Combining K-means and Particle Swarm Optimization for Dynamic Data Clustering Problems

Yucheng Kao
Department of Information Management
Tatung University
Taipei, 104, Taiwan
ykao@ttu.edu.tw

Szu-Yuan Lee
Department of Information Management
Tatung University
Taipei, 104, Taiwan
ericplum@gmail.com

Abstract—**This paper presents a new dynamic data clustering algorithm based on K-means and Combinatorial Particle Swarm Optimization, called KCPSO. Unlike the traditional K-means method, KCPSO does not need a specific number of clusters given before performing the clustering process and is able to find the optimal number of clusters during the clustering process. In each iteration of KCPSO, a discrete PSO is used to optimize the number of clusters with which the K-means is used to find the best clustering result. KCPSO has been developed into a software system and evaluated by testing some datasets. Encouraging results show that KCPSO is an effective algorithm for solving dynamic clustering problems.**

Keywords: Particle Swarm Optimization, K-means, Data clustering, Dynamic clustering

I. INTRODUCTION

The data clustering problem is defined as a problem of grouping data objects into clusters of similar objects without giving any priori knowledge. Data clustering is one of important topics in the fields of data mining and statistical data analysis. The potential applications of data clustering include data mining, pattern recognition, market segmentation, bioinformatics, and so on. Many clustering methods have been developed to solve data clustering problems. In general, these clustering methods can be classified into a couple of major categories, such as hierarchical clustering and nonhierarchical clustering approaches and so on [1].

The hierarchical clustering approaches can be further classified into two categories: agglomerative and divisive approaches. In an agglomerative clustering method, each data point represents a cluster in the beginning, and two appropriate small clusters merge each other to form a larger cluster recursively. On the contrary, a divisive clustering method starts with a large cluster containing all of the data points, and then split the cluster into two smaller but dissimilar clusters repeatedly until an appropriate clustering result presents.

The nonhierarchical clustering approach is also called the partitioning approach. Given a set of n data points, a partitioning clustering method tries to assign n data points into one of k clusters based on the distances between a data point and each cluster center. A data point is assigned to a cluster in which the center has the shortest distance to the data point. Data members of a cluster are similar to each other; while data points belonging to different clusters are different. Therefore, the clustering problem can be viewed as a problem of partitioning a multi-dimensional space into k subspaces. One of the famous nonhierarchical clustering approaches is K-means. The advantages of K-means are fast convergence and easy implementation. However, K-means has two main drawbacks. First, the number of clusters has to be specified a prior; second, the initial condition may affect the clustering results [2].

In order to remedy the drawbacks of K-means, this paper proposes a new dynamic data clustering algorithm based on particle swarm optimization. Dynamic clustering means that the approach has to find a proper number of clusters and also to find the best clustering result for the found number of clusters. In other words, it can automatically find both of the best number of clusters and the best corresponding clustering result through an iterative process. Apparently, the advantage of a dynamic clustering approach is that it does not require the number of clusters to be specified in advanced.

In section 2, the PSO algorithm and some dynamic clustering algorithms are briefly introduced. Section 3 presents the KCPSO algorithm with an illustrative example. Section 4 gives experimental work and a comparative study. In the last section, a conclusion and future work are provided.

II. BACKGROUNDS

Clustering problem is mathematically defined by following equations:

For a set of data points: $\mathbf{X} = \{X_1, X_2, \cdots, X_P, \cdots, X_N\}$, where X_P is a data point, N is the total number of data points. Suppose the number of clusters is K, data set **X** will be divided into K clusters $\{C_1, C_2, ..., C_k\}$ and the following three conditions have to be satisfied：

Condition 1：the union set of all clusters is equal to X：

$$\bigcup_{k=1}^{K} C_k = X \tag{1}$$

978-1-4244-4738-1/09/$25.00 ©2009 IEEE

Condition 2：empty cluster is not allowed：

$$C_k \neq \varnothing, k = 1,..., K \tag{2}$$

Condition 3：each data point is assigned to only one cluster:

$$C_k \cap C_l = \varnothing, \text{where } k \neq l,\ l = 1, ...K \tag{3}$$

As mentioned in the first section, K-means is one of famous partitioning clustering methods. It was proposed by MacQueen in 1967 [3] and is very easy to implement. For a given number of clusters *K*, K-means first randomly selects *K* data points as initial cluster centers. Then each data point is assigned to one of clusters according to the distances between the point and each cluster center. After that, every cluster center is updated based on its cluster members using equation (4). With the new cluster centers, the procedure is repeated until cluster centers do not change any more.

$$Z_k = \frac{1}{|C_k|} \sum_{X_j \in C_k} X_j \tag{4}$$

where Z_k is the center of cluster k, $|C_k|$ is the number of members of cluster k, C_k represents cluster k, and X_j is a member of cluster k.

