Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

A Clustering Comparison Measure Using Density Profiles and its Application to the Discovery of Alternate Clusterings*

Eric Bae · James Bailey · Guozhu Dong

Received: date / Accepted: date

Abstract Data clustering is a fundamental and very popular method of data analysis. Its subjective nature, however, means that different clustering algorithms or different parameter settings can produce widely varying and sometimes conflicting results. This has led to the use of clustering comparison measures to quantify the degree of similarity between alternative clusterings. Existing measures, though, can be limited in their ability to assess similarity and sometimes generate unintuitive results. They also cannot be applied to compare clusterings which contain different data points, an activity which is important for scenarios such as data stream analysis.

In this paper, we introduce a new clustering similarity measure, known as *ADCO*, which aims to address some limitations of existing measures, by allowing greater flexibility of comparison via the use of density profiles to characterize a clustering. In particular, it adopts a 'data mining style' philosophy to clustering comparison, whereby two clusterings are considered to be more similar, if they are likely to give rise to similar types of prediction models.

Furthermore, we show that this new measure can be applied as a highly effective objective function within a new algorithm, known as MAXIMUS, for generating alternate clusterings.

Keywords clustering · cluster analysis · clustering comparison · clustering similarity · alternate clusterings · alternate clustering algorithms

* Part of this work appeared in a preliminary form in [1]. See Section 2 for discussion.

Eric Bae and James Bailey
NICTA Victoria Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia
E-mail: kheb,jbailey@csse.unimelb.edu.au

Guozhu Dong
Department of Computer Science and Engineering, Wright State University, USA
guozhu.dong@wright.edu

1 Introduction

Cluster analysis is a fundamental machine learning and data mining task which discovers patterns, relationships and structures in an unsupervised manner. It is used in a variety of fields, including biomedicine, information retrieval and financial analysis, to discover hidden knowledge and information. However, the process of grouping similar data objects is subjective and highly dependent on the clustering criteria used. For this reason, a vast number of clustering algorithms have been developed, each based on different heuristics. These algorithms often provide very different results. Moreover, even if a single algorithm is used, many different alternative clusterings[1] can still be generated, simply by changing the initial conditions/parameters of the algorithm. Given this situation, researchers often need to compare or measure the similarity between two clusterings. Indeed this is a key component in what is called external validation [2] and it is used to return a quantitative measure of the degree to which two different clusterings are similar or different. Furthermore, clustering comparison techniques are frequently applied to evaluate the quality of clustering algorithm by comparing its results against a gold-standard clustering and to obtain a set of a diverse, high quality alternate clusterings [3, 4].

In this paper, we introduce a new clustering similarity measure, known as *ADCO*, which aims to address some limitations of existing techniques, by allowing greater flexibility of comparison via the use of density profiles to characterize a clustering. In particular, it adopts a 'data mining style' philosophy to clustering comparison, whereby two clusterings are considered to be more similar, if they are likely to give rise to similar types of prediction models.

Furthermore, we show that this new measure can be applied as a highly effective objective function within a new algorithm, known as MAXIMUS, for generating alternate clusterings.

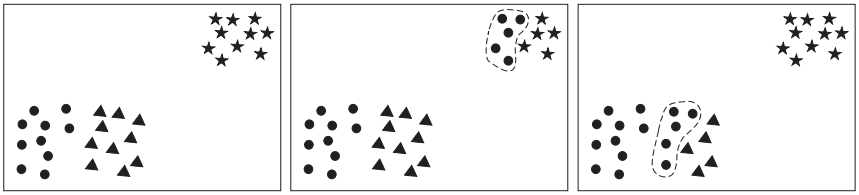
In the following two subsections, we provide background about the limitations of existing comparison measures and also explain further about the alternate clustering generation problem.

