Pattern Recognition 42 (2009) 2557 – 2569

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Enhanced bisecting *k*-means clustering using intermediate cooperation

R. Kashef*, M.S. Kamel

University of Waterloo, Electrical and Computer Engineering Department, Waterloo, Canada

ARTICLE INFO

Article history:
Received 19 March 2008
Received in revised form 15 January 2009
Accepted 5 March 2009

Keywords:
Bisecting clustering
Cooperative clustering
Quality measures

ABSTRACT

Bisecting *k*-means (BKM) is very attractive in many applications as document-retrieval/indexing and gene expression analysis problems. However, in some scenarios when a fraction of the dataset is left behind with no other way to re-cluster it again at each level of the binary tree, a "refinement" is needed to re-cluster the resulting solutions. Current approaches to refine the clustering solutions produced by the BKM employ end-result enhancement using *k*-means (KM) clustering. In this hybrid model, KM *waits* for the former BKM to finish its clustering and then it takes the final set of centroids as initial seeds for a better refinement. In this paper, a cooperative bisecting *k*-means (CBKM) clustering algorithm is presented. The CBKM *concurrently* combines the results of the BKM and KM at each level of the binary hierarchical tree using cooperative and merging matrices. Undertaken experimental results show that the CBKM achieves better clustering quality than that of KM, BKM, and single linkage (SL) algorithms with comparable time performance over a number of artificial, text documents, and gene expression datasets.

1. Introduction

Analysis of data can reveal interesting, and sometimes important, structures or trends in the data that reflect a natural phenomenon. Discovering regularities in data can be used to gain insight, interpret certain phenomena, and ultimately make appropriate decisions in various situations. Data clustering is a data mining technique that enables the abstraction of large amounts of data by forming meaningful groups or categories of objects, formally known as clusters, such that objects in the same cluster are similar to each other, and those in different clusters are dissimilar according to some defined similarity criteria in an unsupervised manner. A cluster of objects indicates a level of similarity between objects such that we can consider them to be in the same category, this simplifying our reasoning about them considerably. Clustering is used in a wide range of applications, such as marketing, biology, psychology, astronomy, information retrieval, image processing, and text mining. For example, in marketing it is used to find groups of customers that share common behavior for the purpose of market segmentation and targeted advertisement. In biology it is used to form taxonomy of species based on their features and to group the set of co-expressed genes together into one group. In image processing it is used to segment texture in images to differentiate between various regions or objects. Clustering is also practically used in many statistical analysis software packages for general-purpose data analysis. The increasing importance of data clustering and the widespread of its applications have led to the development of a variety of algorithms with different quality/complexity tradeoffs [1–9]. They differ in many aspects, such as the types of attributes they use to characterize the dataset, the similarity measure used, and the representation of the clusters.

Bisecting *k*-means (BKM) [7] is a variant of *k*-means (KM) clustering that produces either a partitional or a hierarchical clustering by recursively applying the basic KM method. In [10], it has been shown that the BKM with end-result refinement using the KM produces better results than KM and BKM. A drawback of this end-result enhancement is that KM waits until the former BKM finishes its clustering.

A generalization to the work done in [11], a novel cooperative bisecting *k*-means (CBKM) clustering algorithm is presented. The CBKM combines both the methodology of splitting clusters at each level of the generated hierarchical binary tree as in the traditional BKM and the synchronous intermediate cooperation with KM. The new CBKM is applied on different datasets with different types and configurations. The CBKM obtains the best clustering solutions from both KM and BKM at the intermediate levels based on cooperative contingency and merging matrices. Undertaken experimental results over artificial, document, and gene expression datasets show that the CBKM algorithm achieves better clustering quality than that of the traditional BKM, the KM algorithm, and also the hierarchical single linkage (SL) clustering [1] using different external and internal quality measures.

The rest of this paper is organized as follows: in Section 2, related work to data clustering is given. The proposed CBKM algorithm is

* Corresponding author. Tel.: +1 519722 7157.
E-mail addresses: rkashef@pami.uwaterloo.ca (R. Kashef), mkamel@pami.uwaterloo.ca (M.S. Kamel).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.03.011

Table 1
Symbols and notations.

Symbol	Definition
X	Dataset of objects
$\boldsymbol{x}$	An object represented as a vector of features
d	Dimensionality of the object $\boldsymbol{x}$
n	Number of objects
k	Number of clusters
S_j	The jth cluster
Sb_j	The jth sub-cluster as a result of the cooperation between the KM and BKM
H_i	Histogram of the sub-cluster Sb_i
δ	Similarity threshold
n_{sb}	Number of sub-clusters
R_j	The jth class (external labeling of objects)
c_j	The centroid of cluster j

presented in Section 3. Section 4 provides a formulation of some external and internal quality measures to assess the clustering quality. Experimental results are presented and discussed in Section 5. Finally, we draw some conclusions and outline future work in Section 6.

2. Related work and background

Most clustering algorithms can be classified into two groups: *hierarchical* and *partitional* clustering. The hierarchical techniques produce a nested sequence of partitions, with a single, all-inclusive cluster at the top and single clusters of individual objects at the bottom (leaf nodes) (divisive hierarchical clustering) or a set of singleton clusters at the top and one single partition at the bottom (agglomerative hierarchical clustering). Examples of the hierarchical clustering are the principal direction divisive partitioning (PDDP) [8], BKM [7], hierarchical agglomerative clustering (HAC) [1], collaborative document clustering (CDC) [9], and many others. The partitional clustering approaches partition a collection of objects into a set of groups, so as to maximize the quality of clustering. The KM [5] and fuzzy c-means [6] algorithms are members of the family of partitional clustering algorithms. Some terminologies and notations are best presented at this point to pave the way for discussion of the different concepts and strategies for data clustering and also for the proposed algorithm. Table 1 summarizes the notations and symbols that are used throughout this paper.

2.1. The adopted clustering techniques

The k-means (KM) algorithm selects k objects randomly in the dataset X as initial seeds for the cluster's centroids, and then assigns each objects $\boldsymbol{x}$ to the closest centroid c_i, $i=1,2,\ldots,k$ (*calculation step*). The new centroids are generated for each cluster by calculating the mean of the objects set assigned to each cluster (*updating step*). The iterative KM minimizes a dissimilarity (or distance) function (J)

$$J=\sum_{i=1}^{k}\sum_{\forall \boldsymbol{x}_j\in S_i} DisSim(\boldsymbol{x}_j,c_i) \tag{1}$$

where $DisSim(\boldsymbol{x}_j,c_i)$ is the dissimilarity function between object $\boldsymbol{x}_j$ and the cluster centroids c_i. A common similarity measure that is used specifically in document clustering is the *cosine correlation* measure (used by [12]), defined as

$$CosSim(\boldsymbol{x},\boldsymbol{y})=\frac{\boldsymbol{x}.\boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|} \tag{2}$$

where $(\cdot)$ indicates the vector dot product and $\|\cdot\|$ indicates the length of the vector. There are many varieties of the KM algorithm, to specify which version that is used throughout this paper, we will refer to as naïve KM [5] as shown in Fig. 1.

The BKM algorithm [7] is a variant of KM. It starts by considering the whole dataset to be one cluster. At each step, one cluster V is selected and bisected further into two partitions (V_1 and V_2) using the basic KM algorithm. This process continues until the desired number of clusters or some other specified stopping condition is reached. There are a number of different ways to choose which cluster to split. For example, we can choose: (1) the largest cluster at each step, or (2) the one with the least overall similarity, or (3) a criterion that satisfies both size and overall similarity. Fig. 2 outlines the BKM algorithm.

Recent study [10] concludes that, BKM is better than the standard KM and as good as or better than the hierarchical approaches as it partitions the dataset based on a homogeneity criteria. This is the reason why the bisecting approach is very attractive in many applications as document-retrieval/indexing problems and gene expression analysis. However, in some scenarios when a fraction of the dataset is left behind with no other way to re-cluster it again at each level of the binary hierarchical tree, a "refinement" is needed to re-cluster the resulting solutions.

Combining multiple clustering is considered as an example to further broaden a new progress in the area of data clustering. Combining clusterings is based on the level of cooperation between the clustering algorithms; either they cooperate on the *intermediate level* or at the *end-result* level. Examples of the end-result cooperation are the *ensemble clustering* and the *hybrid clustering* approaches [13–20]. Recent ensemble clustering techniques have been shown to be effective in improving the accuracy and stability of standard clustering algorithms [13–16]. In *hybrid clustering*, cascaded clustering algorithms cooperate together for the goal of refining the clustering solutions produced by a former clustering algorithm(s) [17–20]. Combining clustering with end-result cooperation has been shown to be effective in improving the clustering quality. However, inherent drawbacks of these techniques are the computational complexity of the ensemble clustering and the idle time wasted in the hybrid clustering approaches. Zhao and Karypis [10] showed that the hybrid BKM with end-result refinement using KM produces better results than KM and BKM. In this hybrid model, KM *waits* for the former BKM to finish its clustering and then it takes the final set of centroids as initial seeds for a better refinement.

3. Cooperative bisecting KM (CBKM) clustering

In this section, a novel CBKM clustering is presented. The CBKM combines both the methodology of splitting clusters at each level of the generated hierarchical binary tree as in the traditional BKM and a synchronous intermediate cooperative strategy with KM. The CBKM

Algorithm: *k*-means Clustering: KM(X , *k*)

Input: the dataset X, and the number of clusters *k*.

