Journal of Statistical Software

August 2010, Volume 36, Issue 9. *http://www.jstatsoft.org/*

LinkCluE: A MATLAB Package for Link-Based Cluster Ensembles

Natthakan Iam-on
Aberystwyth University

Simon Garrett
Aberystwyth University

Abstract

Cluster ensembles have emerged as a powerful meta-learning paradigm that provides improved accuracy and robustness by aggregating several input data clusterings. In particular, link-based similarity methods have recently been introduced with superior performance to the conventional co-association approach. This paper presents a MATLAB package, **LinkCluE**, that implements the link-based cluster ensemble framework. A variety of functional methods for evaluating clustering results, based on both internal and external criteria, are also provided. Additionally, the underlying algorithms together with the sample uses of the package with interesting real and synthetic datasets are demonstrated herein.

Keywords: clustering, cluster ensembles, pairwise similarity matrix, cluster relation, link analysis, MATLAB.

1. Introduction

Data clustering is a common task, which plays a crucial role in various application domains such as machine learning, data mining, information retrieval, pattern recognition and bioinformatics. Principally, clustering aims to categorize data into groups or clusters such that data in the same cluster are more similar to each other than to those in different clusters, with the underlying structure of real-world datasets containing a bewildering combination of shape, size and density. Although, a large number of clustering algorithms have been developed for several application areas (Jain *et al.* 1999), the "no free lunch" theorem (Wolpert and Macready 1995) suggests there is no single clustering algorithm that performs best for all datasets (Kuncheva and Hadjitodorov 2004), i.e., unable to discover all types of cluster shapes and structures presented in data (Duda *et al.* 2000; Fred and Jain 2005; Xue *et al.* 2009). Each algorithm has its own strengths and weaknesses. For any given dataset, it is usual

for different algorithms to provide distinct solutions. Indeed, apparent structural differences may occur within the same algorithm, given different parameters. As a result, it is extremely difficult for users to decide *a priori* which algorithm would be the *the most appropriate* for a given set of data.

Recently, the cluster ensemble approach has emerged as an effective solution that is able to overcome these problems. Cluster ensemble methods combine multiple clusterings of the same dataset to yield a single overall clustering. It has been found that such a practice can improve robustness, as well as the quality of clustering results. Thus, the main objective of cluster ensembles is to combine different decisions of various clustering algorithms in such a way to achieve the accuracy superior to those of individual clustering. Examples of well-known ensemble methods are: (i) the feature-based approach that transforms the problem of cluster ensembles to clustering categorical data, i.e., cluster labels (Boulis and Ostendorf 2004; Cristofor and Simovici 2002; Nguyen and Caruana 2007; Topchy *et al.* 2004, 2005), (ii) graph-based algorithms that employ a graph partitioning methodology (Domeniconi and Al-Razgan 2009; Fern and Brodley 2004; Iam-on *et al.* 2010; Strehl and Ghosh 2002), and (iii) the pairwise similarity approach that makes use of co-occurrence relationships between all pairs of data points (Ayad and Kamel 2003; Fern and Brodley 2003; Fred 2001; Fred and Jain 2002, 2003, 2005; Monti *et al.* 2003).

Of particular interest here is the pairwise similarity approach, in which the final data partition is derived based on relations amongst data points represented within the similarity matrix. This is widely known as the *co-association* (CO) matrix (Fred and Jain 2005). This particular matrix denotes co-occurrence statistics between each pair of data points, especially in term of the proportion of base clusterings in which they are assigned to the same cluster. In essence, the CO matrix can be regarded as a new similarity matrix, which is superior to the original distance based counterpart (Jain and Law 2005). It has been wildly applied to various application domains such as gene expression data analysis (Monti *et al.* 2003; Swift *et al.* 2004) and satellite image analysis (Kyrgyzov *et al.* 2007).

This approach has gained popularity and become a practical alternative mainly due to its simplicity. However, it has been criticized because the underlying matrix only considers the similarity of data points at coarse level and completely ignores those existing amongst clusters (Fern and Brodley 2004; Iam-on *et al.* 2008). As a result, by not exploiting available information regarding cluster associations, many relations are unknown, and yet are assigned a similarity value of zero. For this reason, the authors introduced new methods for generating two link-based pairwise similarity matrices, named *connected-triple-based similarity* (CTS) and *SimRank-based similarity* (SRS) matrices, respectively (Iam-on *et al.* 2008). Both methods work on the basic conjecture of taking into consideration as much information, embedded in a cluster ensemble, as possible when finding the similarity between data points. To discover similarity values, they consider both the associations among data points as well as those among clusters in the ensemble using link-based similarity measures (Calado *et al.* 2006; Jeh and Widom 2002; Klink *et al.* 2006). Figure 1 demonstrates the effectiveness of the link-based ensemble approach over the gene expression data of leukemia patients (Armstrong *et al.* 2002). In particular, the link-based ensemble approach can discover clusters (i.e., groups of patients) more accurately than several clustering techniques (SL: single-linkage, CL: complete-linkage, AL: average linkage and k-means) usually used by bioinformaticians.

This paper presents the **LinkCluE** package, which implements the aforementioned link-based methods (both established methods and our own improvements) for solving the cluster ensem-

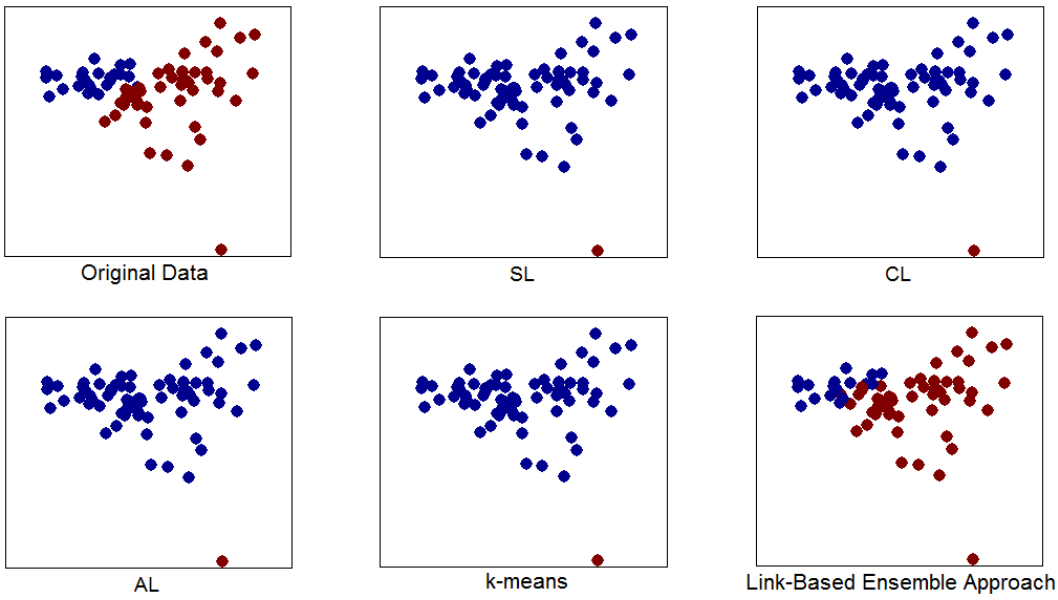


Figure 1: Clusters discovered by different clustering algorithms on the gene expression data of leukemia patients. Note that true clusters are shown by two colors of red and blue, in the illustration of 'Original Data'.

ble problems. It is developed using the mathematical software MATLAB (The MathWorks, Inc. 2007). In addition to the implementation of the link-based similarity algorithms, the package also provides a number of useful functions (by making use of the built-in functions in MATLAB) for other phases in the cluster ensemble framework. A variety of evaluation measures, based on both internal and external criteria, are also offered for assessing the quality of clustering results. The rest of this paper is organized as follows. Section 2 presents a formal definition of the cluster ensemble problem and its general framework. Section 3 provides a review on the pairwise similarity approach using link-based similarity matrices. The package **LinkCluE** is thoroughly introduced in Section 4. Illustrative examples of exploiting the package with real and synthetic data are included in Section 5. The applicability of the package is discussed in Section 6, with perspective of further work and continued expansion of the package.

2. The cluster ensemble problem

2.1. Problem formulation and framework

Let $X = \{x_1, x_2, \ldots, x_N\}$ be a set of N data points and let $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ be a set of M base clustering results, which is referred to as a *cluster ensemble*. Each base clustering result (called an *ensemble member*) returns a set of clusters $\pi_i = \{C_1^i, C_2^i, \ldots, C_{k_i}^i\}$, such that $\bigcup_{j=1}^{k_i} C_j^i = X$, where k_i is the number of clusters in the i-th clustering. For each $x \in X$, $C(x)$ denotes the cluster label to which the data point x belongs. In the i-th clustering, $C(x) = j$ if $x \in C_j^i$. The problem is to find a new partition π^* of a data set X that summarizes the information from the cluster ensemble Π. The general framework of cluster ensembles is

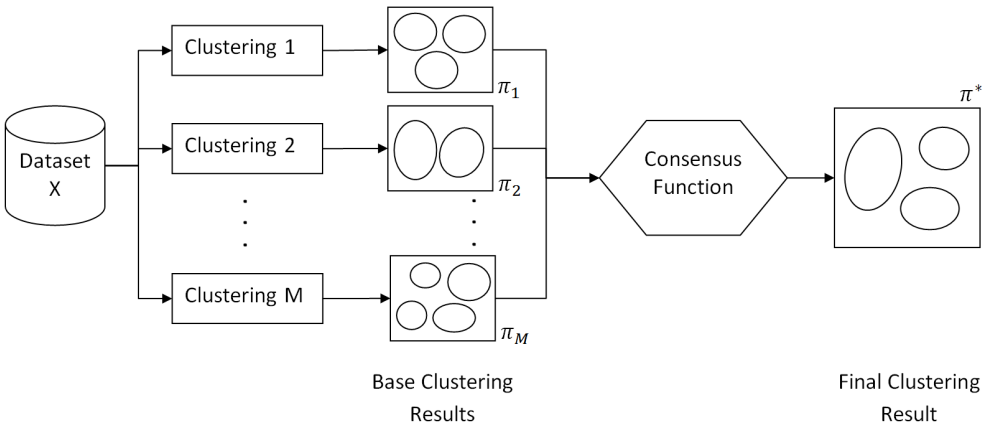


Figure 2: The basic process of cluster ensembles. It first applies multiple base clusterings to a dataset X to obtain diverse clustering decisions ($\pi_1 \dots \pi_M$). Then, these solutions are combined to establish the final clustering result (π^*) using a consensus function.

shown in Figure 2. Accordingly, multiple input clusterings, known as *ensemble members* or *base clusterings*, are intelligently aggregated to form a final data partition. There are two main stages of: (i) generating the cluster ensemble, and (ii) producing the final partition, which is normally referred to as a *consensus function*. See Hornik (2005) for the example of a cluster ensemble framework, with which the implementation in R has also been provided.

2.2. Cluster ensemble generation

It has been shown that ensembles are most effective when constructed from a set of predictors whose errors are distinct (Kittler *et al.* 1998). To a great extent, the diversity amongst ensemble members is introduced to enhance the result of an ensemble (Kuncheva and Vetrov 2006). This appears to be analogous to the Central Limit Theorem in which multiple samples that contain errors/randomness, when combined, reveal the true underlying distribution. Particularly to data clustering, the results obtained with any single algorithm (e.g., k-means Hochbaum and Shmoys 1985 and hierarchical clusterings Jain *et al.* 1999) over many iterations are typically very similar. In such a circumstance where all ensemble members agree on how a dataset should be partitioned, aggregating the base clustering results will show no improvement over any of the constituent members. Several approaches have been proposed to introduce artificial instabilities in clustering algorithms, hence the diversity within a cluster ensemble. The following ensemble generation methods yield different clusterings of the same data, by exploiting different cluster models and different data partitions.

- *Homogeneous ensembles*: Base clusterings are created using the repeated runs of a single clustering algorithm, each with a unique set of parameters. Following this, the k-means technique has often been employed with a random initialization of cluster centers (Fred and Jain 2002, 2003, 2005; Gionis *et al.* 2005; Iam-on *et al.* 2008; Topchy *et al.* 2004). An ensemble of k-means is computational efficient as its time complexity is $O(kNM)$, where k, N and M denote the number of clusters, the number of data points and the number of base clusterings, respectively. In fact, other non-deterministic clustering techniques (whose the results obtained from multiple runs are dissimilar) such as PAM and CLARA

(see Kaufman and Rousseeuw 1990 for details) can also be used to form homogeneous ensembles. However, as compared with k-means, the ensembles of PAM and CLARA are less efficient with time complexity being $O(Mk(N-k)^2)$ and $O(M(ks^2+k(N-k)))$, respectively. Note that s denotes the sample size ($s < N$). Unlike the aforementioned alternatives of base clustering, hierarchical clustering techniques (e.g., SL, CL and AL) are deterministic with the identical result being achieved from multiple runs for any given number of clusters, k. Hence, such methods can not generate diversity within a homogeneous ensemble.