Although the computational time of the K-means algorithm is very short, its clustering quality is easy to be influenced by the selection of initial cluster centers [2]. Some researchers tried to solve the problems by using evolutionary computing techniques like genetic algorithms (GA) [4, 5] and particle swarm optimization (PSO) [6, 7]. Tseng et al. tried to apply GA to solve the clustering problems [8]. In 2000, Maulik and Bandyopadhyay [9] also used GA to solve the clustering problems. Both of the two papers considered traditional partitioning clustering problems, not dynamic clustering problems. In 2002, Maulik and Bandyopadhyay [10] proposed a GA-based algorithm to solve dynamic clustering problems. Their algorithm is called Genetic Clustering for Unknown K (GCUK).

PSO algorithm was introduced by Kennedy and Eberrhart in 1995 [6]. It is a kind of swarm intelligence that a swarm of particles simulate a flock of birds to search food sources. Each particle represents a potential solution and flies through the search space to find optimal solutions. Merwe and Engelbrecht [11] first applied PSO to partitioning clustering problems. In their algorithm, K-means is first used to find a clustering solution as the initial position for each particle. After flying in the search space, a swarm of particles can find final optimal solutions very fast and avoid trapping in local optima. In 2005, Omran *et al.* [12] proposed a PSO-based dynamic clustering algorithm to solve partitioning problems without giving the number of clusters. Their algorithm is called Dynamic Clustering using Particle Swarm Optimization (DCPSO). In 2007, Jarboui *et al.* [13] presented a new discrete PSO algorithm called Combinatorial Particle Swarm Optimization (CPSO) to solve the partitioning clustering problem. They modeled the clustering problem as a combinatorial optimization problem. However, their proposed approach cannot solve dynamic clustering problems.

For the dynamic clustering problem, a grouping criteria, called validate index, is used to evaluate the clustering result. A validate index usually considers the ratio of two criteria: compactness and separation. Compactness means that the members of each cluster should be similar as much as possible. Separation means that any two clusters should be properly separated by a distance. A good validate index has a lower compactness value and a higher separation value. This paper uses DB index that is proposed by Davies and Bouldin in 1979 [14]. A smaller DB index value means a better result of dynamic clustering.

III. KCPSO ALGORITHM

In the proposed algorithm, KCPSO, each particle has two basic tasks: to search the best number of clusters by using a discrete PSO approach based on CPSO proposed in [13] and to find the best set of cluster centers by using the K-means according to the assigned number of clusters.

For the solution representation, each particle represents a potential solution that consists of a set of clusters, $\{C_1, C_2, \ldots, C_{Ki}\}$. Ki is the number of clusters assigned to particle i and its value is between MinK and MaxK. MaxK is the largest possible number of clusters, and MinK is the smallest possible number of clusters. MinK is generally set to two.

In the initialization phase, each particle randomly determines its own Ki value and uses the K-means to find the initial clustering result. The initial result of each particle is evaluated by computing the objective function, DB index, in order to find the initial pbest of the particle. The particle with the best pbest (smallest DB value) is selected as the global best solution (gbest), and its K value is also selected as the global best number of clusters, K_{gbest}.

The main body of KCPSO has four main steps: adjusting Ki value, selecting new clustering centers, performing K-means, updating pbest and gbest.

Step1: adjusting the value of Ki

In the first step, each particle adjusts its own K value by using a discrete CPSO approach. The current K value of particle i is mapped into a position in a virtual space according to equation (5).

$$Y_i^t = \begin{cases} 1 & \text{if } K_i^t = K_{gbest}^t, \\ -1 & \text{if } K_i^t = K_{ipbest}^t, \\ -1 \text{ or } 1 & \text{randomly if } (K_i^t = K_{gbest}^t = K_{ipbest}^t), \\ 0 & \text{otherwise}. \end{cases} \tag{5}$$

where, K_{gbest} is the best number of clusters of all particles, K_{ipbest} is the best number of clusters of particle i.