Output: Set of *k* clusters S=$\{S_0,S_2,..,S_{k-1}\}$

Initialization: S= {}, Randomly, select *k* initial centroids $\{c_i\}_{i=0}^{k-1}$ for the *k* clusters S_i

Begin

Repeat

Step1: Vectors are assigned to the closest centroid

Step2: The objective function *J* (Eq.1) is computed

Step3: New centroids c_i, i=1,..,k are calculated as the mean of the vectors assigned to each cluster S_i, $\{S_i\}_{i=0}^{k-1}$

Until Convergence (or no change in the objective function)

Return S

End

Fig. 1. The KM clustering algorithm.

Algorithm: Bisecting *k*-means Clustering: BKM (X, *ITER*, ζ, *k*)

Input: The dataset X, Number of iterations ITER for the bisecting step, homogeneity criterion ζ, and the desired number of clusters *k*

Output: The set of *k* clusters S=$\{S_0,S_2,..,S_{k-1}\}$

Initialization: - Let V = X, S={ }

Begin

For number of clusters *l* =2 to *k* (*Clustering Step*)

*Step*1 : **For *i*=1 to *ITER*** (*Bisecting Step*)

- Select randomly two initial centroids c_1 and c_2 from the set V.

- Find 2 partitions from the set V using the basic *k*-means algorithm.

End

*Step*2: Take the best of these splits as V_1 and V_2 with the corresponding centroids c_1 and c_2, respectively.

*Step*3: Select the cluster that satisfies the homogeneity criterion ζ as V_1

*Step*4: Assign V to the remaining partition, $V=V_2$

*Step*5: Add V_1 to the set of desired clusters $S=S\cup V_1$

End

Add V_2 to the set of desired clusters $S=S\cup V_2$

Return S

End

Fig. 2. The bisecting *k*-means (BKM) clustering algorithm.

initially starts with the whole dataset as one cluster. *At any level of the tree*, the CBKM is performed in three phases, the *global clustering phase*, the *cooperative clustering phase*, and the *merging phase*. Fig. 3 illustrates the different phases of the CBKM clustering. Each of those phases is discussed in the following sub-sections.

3.1. Global clustering phase

In the *global clustering phase*, one cluster is selected and split into two partitions using the traditional BKM clustering, this reveals a set of clusters $S^{BKM}(l)=\{S_j^{BKM},\ 0\leqslant j\leqslant l-1\}$, $l=2,3,\ldots,k$. Concurrently,

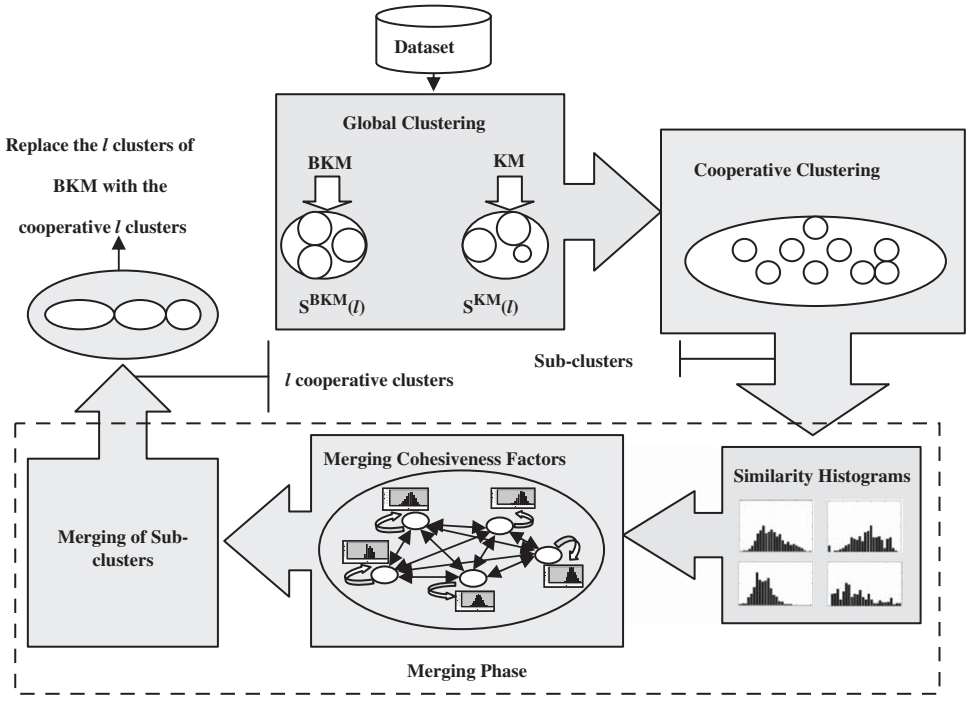

Fig. 3. Cooperative bisecting k-means (CBKM) at each level of the hierarchical tree.

the KM is performed which takes both the whole dataset X and the same number of clusters l as its inputs and generates a new set of l clusters $S^{KM}(l)=\{S_j^{KM},\ 0\leqslant j\leqslant l-1\}$.

3.2. Cooperative clustering phase

In the *cooperative clustering phase*, a cooperative contingency matrix (*CCM*) is constructed using the two sets $S^{KM}(l)$ and $S^{BKM}(l)$. The *CCM* is a two-dimensional matrix of size $l\times l$, where l is the number of clusters at any level of the binary tree, $l=2,3,\ldots,k$. Each element in the *CCM* is defined as: $CCM(S_i,S_j)=$ Number of objects from cluster S_i ($S_i\in S^{KM}(l)$) that belongs to cluster S_j ($S_j\in S^{BKM}(l)$), $i=1,2,\ldots,l$ and $j=1,2,\ldots,l$. This clustering–mapping identifies the set of disjoint sub-clusters, $Sb=\{Sb_i\}_{i=0}^{n_{sb}-1}$, generated by the KM that occurs in the bisecting clusterings (or vice versa). These sub-clusters act as agreement (intersection) between KM and BKM clusterings. The upper bound of number of sub-clusters, n_{sb}, is l^2 (the dimensions of the *CCM* matrix). This means $CCM(S_i,S_j)\neq 0\ \forall i=1,2,\ldots,l$ and $j=1,2,\ldots,l$, where l is the number of clusters. In this case, for each cluster S_i from the KM partitioning, all of its elements are distributed over the whole set of l clusters from the BKM partitioning (or vice versa).

3.3. Merging phase

In the merging phase of the CBKM, we need to obtain the same set of l clusters instead of the n_{sb} sub-clusters, such that the newly generated l clusters have better clustering quality than those of the individual KM and BKM clusterings, $S^{KM}(l)$ and $S^{BKM}(l)$, respectively. Thus, the generated n_{sb} sub-clusters are merged using the cooperative merging matrix (*CMM*). The *CMM* is symmetric matrix, so we need to store only the upper or lower elements of the matrix; the *CMM* is of space complexity of size $(n_{sb}(n_{sb}-1)/2)$. Each element in the *CMM* represents a merging factor between each two sub-clusters Sb_i, Sb_j from the set Sb. The merging factor is called the merging cohesiveness factor, $mcf(Sb_i,Sb_j)$. The *mcf* depends on a new representation of the pair-wise similarity within each sub-clusters, we will refer to this representation as a similarity histogram.

3.3.1. Similarity histogram

Each sub-cluster is represented as concise statistical representation called *similarity histogram* [9]. Similarity histogram H is a concise statistical representation of the set of pair-wise similarities distribution in a collection of objects. A number of bins in the histogram correspond to fixed similarity value intervals. The larger the number of bins the accurate approximation of the distribution of the pair-wise similarity in each sub-clusters. In this paper, we calculate the similarity between any pair of objects using the widely used *cosine* coefficient. Thus the similarity histogram is built over the interval $[-1,1]$ with fixed size of bins, *BinSize*. The number of bins in the histogram is *NumBins* (a user input parameter), thus *BinSize* equals $2/NumBins$. A coherent cluster should have high pair-wise similarities. A *typical* cluster has a histogram where the distribution of similarities is almost a normal distribution, while an *ideal* cluster would have a histogram where all similarities are of maximum values, and a *loose* similarity histogram is a histogram where similarities in the cluster are all of minimum values. A typical histogram of a sub-cluster with $NumBins=20$ is illustrated in Fig. 4. For a fixed bin size, *BinSize*, the $binId^{th}$ bin in the histogram contains the count of similarities that fall in the interval $](binId-(NumBins/2))*BinSize,\ \ (binId-(NumBins/2))*BinSize+BinSize]$. The first bin (i.e. bin with index $=0$) also contains similarities equal to -1. In general, the *Build-Histogram* algorithm is shown in Fig. 5.