- *Selection of k*: The output of any clustering algorithm is dependent on the initial choice of the number of clusters k. To acquire the ensemble diversity, base clusterings are created using randomly selected values of k from a pre-specified interval (see Iam-on *et al.* 2008 and Iam-on *et al.* 2010 for examples). Intuitively, k should be greater than the expected number of clusters and the common rule-of-thumb is $k = \sqrt{N}$ (Fred and Jain 2005; Hadjitodorov *et al.* 2006; Kuncheva and Vetrov 2006). This generation scheme allows a large number of clustering algorithms, both partitioning and hierarchical, to be used as base clusterings. However, k-means is still often employed for the efficiency reason. It is noteworthy that the time complexity of creating cluster ensembles with a hierarchical clustering technique being used as base clusterings is $O(N^2M)$.

- *Data subspacing/sampling*: Cluster ensembles can also be created by applying manifold subsets of initial data to base clusterings. It is intuitively assumed that each clustering algorithm can provide different levels of performance for different partitions of a dataset (Domeniconi and Al-Razgan 2009). Practically, data partitions are obtained by projecting data onto different subspaces (Fern and Brodley 2003; Topchy *et al.* 2003), choosing different subsets of features (Strehl and Ghosh 2002; Yu *et al.* 2007), or data sampling (Dudoit and Fridyand 2003; Fischer and Buhmann 2003; Minaei-Bidgoli *et al.* 2004).

- *Heterogeneous ensembles*: As an alternative, heterogeneous ensembles may be exploited, where the diversity is induced by allowing each base clustering to be generated using a different clustering algorithm (Ayad and Kamel 2003; Hu and Yoo 2004; Law *et al.* 2004).

In addition to using one of these methods, any combination of them can be applied as well (Domeniconi and Al-Razgan 2009; Fred and Jain 2006; Iam-on *et al.* 2008; Monti *et al.* 2003; Nguyen and Caruana 2007; Strehl and Ghosh 2002).

2.3. Consensus functions

Having obtained the cluster ensemble, a variety of *consensus functions* have been developed and made available for generating the ultimate data partition. In general, consensus methods found in the literature can be categorized into: (i) pairwise similarity, (ii) graph-based and (iii) feature-based approaches, respectively.

Pairwise similarity algorithm

This category of cluster ensemble method is based principally on the pairwise similarity amongst data points. In particular, given a dataset $X = \{x_1, x_2, \ldots, x_N\}$, it first gener-

ates a cluster ensemble $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ by applying M base clusterings to the dataset X. Following that, a $N \times N$ similarity matrix is constructed for each ensemble member, denoted as $S_m, m = 1 \ldots M$. Each entry in this matrix represents the relationship between two data points. If they are assigned to the same cluster, the entry will be 1, 0 otherwise. More precisely, the similarity between two data points $x_i, x_j \in X$ from the m-th ensemble member can be computed as follows:

$$S_m(x_i, x_j) = \begin{cases} 1 & \text{if } C(x_i) = C(x_j), \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In essence, M similarity matrices are merged to form a CO matrix (Fred and Jain 2005), various names found in the literature as consensus matrix (Monti *et al.* 2003), similarity matrix (Strehl and Ghosh 2002) or agreement matrix (Swift *et al.* 2004). Each element in the CO matrix represents the similarity degree between any two data points, which is a ratio of a number of ensemble members in which these data points are assigned to the same cluster to the total number of ensemble members. Formally, this similarity between $x_i, x_j \in X$ is defined as

$$CO(x_i, x_j) = \frac{1}{M} \sum_{m=1}^{M} S_m(x_i, x_j). \tag{2}$$

Since the CO matrix is a similarity matrix, any similarity-based clustering algorithm can be applied to this matrix to yield the final partition π^*. Among several existing similarity-based methods, the most well-known technique is agglomerative hierarchical clustering algorithm. Specifically, Fred and Jain (2003, 2005) employ the SL and AL agglomerative clusterings to generate the final partition.

Graph-based methods

The second methodology makes use of the graph representation to solve the cluster ensemble problem (Fern and Brodley 2004; Strehl and Ghosh 2002). Examples of well-known graph-based ensemble methods have been introduced in Strehl and Ghosh 2002 (as CSPA, HGPA and MCLA) and Fern and Brodley 2004 (as HBGF). Firstly, cluster-based similarity partitioning algorithm (CSPA) creates a similarity graph, where vertices represent data points and edges' weight represent similarity scores obtained from the CO matrix. Afterwards, a graph partitioning algorithm called METIS (Karypis and Kumar 1998) is used to partition the similarity graph into k clusters. Hyper-graph partitioning algorithm (HGPA) constructs a hyper-graph, where vertices represent data points and the same-weighted hyper-edges represent clusters in the ensemble. Then, the HMETIS technique (Karypis *et al.* 1999) is applied to partition the underlying hyper-graph into k parts with roughly of the same size.

In addition, meta-clustering algorithm (MCLA) generates a graph that represents the relationships among clusters in the ensemble. In this meta-level graph, each vertex corresponds to each cluster in the ensemble and each edge's weight between any two cluster vertices is computed using the binary Jaccard measure (i.e., the ratio of the intersection to the union of the sets of objects belonging to the two clusters). METIS is also employed to partition this meta-level graph into k meta-clusters. Effectively, each data point has a specific association degree to each meta-cluster. This can be estimated from the number of original clusters, to

which the data point belongs, in the underlying meta-cluster. The final clustering is produced by assigning each data point to the meta-cluster with which it is most frequently associated (i.e., with the highest association degree).

Unlike the previous methods, hybrid bipartite graph formulation (HBGF) exploits of the bipartite graph whose vertices represent both data points and clusters. There is no edge connecting vertices of the same object type, and the weight of an edge between any data point and cluster is 1 if the data point belongs to that cluster, 0 otherwise. The spectral graph partitioning algorithm of Ng *et al.* (2001) and METIS are exploited to obtain the final clustering from this graph.

Feature-based approach

The approach transforms the problem of cluster ensembles to clustering categorical data. Specifically, each base clustering provides a cluster label as a new feature describing each data point, which is utilized to formulate the ultimate solution (Boulis and Ostendorf 2004; Nguyen and Caruana 2007; Topchy *et al.* 2003, 2004). An example of such method is the iterative voting consensus (IVC) algorithm, which was recently introduced in (Nguyen and Caruana 2007). It aims to obtain the consensus partition π^* of data points X from the categorical data induced by a cluster ensemble $\Pi = \{\pi_1, \ldots, \pi_M\}$. Principally, it utilizes the feature vector $Y = \{y_1, y_2, \ldots, y_N\}$, with N denoting the number of data points and $y_i, i = 1 \ldots N$ being specified as

$$y_i = \{\pi_1(x_i), \ldots, \pi_M(x_i)\}, \tag{3}$$

where $\pi_g(x_i)$ represents a label of specific cluster in clustering $\pi_g, g = 1 \ldots M$, to which a data point x_i belongs. In each iteration, IVC first estimates the center of each cluster in π^*. Note that each cluster $C_j, j = 1 \ldots k$ in the target clustering π^* has a cluster center $center_j = \{mode(X_j, \pi_1), \ldots, mode(X_j, \pi_M)\}$, where $X_j \subset X$ is the set of data points belonging to the cluster C_j and $mode(X_j, \pi_g)$ denotes the majority labels (in the clustering π_g) of members of X_j. Having obtains these centers, IVC then reassigns each data point to its closest cluster center. This is possible using the Hamming distance between M-dimensional vectors that represent data points and cluster centers. The iterative process continues until there is no change with the target clustering π^*.

2.4. Evaluating the quality of the data partition

After acquiring the final data partition, its quality is typically assessed using different types of validity measure. One evaluation category includes so-called *internal* validity indices, which evaluate the goodness of a data partition using only quantities and features inherited from the dataset (Jain *et al.* 1999). They are usually employed for the task of class discovery, where true cluster labels are unknown. Examples of such measures are Compactness (Nguyen and Caruana 2007), Davies-Bouldin (Davies and Bouldin 1979) and Dunn (Dunn 1974). Unlike these data-characteristic-based validity indices, another family exploits a prior information of known data partition (Π') or cluster labels of the data. This is similar to the process of cross-validation that is used in evaluating machine learning methods. Given a dataset whose correct clusters are known, it is possible to assess how accurately a clustering method clusters the data relative to this correct clustering. Crucially, however, the clustering method

at no time has access to information about the correct clusters; they are only used to assess the clustering method's performance. This evaluation category includes a number of *external* validity indices such as classification accuracy (Nguyen and Caruana 2007), Rand index (Rand 1971) and adjusted Rand index (Campello 2007). These validity criteria assess the degree of agreement between two data partitions, where one of the partitions is obtained from a clustering algorithm (π^*) and the other is the known partition (Π'). They are usually employed for class prediction (Yu *et al.* 2007) and empirical comparison of different clustering techniques (Campello 2007; Fred and Jain 2005). In order to better understand these validity indices, their details are given below.

Compactness (CP)

It is one of the commonly used measurement criteria, which employ only the information inherent to the dataset. According the description given by Nguyen and Caruana (2007), CP measures the average distance between every pair of data points, which belong to the same cluster. More precisely, it is defined as

$$CP(\pi^*) = \frac{1}{N}\sum_{k=1}^{K} n_k \left(\frac{\sum_{x_i,x_j \in C_k} d(x_i, x_j)}{n_k(n_k-1)/2} \right), \tag{4}$$

where K denotes the number of clusters in the clustering result, n_k is the number of data points belonging to the k-th cluster, $d(x_i, x_j)$ is the distance between data points x_i and x_j, and N is the total number of data points in the dataset. Ideally, the members of each cluster should be as close to each other as possible. Thus, lower value of CP means better cluster configuration.

Davies-Bouldin (DB)

The DB index makes use of similarity measure R_{ij} between the clusters C_i and C_j, which is defined upon a measure of dispersion (s_i) of a cluster C_i and a dissimilarity measure between two clusters (d_{ij}). According to Davies and Bouldin (1979), R_{ij} is formulated as

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}, \tag{5}$$

where d_{ij} and s_i can be estimated by the following equations. Note that v_x denotes the center of cluster C_x and $|C_x|$ is the number of data points in cluster C_x.

$$d_{ij} = d(v_i, v_j), \tag{6}$$

$$s_i = \frac{1}{|C_i|} \sum_{\forall x \in C_i} d(x, v_i). \tag{7}$$

Following that, the DB index is defined as

$$DB(\pi^*) = \frac{1}{k}\sum_{i=1}^{k} R_i, \tag{8}$$

where $R_i = \max_{j=1...k, i \neq j} R_{ij}$.

The DB index measures the average of similarity between each cluster and its most similar one. As the clusters have to be compact and separated, the lower DB index indicates better goodness of a data partition.

Dunn

This validity index is introduced by Dunn (1974). Its purpose is to identify compact and well-separated clusters. For a given number of clusters K, the definition of the Dunn index is given by the following equation.

$$Dunn(\pi^*) = \min_{i=1...K} \left(\min_{j=i+1...K} \left(\frac{d(C_i, C_j)}{\max_{k=1...K}(\text{diam}(C_k))} \right) \right), \tag{9}$$

where $d(C_i, C_j)$ is the distance between two clusters C_i and C_j, which can be defined as

$$d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y). \tag{10}$$

In addition, $\text{diam}(C_i)$ is the diameter of a cluster C_i, which is defined as follows:

$$\text{diam}(C_i) = \max_{x,y \in C_i} d(x, y). \tag{11}$$

In a dataset containing compact and well-separated clusters, the distances between the clusters are expected to be large and the diameters of the clusters are expected to be small. Therefore, a large value of the Dunn index signifies compact and well-separated clusters.

Classification accuracy (CA)

It measures the number of correctly classified data points of a clustering solution compared with known class labels. To compute the CA, each cluster from the clustering result is relabeling with the *majority* cluster label, which most of data points in that cluster come from. Then the accuracy of the new labels is measured by counting the number of correctly labeled data points, in comparison to their known class labels, and dividing by the total number of data in the dataset.

Let m_i is the number of data points with the *majority* cluster label in cluster i, the CA can be regarded as the ratio of the number of correctly classified data points to the total number of data points in the dataset. According to the definition given by Nguyen and Caruana (2007), the CA is defined as

$$CA(\pi^*, \Pi') = \frac{\sum_{i=1}^{K}(m_i)}{N}, \tag{12}$$

where N is the total number of data in the dataset. The CA ranges from 0 to 1. If the clustering result takes value 1 of the CA, it denotes that all data points are clustered correctly and the clustering contains only pure clusters, i.e., each contains data points of the same cluster label.