After all particles are mapped into the virtual space, equation (6) is used to find particles' new velocities. The initial

velocity of particle i is randomly selected from a uniform distribution ranged from -1 to 1. That is, $V_i^{t=0} \sim \pm U(0, 1)$. Through equation (7) and (8), particle i moves to the new position in the virtual space, Y_i^{t+1}.

$$V_i^{t+1} = w^{t+1} * V_i^t + c1 * rand() * (-1 - Y_i^t) + c2 * rand() * (1 - Y_i^t) \quad (6)$$

$$\lambda_i^{t+1} = Y_i^t + V_i^{t+1} \quad (7)$$

$$Y_i^{t+1} = \begin{cases} 1 & if\ \lambda_i^{t+1} > \alpha^{t+1}, \\ -1 & if\ \lambda_i^{t+1} < -\alpha^{t+1}, \\ 0 & otherwise. \end{cases} \quad (8)$$

$$K_i^{t+1} = \begin{cases} K_{gbest}^t & if\ Y_i^{t+1} = 1, \\ K_{ipbest}^t, & if\ Y_i^{t+1} = -1, \\ MinK \le random \le MaxK & otherwise. \end{cases} \quad (9)$$

The λ_i^{t+1} value in equation (7) is a temporary continuous value of particle position in the virtual space. It has to be compared with a threshold value α to determine its true new position. Finally equation (9) is used to transfer the particle from the virtual space back to original integer space and to determine the new number of clusters of particle i.

Step 2: selecting new cluster centers

If the new value of K of a particle is the same as that of the gbest partcle, then it directly copies the cluster centers of the gbest particle as its new cluster centers. In order to explore the search space more, a couple of centers will be randomly chosen and be replaced with randomly selected data points. The same procedure is done for a particle that inherits the number of clusters from its own pbest value. For the third case that the K value has to be randomly changed, the set of cluster centers is re-selected from the data set according to the new K value. This allows the KCPSO algorithm to jump out from a trap of local optimum.

Step 3: performing K-means

Once the number of clusters (K) and the set of cluster centers are determined, each particle uses the K-means to find a better clustering result.

Step 4: updating pbest and gbest

After performing the K-means, the algorithm uses DB index to evaluate the clustering result of each particle. For a particle, if its new DB index is better than before, then its K_{ipbest} value is replaced with the new one. After all particles are evaluated, the iteration-best particle is the particle that has the lowest DB index value. If the DB index value of the iteration-best particle is lower than that of the gbest particle, then the gbest particle will be replaced by the best particle of this iteration.

The above four steps repeat again and again until the termination condition is met. The termination criterion can be the maximum iteration number or the condition that the K values and DB values of all particles are the same.

IV. COMPUTATIONAL EXPERIMENTS

The proposed algorithm has been implemented in Microsoft Visual C# .Net 2005. All experiments conducted in this paper were run in Windows XP on Notebook with Intel Pentium M, 1.5 GHz processor. The performance of three dynamic clustering algorithms, KCPSO, DCPSO and GCUK, were compared.

The parameter settings of these three algorithms were determined by both referring to original papers and performing empirical studies, pleases see table 1. Experiments were conducted using six data sets including four artificial data sets (similar to test cases used in [10]) and two real world data sets selected from UCI machine learning repository [15]. Table 2 provides the information of these six test problems, where Nd is the number of dimensions, N is the total number of data points, Kc is the correct number of clusters, and PPC means the number of points in each cluster of a data set. Figure 1 presents the scatter diagrams of the four artificial data sets. For each of three dynamic clustering algorithms, twenty independent runs were performed for each test data set. DB index was used as the criterion to compare the clustering performance of these three algorithms.

TABLE 1. PARAMETER SETTINGS

KCPSO		DCPSO		GCUK	
# of particles	*10*	*# of particles*	*100*	*# of chromosome*	*50*
MaxIterations	*20*	*MaxIte.: inner loop*	*50*	*MaxIterations*	*100*
c1, c2	*0.5, 0.5*	*c1, c2*	*1.49, 1.49*	*Crossover rate*	*0.8*
W	*0.72*	*W*	*0.72*	*Mutation rate*	*0.001*
α	*1~0.35*	V_{max}	*255*	*MaxK*	*10*
MaxK	*10*	*MaxK (Nc)*	*20*	*MinK*	*2*
MinK	*2*	*MaxIte.: outer loop*	*2*		
		Pini	*0.75*		

TABLE 2. DESCRIPTION OF THE DATA SETS

Case No.	Nd	N	Kc	PPC
Data_3_2	*2*	*76*	*3*	*43, 20, 13*
Data_5_2	*2*	*250*	*5*	*50*
Data_6_2	*2*	*300*	*6*	*50*
Data_4_3	*3*	*400*	*4*	*100*
Iris	*4*	*150*	*3*	*50*
Breast Cancer	*9*	*683*	*2*	*444, 239*