3.3.2. Merging cohesiveness factor (mcf)

Based on the fact that a coherent cluster should have a similarity histogram where most similarities fall close to the maximum range of the similarity interval; while a loose cluster will have most similarities lie on the minimum range of the similarity interval, each pair of sub-clusters Sb_i and Sb_j are assigned a cohesiveness merging factor, $mcf(Sb_i,Sb_j)$. This factor represents the coherency (quality) of merging them into a new coherent cluster. The quality of merging

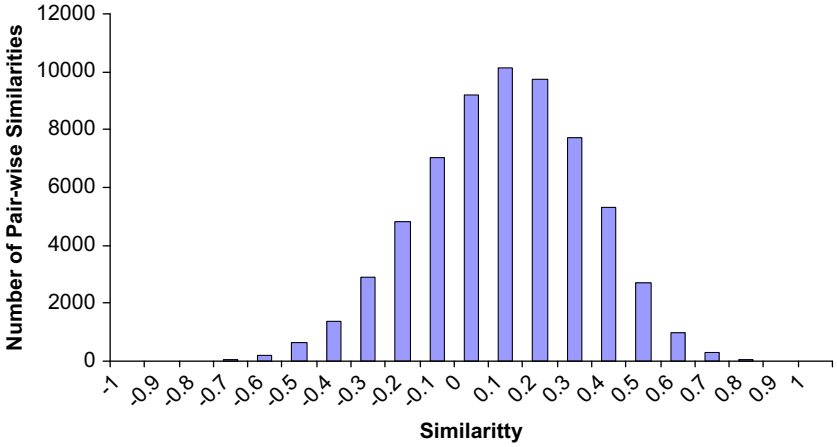

Fig. 4. Similarity histogram of a sub-cluster ($NumBins = 20$).

Algorithm: Build-Histogram (Sb_i, NumBins, SM)

Input: Sub-cluster Sb_i, number of bins *NumBins*, and the Similarity Matrix *SM*.

Output: Similarity Histogram H_i of size *NumBins*

Initializations: Let $H_i(bin)$=0, *bin*=0,1..,*NumBins*-1

Begin

For each pair of objects *x* and *y* in Sb_i

$Sim(\boldsymbol{x},\boldsymbol{y}) = SM(\boldsymbol{x},\boldsymbol{y})$

If ($Sim(\boldsymbol{x},\boldsymbol{y})$=-1) then *binId*=0

Else If($Sim(\boldsymbol{x},\boldsymbol{y})$=1) then *binId*=*NumBins*-1

Else *binId*=-1+ $\lceil Sim(\boldsymbol{x},\boldsymbol{y}) / BinSize \rceil$ + (*NumBins*/2)

Increment $H_i(binId)$ by one

End

Return H_i

End

Fig. 5. Build-histogram.

two sub-clusters is calculated by the coherency of merging the corresponding histograms. We assume that number of bins is the same in each sub-cluster's histogram. The process of merging two sub-clusters reveals a new histogram; this histogram is constructed by adding the corresponding counts of each bin from the two merged histograms, and also by adding the additional pair-wise similarities that are generated as a result of merging the two sub-clusters together that were not calculated in each histogram. The new histogram is constructed as follows:

$$H_{ij}(bin)=\begin{pmatrix}(H_i(bin)+H_j(bin)+|Sim(\boldsymbol{x},\boldsymbol{y})|), \forall \boldsymbol{x}\in Sb_i, \boldsymbol{y}\in Sb_j,\\ bin=0,1,\ldots,NumBins-1\\ \text{Such that}\{((bin-(NumBins/2))*BinSize)<Sim(\boldsymbol{x},\boldsymbol{y})\\ \leqslant((bin-(NumBins/2))*BinSize+BinSize)\}\end{pmatrix} \quad (3)$$

where H_{ij} is the histogram of the new cluster, $H_i(bin)$ is the bin^{th} bin of the similarity histogram H_i, and $|Sim(\boldsymbol{x},\boldsymbol{y})|$ refers to the number of the additional pair-wise similarities due to the merging.

Let $|Sb_i|$, $|Sb_j|$ be the number of objects in sub-clusters Sb_i, Sb_j, respectively. The number of pair-wise similarities in each sub-cluster Sb_i and Sb_j, are $n_{sim}(Sb_i)=|Sb_i|*(|Sb_i|-1)/2$, and $n_{sim}(Sb_j)=|Sb_j|*(|Sb_j|-1)/2$, respectively. The number of additional similarities of merging the two sub-clusters together is $n_{sim}(Sb_i,Sb_j)=(|Sb_i|+|Sb_j|)*(|Sb_i|+|Sb_j|-1)/2$. An *mcf* between any two sub-clusters is computed by calculating the ratio of the count of similarities weighted by the bin similarity above a certain similarity threshold δ to the total count of similarities in the new merged histogram. The higher this ratio, the more cohesive the new generated cluster. The *mcf*, is calculated by the following formula:

$$mcf(Sb_i,Sb_j) = \frac{\sum_{bin=binThreshold}^{numBins-1}(((bin*binSize)-1+(binSize/2))*H_{ij}(bin))}{n_{Sim}(Sb_i,Sb_j)} \quad (4)$$

where *binThreshold* is the bin corresponding to the similarity threshold δ.

Finally, assume initially that each sub-cluster is a cluster by itself. The most similar two sub-clusters (sub-clusters with the maximum value of the *mcf* in the *CMM* matrix) are merged into a new cluster and then the *CMM* is updated based on the new generated cluster.

Algorithm: Cooperative Bisecting *k*-means: CBKM (X,ITER,ζ, *k*,*SM*, *NumBins*)

Input: the dataset X, ITER for the bisecting step, homogeneity criterion ζ, number of clusters k, similarity matrix SM, and number of bins in the histogram, $NumBins$.

Output: set of k clusters, $S=\{S_0,S_2,..,S_{k-1}\}$

Begin

For number of clusters l =2 to k

Phase 1: Global Clustering

$S^{cooperative}(l)=\{\}$, Synchronously generate the two sets $S^{KM}(l)$ and $S^{BKM}(l)$

Phase 2: Cooperative Clustering

Using the two sets $S^{KM}(l)$ and $S^{BKM}(l)$, the $CCM(S^{KM}(l),S^{BKM}(l))$ is constructed and a new set Sb of n_{sb} disjoint sub-clusters is generated .

Phase 3: Merging

*Step*1: for each sub-cluster $Sb_i \in Sb$, *Build-Histogram (Sb_i, NumBins, SM)*

*Step*2: Using the sub-cluster's histograms, each element in the CMM matrix, $mcf(Sb_i,Sb_j)$, is calculated. Initially, $S^{cooperative}(l)=Sb$

*Step*3: ***Repeat***

Merge two sub-clusters from the set $S^{cooperative}(l)$ with the highest similarity merging value (mcf) in CMM; reduce the number of cooperative clusters by one and update the CMM.

Until (number of cooperative clusters=l)

*Step*3: Replace the Bisecting clustering $S^{BKM}(l)$ with the $S^{cooperative}(l)$.

End

Return the final set of k desired clusters $S=S^{cooperative}(k)$

End

Fig. 6. The CBKM algorithm.

These steps are repeated until the number of l cooperative clusters is generated. Then, the BKM clustering's is replaced with the resulting cooperative set of l clusters. The cooperative step is repeated for each number of clusters $l=2,3,\ldots,k$ until the desired number of clusters k is obtained. Fig. 6 shows the CBKM algorithm.

The number of bins is associated with the desired optimum similarity threshold δ as we need to adjust number of bins to accommodate the input threshold. Evaluation of number of bins and the similarity threshold is done experimentally; future work involves theoretical computation of the optimum number of bins and the optimum similarity threshold that maximize the clustering quality. The *send* and *receive* operation between KM and BKM is facilitated using the message passing interface (MPI) routines [21]. The synchronous cooperation between the KM and the BKM attains better time performance than the hybrid model in [10] as there is no idle time wasted for a former algorithm to finish its clustering. In addition, BKM receives a new set of homogenous clusters to be bisected next at the next level; this cooperative strategy enables the BKM to enhance its clustering through intermediate cooperation with KM and to take the advantage of the concurrent execution.

3.4. CBKM complexity analysis

All the basic operations are assumed to have the same unit operation time. The computation complexity of KM is determined by: the number of objects (n), the dimension of each vector (d), the number of clusters (l), and the number of loops ($Lops$). At each loop, the computation complexity of the *calculation step* is dominated by the clustering criterion function J, which has $f(n,l,d)$ operations.

For the updating step, recalculating the centroids needs ld operations. Thus, a $(f(n,l,d)+ld)*Lops$ operations are needed for the KM clustering. For the cosine similarity, $f(n,k,d)=2nld+nl+nd$. The time complexity of the KM is

$$T^{KM}(l)=O(n*l*d*Lops) \tag{5}$$

For the BKM algorithm, let $ITER$ is the number of iterations for each bisecting step, which is usually specified in advance. In this paper, the largest remaining cluster is always split. The computation complexity of the BKM algorithm at each level of the hierarchical tree is determined by the size of the cluster S_j at each bisecting step $|S_j|$, the dimension of the vector (d), the number of clusters (l), the number of loops of KM in each bisecting step ($Loops$), and the number of iterations for each bisecting step ($ITER$). In the bisecting step, $f(|S_j|,2,d)$ operations are required for the calculation step, and $2d$ operations for the centroids updating step. The time complexity of the BKM at each level is

$$T^{BKM}(l)=O(|S_j|*d*Loops*ITER) \tag{6}$$

For the CBKM algorithm, there is an additional messaging cost MC due to sending and receiving operations. In the cooperation step,

constructing the *CCM* takes (n) operations where n is the total number of objects. In the merging phase:

- Building a histogram of a sub-cluster Sb_i needs $(|Sb_i|*(|Sb_i|-1))/2$ operations. Thus $\sum_{i=0}^{n_{sb}-1}|Sb_i|*(|Sb_i|-1)/2$ operations are required to construct the n_{sb} histograms, where $|Sb_i|$ is the size of the sub-cluster Sb_i.
- Calculating the *mcf* for each pair of sub-clusters Sb_i and Sb_j in takes $(NumBins\text{-}\delta)+|Sb_i|*|Sb_j|$ operations.
- Finding the two most homogenous sub-clusters to be merged generate a new cluster S_i is of order $O(n_{sb}^2)$ operations, $n_{sb}\leqslant l^2$.
- Updating the *CMM* with the new added cluster takes $(NumBins*\sum_{j=0}^{n_{sb}-3}|S_i|*|Sb_j|)$ operations.