Rand index (RI)

This validity measure takes into account the number of object pairs that exist in the same and different clusters. More formally, the RI (Rand 1971) can be defined by

$$RI(\pi^*, \Pi') = \frac{n_{11} + n_{00}}{n_{11} + n_{10} + n_{01} + n_{00}}, \tag{13}$$

where n_{11} is the number of pairs of data points that are in the same clusters in both partitions π^* and Π', n_{00} denotes the number of pairs of data points that are placed in the different clusters in both π^* and Π', n_{10} is the number of pairs of data points that belong to the same cluster in π^* but are in the different clusters in Π', and n_{01} indicates the number of pairs of data points that are put in the different clusters in π^* but are in the same cluster in Π'. Intuitively, n_{11} and n_{00} can be interpreted as the quantity of agreements between two partitions, while n_{10} and n_{01} are the number of disagreements. The RI has a value between 0 and 1, with the more the value approximates to 1 the higher the agreement is.

Adjusted Rand index (AR)

To correct the main criticisms of the Rand index, that is, its expected value is not zero when comparing random partitions (Jain and Dubes 1998), Hubert and Arabie (1985) introduce the adjusted Rand index (AR). According to notation denoting the Rand index, the AR index between partition π^* and Π' is defined by the following equation. Note that the higher the AR value is, the greater the agreement becomes.

$$AR(\pi^*, \Pi') = \frac{n_{11} - \frac{(n_{11}+n_{10})(n_{11}+n_{01})}{n_{00}}}{\frac{(n_{11}+n_{10})+(n_{11}+n_{01})}{2} - \frac{(n_{11}+n_{10})(n_{11}+n_{01})}{n_{00}}}. \tag{14}$$

3. The link-based cluster ensemble approach

To enhance the performance of the original pairwise similarity approach (Fred and Jain 2005; Strehl and Ghosh 2002), the authors employed link-based similarity measures to refine the evaluation of similarity values among data points (Iam-on *et al.* 2008). As a result, the connected-triple-based similarity (CTS) and the SimRank-based similarity (SRS) matrices are established with substantially less unknown entries, as compared to the conventional CO matrix. In addition, the approximate SimRank-based similarity (ASRS) matrix is introduced as an efficient variation of the SRS counterpart. The experiment results shown in Iam-on *et al.* (2008) suggest that such techniques can help revealing implicit relationship amongst data points, which is not possible using the original co-occurrence statistical approach. The underlying intuition and formal descriptions of the two link-based similarity methods are thoroughly reviewed herein. Note that a cluster ensemble in this approach is generated using a homogeneous collection of k-means as base clusterings, each with a random initialization of cluster center. In addition, a number of clusters k employed in these base clusterings is either fixed to a specific value ($k = \sqrt{N}$) or varies within a definite range ($k \in \{2, 3, \ldots, \sqrt{N}\}$), where N is the number of data points.

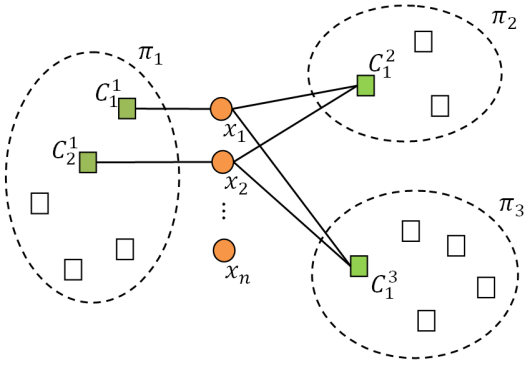


Figure 3: A graphical representation of a cluster ensemble $\Pi = \{\pi_1, \pi_2, \pi_3\}$, where $\pi_1 = \{C_1^1, C_2^1, C_3^1, C_4^1, C_5^1\}$, $\pi_2 = \{C_1^2, C_2^2, C_3^2\}$ and $\pi_3 = \{C_1^3, C_2^3, C_3^3, C_4^3, C_5^3\}$.

3.1. The connected-triple-based similarity (CTS) matrix

The connected-triple method (Klink *et al.* 2006) has been developed to assess the similarity amongst author names and identify possible duplicates (i.e., name pairs with high similarity values) within publication databases. It makes use of a network of co-authoring information $G = (V, E)$, where V is the set of vertices each corresponding to a particular name, and E is the set of edges each connecting two authors if they co-author a publication(s). The similarity of any $v_x, v_y \in V$ can be estimated by counting the number of connected-triples (i.e., triples) they are part in. Formally, a triple, $Triple = (V_{Triple}, E_{Triple})$, is a subgraph of G containing three vertices $V_{Triple} = \{v_x, v_y, v_k\} \subset V$ and two edges $E_{Triple} = \{e_{xk}, e_{yk}\} \subset E$, with $e_{xy} \notin E$. Its specific application to the cluster ensemble problem is illustrated in Figure 3.

In this illustration of a clustering ensemble Π, circle vertices denote data points $x_i, i = 1 \ldots N$, whilst square nodes represent clusters in each clustering $\pi_m, m = 1 \ldots 3$. Additionally, there exists an edge between a data point x_i and a cluster C_j^m if x_i belongs to C_j^m within the base clustering result π_m. In particular, data points x_1 and x_2 are considered to be similar with respect to the clustering results π_2 and π_3, in which they are assigned to the same clusters (clusters C_1^2 and C_1^3, respectively). However, their similarity is perceived as zero using the information given in the first clustering result, π_1, alone. Intuitively, despite being assigned to different clusters, their similarity may be revealed if these clusters are seemingly similar. Using the connected-triple technique, cluster C_1^1 and C_2^1 are justified similar due to the fact that they possess 2 connected-triples in which cluster C_1^2 and C_1^3 are centers of the triples.

Originally, the number of triples associated with any two objects is summed up as a whole number. This simple counting might be sufficient for data points or other indivisible objects. However, to evaluate the similarity between clusters, it is crucial to take into account the characteristics like shared data members among clusters. Inspired by this insight, the new weighted connected-triple algorithm for the problem of cluster ensembles has been introduced.

Weighted connected-triple (WCT) algorithm

Given a cluster ensemble Π of a set of data points X, a weighted graph $G = (V, W)$ can be constructed where V is the set of vertices each representing a cluster in Π and W is a set of weighted edges between clusters. Formally, the weight assigned to the edge w_{ij} connecting clusters $C_i, C_j \in V$ is estimated in accordance with the proportion of their overlapping

members.

$$w_{ij} = \frac{|X_{C_i} \cap X_{C_j}|}{|X_{C_i} \cup X_{C_j}|}, \tag{15}$$

where $X_{C_i} \subset X$ denotes the set of data points belonging to cluster C_i. Instead of counting the number of triples as a whole number, the weighted connected-triple method regards each triple as the minimum weight of the two involving edges.

$$WCT_{ij}^{k} = \min(w_{ik}, w_{jk}), \tag{16}$$

where WCT_{ij}^{k} is the count of the connected-triple between clusters $C_i, C_j \in V$ whose common neighbor is cluster $C_k \in V$. The count of all q $(1 \leq q < \infty)$ triples between cluster C_i and cluster C_j can be calculated as follows:

$$WCT_{ij} = \sum_{k=1}^{q} WCT_{ij}^{k}. \tag{17}$$

Following that, the similarity $Sim^{WCT}(i,j)$ between clusters C_i and C_j can be estimated as follows, where WCT_{max} is the maximum WCT_{xy} value of any two clusters within the cluster ensemble Π.

$$Sim^{WCT}(i,j) = \frac{WCT_{ij}}{WCT_{max}}. \tag{18}$$

Connected-triple-based similarity (CTS) matrix

This adopts the cluster-oriented approach previously described to enhance the quality of the conventional similarity matrix, i.e., co-association. Specifically, for any ensemble member $\pi_m \in \Pi$, $m = 1 \ldots M$, the similarity between data points $x_i, x_j \in X$ is estimated using Equation 19, where $DC \in (0, 1]$ is a constant decay factor (i.e., confidence level of accepting two non-identical objects as being similar).

$$S_m(x_i, x_j) = \begin{cases} 1 & \text{if } C(x_i) = C(x_j), \\ Sim^{WCT}(C(x_i), C(x_j)) \times DC & \text{otherwise.} \end{cases} \tag{19}$$

Following that, each entry in the CTS matrix can be computed as

$$CTS(x_i, x_j) = \frac{1}{M} \sum_{m=1}^{M} S_m(x_i, x_j). \tag{20}$$

3.2. The SimRank-based similarity (SRS) matrix

SimRank (Jeh and Widom 2002) has been considered as the benchmark technique for link-based similarity evaluation (Calado *et al.* 2006). It extends the scope of similarity estimation beyond the local context of adjacent neighbors, with the assumption that *neighbors are similar if their neighbors are similar as well.* Essentially, the similarity of any two vertices, $v_i, v_j \in V$

in a graph $G = (V, E)$, where V and E are sets of vertices and edges, respectively, can be calculated as follows:

$$s(v_i, v_j) = \frac{DC}{|N_{v_i}||N_{v_j}|} \sum_{x=1}^{|N_{v_i}|} \sum_{y=1}^{|N_{v_j}|} s(N_{v_i}^x, N_{v_j}^y), \tag{21}$$

where $DC \in (0, 1]$ is a decay factor, $N_{v_i} \subset V$ and $N_{v_j} \subset V$ are the neighbor sets whose members are directly linked to vertices v_i and v_j, respectively. Individual neighbors are specified as $N_{v_i}^x$ and $N_{v_j}^y$, for $1 \leq x \leq |N_{v_i}|$ and $1 \leq y \leq |N_{v_j}|$. Note that $s(v_i, v_j) = 0$ when $N_{v_i} = \emptyset$ or $N_{v_j} = \emptyset$. It is suggested by Jeh and Widom (2002) that the optimal similarity measures could be obtained through the iterative refinement of similarity values to a fixed-point (i.e., after t iterations).

$$\lim_{t \to \infty} R_t(v_i, v_j) = s(v_i, v_j). \tag{22}$$

This can be simplified as

$$R_{t+1}(v_i, v_j) = \frac{DC}{|N_{v_i}||N_{v_j}|} \sum_{x=1}^{|N_{v_i}|} \sum_{y=1}^{|N_{v_j}|} R_t(N_{v_i}^x, N_{v_j}^y). \tag{23}$$

At the outset, this iterative process starts off using the lower bound of: $R_0(v_i, v_j) = 1$ if $v_i = v_j$, and 0 otherwise.

Applying SimRank to the cluster ensemble problem

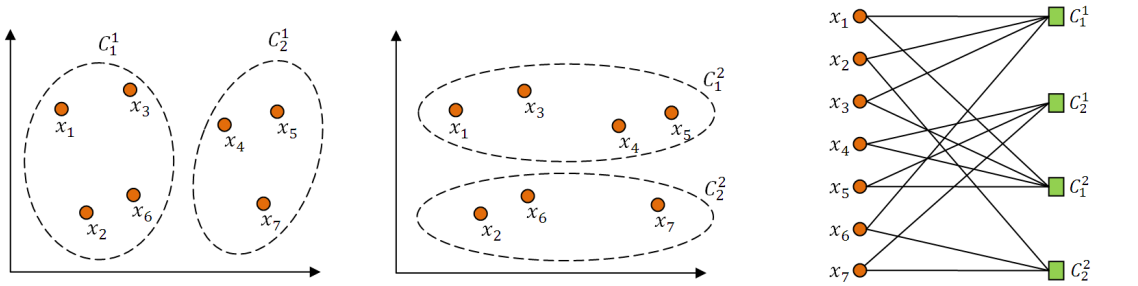
Besides considering a cluster ensemble as a network of clusters only (as with the CTS algorithm), a bipartite representation can be utilized to reveal more hidden relations. Figure 4(a) and 4(b) depict the cluster results of two base clusterings (i.e., π_1 and π_2), and the corresponding bipartite graph is presented in Figure 4(c). Given a cluster ensemble Π, a graph $G = (V, E)$ can be constructed, where V is a set of vertices representing both data points and clusters in the ensemble, and E denotes a set of edges between data points and the clusters to which they are assigned. Let $SRS(a, b)$ be the entry in the SRS matrix, which represents the similarity between any pair of data points or the similarity between any two clusters in the ensemble. For $a = b, SRS(a, b) = 1$. Otherwise,

$$SRS(a, b) = \frac{DC}{|N_a||N_b|} \sum_{a' \in N_a} \sum_{b' \in N_b} SRS(a', b'), \tag{24}$$

where DC is constant decay factor within the interval (0, 1], $N_x \subset V$ denotes the set of vertices connecting to $x \in V$. Accordingly, the similarity between data points x_i and x_j is the average similarity between the clusters to which they belong, and likewise, the similarity between clusters is the average similarity between their members.