1.1 Problems with Existing Methods and Motivations

Whilst there already exist a number of clustering comparison measures, all of them use the *membership of points to clusters* as the primary factor in their similarity calculation. Although this can be an important determinant for clustering similarity, it neglects some other aspects. There are two main limitations faced by this type of approach:

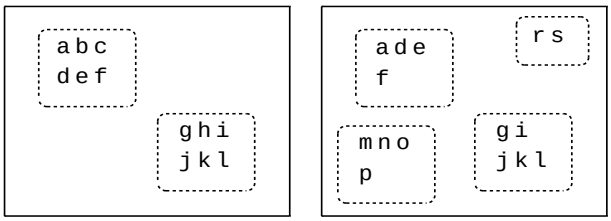
- **Inability to Detect Structural Dissimilarity** : We illustrate this using Figure 1, which shows three clusterings, each with three clusters (represented respectively by circles, triangles and stars). Figure 1(a) is the pre-defined (gold-standard) clustering and 1(b) and 1(c) are two other possible clusterings of the same data set. Suppose we wish to separately compare 1(a) to 1(b) and 1(a) to 1(c), to check which of 1(b) and 1(c) is more similar to 1(a). As indicated by the dotted lines, each of clusterings 1(b) and 1(c) has five points clustered differently compared to 1(a).

[1] A clustering is a set of clusters.



(a) pre-defined clustering (b) clustering P (c) clustering S

Fig. 1 3 clusterings, each containing 3 clusters.



(a) clustering A (b) clustering B

Fig. 2 Two clusterings, each for data from a different window of a data stream. Each letter represents a data object and the objects are assumed to arrive in alphabetical order. These two clusterings cannot be accurately compared using existing membership based measures.

Existing clustering measures, would report that the similarity between 1(a) and 1(b) is exactly the same as the similarity between 1(a) and 1(c) (in fact a similarity of 0.44 for both comparisons when using the the Rand index [5]).
However, we argue that 1(a) and 1(c) are actually more similar than 1(a) and 1(b). This is based on the following observations:

- The cluster centroids of 1(a) are more similar to the cluster centroids of 1(c), than they are to the cluster centroids of 1(b).
- The cluster shapes in 1(a) are more similar to the cluster shapes in 1(c), than they are to the cluster shapes in 1(b).
- Suppose we were to learn predictive models, one for each cluster, each summarising the cluster's common properties and being able to predict the likelihood of an unseen point being a cluster member. It is likely that the cluster models of 1(a) would be more similar to the cluster models of 1(c), than they would be to the cluster models of 1(b). This is a reflection of the fact that the spatial distribution statistics for each cluster share more similarity between 1(a) and 1(c), then they do between 1(a) and 1(b).

– **Inability to Compare Clusterings of Non-Overlapping Points** : One often needs to compare two clusterings, each derived from a different dataset. This is particularly true for evolving data, such as stream datasets for stock market information or network traffic information, where the data is divided into time based snapshots and clusterings of different snapshots are compared.
However, it is not possible to use existing membership-based comparison measures for clustering comparison in such scenarios. Consider Figure 2. Two clusterings are shown in 2(a) and 2(b). Assume each clustering uses data from a different time period (window) of the data stream. Let each letter represent a data object and suppose the objects arrive in alphabetical order. If existing membership-based com-

parison measures are used to evaluate the similarity between these two clusterings, they can only use the points common in both clusterings ('a', 'd', 'e', 'f', 'g', 'i', 'j', 'k', 'l') and would discount any of the points not present in both clusterings ('b', 'c', 'h', 'm', 'n', 'o', 'p', 'r', 's'), giving a misleading and inaccurate comparison result.

1.2 A Clustering Similarity Measure for Use in Discovering Alternate Clusterings

In existing literature, similarity measures for comparing clusterings have been used as important tools for performing external validation and for exploring degrees of similarity amongst *existing* clusterings.

However, in this paper, we also identify a novel and unexplored use of a clustering similarity measure - as a tool for alternate clustering generation. Here, the goal is to generate a distinctively different and high quality alternate clustering, given an input initial clustering. The key idea is that the clustering similarity measure itself can be used as an objective function to drive the creation of an alternate clustering.

