歸屬的群集錯誤，則錯誤率為 2/5，即 40%。

$$ER = \left(\sum_{i=1}^{n} (if \quad (A_i = B_i) \quad then \quad 0 \quad else \quad 1) \div n \right) \times 100 \qquad (10)$$

表 3 為各項實驗所需參數之設定表，其中參數 N 如公式(6)所示。表 4 與表 5 中的結果皆為執行 20 次之平均，每次執行均分別對六個多維

問題的資料集執行 $10N$ 次迭代，N 如公式(6)所示。將迭代次數設為 $10N$ 是根據許多根據文獻而設定，其在效率與效果方面都有良好的表現[8]。

文獻之五種演算法(K-means、PSO、NM-PSO、K-PSO 和 K-NM-PSO) [8] 與 PSO-TVAC 及 APSO-TVAC 的執行結果如表 4 所示，表中分別列出平均值、標準差(Standard deviations)以及最佳值。平均值為執行 20 次後所得的群集內距離總和之平均，最佳值為演算法所求得之最佳解。括號中的值為解範圍內之標準差，由於標準差能顯示每次實驗結果的離散程度，若標準差的值較大，則代表大部份實驗數據與其平均值有較大差異；反之則表示大部份的實驗數據與平均值較接近且差異較小，故標準差越小越好。因此，可藉由實驗結果的標準差數值來判定分群演算法之求解穩定性。由實驗結果得知，PSO-TVAC 在所有資料集之結果皆較 PSO 佳，而在 Wine 與 CMC 資料集甚至比結合 Nelder-

表 2：錯誤率計算方式

i	資料點	A_i	B_i	正確分群(0)/錯誤分群(1)
1	(2, 6)	2	2	0
2	(6, 3)	2	1	1
3	(1, 7)	1	1	0
4	(5, 4)	2	1	1
5	(8, 7)	1	1	0
被分錯群集的資料點個數：				2

表 3：實驗參數設定表

	各項設定
迭代次數	$10N$
族群大小	$3N$
實驗總次數	20
評估效能之標準	群集內距離總和、錯誤率
評估效率之標準	計算適應函數總數

表 4：七種分群演算法的群集內距離比較表

資料集	評估標準	K-means [8]	PSO [8]	NM-PSO [8]	K-PSO [8]	K-NM-PSO [8]	PSO-TVAC	APSO-TVAC
Iris	平均值	106.05	103.51	100.72	96.76	96.67	101.67	**96.66**
	(標準差)	(14.11)	(9.69)	(5.82)	(0.07)	(0.008)	(4.24)	(0.005)
	最佳值	97.33	96.66	96.66	96.66	96.66	96.95	96.66
CMC	平均值	5693.60	5734.20	5563.40	5532.90	5532.70	5554.74	**5532.20**
	(標準差)	(473.14)	(289.00)	(30.27)	(0.09)	(0.23)	(10.77)	(0.01)
	最佳值	5542.20	5538.50	5537.30	5532.88	5532.40	5538.8	5532.19
Cancer	平均值	2988.30	3334.60	2977.70	2965.80	2964.70	2986.92	**2964.42**
	(標準差)	(0.46)	(357.66)	(13.73)	(1.63)	(0.15)	(10.97)	(0.04)
	最佳值	2987	2976.30	2965.59	2964.50	2964.50	2967.76	2964.39
Wine	平均值	18061.00	16311.00	16303.00	16294.00	16293.00	16297.10	**16292.50**
	(標準差)	(793.21)	(22.98)	(4.28)	(1.70)	(0.46)	(2.42)	(0.24)
	最佳值	16555.68	16294.00	16292.00	16292.00	16292.00	16293.80	16292.19
Vowel	平均值	159242.87	168477.00	151983.91	149375.70	149141.40	155234.42	**149044.12**
	(標準差)	(916)	(3715.73)	(4386.43)	(155.56)	(120.38)	(4369.52)	(74.73)
	最佳值	149422.26	163882.00	149240.02	149206.10	149005.00	149681.91	148979.36
Crude Oil	平均值	287.36	285.51	277.59	277.77	277.29	278.74	**277.23**
	(標準差)	(25.41)	(10.31)	(0.37)	(0.33)	(0.095)	(0.807)	(0.031)
	最佳值	279.20	279.07	277.19	277.45	277.15	277.56	277.21