Iterative refinement of SimRank measure

The similarity measure between any pair of vertices can be computed through the iterative refinement process. Similar to Equation 22, the similarity $SRS(a, b)$ between vertices $a, b \in V$ can be found by



(a) Clustering Result 1, π_1. (b) Clustering Result 2, π_2. (c) Bipartite Graph of the ensemble.

Figure 4: A bipartite-graph representation of cluster ensemble $\Pi = \{\pi_1, \pi_2\}$, where $\pi_1 = \{C_1^1, C_2^1\}$, $\pi_2 = \{C_1^2, C_2^2\}$ and $X = \{x_1, \ldots, x_7\}$.

$$\lim_{r\to\infty} SRS_r(a,b) = SRS(a,b). \tag{25}$$

In particular, let $SRS_r(a,b)$ be a similarity degree at iteration r, the estimation of the similarity score at the next iteration $r+1$ is defined as follows:

$$SRS_{r+1}(a,b) = \frac{DC}{|N_a||N_b|} \sum_{a'\in N_a} \sum_{b'\in N_b} SRS_r(a',b'). \tag{26}$$

Note that, initially, $SRS_0(a,b) = 1$ if $a = b$ and 0 otherwise.

3.3. The approximate SimRank-based similarity (ASRS) matrix

To improve the applicability of the SRS approach, the ASRS (approximate SRS) method is introduced as a more efficient variation of the SRS, without the iterative process of similarity refinement. Formally, a bipartite graph $G = (V, E)$ is constructed to represent a cluster ensemble Π, where V is a set of vertices representing both data points and clusters in the ensemble and E denotes a set of edges between data points and their clusters. Let $ASRS(a,b)$ be the entry in the ASRS matrix, which represents the similarity between data points $a, b \in V$. For $a = b$, $ASRS(a,b) = 1$. Otherwise,

$$ASRS(a,b) = \frac{1}{|N_a||N_b|} \sum_{a'\in N_a} \sum_{b'\in N_b} Sim^{Clus}(a',b'), \tag{27}$$

where $N_x \subset V$ denotes the set of vertices connecting to data point $x \in V$ (i.e., a set of clusters to which x belongs) and $Sim^{Clus}(y,z)$ is a similarity value between clusters y and z, which can be obtained using the weighted SimRank algorithm described below.

Weighted SimRank (wSR)

Given a cluster ensemble Π, a graph $G = (V, W)$ can be constructed where V is the set of vertices each representing a cluster in Π and W is a set of weighted edges between clusters. Formally, the weight assigned to the edge w_{ij} connecting clusters $i, j \in V$ is estimated in accordance with the proportion of their overlapping members.

$$w_{ij} = \frac{|X_i \cap X_j|}{|X_i \cup X_j|}, \tag{28}$$

where $X_p \subset X$ denotes the set of data points belonging to cluster $p \in V$. Let $Sim^{Clus}(y, z)$ be a similarity between any two clusters. For $y = z, Sim^{Clus}(y, z) = 1$; otherwise, it can be estimated as follows.

$$Sim^{Clus}(y, z) = \frac{wSR(y, z)}{wSR_{max}} \times DC, \tag{29}$$

where $DC \in (0, 1]$ is the confidence level to accept two non-identical clusters to be similar, and wSR_{max} is the maximum wSR value of any two clusters y and z, being defined as

$$wSR(y, z) = \frac{1}{|N_y||N_z|} \sum_{y' \in N_y} \sum_{z' \in N_z} (w_{yy'} \times w_{zz'}), \tag{30}$$

where $N_y, N_z \subset V$ are the set of clusters to which clusters y and z are linked (i.e., sharing data points), respectively.

3.4. Time complexity analysis

CTS matrix. Given N data points, a cluster ensemble of M ensemble members (i.e., base clusterings) and C clusters (i.e., a total number of clusters across all ensemble members), the time complexity of creating the CTS matrix is $O(N^2M + C^2T_1)$, where T_1 is the average of $|L_x|$ in the network of clusters, L_x denotes the set of clusters each directly links to the cluster x, and $|g|$ represents the size of any set g, respectively.

SRS matrix. In addition, the time requirement of generating the SRS matrix is $O(R(N^2T_2 + C^2T_3))$, where T_2 is the average of $|G_a|.|G_b|$ over all pairs of data points (a, b) in a bipartite network, G_a and G_b are the set of clusters linked to data points a and b, respectively. Similarly, T_3 denotes the average of $|G_c|.|G_d|$ over all pairs of clusters (c, d), G_c and G_d are the set of data points linked to clusters c and d. With the SimRank algorithm, R is the number of iterations of refining similarity values.

ASRS matrix. As for the ASRS matrix, the time complexity required for estimating the pairwise similarity amongst data points is reduced from $O(R(N^2T_2 + C^2T_3))$, with the SRS method, to $O(N^2T_2 + C^2T_4)$, where T_4 is the average of $|L_x|.|L_y|$ in the network of clusters, L_x and L_y denote the sets of clusters directly linked to clusters x and y, respectively. Note that T_3 measured in a bipartite network is typically greater than T_4 estimated in the single-object network of clusters. Despite such improvement, the CTS method is more efficient than the ASRS, since T_1 is typically smaller than T_4 and T_2 is usually greater than M.

4. The LinkCluE package

The objective of **LinkCluE** package is to provide MATLAB functions for generating and analyzing cluster ensembles using three link-based similarity matrices (CTS, SRS and ASRS) described in Section 3. All functions included in the **LinkCluE** package are user-callable, like any standard function in the MATLAB environment. Figure 5 illustrates the main functions

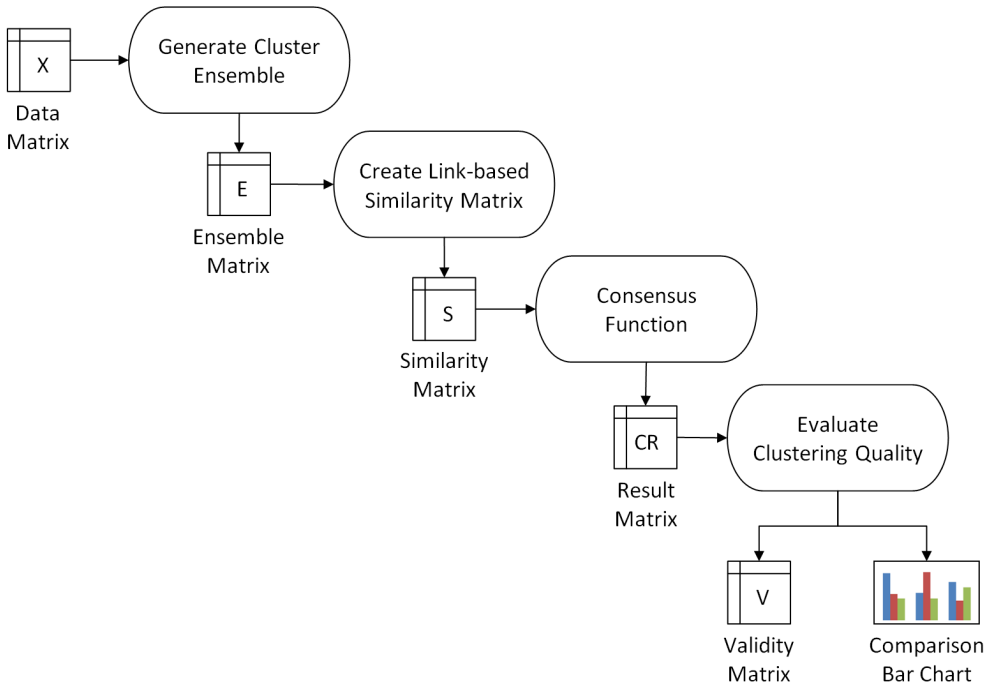


Figure 5: Main functions of the **LinkCluE** package.

of the underlying package (generating a cluster ensemble, creating link-based similarity matrices, consensus functions and evaluating clustering results, respectively), each of which is further described below.

4.1. Generating the cluster ensemble

The first step of the cluster ensemble framework is to establish a cluster ensemble $\Pi = \{\pi_1, \ldots, \pi_M\}$, which is a collection of M base clustering results (i.e., *ensemble members*). In the **LinkCluE** package, the `crEnsemble` function is used to create a matrix of cluster ensemble using k-means (with random initializations) as a base clustering algorithm. The input argument set consists of `X`, `M`, `k` and `scheme`, which are the data matrix, the number of base clusterings, the number of preferred clusters in base clusterings and the generation scheme (i.e., 1 for Fixed `k` and 2 for Random `k`), respectively. In particular, `X` is a $N \times d$ matrix of data, whose rows correspond to N observations (i.e., data points) and columns correspond to d attributes (see example of the Four-Gaussian dataset, `FGD.mat`, in `SampleData` directory or Table 1).

For the first ensemble generation scheme (i.e., Fixed `k`), a constant value `k` is used as the number of clusters across all `M` k-means base clusterings, whilst a random number within the range of [2, `k`] is employed for the other generation scheme (i.e., Random `k`). The output `E` produced from this function is an $N \times M$ matrix of cluster labels for N data points from `M` base clusterings. In practice, the `crEnsemble` function is executed as follows:

```
> E = crEnsemble(X, M, k, scheme)
```

Note that, `crEnsemble` is a non-deterministic function, such that each call may result in

different output `E`. This is due to random initializations in the k-means algorithm and the random number of `k` with Random `k` scheme. In addition, users can also import their own cluster ensembles (e.g., created by using other clustering algorithms) to the **LinkCluE** package, but they must comply with format previously set out (see Table 2 for an example).

4.2. Creating link-based similarity matrix

With the cluster ensemble produced from Section 4.1, the relationship between any pair of data points can be calculated using link-based measures. The **LinkCluE** package provides three functions for creating such similarity matrix: `cts`, `srs` and `asrs`. Their main input argument is a matrix of cluster ensemble `E`, which can either be obtained using any base clustering algorithms or the `crEnsemble` function. A similarity matrix `S`, acquired as the output of `cts`, `srs` or `asrs`, is used together with any similarity-based clustering algorithm (e.g., hierarchical agglomerative methods and METIS Karypis and Kumar 1998) to generate the final clustering result.

- `cts` **function:** Input arguments for the `cts` function consists of `E` and `dc`, which are a matrix of cluster ensemble and a decay factor, respectively. The first argument, `E`, is a $N \times M$ matrix of cluster labels for each data point obtained from base clusterings, where N is the number of data points and M is the number of base clusterings. The other argument, `dc` $\in (0, 1]$, is a constant decay factor (i.e., confidence level of accepting two non-identical clusters as being similar). The output produced by the `cts` function is an $N \times N$ matrix, `S`, of pair-wise similarity measures amongst data points. Accordingly, the `cts` function can be called using the following command:

  ```
  > S = cts(E, dc)
  ```

- `srs` **function:** While the first two input arguments of the `srs` function (i.e., `E` and `dc`) are similar to those of the `cts` function, the additional input variable `R` is require to determine the number of iterations for SimRank similarity refinement. The output of this function is also an $N \times N$ matrix of similarity values, similar to that achieved with the `cts` function. The command used to execute the `srs` function is:

  ```
  > S = srs(E, dc, R)
  ```

- `asrs` **function:** This function requires the similar input arguments as the `cts` function, `E` and `dc`, respectively. It also produces the similar output, an $N \times N$ matrix of similarity values. The `asrs` function can be executed using the following command:

  ```
  > S = asrs(E, dc)
  ```

4.3. Consensus functions

Having obtained link-based similarity matrices, they can then be input to any similarity-based clustering algorithms to produce a final clustering. In particular to the **LinkCluE** package, the `clHC` function is provided to perform three different hierarchical agglomerative clustering methods of: SL, CL and AL, respectively. It accepts a pair-wise similarity matrix `S` and the number of clusters in the final data partition (`K`) as the inputs, and delivers the final clustering

decisions, `CR`, which is an $N \times 3$ matrix (of cluster labels for N data points) produced by 3 different methods (SL, CL and AL). Formally, in order to apply these consensus functions to a given similarity matrix `S`, the `clHC` function is employed as follows:

```
> CR = clHC(S, K)
```

4.4. Evaluating clustering quality

After acquiring the final data partition, users may need to assess and compare their quality for further analysis or decision making. In the **LinkCluE** package, a function `cleval` is provided for such assessment. This function makes uses of both internal and external validity criteria (see Section 2 for details). In particular, three internal validity indices (CP, DB and Dunn) and three other external validity indices (CA, RI and AR) are employed herein. Note that low values of CP and DB indices signify good cluster structures, whilst high values of Dunn, AR, RI and CA indicate good cluster quality.