Moreover, this task also motivates us to develop a new type of clustering constraint, we call the *distribution constraint*. This can be viewed as supplementary knowledge that can be used to drive the construction of a clustering, in an analogous way to must-link or cannot-link constraints [6].

We show that our *ADCO* similarity measure is well suited to these circumstances. In fact, using *ADCO* as an objective function allows the formulation of alternate clustering generation as an integer linear program of moderate complexity. The possibility of using other existing clustering similarity measures (e.g. Rand Index) for this task is an open question, but appears problematic, as the natural encoding would result in non linear integer programs, which are considerably more difficult to efficiently solve.

1.3 Contributions

Our main contribution in this paper is a new clustering similarity measure known as *ADCO* (**A**ttribute **D**istribution **C**lustering **O**rthogonality), which is designed to address the limitations of existing clustering comparison measures. By representing clusterings as *density profiles*, *ADCO* is able to take into account feature distribution information, as well as point membership information. At a more detailed level, our contributions are:

- **Detecting structural dissimilarity** : *ADCO* incorporates distribution information of data points along each attribute, considering the structures or density profiles of the clusters. This provides the ability to compare two clusterings in terms of their feature distributions, and this means that the comparison is a reflection of their similarity as "hypotheses", or as their similarity for deriving predictors. This type of comparison is not possible with existing membership based measures, such as the Rand [5] index or Jaccard index [7].
- **Comparing non-overlapping clusterings** : Since *ADCO* takes into account the density of points, it can achieve more flexibility in the type of clustering comparison. In particular, using *ADCO* it is possible to compare clusterings which do not share any common data points. This has important applications in situations where the data is evolving, such as stream clustering.

Table 1 Definitions of Five Existing Clustering Comparison Measures

Rand index (RI)	$RI(C,C') = \frac{N_{11}+N_{00}}{N_{11}+N_{10}+N_{01}+N_{00}}$
Jaccard index (JI)	$JI(C,C') = \frac{N_{11}}{N_{11}+N_{10}+N_{01}}$
Clustering error (CE)	$CE(C,C') = 1 - \frac{1}{n} max \sum_{k=1}^{K} n_{k,\sigma(k)}$
Variation of information (VI)	$VI(C,C') = 2H(C,C') - H(C) - H(C')$
Normalized Mutual Information (NMI)	$\phi^{NMI}(C,C') = \frac{2}{n} \sum_{l=1}^{K} \sum_{h=1}^{K'} n_l^{(h)} log_{K \times K'} \left(\frac{n_l^{(h)} n}{n^{(h)} n_l} \right)$

– **Application in alternate clustering algorithms** : We propose a novel algorithm, MAXIMUS, for alternate clustering generation, which uses the *ADCO* measure as an objective function. Given an input clustering, MAXIMUS discovers multiple alternate clusterings, each of high quality, yet each distinctively different. We compare MAXIMUS to other alternate clustering algorithms and demonstrate its capacity to efficiently generate high quality solutions.

2 Related Work

This paper is an expanded version of work in [1], which first described the *ADCO* measure. Compared to that work, this paper contains the following additional material: i) gives a deeper analysis of the philosophy behind *ADCO* and proves a number of formal properties of the measure, ii) presents a more comprehensive experimental analysis of its accuracy, compared to other measures, using more datasets, iii) analyses *ADCO*'s runtime complexity both formally and experimentally, iv) shows how *ADCO* can be used in a novel way, as an objective measure for a powerful new alternate clustering algorithm called MAXIMUS.

There are three main types of existing clustering comparison methods, which are discussed below and also summarized in Table 1.