註:粗體表示七種分群演算法執行相同資料集後，群集內距離的平均值為七種分群演算法中最小之方法。

表 5：七種分群演算法的錯誤率比較表

資料集	評估標準	K-means (%)[8]	PSO (%)[8]	NM-PSO (%)[8]	K-PSO (%)[8]	K-NM-PSO (%)[8]	PSO-TVAC (%)	APSO-TVAC (%)
Iris	平均值	17.80	12.53	11.13	10.20	10.07	10.47	**10.00**
	(標準差)	(10.72)	(5.38)	(3.02)	(0.32)	(0.21)	(2.08)	(0.00)
	最佳值	10.67	10.00	8.00	10.00	10.00	6.67	10.00
CMC	平均值	54.49	54.41	54.47	**54.38**	**54.38**	54.82	**54.38**
	(標準差)	(0.04)	(0.13)	(0.06)	(0.00)	(0.054)	(0.396)	(0.00)
	最佳值	54.45	54.24	54.38	54.38	54.31	54.31	54.38
Cancer	平均值	4.08	5.11	4.28	3.66	3.66	3.64	**3.51**
	(標準差)	(0.46)	(1.32)	(1.10)	(0.00)	(0.00)	(0.21)	(9.1E-16)
	最佳值	3.95	3.66	3.66	3.66	3.66	3.37	3.51
Wine	平均值	31.12	28.71	28.48	28.48	28.37	28.82	**28.23**
	(標準差)	(0.71)	(0.27)	(0.27)	(0.40)	(0.27)	(0.41)	(0.24)
	最佳值	29.78	28.09	28.09	28.09	28.09	28.09	28.09
Vowel	平均值	44.26	44.65	41.96	42.24	41.94	42.02	**41.87**
	(標準差)	(2.15)	(2.55)	(0.98)	(0.95)	(0.95)	(1.71)	(0.18)
	最佳值	42.02	41.45	40.07	40.64	40.64	39.15	41.33
Crude Oil	平均值	24.46	24.64	24.29	24.29	**23.93**	26.34	26.60
	(標準差)	(1.21)	(1.73)	(0.75)	(0.92)	(0.72)	(0.79)	(0.55)
	最佳值	23.21	23.21	23.21	23.21	23.21	25	25.00

註:粗體表示七種分群演算法執行相同資料集後，錯誤率的平均值為七種分群演算法中最低之方法。

Mead 單體法 (Nelder-Mead simplex searchmethod)[15]的 NM-PSO 佳。與文獻中所有分群法比較，結合加速策略的 APSO-TVAC 演算法之平均值與標準差在所有資料集，不僅較 PSO-TVAC 佳，亦比文獻中效能最佳之 K-NM-PSO 分群法的結果佳，顯示 APSO-TVAC 在求解分群問題時確實較其他演算法來得可靠且穩定。

表 5 為 PSO-TVAC、APSO-TVAC 與文獻上五種分群法對資料集進行分群後的錯誤率比較表。表中分別列出平均值、標準差以及最佳值。平均值為執行 20 次後每次分群錯誤率之平均，最佳值為錯誤率最低的解。括號中的值為解範圍內之標準差，我們亦透過錯誤率之標準差來判定分群演算法的求解穩定性。經實驗結果得知，除 Crude Oil 資料集，APSO-TVAC 分群法在其餘五個資料集之錯誤率都較 PSO-TVAC 與文獻中效能最佳的 K-NM-PSO 分群法之結果低，實驗結果顯示出在大部份資料集中，APSO-TVAC 明顯優於文獻[8]的所有分群法，不僅平均值獲得更佳的數據且於標準差表示之穩定性方面亦有顯著的改善，證明 APSO-TVAC 更能準確將資料規屬於正確的群集。

表 6 列出五種分群演算法及本研究方法的計算適應函數總數，其目的為表示演算法之運算速度，計算適應函數總數越低，則表示演算法的運算量越低，而演算法之運算量越低越能提高演算法的執行效率。由表 6 可知所有資料集的計算適應函數總數之平均值為 K-means 演算法最低，但 K-means 演算法不屬於進化式演算法，且通常結果比其他五種演算法差。除 K-means 演算法之外，PSO-TVAC 及 APSO-TVAC 演算法所需計算適應函數總數最少，且 APSO-TVAC 演算法其群集內距離為七種分群演算法中最小之分群演算