Thus the merging phase is of order $O(n_{sb}^2+|S_i|*|Sb_j|)\ \forall i,j=1,\ldots,n_{sb}\lll O(n^2)$ for small number of clusters l. The number of sub-clusters $n_{sb}\leqslant l^2$, $l\leqslant k$ and the size of each sub-cluster determine the cost of generating the *CMM* matrix.

The time complexity of the CBKM for l partitions at a given level of the hierarchal tree is computed as

$$T^{CBKM}(l)=O(Max(T^{KM}(l),T^{BKM}(l)))+O(MC_l+n+n_{sb}^2+|S_i|*|Sb_j|) \quad (7)$$

The time complexity of the CBKM is based on the algorithm with the maximum running time (KM or BKM) for each number of clusters $l=2,3,\ldots,k$, number of sub-clusters, and the size of each sub-cluster.

4. Clustering quality measures

The clustering results of any clustering algorithm should be evaluated using an informative quality measure(s) that reflects the "goodness" of the resulting clusters. External quality measures include *F*-measure, entropy, purity [9], and normalized mutual information (NMI) [22,23] are used, which assume that a prior knowledge about the data objects (i.e. class labels) is given. The separation index (SI) [24] is used as internal quality measure, which does not require a prior knowledge about the objects.

4.1. F-measure

F-measure combines the *precision* and *recall* ideas from the information retrieval literature. The precision and recall of a cluster S_j with respect to a class R_i, $i,j=1,2,\ldots,k$, are defined as

$$recall(R_i,S_j)=\frac{L_{ij}}{|R_i|} \quad (8)$$

$$precision(R_i,S_j)=\frac{L_{ij}}{|S_j|} \quad (9)$$

where L_{ij} is the number of objects of class R_i in cluster S_j, $|R_i|$ is the number of objects in class R_i, and $|S_j|$ is the number of objects in cluster S_j. The *F*-measure of a class R_i is defined as

$$F(R_i)=\max_j\frac{2*precision(R_i,S_j)*recall(R_i,S_j)}{precision(R_i,S_j)+recall(R_i,S_j)} \quad (10)$$

With respect to class R_i we consider the cluster with the highest *F*-measure to be the cluster S_j that is mapped to class R_i, and that *F*-measure becomes the score for class R_i. The overall *F*-measure for the clustering result of k clusters is the weighted average of the *F*-measure for each class R_i:

$$F\text{-}measure=\frac{\sum_{i=0}^{k-1}(|R_i|\times F(R_i))}{\sum_{i=0}^{k-1}|R_i|} \quad (11)$$

The higher the *F-measure*, the better the clustering due to the higher accuracy of the resulting clusters mapping to the original classes.

4.2. Entropy

Entropy tells us how *homogenous* a cluster is. Assume a partitioning result of a clustering algorithm consisting of k clusters. For every cluster S_j we compute pr_{ij}, the probability that a member of cluster S_j belongs to class R_i. The entropy of each cluster S_j is calculated as

$$E(S_j)=-\sum_{i=0}^{k-1}pr_{ij}\log(pr_{ij}),\quad j=0,1,\ldots,k-1 \quad (12)$$

The overall entropy for a set of k clusters is the sum of entropies for each cluster weighted by the size of each cluster:

$$entropy=\sum_{j=0}^{k-1}\left(\frac{|S_j|}{n}\right)\times E(S_j) \quad (13)$$

The lower the entropy, the more the *homogeneity* (or *similarity*) of objects within clusters.

4.3. Purity

The purity of a clustering solution is the average precision of the clusters relative to their best matching classes. For a single cluster S_j, purity is defined as the ratio of the number of objects in the dominant cluster to the total number of objects in the cluster:

$$P(S_j)=\frac{1}{|S_j|}\max_{i=0,1,\ldots k-1 j\neq i}(L_{ij}) \quad (14)$$

where L_{ij} is the number of objects from class R_i into cluster S_j, and $|S_j|$ is the number of objects in cluster S_j. To evaluate the total purity for the entire k clustering, the cluster-wise purities are weighted by the cluster size as and the average value is calculated as

$$purity(k)=\frac{1}{|n|}\sum_{j=0}^{k-1}\max_{i=0,1\ldots k-1,j\neq i}(L_{ij}) \quad (15)$$

Higher values of the purity measure indicate better partitioning of objects.

4.4. Normalized mutual information

A partition $S=\{S_0,S_1,\ldots,S_{k-1}\}$ describes a labeling of the n patterns in the dataset X, into k clusters. Taking frequency counts as approximations for probabilities, the entropy [13] of the data partition S is expressed as

$$H(S)=-\sum_{i=0}^{k-1}\frac{|S_i|}{n}\log\left(\frac{|S_i|}{n}\right) \quad (16)$$

where $|S_i|$ represents the number of objects in clusters S_i. The agreement between two partitions S and R is measured by the mutual information; $I(S;R)$, as proposed by Strehl and Ghosh [13]:

$$I(S;R)=\sum_{i=0}^{k-1}\sum_{j=0}^{k-1}\frac{L_{ij}}{n}\log\left(\frac{\frac{L_{ij}}{n}}{\frac{|S_i|}{n}.\frac{|R_j|}{n}}\right)=\sum_{i=0}^{k-1}\sum_{j=0}^{k-1}\frac{L_{ij}}{n}\log\left(\frac{L_{ij}*n}{|S_i|.|R_j|}\right) \quad (17)$$

where L_{ij} denoting the number of shared patterns between clusters $S_i\in S$ and $R_j\in R$.

From the definition of mutual information [23], it is easy to demonstrate that $I(S;R)\leqslant(H(S)+H(R))/2$. We define the NMI between two partitions S and R as

$$NMI(S;R)=\frac{2*I(S;R)}{H(S)+H(R)} \quad (18)$$

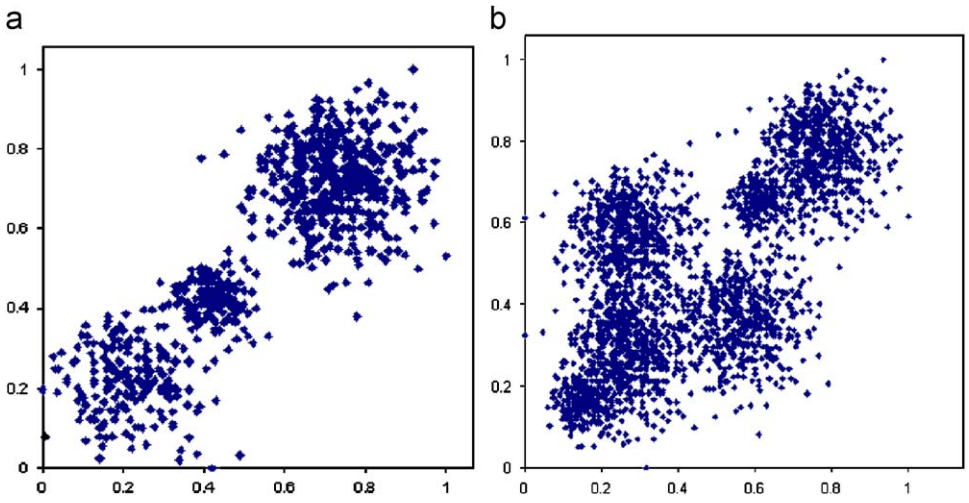

Fig. 7. (a) *DS* 1: three clusters and (b) *DS* 2: six clusters.

which after simplification, leads to

$$I(S;R)=\frac{(2/n)*\sum_{i=1}^{k}\sum_{j=1}^{k}L_{ij}\log\left(\frac{L_{ij}*n}{|S_i|.|R_j|}\right)}{\sum_{i=1}^{k}|S_i|\log(|S_i|/n)+\sum_{j=1}^{k}|R_j|\log(|R_j|/n)} \tag{19}$$

Note that $0\leqslant \mathrm{NMI}(S,R)\leqslant 1$.

4.5. Separation index (SI)

SI is a cluster validity measure that utilizes cluster centroids to measure the dissimilarity between clusters, as well as between points in a cluster to their respective cluster centroid. It is defined as the ratio of average within-cluster variance (cluster scatter) to the minimum pair-wise dissimilarity (measured by the *cosine correlation* measure) between clusters:

$$SI(k)=\frac{\sum_{i=1}^{k}\sum_{\forall \boldsymbol{x}_j\in S_i}1-CosSim(\boldsymbol{x}_j,c_i)}{n*\min_{r,s=1,\ldots,k,r\neq s}\{1-CosSim(c_r,c_s)\}} \tag{20}$$

The smaller the SI the more separate are the clusters. Thus, the smallest SI indeed indicates a valid optimal partition.

5. Experimental results

The proposed CBKM algorithm has been evaluated on nine datasets: two artificial datasets, four document datasets, and three gene expression datasets. The main measures of evaluation are the external and internal quality of the output clusters generated by the proposed algorithm. Another measure that was also evaluated is the total running time.