The comparison results are summarized in table 3, where "correctness times" is the times that the algorithm finds the correct number of clusters (Kc) and MCT means mean computational time. For each test problem the best results among three algorithms are indicated with a star mark. For the four artificial data sets, both KCPSO and DCPSO provide better results in terms of the DB index values and in the correctness of the number of clusters. Compared to KCPSO, GCUK found worse DB Index values and did not find the correct number of clusters every time among 20 independent runs. In addition, KCPSO spent much less computational times than DCPSO and GCUK. From the experimental result, it can be concluded that the proposed algorithm outperforms the other two algorithms for these test problems.

For the first real data set, Iris, the best number of clusters found by these three algorithms was two, not three, in all 20 independent runs. The DB index value obtained by using DCPSO is better than the other two algorithms. For the test case of breast cancer, KCPSO provided better results on the objective function values and the correctness of the number of clusters. GCUK also found correct number of clusters in all 20 independent runs with almost the same objective function values. The results obtained by DCPSO were not as good as those of KCPSO. In this test case, KCPSO also took less CPU times as it did in the case of iris.

TABLE 3. PERFORMANCE COMPARISONS FOR THREE DYNAMIC CLUSTERING ALGORITHMS

Case No.	Algorithm	Mean DB Index	Correctness times	MCT (Seconds)
Data_3_2	GCUK	*0.4428*	*17/20*	*0.6995*
	DCPSO	*0.4299*	*20/20**	*1.5748*
	KCPSO	*0.4238**	*20/20**	*0.0951**
Data_5_2	GCUK	*0.5395*	*16/20*	*2.4050*
	DCPSO	*0.5257**	*20/20**	*4.7654*
	KCPSO	*0.5257**	*20/20**	*0.4266**
Data_6_2	GCUK	*0.3851*	*8/20*	*2.8766*
	DCPSO	*0.3746**	*20/20**	*5.4804*
	KCPSO	*0.3746**	*20/20**	*0.4446**
Data_4_3	GCUK	*0.4817*	*18/20*	*3.4399*
	DCPSO	*0.4745**	*20/20**	*7.3431*
	KCPSO	*0.4745**	*20/20**	*0.8672**
Iris	GCUK	*0.4043*	*0/20*	*1.2233*
	DCPSO	*0.3828**	*0/20*	*3.5326*
	KCPSO	*0.4043*	*0/20*	*0.3760**
Breast Cancer	GCUK	*0.7572*	*20/20**	*7.4788*
	DCPSO	*0.7775*	*18/20*	*19.8811*
	KCPSO	*0.7570**	*20/20**	*3.9742**

V. CONCLUSION

The dynamic clustering problem is divided into two parts in this paper: optimizing the number of clusters and finding the best cluster configuration under a specified number of clusters. In the proposed approach, the first part is solved by using combinatorial particle swarm optimization and the second part is solved by using the K-means under the found number of clusters. The experimental results show that KCPSO can effectively solve dynamic clustering problems. For most of the test problems, KCPSO can find the correct number of clusters and avoid local optimum to find the best clustering results within a very short time period. The performance of KCPSO has been compared with the other two dynamic clustering algorithms. The comparison result shows that KCPSO can find better or equal clustering results with less computational times.

REFERENCES

[1] J. Han and M. Kamber, "Data mining: Concepts and Techniques." San Francisco: Morgan Kaufmann Publisher, 2001.

[2] J. M. Peña, J. A. Lozano, and P. Larrañaga," An empirical comparison of four initialization methods for the K-Means algorithm." Pattern Recognition Letters 20 1027-1040,1999.

[3] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations." Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297, 1967.

[4] D. E. Goldberg, "Genetic Algorithm in Search." Optimization and Machine Learning.", Addision-Wesley, New York, 1989.

[5] L. Davis, "Handbook of Genetic Algorithm." Van Nostrand Reinhold, New York, 1991.

[6] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory." Proc. Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp.39-43, 1995.

[7] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources" Proc. IEEE Int. Conf. On Evolutionary Computation, pp.81-86, 2001.

[8] L. Y. Tseng, "Genetic algorithm for clustering, feature selection and Classification." 1997 IEEE International Conference on Neural Networks, vol, 3, pp. 1612-1616 1997.

[9] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique." Pattern Recognition vol. 33, pp.1455-1465, 2000.