The `cleval` function produces a matrix of validity values and a comparison bar chart. It normally requires three input arguments of `X`, `CR` and `methods`. The first argument is a data matrix, while the second, `CR`, can be either vector or matrix of cluster labels for N data points. The argument `methods` is a cell of strings specifying methods that used to produce the clustering result matrix `CR`. For example, `{'CTS-SL', 'CTS-AL'}` refers to the CTS matrices with AL and AL, respectively. This set of strings is used to show as legends in a bar chart. In addition, the `cleval` function also has an optional input, `truelabels`, which can be specified only when pre-known cluster labels are available. Thus, to evaluate the clustering results, this function can be called as either

```
> V = cleval(X, CR, methods)
```

or

```
> V = cleval(X, CR, methods, truelabels)
```

Note that when users specify the argument `truelabels`, the values of all criteria measures (including three external indices of CA, AR and RI) will be shown. Otherwise, the function presents only the three internal indices (i.e., CP, DB and Dunn).

4.5. Using a single command to combine all functions

The package **LinkCluE** also provides the `LinkCluE` function to guide users through functional utility. It is an one-stop function which combines all procedures for solving a cluster ensemble problem. In practice, the `LinkCluE` function is executed as follows:

```
> [CR, V] = LinkCluE(X, M, k, scheme, K, dcCTS, dcSRS, R, dcASRS, truelabels)
```

This function first create cluster ensemble (`E`), then generate three link-based similarity matrices. Following that, it produces clustering results (`CR`) using `clHC` function and also evaluates quality of the result (`V`) using `cleval` function. Note that `truelabels` is an option input argument.

5. Illustrative examples

This section presents illustrative examples of using the **LinkCluE** package to solve cluster ensemble problems, over both synthetic and real datasets.

5.1. Four-Gaussian dataset

This synthetic dataset, acquired from Kuncheva and Vetrov (2006), is initially created in two dimensions and later added with ten more dimensions of noise. The package **LinkCluE** contains file `FGD.mat` and `FGT.mat` (within the `SampleData` directory) for its data content and true cluster labels. These can be simply imported into the MATLAB environment using the `load` built-in MATLAB function. The dataset is graphically shown in Figure 6, where only values of the two non-noise attributes are employed. Its corresponding data matrix `X` is also given in Table 1.

```
> load SampleData\FGD.mat
> load SampleData\textbackslash FGT.mat
> h = scatter(FGD(:, 1), FGD(:, 2), 50, FGT, `filled')
```

Having obtained the data matrix `X`, the cluster ensemble `E` can be created using the `crEnsemble` function. The following commands demonstrate a sample of ensemble generation, in which the input argument `k`, i.e., the number of clusters in base clusterings, is simply set to $\sqrt{N}$. Intuitively, in order to create diversity in an ensemble, `k` should be greater than the expected

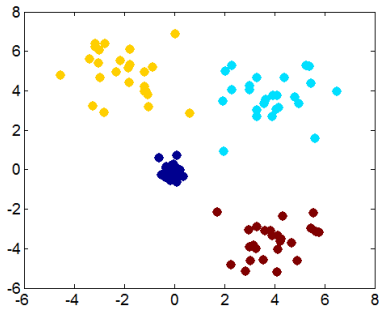


Figure 6: The Four-Gaussian dataset.

	attr_1	attr_2	attr_3	attr_4	attr_5	attr_6	attr_7	attr_8	...	attr_{12}
data_1	-0.331	-0.411	1.810	3.880	1.700	0.185	1.570	1.390	...	3.250
data_2	0.006	-0.004	0.177	2.370	1.520	2.380	2.570	2.250	...	1.710
data_3	-0.050	-0.339	2.530	0.598	2.660	2.670	0.638	2.090	...	1.920
data_4	0.101	0.693	2.060	2.030	2.610	3.770	2.380	1.690	...	0.028
data_5	-0.160	-0.576	2.930	1.550	3.800	1.760	2.890	0.359	...	0.318
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
data_{100}	4.140	-3.340	1.520	1.510	2.230	0.102	0.496	2.670	...	2.600

Table 1: A sample of $N \times d$ matrix `X` of Four-Gaussian data, where $N = 100$ and $d = 12$.

	clustering_1	clustering_2	clustering_3	clustering_4	...	clustering_{10}
data_1	3	4	9	5	...	4
data_2	5	4	4	4	...	4
data_3	5	4	4	4	...	4
data_4	5	4	4	4	...	4
data_5	5	4	4	4	...	4
⋮	⋮	⋮	⋮	⋮	⋱	⋮
data_{100}	8	8	5	2	...	1

Table 2: A sample of $N \times M$ cluster ensemble matrix `E`, with $N = 100$ and $M = 10$.

number of clusters. The common rule-of-thumb is `k` $= \sqrt{N}$ (Fred and Jain 2005; Kuncheva and Vetrov 2006; Hadjitodorov *et al.* 2006).

```
> k = ceil(sqrt(size(FGD, 1)))
> E = crEnsemble(FGD, 10, k, 1)
```

As a result, this will generate the cluster ensemble `E` from the data matrix `X`, using 10 k-means base clusterings each with Fixed `k` scheme (where `k` $= 10$). The sample of the output 100×10 matrix `E` is shown in Table 2. Each entry in `E` represents a label of the cluster to a particular data point belongs. Note that, for any data point, the labels acquired from distinct base clusterings may not be similar.

Subsequently, the cluster ensemble `E` is utilized to produce link-based similarity matrices. In order to construct the CTS matrix, named `Scts` in the following example, with a decay factor `dc` of 0.8, the required command is:

```
> Scts = cts(E, 0.8)
```

Similarly, the SRS and ASRS matrices, named `Ssrs` and `Sasrs` here, can be created as follows. The decay factor `dc` is set to be 0.8 for both matrices and the number of refinement iteration `R` for the SRS matrix is set to 3. Note that the output matrices (`Scts`, `Ssrs` and `Sasrs`) are with the same format as demonstrated in Table 3.

```
> Ssrs = srs(E, 0.8, 3)
> Sasrs = asrs(E, 0.8)
```

Then, to obtain the final clustering result, any similarity-based clustering algorithm can be applied to the aforementioned link-based similarity matrices. For such purpose, the **LinkCluE** package provides the `clHC` function that makes use of three different agglomerative clustering methods (SL, CL and AL) as consensus functions. In practice, with 4 being number of desired clusters (`K`), the following command constructs the matrix `CR`, which contains 9 clustering results each corresponds to a unique combination of the similarity matrix (CTS, SRS or ASRS) and the consensus function (SL, CL or AL). Table 4 presents a sample of the `CR` matrix.

	data_1	data_2	data_3	data_4	data_5	...	data_{100}
data_1	1	0.6315	0.5862	0.5862	0.5862	...	0.0101
data_2	0.6315	1	0.9547	0.9547	0.9547	...	0.0101
data_3	0.5862	0.9547	1	1	1	...	0.0101
data_4	0.5862	0.9547	1	1	1	...	0.0101
data_5	0.5862	0.9547	1	1	1	...	0.0101
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
data_{100}	0.0101	0.0101	0.0101	0.0101	0.0101	...	1

Table 3: A sample of $N \times N$ similarity matrix, where $N = 100$.

	CTS-SL	CTS-CL	CTS-AL	SRS-SL	SRS-CL	SRS-AL	...	ASRS-AL
data_1	1	1	1	1	3	2	...	2
data_2	1	1	1	1	3	2	...	2
data_3	1	1	1	1	3	2	...	2
data_4	1	1	1	1	3	2	...	2
data_5	1	1	1	1	3	2	...	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
data_{100}	4	3	4	4	1	4	...	4

Table 4: A sample of $N \times 9$ matrix of clustering results `CR` from 9 combinations of similarity matrices and consensus functions (CTS-SL, CTS-CL, CTS-AL, SRS-SL, SRS-CL, SRS-AL, ASRS-SL, ASRS-CL and ASRS-AL), where $N = 100$. Note that the clustering results of ASRS-SL and ASRS-CL are omitted for the presentation simplicity.

Validity index	CTS-SL	CTS-CL	CTS-AL	SRS-SL	SRS-CL	...	ASRS-AL
CP	5.4115	5.4030	5.4030	5.4115	5.4115	...	5.3995
DB	1.1441	1.2287	1.2287	1.1441	1.3677	...	1.2600
Dunn	1.3570	1.3570	1.3570	1.3570	1.3570	...	1.3887

Table 5: A sample of validity matrix `V`, containing the evaluation of clustering results achieved with nine cluster ensemble methods (CTS-SL, CTS-CL, CTS-AL, SRS-SL, SRS-CL, SRS-AL, ASRS-SL, ASRS-CL and ASRS-AL), using three internal validity criteria (CP, DB, and Dunn). Note that the validity scores of SRS-AL, ASRS-SL and ASRS-CL are omitted for the presentation simplicity.

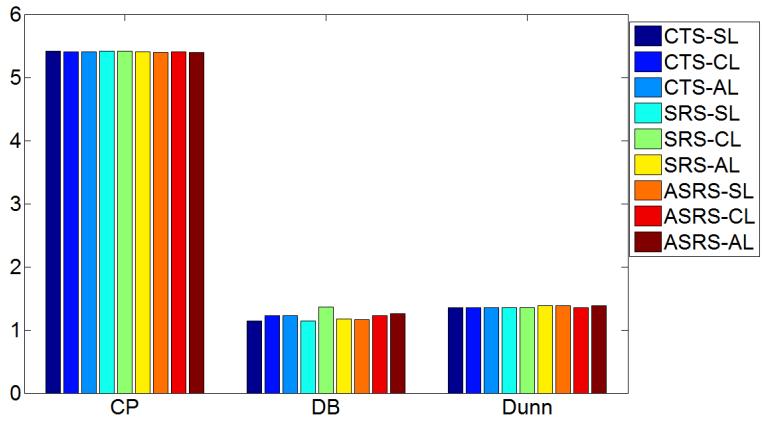


Figure 7: A bar chart that compares performance of nine link-based ensemble methods, in accordance with three internal validity indices of CP, DB and Dunn, respectively.

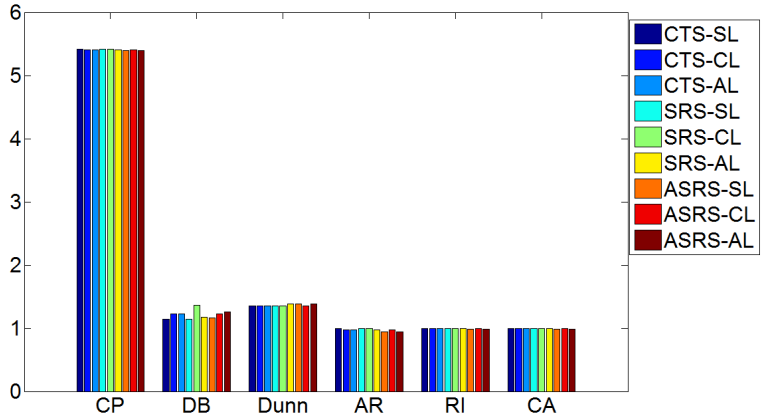


Figure 8: A bar chart that compares performance of nine link-based ensemble methods, in accordance with three internal validity indices (CP, DB and Dunn) and three external validity indices (AR, RI and CA), respectively.

```
> CR = [clHC(Scts, 4) clHC(Ssrs, 4) clHC(Sasrs, 4)]
```

After obtaining the clustering results, users can evaluate the quality or goodness of those outcomes using the `cleval` function in the **LinkCluE** package. When true cluster labels are unknown, users can use the following command to launch the `cleval` function.

```
> methods = {`CTS-SL', `CTS-CL', `CTS-AL', `SRS-SL', `SRS-CL', `SRS-AL',
    `ASRS-SL', `ASRS-CL', `ASRS-AL'}
> V = cleval(FGD, CR, methods)
```

The clustering results in `CR` will be assessed using three internal validity criteria, and the matrix of validity measures `V` (similar to that shown in Table 5) together with a comparison bar chart (see Figure 7 for an example) are produced for further analysis.

On the other hand, if true cluster labels are available, users can assess the quality of clustering results using both internal and external validity criteria. In such case, the optional

	true cluster label
data_1	1
data_2	1
data_3	1
data_4	1
data_5	1
$\vdots$	$\vdots$
data_{100}	4

Table 6: A sample of $N \times 1$ true cluster labels vector, `truelabels`, with $N = 100$.

Validity index	CTS-SL	CTS-CL	CTS-AL	SRS-SL	SRS-CL	...	ASRS-AL
CP	5.4115	5.4030	5.4030	5.4115	5.4115	...	5.3995
DB	1.1441	1.2287	1.2287	1.1441	1.3677	...	1.2600
Dunn	1.3570	1.3570	1.3570	1.3570	1.3570	...	1.3887
AR	1.0000	0.9731	0.9731	1.0000	1.0000	...	0.9456
RI	1.0000	0.9901	0.9901	1.0000	1.0000	...	0.9800
CA	1.0000	0.9900	0.9900	1.0000	1.0000	...	0.9800

Table 7: A sample of validity matrix `V`, containing the evaluation results achieved with nine cluster ensemble methods (CTS-SL, CTS-CL, CTS-AL, SRS-SL, SRS-CL, SRS-AL, ASRS-SL, ASRS-CL and ASRS-AL) using six validity criteria (CP, DB, Dunn, AR, RI and CA). Note that the validity scores of SRS-AL, ASRS-SL and ASRS-CL are omitted for the presentation simplicity.