5.1. Artificial datasets

Two artificial datasets are used. The datasets are *DS1* and *DS2*. *DS1* is a two-dimensional datasets of 811 records, *DS1* forms three spherical-shaped clusters. *DS2* has six globular-shaped clusters of 2530 objects. Both *DS1* and *DS2* are available at [25] (Fig. 7).

5.2. Document datasets

In order to cluster documents, the type of the attributes (e.g. words or phrases) of the documents must be first chosen on which the clustering algorithm will be based and their representation. We used the *vector space model* (VSM) [26], which is the most common document representation model used in text mining; another document representations can be found in [9,27]. In VSM each document is represented by a vector $\boldsymbol{x}$, in the term space, $\boldsymbol{x}=[tf_1, tf_2 \ldots tf_d]$, where tf_i, $i=1,\ldots,d$ is the term frequency in the document, or the number of occurrences of the term t_i in a document $\boldsymbol{x}$. To represent every document with the same set of terms, we have to extract all the terms found in the documents and use them as our feature vector.[1] Sometimes another method is used which combines the term frequency with the inverse document frequency (*TF-IDF*) as used in [9]. The document frequency df_i is the number of documents in a collection of n documents in which the term t_i occurs. A typical inverse document frequency (*idf*) factor of this type is given by $\log(n/df_i)$. The *weight* of a term t_i in a document is given by

$$weight_i=tf_i\times\log\left(\frac{n}{df_i}\right) \tag{21}$$

Four documents datasets are used; the *UW*, *Yahoo*, *SN*, and 20*NG* document datasets. No feature selection is assumed for both the *UW* and *SN* dataset, but for the *Yahoo* and 20*NG* datasets features with term frequency equals one are eliminated. The words are tokenized in the *UW*, *Yahoo*, *SN*, and 20*NG* datasets in the following way:

1. Words consisting of numbers only are removed.
2. All words are converted to lower case letters.
3. Stop words are removed.
4. Words of length less than three are removed.

The *UW* dataset contains manually collected documents from the University of Waterloo[2] various web sites. The *UW* dataset was primarily used for the work presented in [9]. The *Yahoo* dataset is a collection of Reuter's news articles from the Yahoo! News website. The *Yahoo* dataset was used by Boley et al. [8,28].

The *SN* dataset is a dataset of 2371 metadata records collected from the Canada's SchoolNet[3] website. Specifically, the data were collected from the "Curriculum Area" of the web site. The 20*NG* is the standard 20-newsgroup dataset,[4] which contains 18,828 documents

[1] Obviously the dimensionality of the feature vector is always very high, in the range of hundreds and sometimes thousands.

[2] http://uwaterloo.ca

[3] http://schoolnet.ca

[4] http://people.csail.mit.edu/jrennie/20Newsgroups/

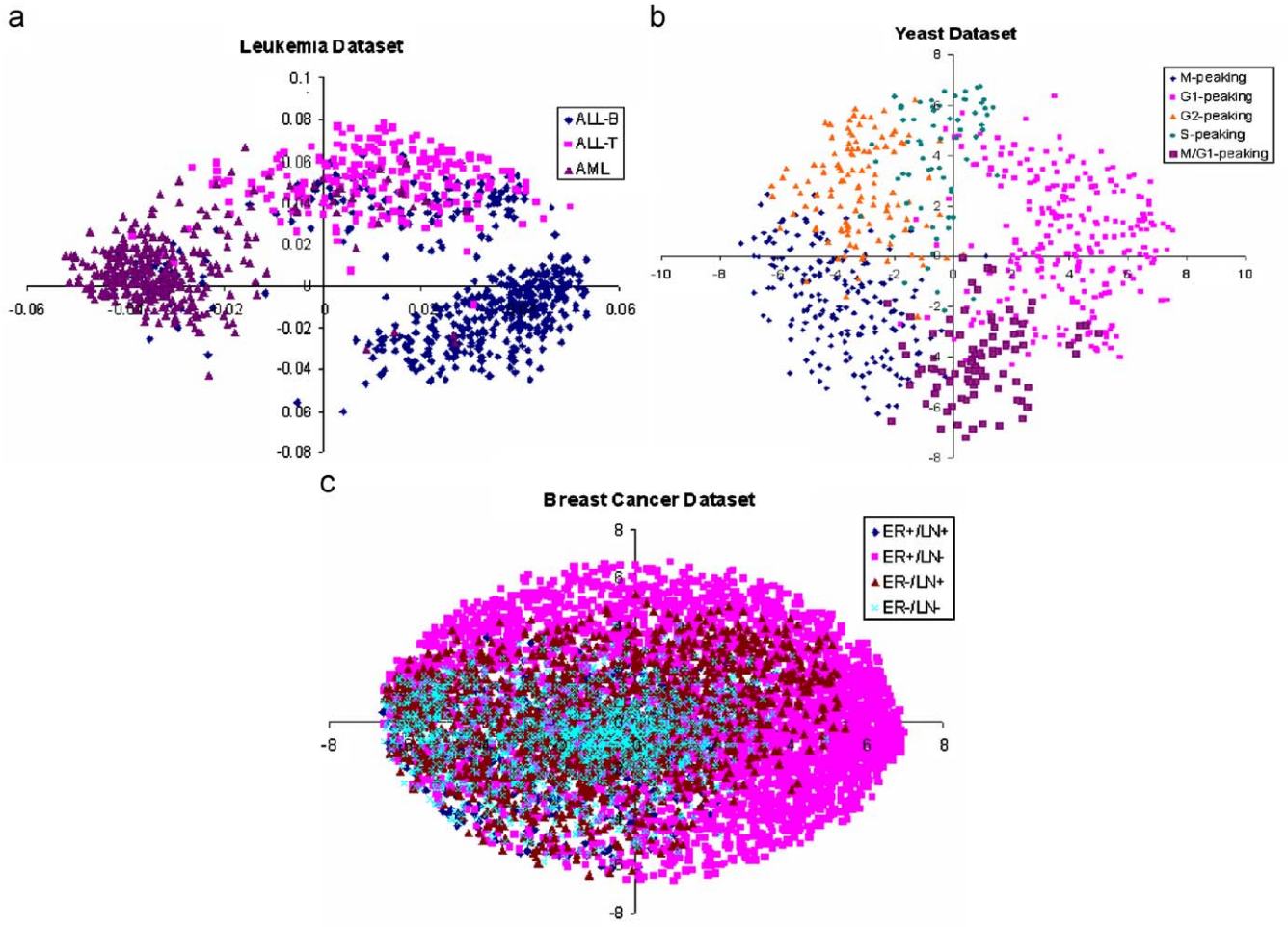

Fig. 8. Coefficients of SVD modes for (a) leukemia dataset (b) yeast dataset, and (c) breast cancer dataset.

Table 2
Summary of the datasets.

Dataset	*n*	*k*	*d*
DS 1	811	3	2
DS 2	2530	6	2
UW	314	10	15,134
SN	2371	17	7167
Yahoo	2340	20	28,298
20*NG*	18,828	20	91,652
Leukemia	999	3	38
Yeast	703	5	73
Breast cancer	7129	4	49

from 20 Usenet newsgroups divided into 20 balanced categories with an average of 151 words per document.

5.3. Gene expression datasets

Three gene expression datasets are used, the datasets are: *leukemia* dataset [29], *yeast* gene expression dataset [30], and *breast cancer* dataset [31]. The classification model (i.e. class labels) for both the leukemia and the breast cancer datasets is discovered using the same approach as in [32].

Leukemia dataset is an example of a non-temporal gene expression set. The dataset contains the expression of 999 genes along 38 samples obtained from ALL (27 samples) and AML (11 samples). Furthermore, the ALL samples are arranged in 18 B lineage and 9 T lineage samples. The order of samples along the dataset columns is: ALL B lineage, ALL T lineage and AML.

The yeast cell cycle time series dataset contains the expression levels of 6218 gene transcripts (identified as ORFs) measured at 10-min intervals over two cell cycles (160 min). The same filtering and data normalization procedures of [33] are applied resulting in a set of 703 genes. Based on the analysis conducted by Spellman et al. [34], the five main clusters are G1-peaking genes, S-peaking genes, G2-peaking genes, M-peaking genes, and M/G1-peaking genes.

The breast cancer dataset was primary used in the work done by [31]. The dataset contains 7129 gene expression and 49 tumors. The tumor samples are given labels ER+/LN+, ER+/LN−, ER−/LN+, and ER−/LN−.