– **Pair counting** : Methods in this category are based on counting pairs of points and comparing the 'agreement' and the 'disagreement' between two clusterings. Pairs of points are classified into four types - N_{11}, N_{10}, N_{01} and N_{00} - where N_{11} is the number of pairs of points which belong to the same cluster in both clusterings, N_{10} and N_{01} are numbers of pairs which belong to the same cluster in one of the clusterings but not the other, and N_{00} is the number of point pairs belonging to different clusters in both clusterings. N_{11} and N_{00} are treated as 'agreements' and N_{10} and N_{01} are treated as 'disagreements' between the two clusterings. Popular pair counting methods are the Rand index [5] and Jaccard index [7] (defined in Table 1) and also the Wallace indices [8] and their extensions [9,10].
– **Set matching** : This category of methods is based on measuring the shared set cardinality between two clusterings. The simplest form of set matching technique is called 'clustering error' [11], which is defined in Table 1 (where n is the number of objects and K is the number of clusters in each clustering, and $n_{k,\sigma(k)}$ finds the 'best match' between clusters); it computes the best matches between clusters (in terms of shared points) from each of the two clusterings. It returns a value equal to the total number of points shared between pairs of matched clusters. Other related techniques have also been developed by Larsen [12] and Meila [13].

– **Information theoretic measures** : Examples of these are the (NMI) [14] and 'variation of information' (VI) measures [15]. Both measures utilize the mutual information between two clusterings, which is determined by the conditional probabilities resulting from the number of points shared between clusters of the two clusterings. The mutual information essentially signifies the amount of information one clustering provides about the other. While NMI normalizes the mutual information with the sum of the two clusterings' entropies, VI uses a different comparison criterion to give the final value. (NMI and VI are defined in Table 1, where n is the number of objects, $n_l^{(h)}$ indicates the number of points shared by l-th cluster of C and h-th cluster of C', n_l indicates the number of points in l-th cluster of C and $n^{(h)}$ indicates the number of points in h-th cluster of C'.)

Clustering comparison methods are frequently applied as part of the process of *ensemble clustering*, in which several clusterings are merged to form a consensus clustering. A popular technique for merging is called 'majority voting' [16], which is a pair-counting method extended over multiple clusterings. In [17], a 'hypergraph partitioning algorithm' [18] is applied to find a consensus clustering where the underlying idea is to find dense intersections between the clusterings based on point membership information. Comparison methods are also used in stream data clustering [19, 10], where clusters are generated and consistently evolve as new data arrives. The idea is that studying this evolution can uncover valuable information and detect sudden structural changes within the data. In [19], clusterings at different time periods are compared by observing any newly formed, removed or modified clusters. The technique used is membership-based and assumes that the clusterings have at least some non-empty overlap of data points. However, it would not work if the two clusterings share no common points at all. Indeed in this circumstance, the existing comparison measures are not applicable.

We will discuss related work about alternate clustering generation in Section 7.1.

3 The *ADCO* Similarity Measure

Let us introduce the following terminology to describe *ADCO*. Let D be a data set of N objects containing R attributes. Also assume $C = \{c_1, \ldots, c_K\}$ and $C' = \{c'_1, \ldots, c'_{K'}\}$ are two (hard) clusterings that are to be compared. The *ADCO* similarity value between the two clusterings is denoted as $ADCO(C, C')$, where higher values of the measure indicate higher similarity (less dissimilarity).

3.1 Defining *ADCO*

The *ADCO* measure determines the similarity between two clusterings based on their density profiles along each attribute. Essentially, each attribute's range is divided into a number of intervals, and the similarity between two clusters corresponds to how closely the point sets from each cluster are distributed across these intervals. The similarity between two clusterings then corresponds to the amount of similarity between their component clusters. We begin by defining some terms that we need for describing density.