表 6：七種分群演算法的計算適應函數總數比較表

資料集	K-means [8]	PSO [8]	NM-PSO [8]	K-PSO [8]	K-NM-PSO [8]	PSO-TVAC	APSO-TVAC
Vowel	180	16,290	10,501	15,133	9,291	9,720	9,720
Iris	120	7,260	4,836	6,906	4,556	4,320	4,320
Crude Oil	150	11,325	7,394	10,807	7,057	6,750	6,750
CMC	270	36,585	23,027	34,843	21,597	21,870	21,870
Cancer	180	16,290	10,485	15,756	10,149	9,720	9,720
Wine	390	73,245	47,309	74,305	46,459	45,630	45,630
平均值	215	26,833	17,259	26,292	16,519	16,335	16,335

表 7：GA、KGA 與 APSO-TVAC 的群集內距離比較表

資料集	評估標準	GA[14]	KGA[2]	APSO-TVAC
Iris	平均值	135.40	97.10	**96.66**
	最佳值	124.13	97.10	**96.66**
Vowel	平均值	390088.24	149368.45	**149241.36**
	最佳值	383484.15	149356.01	**148967.31**
Crude Oil	平均值	308.16	278.97	**277.27**
	最佳值	297.05	278.97	**277.21**

註:粗體表示執行相同資料集後，群集內距離的平均值為三種分群演算法中最小之方法。

法。

表 7 列出 GA [14]和整合 K-means 演算法及 GA 的 KGA 演算法[2]其群集內距離總和之數據，其中包含平均值與最佳值。由於文獻[2, 14]僅有 Vowel 、Iris 與 Crude Oil 等三個資料集的群集內距離總和，故僅針對此三個資料集與 APSO-TVAC 作比較。由表 7 可以發現於 Vowel 、Iris 與 Crude Oil 資料集，本文所提出之方法較 GA [14]和 KGA [2]之群集內距離的結果佳。GA 效能較 APSO-TVAC 差，主要為 GA 在經過多次迭代後，染色體的排列組合會越來越相似，導致再經過多次迭代後，結果只會得到一組相同的最佳解；而本文所提出之方法不僅利用 TVAC 來避免 PSO 提早收斂，且執行一加速策略來提高效率。實驗結果亦顯示 APSO-TVAC 在求解分群問題的效能上確實較 GA 及 KGA 優異。

(三) 討論

在本研究中，我們以群集內距離總和與錯誤率作為評估分群問題的準則。從表 3 與表 4 可發現，全部資料集在群集內距離部份皆以 APSO-TVAC 分群法最佳,錯誤率方面除了 Crude Oil 資料集，其餘資料集皆為 APSO-TVAC 分群法最低；Crude Oil 資料集在群集內距離部份為

APSO-TVAC 最佳，然而錯誤率卻是 K-NM-PSO 分群法最低。一般而言，群集內距離總和與錯誤率兩者之間並沒有絕對關係，因實際資料庫之資料分佈情形並不一定呈規則性分佈，因此距離總和佳，錯誤率不一定低[8]。

雖然 K-means 演算法計算較為快速，但其穩定性不佳且容易受到雜訊或離群資料的影響進而落入區域最佳解，導致結果之正確性降低。因此本文利用 PSO 最佳化演算法使其跳脫區域最佳解，並加入 TVAC 針對學習因子進行動態調適，以避免 PSO 過早收斂，並使 PSO 於解空間中有較廣泛之搜尋能力。除此之外針對 PSO 收斂速度較慢的問題[8]，本研究亦結合一種類似 K-means 演算法的加速策略於 PSO-TVAC 演算法中。APSO-TVAC 與混合 K-means 演算法及 PSO 的 K-PSO 演算法比較，其差異為加速策略不需執行完整的 K-means 演算法，僅需針對初始族群部份粒子進行中心點重置動作，利用這些粒子的多種可能性來找尋較佳之群集中心點，並透過算術平均中心點取代隨機產生之群集中心點的方式來加快收斂速度，藉此改善分群問題的效率。

文獻中效能最佳之 K-NM-PSO[8]為混合式演算法，其結合 K-means 演算法、Nelder-Mead 單體法(Nelder-Mead simplex search method)[15]與 PSO 演算法，K-NM-PSO 利用 Nelder-Mead 單體法進行區域搜尋，以改善 PSO 收斂速度較慢之情形。倘若遇到問題區域解太多，Nelder-Mead 單體法不僅很難找到全域最佳解，且相當容易陷入區域最佳解，得依靠 PSO 跳脫區域最佳解及找尋最佳解，但 PSO 搜尋全域最佳解的能力仍不夠強建。本研究透過 TVAC 來增強搜尋全域最佳解之能力，並利用加速策略之步驟取代混合 K-means 演算法。經由實驗結果得知，APSO-TVAC 演算法在表示分群演算法效能之群集內距離總和與錯誤率兩方面，其結果皆較文獻中之多種混合式分群演算法佳，且在演算法速度之計算適應函數總數方面，APSO-TVAC 演算法總數為最低，顯示本方法之效能及效率皆優於文獻上之分群演算法。