Table 3
Performance of KM, BKM, SL, and CBKM for the *DS*1 ($k=3$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.6324 \pm (0.027)$, $t_1 = 9.63$, H_0 : *Rejected*	$0.5821 \pm (0.032)$, $t_2 = 13.93$, H_0 : *Rejected*	0.6210	$0.7209 \pm (0.031)$
Entropy ↓	$0.5324 \pm (0.015)$, $t_1 = 19.57$, H_0 : *Rejected*	$0.5854 \pm (0.035)$, $t_2 = 16.94$, H_0 : *Rejected*	0.5691	$0.4426 \pm (0.014)$
Purity ↑	$0.4825 \pm (0.039)$, $t_1 = 7.63$, H_0 : *Rejected*	$0.4123 \pm (0.027)$, $t_2 = 16.14$, H_0 : *Rejected*	0.4737	$0.5719 \pm (0.035)$
NMI ↑	$0.7382 \pm (0.018)$, $t_1 = 6.81$, H_0 : *Rejected*	$0.6948 \pm (0.029)$, $t_2 = 10.62$, H_0 : *Rejected*	0.7129	$0.7827 \pm (0.023)$
SI ↓	$0.2645 \pm (0.026)$, $t_1 = 7.62$, H_0 : *Rejected*	$0.2941 \pm (0.022)$, $t_2 = 12.35$, H_0 : *Rejected*	0.2738	$0.2042 \pm (0.024)$
Time (s) ↓	$0.1243 \pm (0.032)$, $t_1 = 3.83$, H_0 : *Rejected*	$0.1321 \pm (0.026)$, $t_2 = 3.34$, H_0 : *Rejected*	0.3246	$0.1607 \pm (0.028)$

Table 4
Performance of KM, BKM, SL, and CBKM for *DS*2, ($k=6$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.6854 \pm (0.041)$, $t_1 = 14.84$, H_0 : *Rejected*	$0.7428 \pm (0.027)$, $t_2 = 12.60$, H_0 : *Rejected*	0.7234	$0.8327 \pm (0.017)$
Entropy ↓	$0.4823 \pm (0.023)$, $t_1 = 14.56$, H_0 : *Rejected*	$0.4266 \pm (0.032)$, $t_2 = 5.16$, H_0 : *Rejected*	0.4555	$0.3830 \pm (0.020)$
Purity ↑	$0.5624 \pm (0.024)$, $t_1 = 14.11$, H_0 : *Rejected*	$0.6124 \pm (0.015)$, $t_2 = 9.72$, H_0 : *Rejected*	0.5798	$0.6912 \pm (0.033)$
NMI ↑	$0.6635 \pm (0.019)$, $t_1 = 13.79$, H_0 : *Rejected*	$0.7294 \pm (0.029)$, $t_1 = 5.42$, H_0 : *Rejected*	0.6984	$0.7836 \pm (0.034)$
SI ↓	$0.2234 \pm (0.019)$, $t_1 = 11.02$, H_0 : *Rejected*	$0.1845 \pm (0.031)$, $t_2 = 4.63$, H_0 : *Rejected*	0.2014	$0.1420 \pm (0.027)$
Time (s) ↓	$0.2843 \pm (0.039)$, $t_1 = 8.83$, H_0 : *Rejected*	$0.3114 \pm (0.042)$, $t_2 = 6.01$, H_0 : *Rejected*	0.5236	$0.3792 \pm (0.028)$

Table 5
Performance of KM, BKM, SL, and CBKM for the *UW dataset* ($k=10$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.6988 \pm (0.027)$, $t_1 = 14.69$, H_0 : *Rejected*	$0.7520 \pm (0.031)$, $t_2 = 8.92$, H_0 : *Rejected*	0.4996	$0.8467 \pm (0.036)$
Entropy ↓	$0.2579 \pm (0.022)$, $t_1 = 14.06$, H_0 : *Rejected*	$0.2281 \pm (0.024)$, $t_2 = 10.06$, H_0 : *Rejected*	0.4747	$0.1434 \pm (0.029)$
Purity ↑	$0.6879 \pm (0.031)$, $t_1 = 20.28$, H_0 : *Rejected*	$0.7334 \pm (0.024)$, $t_2 = 17.47$, H_0 : *Rejected*	0.4268	$0.8483 \pm (0.017)$
NMI ↑	$0.7914 \pm (0.027)$, $t_1 = 10.52$, H_0 : *Rejected*	$0.8402 \pm (0.032)$, $t_2 = 4.49$, H_0 : *Rejected*	0.4895	$0.9296 \pm (0.028)$
SI ↓	$1.6921 \pm (0.152)$, $t_1 = 14.52$, H_0 : *Rejected*	$1.3772 \pm (0.140)$, $t_2 = 7.81$, H_0 : *Rejected*	1.8249	$1.0446 \pm (0.129)$
Time (s) ↓	$15.9383 \pm (1.294)$, $t_1 = 29.56$, H_0 : *Rejected*	$21.7241 \pm (2.345)$, $t_2 = 10.60$, H_0 : *Rejected*	34.938	$28.087 \pm (1.305)$

Table 6
Performance of KM, BKM, SL, and CBKM for the *SN* dataset ($k=17$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.4829 \pm (0.029)$, $t_1 = 26.13$, H_0 : *Rejected*	$0.5281 \pm (0.037)$, $t_2 = 17.24$, H_0 : *Rejected*	0.6138	$0.6921 \pm (0.021)$
Entropy ↓	$0.3787 \pm (0.018)$, $t_1 = 14.26$, H_0 : *Rejected*	$0.3585 \pm (0.022)$, $t_2 = 9.87$, H_0 : *Rejected*	0.3193	$0.2929 \pm (0.020)$
Purity ↑	$0.6449 \pm (0.025)$, $t_1 = 19.89$, H_0 : *Rejected*	$0.6867 \pm (0.024)$, $t_2 = 14.30$, H_0 : *Rejected*	0.7329	$0.7846 \pm (0.019)$
NMI ↑	$0.4238 \pm (0.019)$, $t_1 = 27.56$, H_0 : *Rejected*	$0.4891 \pm (0.023)$, $t_2 = 17.42$, H_0 : *Rejected*	0.5447	$0.6273 \pm (0.027)$
SI ↓	$1.7441 \pm (0.215)$, $t_1 = 16.66$, H_0 : *Rejected*	$1.2876 \pm (0.146)$, $t_2 = 10.77$, H_0 : *Rejected*	1.0218	$0.8653 \pm (0.097)$
Time (s) ↓	$38.1973 \pm (2.479)$, $t_1 = 32.56$, H_0 : *Rejected*	$56.4298 \pm (3.184)$, $t_2 = 7.65$, H_0 : *Rejected*	72.92	$63.2394 \pm (2.384)$

Table 7
Performance of KM, BKM, SL, and CBKM for the *Yahoo* dataset ($k=20$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.4585 \pm (0.011)$, $t_1 = 29.81$, H_0 : *Rejected*	$0.5501 \pm (0.025)$, $t_2 = 14.34$, H_0 : *Rejected*	0.5919	$0.6778 \pm (0.031)$
Entropy ↓	$0.3815 \pm (0.031)$, $t_1 = 19.79$, H_0 : *Rejected*	$0.3128 \pm (0.029)$, $t_2 = 12.35$, H_0 : *Rejected*	0.2837	$0.2106 \pm (0.023)$
Purity ↑	$0.6192 \pm (0.044)$, $t_1 = 23.16$, H_0 : *Rejected*	$0.7171 \pm (0.028)$, $t_2 = 19.24$, H_0 : *Rejected*	0.7629	$0.9237 \pm (0.039)$
NMI ↑	$0.4920 \pm (0.027)$, $t_1 = 21.69$, H_0 : *Rejected*	$0.5482 \pm (0.021)$, $t_2 = 16.36$, H_0 : *Rejected*	0.5839	$0.6494 \pm (0.018)$
SI ↓	$2.3246 \pm (0.134)$, $t_1 = 35.13$, H_0 : *Rejected*	$1.6641 \pm (0.089)$, $t_2 = 24.41$, H_0 : *Rejected*	1.2938	$1.0358 \pm (0.073)$
Time (s) ↓	$28.3733 \pm (1.923)$, $t_1 = 12.00$, H_0 : *Rejected*	$31.9359 \pm (2.883)$, $t_2 = 6.83$, H_0 : *Rejected*	45.293	$38.6125 \pm (3.291)$

Table 8
Performance of KM, BKM, SL, and CBKM for the 20*NG* dataset ($k=20$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.4352 \pm (0.011)$,$t_1 = 15.02$,H_0 : *Rejected*	$0.4167 \pm (0.018)$,$t_2 = 15.46$,H_0 : *Rejected*	0.4738	$0.4977 \pm (0.015)$
Entropy ↓	$0.6765 \pm (0.034)$,$t_1 = 17.75$,H_0 : *Rejected*	$0.7223 \pm (0.039)$,$t_2 = 18.74$,H_0 : *Rejected*	0.6219	$0.5135 \pm (0.023)$
Purity ↑	$0.5120 \pm (0.033)$,$t_1 = 17.56$,H_0 : *Rejected*	$0.4827 \pm (0.040)$,$t_2 = 18.17$,H_0 : *Rejected*	0.5428	$0.6723 \pm (0.024)$
NMI ↑	$0.4329 \pm (0.041)$,$t_1 = 14.50$,H_0 : *Rejected*	$0.3854 \pm (0.028)$,$t_2 = 24.91$,H_0 : *Rejected*	0.4691	$0.5838 \pm (0.022)$
SI ↓	$0.5436 \pm (0.029)$,$t_1 = 16.36$,H_0 : *Rejected*	$0.6472 \pm (0.032)$,$t_2 = 26.55$,H_0 : *Rejected*	0.4474	$0.3986 \pm (0.027)$
Time (s) ↓	$619.1264 \pm (12.33)$, $t_1 = 50.11$, H_0 : *Rejected*	$767.8723 \pm (14.29)$, $t_2 = 14.14$, H_0 : *Rejected*	986.924	$831.014 \pm (13.94)$