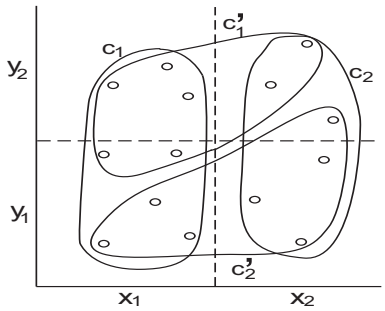


Fig. 3 Two Clusterings $C = \{c_1, c_2\}$ and $C' = \{c'_1, c'_2\}$ with attributes X and Y each binned into 2 intervals

Definition 1 *Given an attribute/feature space $A = \{a_1, a_2, .., a_R\}$, let the range of each attribute a_i be divided into Q bins. An **attribute-bin region** is a pair denoted as (i, j), which corresponds the j-th bin of the i-th attribute. (So there are a total of RQ regions.) The **density of an attribute-bin region** (i, j) is denoted as $dens(i, j)$ and refers to the number of points in that region expressed as*

$$dens(i, j) = |\{d \in D \mid d[a_i] \in (i, j)\}| \tag{1}$$

*where $d[a_i]$ is the projection of instance d on attribute a_i. Additionally, the **density of an attribute-bin region for cluster** c_k **in clustering** C, denoted as $dens_C(k, i, j)$, refers to the number of points in the region (i, j), which belongs to the cluster c_k of clustering C.*

The values of $dens_C(k, i, j)$ for all possible k, i, j form the building blocks of a clustering's 'density profile vector'; in the vector those values are listed in a lexicographical ordering imposed on all attribute-bin regions. For example, in Figure 3, the data set contains two attributes X and Y, which are both divided into two bins. The ordering applied to its attribute-regions is $(X, x_1), (X, x_2), (Y, y_1), (Y, y_2)$. The density profile vector is generated using the density profile function defined below.

Definition 2 *The **density profile** of a clustering C is the following **density profile vector** of C:*

$$\begin{aligned} V_C = (&dens_C(1,1,1), dens_C(1,1,2), .., dens_C(1,1,Q), dens_C(1,2,1), .., \\ &dens_C(1,R,Q), dens_C(2,1,1), .., dens_C(K,R,Q)) \end{aligned}$$

For example, in Figure 3, the density profiles of clusterings C and C' are $V_C = (8, 0, 5, 3, 0, 6, 3, 3)$ and $V_{C'} = (5, 2, 2, 5, 3, 4, 6, 1)$.

Suppose C and C' are clusterings with respectively K and K' clusters. We use the following formula on their density profile vectors to determine the degree of similarity between C and C':

$$sim(C, C') = V_C \cdot \rho(V_{C'}) \tag{2}$$

$$= \max_{\rho} \sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j) \times dens_{C'}(\rho(k), i, j) \tag{3}$$

where ρ ranges over permutations over the cluster IDs of C' and $K_{min} = \min(K, K')$. By considering all possible permutations ρ, we consider all possible pairings of clusters and select the maximum dot product value corresponding to the best match. The best cluster match may not be the match where the kth cluster in C is matched with the kth cluster in C'. This ensures that the similarity is independent of the assigned cluster labels. For example, in Figure 3, the pairing of (c_1, c'_1) and (c_2, c'_2) gives a higher value (110) compared to the value (90) given by the pairing (c_1, c'_2) and (c_2, c'_1). Regarding computation, rather than iterating through all possible permutations of C', we can consider equation 2 as an assignment problem between clusters of C and clusters of C' where the aim is to maximize the scalar product value. This can be solved in polynomial time of $O(K_{min}^3)$ by the widely used Hungarian algorithm [20] for solving assignment problems; the input to the algorithm is a $K \times K'$ cost matrix M between C and C', where the (k, k')-element is the scalar product of the density values between the kth cluster of C and the k'th cluster of C'.