[10] S. Bandyopadhyay and U. Maulik, "Genetic Clustering for Automatic Evolution of Clusters and Application to Image Classification.", Journal of the Pattern Recognition, vol. 35, no. 6, pp.1197-1208, 2002.

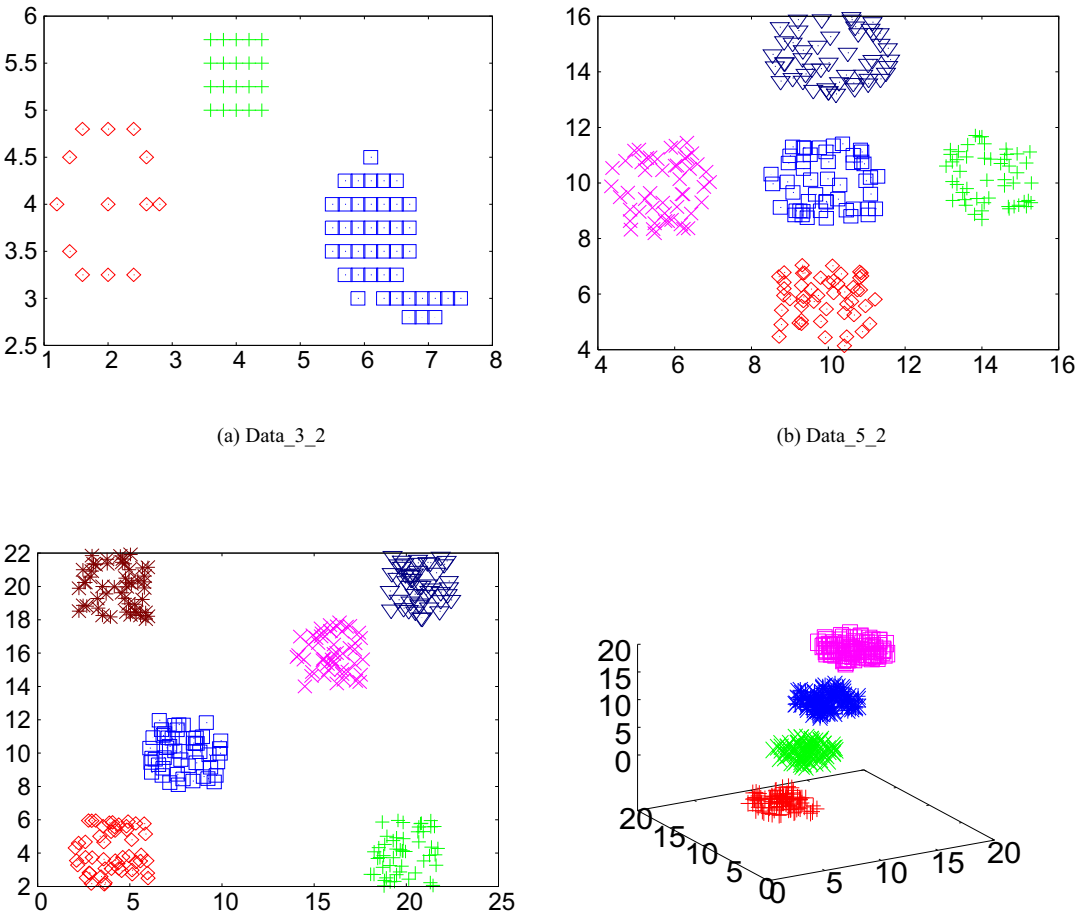
[11] D. W. V. D. Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization." Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), pp. 215-220, 2003.

[12] M. G. H. Omran, A. P. Engelbrecht and A. Salman, "Dynamic Clustering using Particle Swarm Optimization with Application in Unsupervised Image Classification.", TRANSACTIONS ON ENGINEERING, COMPUTING AND TECHNOLOGY, 9,199-204, 2005.

[13] B. Jarboui, M. Cheikh, P. Siarry and A. Rebai, "Combinatorial particle swarm optimization (CPSO) for partitional clustering problem.", Applied Mathematics and Computation, vol. 192, pp. 337-345, 2007.

[14] D. I. Davies and D. W. Bouldin, "A cluster separation measure.", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 1, No2, 1979.

[15] UCI Repository of Machine Learning Databases retrieved from the World Wide Web: http://www.ics.uci.edu/~mlearn/MLRepository.html.



(a) Data_3_2

(b) Data_5_2

(c) Data_6_2

(d) Data_4_3

Fig. 1. The scatter diagrams of artificial data sets.