Table 9
Performance of KM, BKM, SL, and CBKM for the *leukemia* dataset ($k=3$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.8366 \pm (0.021)$, $t_1 = 14.09$, H_0 : *Rejected*	$0.8073 \pm (0.022)$, $t_2 = 17.75$, H_0 : *Rejected*	0.8301	$0.9395 \pm (0.025)$
Entropy ↓	$0.4086 \pm (0.032)$, t_1=8.33 H_0 : *Rejected*	$0.4738 \pm (0.011)$, t_2=20.99 H_0 : *Rejected*	0.4264	$0.3282 \pm (0.029)$
Purity ↑	$0.8328 \pm (0.036)$ $t_1 = 4.35$, H_0 : *Rejected*	$0.8048 \pm (0.028)$ $t_2 = 8.58$, H_0 : *Rejected*	0.8258	$0.8744 \pm (0.023)$
NMI ↑	$0.6026 \pm (0.035)$, $t_1 = 15.76$, H_0 : *Rejected*	$0.5022 \pm (0.031)$ $t_2 = 28.95$, H_0 : *Rejected*	0.5833	$0.7483 \pm (0.022)$
SI ↓	$0.3921 \pm (0.034)$, $t_1 = 10.02$, H_0 : *Rejected*	$0.4502 \pm (0.025)$, $t_2 = 15.91$, H_0 : *Rejected*	0.4121	$0.2639 \pm (0.046)$
Time (s) ↓	$0.1524 \pm (0.024)$, $t_1 = 20.87$, H_0 : *Rejected*	$0.2151 \pm (0.011)$, $t_2 = 16.47$, H_0 : *Rejected*	0.4575	$0.2897 \pm (0.017)$

Table 10
Performance of KM, BKM, SL, and CBKM for the *yeast* dataset ($k=5$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.6301 \pm (0.040)$, $t_1 = 15.31$, H_0 : *Rejected*	$0.6784 \pm (0.031)$, $t_2 = 13.14$, H_0 : *Rejected*	0.6218	$0.8742 \pm (0.059)$
Entropy ↓	$0.4351 \pm (0.032)$, $t_1 = 19.05$, H_0 : *Rejected*	$0.4136 \pm (0.024)$, $t_2 = 18.80$, H_0 : *Rejected*	0.4121	$0.2102 \pm (0.042)$
Purity ↑	$0.6715 \pm (0.031)$, $t_1 = 24.82$, H_0 : *Rejected*	$0.7496 \pm (0.026)$, $t_2 = 18.08$, H_0 : *Rejected*	0.6375	$0.9071 \pm (0.029)$
NMI ↑	$0.8209 \pm (0.010)$, $t_1 = 14.72$, H_0 : *Rejected*	$0.8652 \pm (0.013)$, $t_2 = 5.13$, H_0 : *Rejected*	0.7531	$0.9416 \pm (0.019)$
SI ↓	$1.6303 \pm (0.206)$, $t_1 = 21.07$, H_0 : *Rejected*	$1.2991 \pm (0.185)$, $t_2 = 15.76$, H_0 : *Rejected*	1.6477	$0.5856 \pm (0.082)$
Time (s) ↓	$0.3552 \pm (0.019)$, $t_1 = 29.02$, H_0 : *Rejected*	$0.4556 \pm (0.023)$, $t_2 = 13.99$, H_0 : *Rejected*	0.8654	$0.5642 \pm (0.026)$

Table 11
Performance of KM, BKM, SL, and CBKM for the *breast cancer* dataset ($k=4$).

	KM	BKM	SL	CBKM
F-measure ↑	$0.4271 \pm (0.011)$, $t_1 = 16.10$, H_0 : *Rejected*	$0.4355 \pm (0.020)$, $t_2 = 13.23$, H_0 : *Rejected*	0.4089	$0.5422 \pm (0.030)$
Entropy ↓	$0.8031 \pm (0.032)$, $t_1 = 16.94$, H_0 : *Rejected*	$0.7948 \pm (0.041)$, $t_2 = 13.41$, H_0 : *Rejected*	0.8328	$0.6538 \pm (0.023)$
Purity ↑	$0.5734 \pm (0.048)$, $t_1 = 13.81$, H_0 : *Rejected*	$0.5825 \pm (0.036)$, $t_2 = 16.49$, H_0 : *Rejected*	0.5312	$0.7329 \pm (0.019)$
NMI ↑	$0.4537 \pm (0.022)$, $t_1 = 12.64$, H_0 : *Rejected*	$0.4793 \pm (0.017)$, $t_2 = 9.99$, H_0 : *Rejected*	0.4129	$0.5397 \pm (0.021)$
SI ↓	$0.7578 \pm (0.024)$, $t_1 = 25.16$, H_0 : *Rejected*	$0.7363 \pm (0.038)$, $t_2 = 18.65$, H_0 : *Rejected*	0.8332	$0.5236 \pm (0.034)$
Time (s) ↓	$0.5423 \pm (0.092)$, $t_1 = 31.48$, H_0 : *Rejected*	$1.8131 \pm (0.107)$, $t_2 = 9.33$, H_0 : *Rejected*	3.7143	$2.3652 \pm (0.242)$

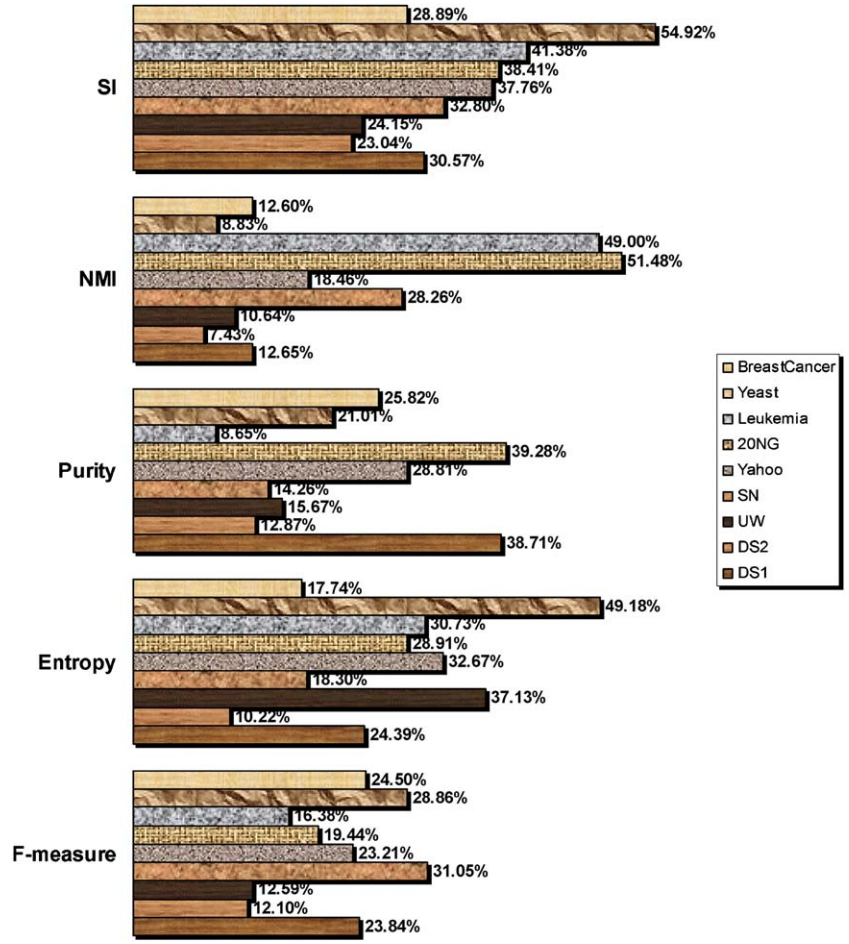

Fig. 9. Percentage of improvement in the clustering quality using CBKM compared to the traditional BKM.

The singular value decomposition (SVD) representation of each gene expression dataset is illustrated in Fig. 8. Table 2 summarizes the adopted artificial, documents, and gene expression datasets.

5.4. Significance of results

In order to illustrate the significance of the obtained results, the *t-test* is used to determine if there are significant differences among the mean values of each measure. The *null hypothesis* H_0 is "no significance difference". Within 95% confidence interval, the critical value of t, $t_{critical}$, is 2.024 at degree of freedom ($df = 38$). If the calculated $t < t_{critical}$ then the null hypothesis is accepted otherwise it is rejected and that means there is a significance difference between the results.

5.5. CBKM performance evaluation

Tables 3–11 present the clustering quality and time performance of 20 runs of KM, BKM, and the CBKM algorithm for the nine datasets. The quality and time performances of the CBKM algorithm are compared with those of KM, BKM, and hierarchical SL [1], algorithms using both the external and internal quality measures defined in Section 4. The value of the similarity threshold $\delta \in [0.1, 0.25]$ for the gene expression datasets and $\delta \in [0.2, 0.3]$ for the document datasets. Let q be any comparison measure, (e.g. *F*-measure, entropy, or time), in each table:

- $\bar{q}$: The average value of the variable q over 20 run.
- $\pm sd$: The standard deviation of the variable q over the calculated 20 runs.
- t_1: The t-test value between the results of the CBKM and the results of KM.
- t_2: The t-test value between the results of the CBKM and the results of BKM.