We note that $sim(C, C) = \sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j)^2$. Indeed, for arbitrary permutations ρ, we have

$$\sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j)^2 \geq \sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j) \times dens_C(\rho(k), i, j)$$

by the following reasoning:

$$\begin{aligned} 0 &\leq \sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=1}^{Q} (dens_C(k, i, j) - dens_C(\rho(k), i, j))^2 \\ &= \sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=1}^{Q} (dens_C(k, i, j)^2 + dens_C(\rho(k), i, j)^2 - 2dens_C(k, i, j) * dens_C(\rho(k), i, j)) \\ &= \sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=1}^{Q} (2dens_C(k, i, j)^2 - 2dens_C(k, i, j) * dens_C(\rho(k), i, j)) \end{aligned} \tag{4}$$

which proves that $sim(C, C) = \sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j)^2$. Lastly, we also need a normalization factor for computing the $ADCO$ measure, which corresponds to the maximum achievable similarity when using either of the two clusterings. This is given in equation 5. The $ADCO(C, C')$ measure is then shown in equation 6.

$$NF(C, C') = \max\left[sim(C, C), sim(C', C')\right] \tag{5}$$

$$ADCO(C, C') = \frac{sim(C, C')}{NF(C, C')} \tag{6}$$

The value of $ADCO$ ranges from 0 to 1, where a lower value indicates higher dissimilarity and a higher value indicates higher similarity. For the example in Figure 3, $NF(C, C') = \max[152, 120] = 152$ and $ADCO(C, C') = \frac{110}{152} = 0.72$.

Algorithm 1 summarizes the calculation steps of $ADCO$. When clusterings C and C' do not share the same number of clusters, $ADCO$ simply finds the best matching, similar to the clustering error metric described earlier. Note that by varying the Q parameter (the number of bins), one can trade off between the granularity of the density profile and the complexity of computing the $ADCO$ value. We have found

$Q = 10$ to work well as suggested in [21]. Unless mentioned otherwise, we assume it as a default setting that will be used throughout the rest of the paper.

The method of defining the bins can be determined by any discretization techniques such as 'maximal marginal entropy' [22], 'StatDisc' [23] or those outlined in [24–26]. For $ADCO$, we have applied a simple equi-width method, which divides an attribute range into equi-width intervals. In Section 4.4, we further investigate the effect of discretization choice on the $ADCO$ value.

Algorithm 1 Calculation steps for $ADCO$

1: Scan the clusterings to compute V_C and $V_{C'}$
2: $sim(C, C') = HungarianAlgorithm(C \cdot C')$ {the best matching pairs of clusters are determined by the Hungarian algorithm [20]}
3: $NF = \max[sim(C, C), sim(C', C')]$
4: $ADCO(C, C') = \frac{sim(C,C')}{NF}$

3.2 Vector Interpretation of $ADCO$

We now present a geometric interpretation of the $ADCO$ measure.

In our model each clustering C has been represented as a vector V_C. By overloading notation, we will now use both C and its density profile V_C synonymously. We will also use the notation $|C|$ to denote the (Euclidean) magnitude of (the density profile vector of) clustering C. Observe that $|C| = \sqrt{sim(C, C)}$.

Consider two clusterings, $C1$ and $C2$, where $|C2| \geq |C1|$ and the angle between $C1$ and $C2$ is x. Then we can derive the following expression for $ADCO$:

$$\begin{aligned} ADCO(C1, C2) &= \frac{sim(C1, C2)}{NF(C1, C2)} \\ &= \frac{sim(C1, C2)}{sim(C2, C2)} \\ &= \frac{sim(C1, C2)}{|C2|^2} \\ &= \frac{C1 \cdot C2}{|C2|^2} \quad \text{(For ease of explanation, we disregard any permutation } \rho\text{, with the dot product)} \\ &= \frac{|C1||C2|cos(x)}{|C2|^2} \\ &= \frac{|C1|cos(x)}{|C2|} \end{aligned} \tag{7}$$