Validity index	SL	CL	AL	k-means
CP	7.0209	5.5076	5.4115	5.9508
DB	1.5122	1.4101	1.3454	1.6946
Dunn	0.6631	1.2631	1.3570	0.6502
AR	0.2888	0.8725	1.0000	0.5683
RI	0.6141	0.9529	1.0000	0.8248
CA	0.5100	0.9500	1.0000	0.7300

Table 8: A validity matrix, containing the evaluation results achieved with four single-run clustering techniques (SL, CL, AL and k-means) using six validity criteria (CP, DB, Dunn, AR, RI and CA).

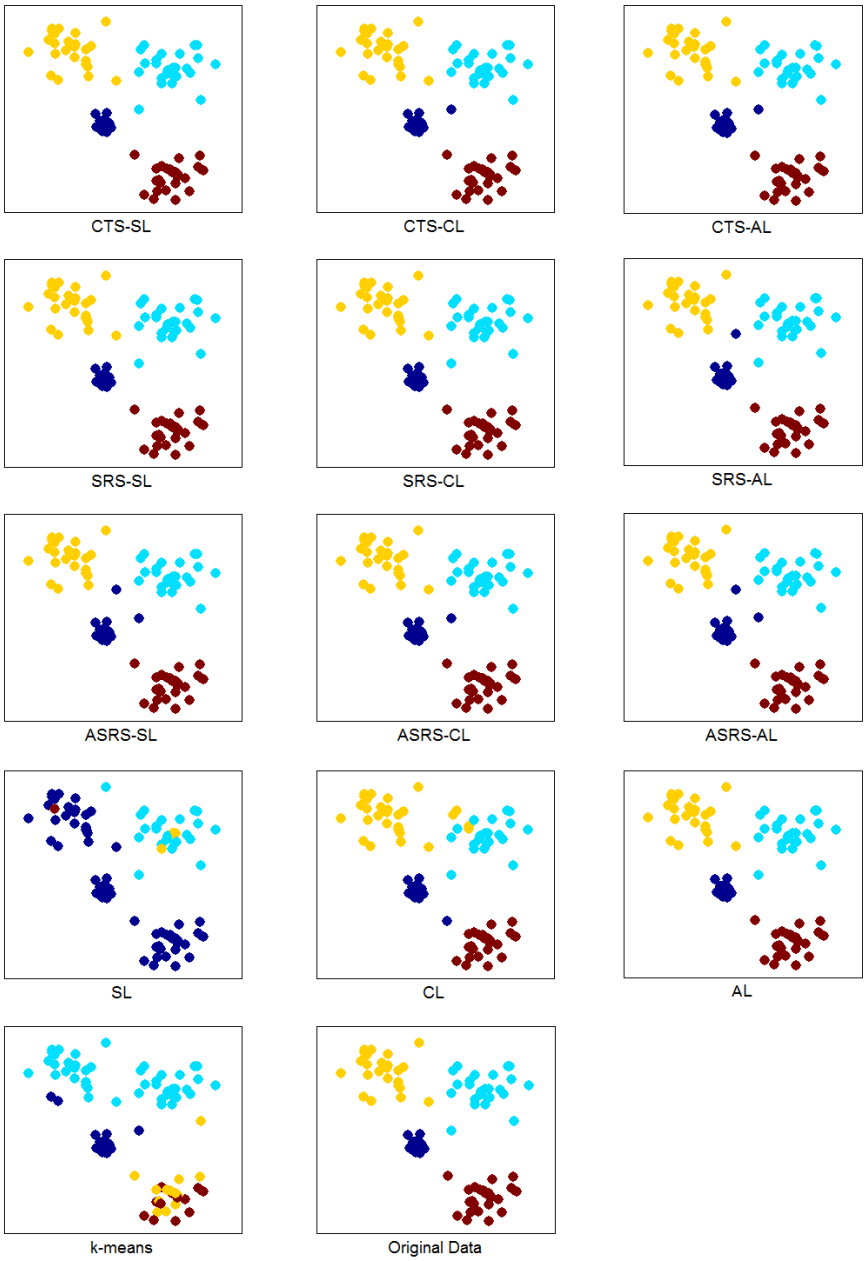


Figure 9: Nine clustering results for the Four-Gaussian dataset using Package **LinkCluE**, compared with those of SL, CL, AL, k-means and original data, respectively.

`truelabels` input argument ($N \times 1$ vector) must be specified prior to the execution of the `cleval` function. For the example illustrated so far, the true cluster labels for Four-Gaussian data is provided as the file `FGT.mat` within the `SampleData` directory. This vector can be simply imported into the MATLAB environment using the `load` built-in MATLAB function. The format of this `truelabels` vector is shown in Table 6.

The following command is employed in order to launch the `cleval` function, such that the vector of known cluster labels is also exploited for quality evaluation.

```
> V = cleval(FGD, CR, methods, FGT)
```

Effectively, this will assess the quality of clustering results `CR` using both internal and external validity indices. It also produces the validity matrix `V` (as shown in Table 7) and a comparison bar chart (see Figure 8).

To demonstrate the effectiveness of the cluster ensemble approach, the results shown in Table 7 are compared with similar evaluation measures of applying four single-run clustering methods (SL, CL, AL and k-means) to the Four-Gaussian dataset (see Table 8). In addition, Figure 9 provides a graphical means for this comparison. Accordingly, it is clearly seen that package **LinkCluE** can provide almost perfect clustering results for Four-Gaussian dataset and drastically improve performance of the base clustering (i.e., k-means).

5.2. Leukemia dataset

This real data is exploited in (de Souto *et al.* 2008) for gene expression data clustering. It contains expression values of 1,081 genes collected from the Affymetrix U95A chip and 72 blood samples of leukemia patients at the time of diagnosis or relapse. These samples are categorized into 2 classes of 24 acute lymphoblastic leukemia (ALL) samples and 48 samples of lymphoblastic leukemias with MLL translocations (MLL). Further biological details of this dataset can be found in (Armstrong *et al.* 2002). The package **LinkCluE** includes file `LD.mat` and `LT.mat` (within the `SampleData` directory) for its data content and true cluster labels.

```
> load SampleData\LD.mat
> load SampleData\LT.mat
```

In a real world dataset, variables can be measured against different scales. For instance, one variable can measure the blood pressure and another variable can measure heart rate. These discrepancies can distort the proximity calculation of any clustering technique. Hence, variables are usually normalized before being employed in a machine learning model. Specifically to the Leukemia dataset, MATLAB's built-in `zscore` function is exploited to transform all the attribute values to those expressed on the uniform scale.

```
> LD = zscore(LD)
```

To obtain clustering results from Package **LinkCluE**, the one-stop function `LinkCluE` can be conveniently used as follows:

```
> k = ceil(sqrt(size(LD, 1)))
> [CR, V] = LinkCluE(LD, 10, k, 1, 2, 0.8, 0.8, 3, 0.8, LT)
```

This will first generate the cluster ensemble `E` from the data matrix `LD`, using 10 k-means base clusterings each with Fixed `k` scheme (where `k` = 9). Afterward, three link-based similarity matrices (CTS, SRS and ASRS) are constructed from the cluster ensemble `E` (all with a decay factor `dc` of 0.8 and the number of iterations `R` = 3 for the SRS matrix. The final nine clustering results `CR` are subsequently produced by the `clHC` function, using `K = 2`. In particular, the optional input argument of true cluster labels `LT`, is specified. Therefore, clustering results `CR` are evaluated using both internal and external validity criteria. Finally, the function will

Validity index	CTS-SL	CTS-CL	CTS-AL	SRS-SL	SRS-CL	...	ASRS-AL
CP	45.6950	45.1790	44.6520	45.6950	44.6740	...	44.6520
DB	2.8559	4.7783	3.0839	2.8559	2.7857	...	3.0839
Dunn	0.6473	0.3838	0.6430	0.6473	0.6967	...	0.6430
AR	0.1178	-0.0030	0.6343	0.1178	0.4534	...	0.6343
RI	0.5931	0.4992	0.8220	0.5931	0.7375	...	0.8220
CA	0.7222	0.6667	0.9028	0.7222	0.8472	...	0.9028

Table 9: A validity matrix `V` of Leukemia dataset, containing the evaluation results achieved with nine cluster ensemble methods (CTS-SL, CTS-CL, CTS-AL, SRS-SL, SRS-CL, SRS-AL, ASRS-SL, ASRS-CL and ASRS-AL) using six validity criteria (CP, DB, Dunn, AR, RI and CA). Note that the validity scores of SRS-AL, ASRS-SL and ASRS-CL are omitted for the presentation simplicity.

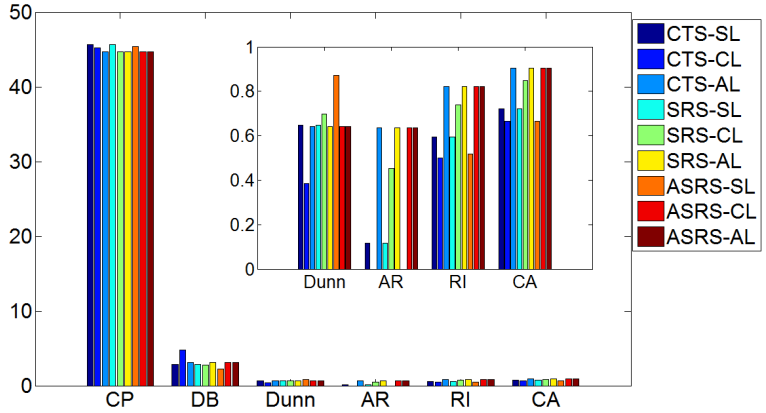


Figure 10: A bar chart that compares performance of nine link-based ensemble methods for Leukemia dataset, in accordance with three internal validity indices (CP, DB and Dunn) and three external validity indices (AR, RI and CA), respectively. Note that the enlarged sub-chart is for presentational purpose in this paper only.

Validity index	SL	CL	AL	k-means
CP	44.7370	44.7370	44.7370	44.7370
DB	40.9750	40.9750	40.9750	40.9750
Dunn	0.0244	0.0244	0.0244	0.0244
AR	-0.0138	-0.0138	-0.0138	-0.0138
RI	0.5403	0.5403	0.5403	0.5403
CA	0.6667	0.6667	0.6667	0.6667

Table 10: A validity matrix of Leukemia dataset, containing the evaluation results achieved with four single-run clustering techniques (SL, CL, AL and k-means) using six validity criteria (CP, DB, Dunn, AR, RI and CA).

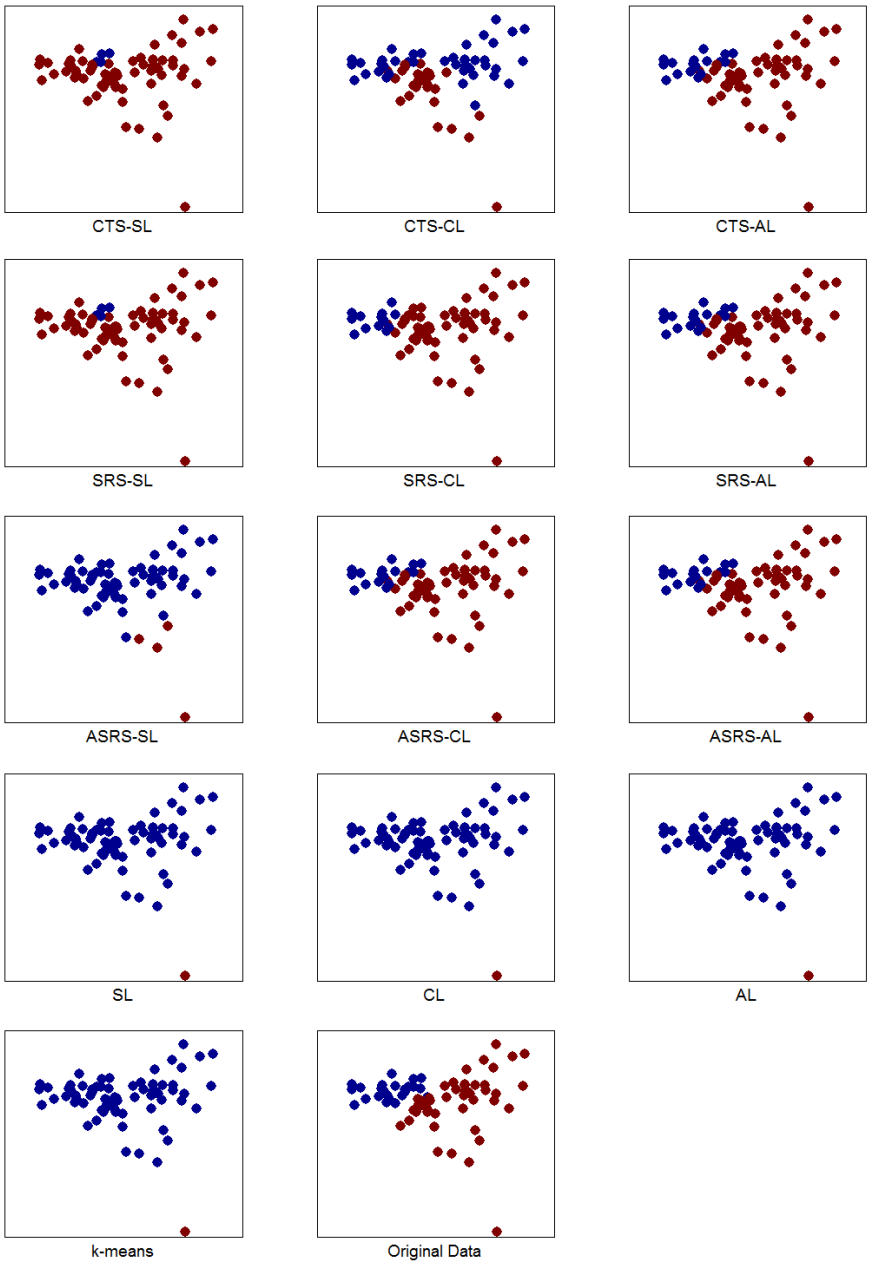


Figure 11: Nine clustering results for Leukemia dataset using Package **LinkCluE** compared with those of SL, CL, AL, k-means and original data, respectively.