The SL hierarchical algorithm generates one solution, as it does not depend on any randomization, thus no average value is taken. In each table, t_1 and $t_2 > 2.024$, thus the null hypothesis H_0 is rejected and that means the obtained results from KM and BKM are significantly different from those obtained from the CBKM using the t-test values, t_1, t_2, respectively. Tables 3–11 illustrate that the CBKM outperforms KM, BKM, and SL measured by higher values for *F*-measure, purity, and NMI measures and lower value for entropy and SI. The main reason for this enhancement is that, the clustering solutions to KM and BKM are merged together into a new set of homogenous clusters with higher homogeneity between their original sub-clusters. Thus the CBKM takes the advantages of each of the clustering algorithms using intermediate cooperation and tries to overcome the deficiency of the traditional BKM that leaves portions of the dataset at each level of the tree with no way of re-clustering it again. In addition, the new set of cooperative clusters at a given level of the tree is fed back to the CBKM (as input to the next level of the tree) to be further clustered, which enables further partitioning of more homogeneous clusters. Fig. 9 shows the percentage of improvement in the clustering quality using the CBKM compared to that of the traditional BKM.

We can see that the CBKM achieves improvement in *F*-measure (up to 31% for the *SN* dataset), entropy (up to 49% for the *yeast* dataset), purity (up to 39% for the 20*NG* dataset), NMI (up to 51% for the 20*NG* dataset), and SI (up to 55% for the *yeast* dataset).

The CBKM takes slightly more time than the algorithm with the maximum running time, KM or BKM. For example, for the 20*NG* dataset, the CBKM has a penalty of only 8.34% increase in time than the BKM approach, which is much lower than the time complexity taken by the hybrid approach. Also in the *leukemia dataset*, an increase of time of up to 34% compared to KM and BKM, which is still lower than the time taken by the hybrid approaches, in which the total time complexity is the sum of the computational time of KM and BKM.

We also can notice that the clustering quality of the CBKM is better than that of the SL algorithm with much lower time complexity. Thus the intermediate cooperation between KM and BKM provides clustering solutions of better homogeneity than the individual approaches and even better than the hierarchical SL clustering with comparable time performance.

6. Conclusions

In this work, the CBKM was presented targeting better clustering quality and time performances than the non-cooperative KM and BKM algorithms. The cooperative algorithm is based on intermediate cooperation at each level of the hierarchical tree between the KM and the bisecting clustering algorithms.

The CBKM was effectively applied for a number of artificial datasets, document datasets as well as a number of gene-expression datasets. The proposed algorithm was able to group the co-related objects into the same cluster together into more homogenous groups based on combing the clustering solutions of both the KM and BKM clustering at the intermediate levels of the hierarchical tree. It achieves a superior improvement in the *F*-measure (up to 31%), entropy (up to 49%), purity (up to 39%), NMI (up to 51%), and SI (up to 55%), whereas the accompanying time penalty was as low as of 34%. Thus, the CBKM was utilized with improved clustering quality than the non-cooperative algorithms. Future work includes developing a distributed version of the CBKM algorithm to be applied on distributed datasets.

References

[1] A. Jain, M. Murty, P. Flynn, Data clustering: a review, ACM Computing Surveys 31 (1999) 264–323.
[2] R. Xu, Survey of clustering algorithms, IEEE Transactions on Neural Networks 16 (3) (2005) 645–678.
[3] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, in: Proceeding of the KDD Workshop on Text Mining, 2000, pp. 109–110.
[4] R. Duda, P. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[5] J. Hartigan, M. Wong, A *k*-means clustering algorithm, Applied Statistics 28 (1979) 100–108.
[6] J. Bezdek, R. Ehrlich, W. Full, The fuzzy C-means clustering algorithm, Computers and Geosciences 10 (1984) 191–203.
[7] S.M. Savaresi, D. Boley, On the performance of bisecting *k*-means and PDDP, in: Proceedings of the 1st SIAM International Conference on Data Mining, 2001, pp. 1–14.
[8] D. Boley, Principal direction divisive partitioning, Data Mining and Knowledge Discovery 2 (4) (1998) 325–344.
[9] K. Hammouda, M. Kamel, Collaborative document clustering, in: 2006 SIAM Conference on Data Mining (SDM06), 2006, pp. 453–463.
[10] Y. Zhao, G. Karypis, Criterion functions for document clustering: experiments and analysis, Technical Report, 2002.
[11] R. Kashef, M.S. Kamel, Cooperative partitional-divisive clustering and its application in gene expression analysis, in: IEEE 7th International Conference on BioInformatics and BioEngineering (BIBE07), 2007, pp. 116–122.
[12] G. Salton, Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley, Reading, MA, 1989.
[13] A. Strehl, J. Ghosh, Cluster ensembles–knowledge reuse framework for combining partitionings, Conference on Artificial Intelligence (AAAI 2002), AAAI/MIT Press, Cambridge, MA, 2002, pp. 93–98.
[14] Y. Qian, C. Suen, Clustering combination method, in: International Conference on Pattern Recognition, ICPR 2000, vol. 2, 2000, pp. 732–735.
[15] D. Greene, P. Cunningham, Efficient ensemble methods for document clustering, Technical Report, Trinity College Dublin, Computer Science Department, 2006.
[16] H. Ayad, M. Kamel, Cumulative voting consensus method for partitions with variable number of clusters, IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society Digital Library, IEEE Computer Society, 2007.
[17] C. Lin, M. Chen, Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging, IEEE Transactions on Knowledge and Data Engineering 17 (2) (2005) 145–159.
[18] Y. Eng, C. Kwoh, Z. Zhou, On the two-level hybrid clustering algorithm, in: AISAT04—International Conference on Artificial Intelligence in Science and Technology, 2004, pp. 138–142.

[19] S. Xu, J. Zhang, A hybrid parallel web document clustering algorithm and its performance study, Journal of Supercomputing 30 (2) (2004) 117–131.
[20] M. Ismail, M. Kamel, Multidimensional data clustering utilizing hybrid search strategies, Pattern Recognition 22 (1989) 75–89.
[21] W. Gropp, E. Lusk, A. Skjellum, Using MPI, Portable Parallel Programming with Message Passing Interface, The MIT Press, Cambridge, MA, 1996.
[22] J. Pluim, J. Maintz, M. Viergever, Mutual information based registration of medical images: a survey, IEEE Transaction on Medical Imaging, 2003.
[23] T. Cover, J. Thomas, Elements of Information Theory, Wiley, New York, 1991.
[24] U. Maulik, S. Bandyopadhyay, Performance evaluation of some clustering algorithms and validity indices, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (12) (2002) 1650–1654.
[25] ⟨http://webmining.spd.louisville.edu/NSF_Career/datasets.htm⟩.
[26] G. Salton, A. Wong, C. Yang, A vector space model for automatic indexing, Communications of the ACM 18 (11) (1975) 613–620.
[27] D. Cai, X. He, J. Han, Document clustering using locality preserving indexing, IEEE Transactions on Knowledge and Data Engineering 17 (12) (2005) 1624–1637.
[28] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, J. Moore, Partitioning-based clustering for web document categorization, Decision Support Systems 27 (1999) 329–341.
[29] S. Monti, P. Tamayo, J. Mesirov, T. Golub, Consensus Clustering: A Resampling-based Method for Class Discovery and Visualization of Gene Expression Microarray Data, Kluwer Academic Publishers, Dordrecht, 2003.
[30] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, G. Church, Systematic determination of genetic network architecture, Nature Genetics 22 (1999) 281–285.
[31] M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. Olson, J. Marks, J. Nevins, Predicting the clinical status of human breast cancer by using gene expression profiles, Proceedings of the National Academy of Sciences of the United States of America 98 (2001) 11462–11467.
[32] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M.A. Caligiuri, C. Bloomfield, E. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 (5439) (1999) 531–537.
[33] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, T. Golub, Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation, Proceedings of the National Academy of Sciences of the United States of America 96 (1999) 2907–2912.
[34] M. Eisen, P. Spellman, P. Brown, D. Botstein, Cluster analysis and display of genome-wide expression patterns, Proceedings of the National Academy of Sciences of the United States of America 95 (25) (1998) 14863–14868.

About the Author—RASHA KASHEF received the B.Sc. degree in Computer Engineering from the Faculty of Engineering, Alexandria University, Egypt, in 2001. She received the M.A.Sc. degree from the Department of Computer Engineering, Arab Academy for Science and Technology, Egypt, in 2004. From 2001 to 2005, she was with the Department of Computer Engineering, Faculty of Engineering, Arab Academy for Science and Technology, as a teaching assistant. She received her Ph.D. from the University of Waterloo, Department of Electrical and Computer Engineering in September 2008. Currently, she is hired as an assistant professor at the Arab Academy for Science and Technology. Her research interests are in data mining, especially cooperative clustering and distributed cooperative clustering.

About the Author—MOHAMED S. KAMEL received the Ph.D. degree in Computer Science from the University of Toronto, Canada. He is at present a professor and director of the Pattern Analysis and Machine Intelligence Laboratory at the Department of Systems Design Engineering, University of Waterloo, Canada. Dr. Kamel holds a Canada Research Chair in Cooperative Intelligent Systems. He has authored and coauthored more than 180 papers in journals and conference proceedings, two patents, and numerous technical and industrial project reports. Under his supervision, 44 Ph.D. and M.A.Sc. students have completed their degrees. Dr. Kamel is a member of the ACM, the AAAI, the CIPS, the APEO, and a senior member of the IEEE, and is editor-in-chief of the International Journal of Robotics and Automation, associate editor of four international journals, and guest editor for special issues in four journals. He is a member of the board of directors and cofounder of Virtek Vision International in Waterloo. He is a consultant to many companies including NCR, IBM, Nortel, VRP, and CSA.