五、結論

在本研究中，我們提出 APSO-TVAC 演算法解決資料分群之問題，並以六個真實資料集進行實驗，再與 K-means、PSO、NM-PSO、K-PSO、K-NM-PSO 與 PSO-TVAC 等六種分群演算法進行比較，以驗證 APSO-TVAC 之效能與效率。APSO-TVAC 隨著迭代次數增加而動態調整學習因子大小，以增強 PSO 於複雜空間上搜尋全域最佳解之能力，並執行一種與 K-means 演算法相似之加速策略來解決 PSO 收斂速度較慢之缺點。綜觀本研究之實驗結果，APSO-TVAC 演算法不只群集內距離與錯誤率較其他方法優異，於代表效率之計算適應函數總數方面 APSO-TVAC 演算法亦為最少，顯示本研究方法具有同時兼顧效能與效率之能力。未來研究將應用 APSO-TVAC 於生物資訊領域中之各種不同分群問題，並發展結合不同演算法來提高 APSO-TVAC 效能之可能性。

六、參考文獻

[1] M. R. Anderberg, “Cluster analysis for applications”, Academic Press, New York, 1973.

[2] S. Bandyopadhyay and U. Maulik, “An evolutionary technique based on K-Means algorithm for optimal clustering in RN”, Information Sciences, Vol. 146, pp. 221-237, 2002.

[3] C. Y. Chen and F. Ye, “Particle swarm optimization algorithm and its application to clustering analysis”, 2004 IEEE International Conference on Networking, Sensing and Control, pp.789-794, 2004.

[4] R. C. Eberhart and Y. Shi, “Tracking and optimizing dynamic systems with particle

swarms", Proceedings of the Congress on Evolutionary Computation, Vol. 1, pp. 94-100, 2001.

[5] S. Guha, R. Rastogi and K. Shim, "Cure: An efficient clustering algorithm for large databases", SIGMOD, pp.73-84, 1998.

[6] J. Han and M. Kamber, "Data mining: concepts and techniques", Morgan Kaufmann, 2000.

[7] X. Hu and R. C. Eberhart, "Tracking dynamic systems with PSO: where's the cheese?", Proceedings of the workshop on particle swarm optimization, 2001.

[8] 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.

[9] G. Karypis, H. Eui-Hong and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling", Computer, Vol. 32, pp. 68-75, 1999.

[10] L. Kaufman and P. J. Rousseeuw, "Finding Groups in Data: an Introduction to Cluster Analysis", John Wiley & Sons, 1990.

[11] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", Proceedings of the IEEE International Conference on Neural Networks, Vol. 4, pp.1942-1948, 1995.

[12] J. Kennedy, R. C. Eberhart and Y. Shi, "Swarm Intelligenc", Morgan Kaufmann Publishers, 2001.

[13] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations", Proceedings of the Fifth Berkeley Symp. Math. Stat. Prob., pp. 281-297, 1967.

[14] C. A. Murthy and N. Chowdhury, "In search of optimal clusters using genetic algorithms", Pattern Recognition Letters, Vol. 17, pp.825-832, 1996.

[15] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization", The Computer Journal, Vol. 7, pp.308-313, 1965.

[16] K. E. Parsopoulos and M. N. "Vrahatis, Parameter selection and adaptation in Unified Particle Swarm Optimization", Mathematical and Computer Modelling, Vol. 46, pp.198-213, 2007.

[17] S. Paterlini and T. Krink, "Differential evolution and particle swarm optimisation in partitional clustering", Computational Statistics & Data Analysis, Vol. 50, pp.1220-1247, 2006.

[18] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients", IEEE Transactions on Evolutionary Computation, Vol. 8, pp.240-255, 2004.

[19] S. Z. Selim and M. A. Ismail, "K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 6, pp.81-87, 1984.

[20] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization", Proceedings of Congress on Evolutionary Computation, pp.1945-1949, 1999.

[21] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator", Proceedings of the 1999 Congress of Evolutionary Computation, pp.1958–1962, 1999.

[22] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", SIGMOD, pp.103-114, 1996.