deliver clustering results `CR`, validity matrix `V` (see Table 9) and a comparison bar (shown in Figure 10), respectively.

Similar to the previous example of Four-Gaussian data, the clustering results obtained by the package **LinkCluE** are compared to those of four single-run clustering methods (SL, CL, AL and k-means). The validity matrix of these simple techniques is presented in Table 10.

Due to its high dimensionality, the principal components analysis (PCA) method (Denvijver and Kittler 1982) is used to help visualizing Leukemia data. The PCA method generates

a new set of variables, called principal components, each as a linear combination of the original variables. All the principal components are orthogonal to each other, thus there is no redundant information. The set of principal components forms an orthogonal basis for the space of the underlying data. The following command executes the built-in function `princomp` that generates a set of principal components for the Leukemia dataset (`LD`).

```
> [COEFF, SCORE] = princomp(LD)
```

Following that, two principal components with the highest score (i.e., the first two columns of the `SCORE` matrix, whose rows correspond to original data points) are used to create a simple visualization of the Leukemia dataset. This is achieved by executing the following command.

```
> h = scatter(SCORE(:, 1), SCORE(:, 2), 50, LT, `filled')
```

With the original dataset, Figure 11 compares nine clustering results obtained from the package **LinkCluE** with those acquired by using SL, CL, AL and k-means algorithms.

6. Discussion

The **LinkCluE** package is thoroughly tested on a workstation (Intel Core2 CPU 6600 @2.40GHz, 2GB RAM) with MATLAB version 7.8.0(R2009a). Here it has been shown that **LinkCluE** package is effective on real world datasets, such as the Leukemia dataset, for which it produced excellent results. Nevertheless, there are some limitations of the SRS matrix due to its operating time complexity. The ASRS method can improve on this to some extent (by removing the iteration process of SimRank), whilst still being more expensive than the CTS counterpart.

6.1. Scalability analysis

In order to illustrate the scalability of these link-based techniques, synthesized cluster ensembles are employed to assess the computational time requirement of generating three link-based matrices. Similar to Cristofor and Simovici (2002), these ensembles are created as: for each base-clustering (i.e., ensemble member) $\pi_t \in \{\pi_1, \ldots, \pi_M\}$, a data point $x_i, i = 1, \ldots, N$ is randomly given a cluster label $L_j, j = 1, \ldots, c$. In the current research, these parameters are: $N \in \{500, 1,000, \ldots, 5,000\}$, $c = 10$ and $M \in \{2, 3, \ldots, 10\}$. Note that C (i.e., the number of all clusters in an ensemble) can be estimated by $C = c \times M$.

Since the creation of the link-based matrices depends principally on the magnitudes of N and C, two types of scalability are assessed: (i) the scalability against the number of data points (N) for a given value of C and (ii) the scalability against the number of clusters C for a given value of N. Figure 12 shows the run times (in seconds) for computing the three link-based matrices over synthesized cluster ensembles with $C = 100$ and ten distinct numbers of data points ($N \in \{500, 1,000, \ldots, 5,000\}$). The important observation from this figure is that the run time of generating all three matrices tends to increase quadratically as the number of data points is increased. In particular to the CTS matrix, the fitted curve, with $R^2 = 1$, is $y = 0.003x^2 + 0.026x + 0.094$. The fitted curves for ASRS and SRS matrices are $y = 0.040x^2 + 0.021x + 0.125$, with $R^2 = 1$ and $y = 0.094x^2 + 0.298x - 1.305$, with $R^2 = 0.999$, respectively. The quantitative measure R^2 is known as the 'goodness' of fit. It is computed

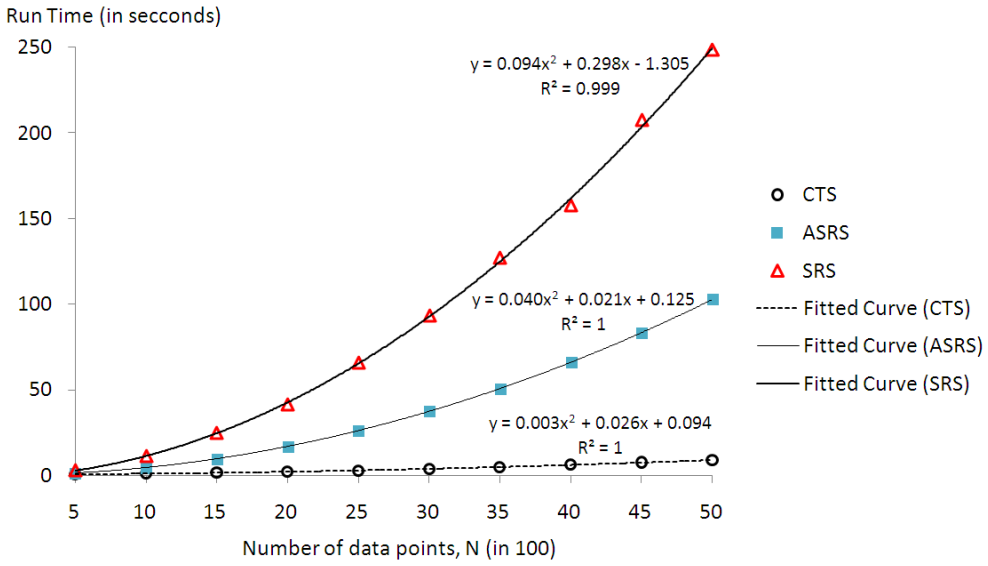


Figure 12: Scalability of link-based matrices creation to the number of data points when C of synthesized cluster ensembles is 100. The fitted lines are given to illustrate quadratic relations.

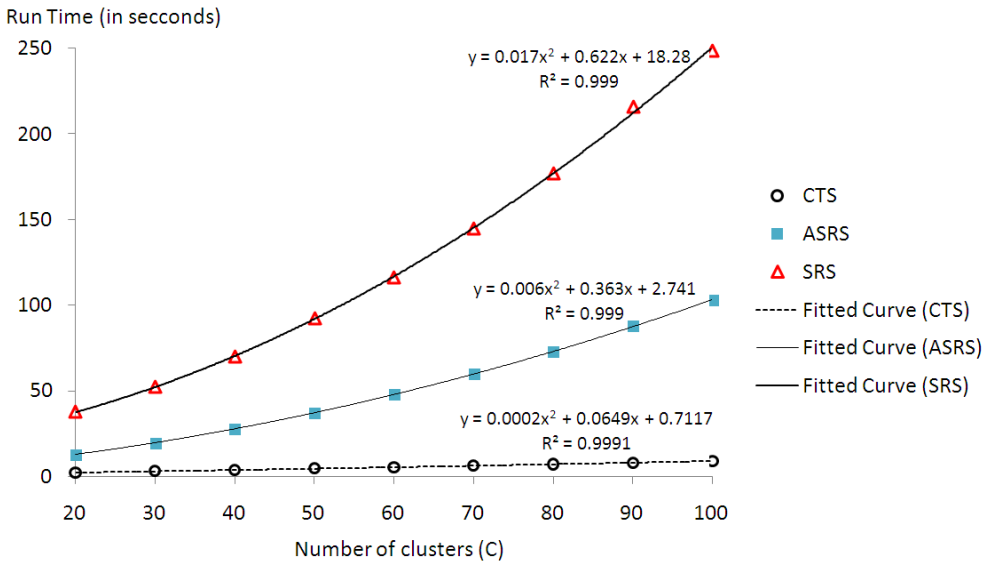


Figure 13: Scalability of link-based matrices creation to the number of clusters when N of synthesized cluster ensembles is 5,000. The fitted curves suggest a strong quadratic relationship.

as the fraction of the total variation of the Y values of data points that is attributable to the assumed fitted curve. Its values typically range from 0 to 1, with values close to 1 indicating a good fit (Draper and Smith 1998).

In addition, Figure 13 illustrates the execution times over synthesized ensembles with $N = 5,000$ and nine different numbers of C ($C \in \{20, 30, \ldots, 100\}$). Similar observations have been obtained, the run times have quadratic relationship with the values of C. The fitted

curve for the CTS matrix is $y = 0.0002x^2 + 0.0649x + 0.7117$, with $R^2 = 0.9991$. Likewise, the fitted curves, both with $R^2 = 0.999$, are $y = 0.006x^2 + 0.363x + 2.741$ and $y = 0.017x^2 + 0.622x + 18.28$, for ASRS and SRS matrices, respectively.

6.2. Further improvement

In spite of their effectiveness, the implementation of link-based similarity methods (even the CTS) similarly suffer from high computational time requirements. This drawback originates within the algorithms, whose simplified variation may not be able to maintain the original performance. Hence, the possible solution is to rely on programming language and hardware technology that may allow the underlying algorithms to be executed more efficiently. Recently, the multicore-processor architecture has emerged as a new standard of work station with enhanced capability. To take a full advantage of such innovation, MATLAB and **Parallel Computing Toolbox** (included in MATLAB 7.4, R2007a, and higher) address the challenge of designing a programming language that works well in a multicore system (Moler 2007). In particular, the two most common paradigms of parallel programming are 'thread' and 'parallel for-loop', respectively.

Based on the empirical investigation of Luszczek (2008), threading is less efficient in a multicore system, as compared to the parallel 'for-loop' (named `parfor`) provided by MATLAB. In addition, the number of threads should not exceed the actual number of processing core. This constraint is invalid using the `parfor` function. A definite precaution of employing this function is that the results within one iteration should be independent to those of others. Particularly to the link-based approach discussed thus far, the following example demonstrated how the `parfor` function can be used to implement the generation process of the CTS matrix.

Two processing tasks are required to obtain the CTS matrix: (i) first the $C \times C$ matrix of cluster similarity (CLUS) is created by the function `WCT1`, then (ii) entries in the CLUS matrix are used to estimated entries of the CTS matrix ($N \times N$ matrix of similarity amongst data points), by the function `WCT2`. The following code roughly illustrates the way in which the `parfor` can be used, where `C` and `N` denote the number of clusters and that of data points, respectively.

Pseudo code for generating the CLUS matrix:

```
parfor x = 1 : (C - 1)
    parfor y = (x + 1) : C
        CLUS(x, y) = WCT1(x, y);
    end
end
```

Pseudo code for generating the CTS matrix:

```
parfor a = 1 : (N - 1)
    parfor b = (x + 1) : N}
        CTS(a, b) = WCT2(CLUS, a, b);}
    end
end
```

With this example, entries in the CLUS matrix (also those of the CTS matrix) are created separately, by executing the `WCT1` (or `WCT2` for the case of CTS matrix) on a number of

different 'labs' (MATLAB sessions). These labs are run on processor cores, but the number of labs does not have to match the number of cores. Unlike threads, labs do not share memory with each other, thus allowing them to be execute on several systems connected via a network. The programming technique displayed in this example can also be applied to implement the generation of SRS and ASRS matrices.

Future versions of the package will be made available via the web site at `http://users.aber.ac.uk/nii07/`. These will include new approximate methodologies that aim to reduce the computational complexity of link-based similarity measures and extend their applicability to large datasets. Moreover, a graphical user interface tool will also be provided for efficient comparison and analysis.