We can interpret this result in terms of vector algebra. Clustering $C1$ can be expressed as the sum of two component vectors. One of them parallel to $C2$ and the other orthogonal to $C2$. The magnitude of the former is known as the scalar projection of $C1$ onto $C2$. This is shown in Figure 4, where we see that the numerator $|C1|cos(x)$ of equation 7 is equal to the scalar projection of $C1$ onto $C2$. The denominator of equation 7 is equal to the length of $C2$. Hence $ADCO(C1, C2)$ measures the ratio between the scalar projection of $C1$ onto $C2$, and the magnitude of $C2$. In other words, $ADCO$

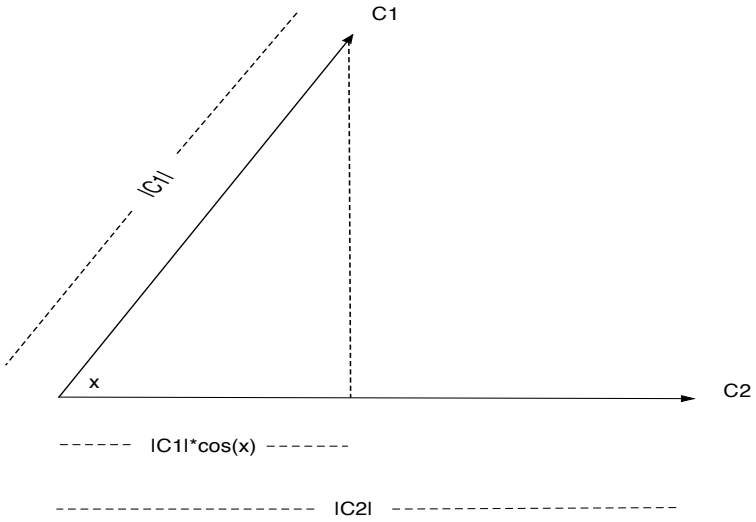


Fig. 4 Interpreting *ADCO* in terms of scalar vector projection. The *ADCO* value equals length of the projection of $C1$ onto $C2$, divided by the length of $C2$. i.e. $|C1|cos(x)/|C2|$

is assessing a type of containment judgement between $C1$ and $C2$, which might be expressed as *"How much of clustering $C2$ is contained in clustering $C1$?"* or , *"What percentage of clustering $C2$ is contained in clustering $C1$?"*.

Interestingly, the use of containment judgements for measuring similarity has been studied in the field of psychometrics. Work in [27] used similar kinds of measures in a study where subjects were being asked to assess the similarity between different multidimensional stimuli. Subjects were asked questions such as "How much of this blue is contained in this red ?". The broader use of containment measures for similarity in psychometrics is also discussed in [28] and [29].

3.3 Mathematical Properties of *ADCO*

We next establish a number of mathematical properties of *ADCO*, investigating how it behaves as a similarity measure and also how it might be transformed into a distance function.

We begin by analysing its properties as a similarity function:
Non-negativity : $ADCO(C, C') \geq 0$ for any two clusterings C and C'.

Proof : The similarity of C and C' is calculated using equations 2 and 5, involving the density profile vectors V_C and $V_{C'}$. Since values in density profile vectors cannot be negative, the product of values in equations 2 and 5, and hence the resultant $ADCO$ value, cannot be negative. □

Identity of indiscernibles : We show this subject to some extra assumptions about what it means for two clusterings to be indiscernible or equivalent. Let C and C' be clusterings where both C and C' have K clusters for some $K > 0$. The density profiles V_C and $V_{C'}$ are said to be equivalent if there is a permutation ρ over $\{1, ..., K\}$ such that $dens_C(k, i, j) = dens_{C'}(\rho(k), i, j)$ for all i, j, k. We need to prove that V_C and $V_{C'}$ are equivalent iff $ADCO(C, C') = 1$.