6.3. Further application

The illustrative examples given in this paper (Section 5) for demonstrating the exploitation of **LinkCluE** package focus on using built-in MATLAB functions for generating an ensemble (i.e., `kmeans` as base clustering technique) and as a consensus function (i.e., `linkage` as final clustering function). But, in fact, `cts`, `srs` and `asrs` functions are generic such that they can be used with any user-generated cluster ensemble `E`. For instance, an ensemble may be created from heterogeneous base clusterings or the homogeneous collection each with different data subspaces.

The resulting matrices (CTS, SRS and ASRS) can be input to any similarity-based clustering method to derive the final data partition. In particular, the matrix can be transformed into a weighted graph $G = (V, W)$, where V is the set of vertices each corresponds to a specific data point, and W denotes the set of edges'weight between any two vertices. These weights can be obtained directly from a given similarity matrix. For instance, with the CTS matrix, $w_{ij} \in W$ (of the edge connecting vertices $v_i, v_j \in V$, which correspond to data points x_i and x_j, respectively) can be acquired directly from the entry $CTS(i, j)$. Having achieved such graph, a graph partitioning algorithm (such as METIS Karypis and Kumar 1998) can be employed to generate the final data partition.

Apart from the application to numerical datasets, the **LinkCluE** package can also be used to analyze categorical data. Conceptually, an ensemble `E` is constructed using any clustering algorithm for categorical data (e.g., k-modes Huang 1998). Then, `cts`, `srs`, `asrs` and other functions in this package can be exploited to create similarity matrices, derive the final clustering results and their evaluation measures.

Acknowledgments

The authors would like to thank the reviewer and the associate editor for their useful comments, and Dr. Tossapon Boongoen for his support and suggestions.

References

Armstrong SA, Staunton JE, Silverman LB, Pieters R, den Boer ML, Minden MD, Sallan SE, Lander ES, Golub TR, Korsmeyer SJ (2002). "MLL Translocations Apecify a Distinct

Gene Expression Profile that Distinguishes a Unique Leukemia." *Nature Genetics*, **30**(1), 41–47.

Ayad H, Kamel M (2003). "Finding Natural Clusters Using Multi-Clusterer Combiner Based on Shared Nearest Neighbors." In *Proceedings of International Workshop on Multiple Classifier Systems*, pp. 166–175. Springer-Verlag, Berlin.

Boulis C, Ostendorf M (2004). "Combining Multiple Clustering Systems." In *Proceedings of European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 63–74. Springer-Verlag, New York.

Calado P, Cristo M, Gonçalves MA, de Moura ES, Ribeiro-Neto BA, Ziviani N (2006). "Link-Based Similarity Measures for the Classification of Web Documents." *Journal of American Society for Information Science and Technology*, **57**(2), 208–221.

Campello R (2007). "A Fuzzy Extension of the Rand Index and Other Related Indexes for Clustering and Classification Assessment." *Pattern Recognition Letters*, **28**(7), 833–841.

Cristofor D, Simovici D (2002). "Finding Median Partitions Using Information-Theoretical-Based Genetic Algorithms." *Journal of Universal Computer Science*, **8**(2), 153–172.

Davies DL, Bouldin DW (1979). "A Cluster Separation Measure." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **1**(2), 224–227.

de Souto MC, Costa IG, de Araujo DSA, Ludermir TB, Schliep A (2008). "Clustering Cancer Gene Expression Data: A Comparative Study." *BMC Bioinformatics*, **9**, 497.

Denvijver PA, Kittler J (1982). *Pattern Recognition: A Statictical Approach*. Prentice Hall, Englewood Cliffs.

Domeniconi C, Al-Razgan M (2009). "Weighted Cluster Ensembles: Methods and Analysis." *ACM Transactions on Knowledge Discovery from Data*, **2**(4), 1–40.

Draper NR, Smith H (1998). *Applied Regression Analysis*. 3rd edition. John Wiley & Sons, New York.

Duda RO, Hart PE, Stork DG (2000). *Pattern Classification*. 2nd edition. John Wiley & Sons, New York.

Dudoit S, Fridyand J (2003). "Bagging to Improve the Accuracy of a Clustering Procedure." *Bioinformatics*, **19**(9), 1090–1099.

Dunn JC (1974). "Well Separated Clusters and Optimal Fuzzy Partitions." *Cybernetics and Systems*, **4**(1), 95–104.

Fern XZ, Brodley CE (2003). "Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach." In *Proceedings of International Conference on Machine Learning*, pp. 186–193. AAAI Press, Washington, DC.

Fern XZ, Brodley CE (2004). "Solving Cluster Ensemble Problems by Bipartite Graph Partitioning." In *Proceedings of International Conference on Machine Learning*, pp. 36–43. ACM, New York.

Fischer B, Buhmann JM (2003). "Bagging for Path-Based Clustering." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**(11), 1411–1415.

Fred ALN (2001). "Finding Consistent Clusters in Data Partitions." In *Proceedings of Second International Workshop on Multiple Classifier Systems*, pp. 309–318. Springer-Verlag, London.

Fred ALN, Jain AK (2002). "Data Clustering Using Evidence Accumulation." In *Proceedings of International Conference on Pattern Recognition*, pp. 276–280. IEEE Computer Society, Washington, DC.

Fred ALN, Jain AK (2003). "Robust Data Clustering." In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 128–136. IEEE Computer Society, Los Alamitos.

Fred ALN, Jain AK (2005). "Combining Multiple Clusterings Using Evidence Accumulation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(6), 835–850.

Fred ALN, Jain AK (2006). "Learning Pairwise Similarity for Data Clustering." In *Proceedings of International Conference on Pattern Recognition*, pp. 925–928. IEEE Computer Society, Washington, DC.

Gionis A, Mannila H, Tsaparas P (2005). "Clustering Aggregation." In *Proceedings of International Conference on Data Engineering*, pp. 341–352. IEEE Computer Society, Los Alamitos.

Hadjitodorov ST, Kuncheva LI, Todorova LP (2006). "Moderate Diversity for Better Cluster Ensembles." *Information Fusion*, **7**(3), 264–275.

Hochbaum DS, Shmoys DB (1985). "A Best Possible Heuristic for the k-Center Problem." *Mathematics of Operations Research*, **10**(2), 180–184.

Hornik K (2005). "A CLUE for CLUster Ensembles." *Journal of Statistical Software*, **14**(12), 1–25. URL http://www.jstatsoft.org/v14/i12/.

Hu X, Yoo I (2004). "Cluster Ensemble and its Applications in Gene Expression Analysis." In *Proceedings of Asia-Pacific Bioinformatics Conference*, pp. 297–302. Australian Computer Society, Darlinghurst.

Huang Z (1998). "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values." *Data Mining Knowledge Discovery*, **2**, 283–304.

Hubert L, Arabie P (1985). "Comparing Partitions." *Journal of Classification*, **2**, 193–218.

Iam-on N, Boongoen T, Garrett S (2008). "Refining Pairwise Similarity Matrix for Cluster Ensemble Problem with Cluster Relations." In *Proceedings of Eleventh International Conference on Discovery Science*, pp. 222–233. Springer-Verlag, Berlin.

Iam-on N, Boongoen T, Garrett S (2010). "LCE: A Link-Based Cluster Ensemble Method for Improved Gene Expression Data Analysis." *Bioinformatics*, **26**(12), 1513–1519.

Jain AK, Dubes RC (1998). *Algorithms for Clustering Data*. Prentice-Hall, New Jersey.

Jain AK, Law MHC (2005). "Data Clustering: A User's Dilemma." In *Proceedings of International Conference on Pattern Recognition and Machine Intelligence*, pp. 1–10. Springer-Verlag, Berlin.

Jain AK, Murty MN, Flynn PJ (1999). "Data Clustering: A Review." *ACM Computing Survey*, **31**(3), 264–323.

Jeh G, Widom J (2002). "SimRank: A Measure of Structural-Context Similarity." In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 538–543. ACM, New York.

Karypis G, Aggarwal R, Kumar V, Shekhar S (1999). "Multilevel Hypergraph Partitioning: Applications in VLSI Domain." *IEEE Transaction on VLSI System*, **7**(1), 69–79.

Karypis G, Kumar V (1998). "Multilevel k-Way Partitioning Scheme for Irregular Graphs." *Journal of Parallel Distributed Computing*, **48**(1), 96–129.

Kaufman L, Rousseeuw PJ (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York.

Kittler J, Hatef M, Duin R, Matas J (1998). "On Combining Classifiers." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(3), 226–239.

Klink S, Reuther P, Weber A, Walter B, Ley M (2006). "Analysing Social Networks within Bibliographical Data." In *Proceedings of International Conference on Database and Expert Systems Applications*, pp. 234–243. Springer-Verlag, Berlin.

Kuncheva LI, Hadjitodorov ST (2004). "Using Diversity in Cluster Ensembles." In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, pp. 1214–1219. IEEE Computer Society, Washington, DC.

Kuncheva LI, Vetrov D (2006). "Evaluation of Stability of k-Means Cluster Ensembles with Respect to Random Initialization." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(11), 1798–1808.

Kyrgyzov IO, Maitre H, Campedel M (2007). "A Method of Clustering Combination Applied to Satellite Image Analysis." In *Proceedings of International Conference on Image Analysis and Processing*, pp. 81–86. IEEE Computer Society, Washington, DC.

Law M, Topchy A, Jain AK (2004). "Multiobjective Data Clustering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pp. 424–430. IEEE Computer Society, Los Alamitos.

Luszczek P (2008). "Enhancing Multicore System Performance Using Parallel Computing with MATLAB." *MATLAB Digest*, **17**(5). URL http://www.mathworks.com/company/newsletters/digest/2008/sept/parallel-computing.html.

Minaei-Bidgoli B, Topchy A, Punch W (2004). "A Comparison of Resampling Methods for Clustering Ensembles." In *Proceedings of the International Conference on Machine Learning: Models, Technologies and Applications*, pp. 939–945. CSREA Press.

Moler C (2007). "Parallel MATLAB: Multiple Processors and Multiple Cores." *The MathWorks News and Notes*. URL http://www.mathworks.com/company/newsletters/news_notes/june07/clevescorner.html.

Monti S, Tamayo P, Mesirov JP, Golub TR (2003). "Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data." *Machine Learning*, **52**(1-2), 91–118.

Ng A, Jordan M, Weiss Y (2001). "On Spectral Clustering: Analysis and an Algorithm." *Advances in Neural Information Processing Systems*, **14**, 849–856.

Nguyen N, Caruana R (2007). "Consensus Clusterings." In *Proceedings of IEEE International Conference on Data Mining*, pp. 607–612. IEEE Computer Society, Washington, DC.

Rand WM (1971). "Objective Criteria for the Evaluation of Clustering Methods." *Journal of the American Statistical Association*, **66**, 846–850.

Strehl A, Ghosh J (2002). "Cluster Ensembles: A Knowledge Reuse Framework for Combining Multiple Partitions." *Journal of Machine Learning Research*, **3**, 583–617.

Swift S, Tucker A, Vinciotti V, Martin N, Orengo C, Liu X, Kellam P (2004). "Consensus Clustering and Functional Interpretation of Gene-Expression Data." *Genome Biology*, **5**, R94.

The MathWorks, Inc (2007). *MATLAB – The Language of Technical Computing, Version 7.5*. The MathWorks, Inc., Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Topchy AP, Jain AK, Punch WF (2003). "Combining Multiple Weak Clusterings." In *Proceedings of IEEE International Conference on Data Mining*, pp. 331–338. IEEE Computer Society, Washington, DC.

Topchy AP, Jain AK, Punch WF (2004). "A Mixture Model for Clustering Ensembles." In *Proceedings of SIAM International Conference on Data Mining*, pp. 379–390. SIAM, Philadelphia.

Topchy AP, Jain AK, Punch WF (2005). "Clustering Ensembles: Models of Consensus and Weak Partitions." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(12), 1866–1881.

Wolpert DH, Macready WG (1995). "No Free Lunch Theorems for Search." *Technical Report SFI-TR-95-02-010*, Santa Fe Institute.

Xue H, Chen S, Yang Q (2009). "Discriminatively Regularized Least-Squares Classification." *Pattern Recognition*, **42**(1), 93–104.

Yu Z, Wong HS, Wang H (2007). "Graph-Based Consensus Clustering for Class Discovery from Gene Expression Data." *Bioinformatics*, **23**(21), 2888–2896.

Affiliation:

Natthakan Iam-on
Department of Computer Science
Penglais Campus
Aberystwyth University
Aberystwyth, Wales, United Kingdom
E-mail: nii07@aber.ac.uk
URL: http://users.aber.ac.uk/nii07/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 36, Issue 9 *Submitted:* 2009-05-11
August 2010 *Accepted:* 2010-06-24