Proof : (If): Suppose V_C and $V_{C'}$ are equivalent. Then

$$\sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j)^2 = \max_{\rho} \sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j) \times dens_{C'}(\rho(k), i, j)$$

Hence $sim(C, C) = sim(C, C')$. Similarly, $sim(C', C') = sim(C, C')$.
Therefore $max\{sim(C, C), sim(C', C')\} = sim(C, C')$ and $ADCO(C, C') = 1$.
(Only-if): Assume that $|C'| \geq |C|$. If $ADCO(C, C') = 1$, then $V_C \cdot \rho(V_{C'}) = V_{C'} \cdot V_{C'}$. Now for any C and any specific permutation ρ_n, we have $V_C \cdot V_C = \rho_n(V_C) \cdot \rho_n(V_C)$. So we can therefore conclude that there is some permutation ρ_n for which $V_C = \rho_n(V_{C'})$. This means that V_C and $V_{C'}$ are equivalent. □

Clearly there are many clusterings whose density profiles are equivalent to each other. The $ADCO$ measure will treat these clusterings as indiscernible. In contrast, the measures described in Section 2 will only consider two clusterings as equal if memberships of points to clusters in the two clusterings are identical. We will discuss this further in Section 3.4.

Symmetry : $ADCO(C, C') = ADCO(C', C)$ for any two clusterings C and C'.

Proof : It suffices to show that $sim(C, C') = sim(C', C)$. Let ρ be a permutation satisfying

$$sim(C, C') = \sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j) \times dens_{C'}(\rho(k), i, j)$$

Then

$$\sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j) \times dens_{C'}(\rho(k), i, j) =$$
$$max_{\rho'} \sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_C(k, i, j) \times dens_{C'}(\rho'(k), i, j)$$

It is easy to see that ρ^{-1} is a permutation satisfying $\sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_{C'}(k, i, j) \times dens_C(\rho^{-1}(k), i, j) = max_{\rho'} \sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_{C'}(k, i, j) \times dens_C(\rho'(k), i, j)$. So

$$sim(C', C) = \sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_{C'}(k, i, j) \times dens_C(\rho^{-1}(k), i, j) = sim(C, C')$$

□

Using *ADCO* as the basis for a distance function: We next consider how one might employ a modification of *ADCO* as a distance function. The distance between two clusterings should monotonically decrease as their similarity increases.

Consider the following natural proposal for a distance function based on *ADCO*.

$$D_{ADCO}(C1, C2) = 1 - ADCO(C1, C2) \tag{8}$$

For D_{ADCO} to be a metric, the following properties are required to hold:

1. $D_{ADCO}(C1, C2) \geq 0$
2. $D_{ADCO}(C1, C2) = D_{ADCO}(C2, C1)$
3. $D_{ADCO}(C1, C2) = 0$ iff $C1 = C2$
4. $D_{ADCO}(C1, C2) \leq D(C1, C3) + D_{ADCO}(C3, C2)$

Properties 1,2 and 3 can be straightforwardly deduced from the nonnegativity, symmetry and identity of indiscernibles that we have already proved for *ADCO*. Property 4 is not true in general though and a counter example is (for profiles with 1 attribute and 4 bins): $C1 = (3, 0, 0, 4)$, $C2 = (3, 3, 0, 1)$ and $C3 = (1, 0, 2, 4)$.

We can, however, use standard techniques from multidimensional scaling (see e.g. [30] and [29]) to modify D_{ADCO} so that it does satisfy the triangle inequality and thus become a metric. The basic idea is to increase the value of the distance between all discernible clusterings by some constant amount (in this case one further unit). This effectively 'repairs' the violations of the triangle inequality that occurred for D_{ADCO}. The revised distance function can be defined as follows:

$$\begin{aligned} D'_{ADCO}(C1, C2) &= 2 - ADCO(C1, C2), \quad \text{if } C_1 \neq C_2 \ \ (V_{C1} \neq V_{C2}) \\ &= 0, \qquad\qquad\qquad\qquad\quad \text{otherwise} \end{aligned}$$

Theorem 1 *D'_{ADCO} is a metric*

Proof: See appendix.

3.4 Philosophy behind the *ADCO* Measure

Now that we have defined and established some formal properties of the *ADCO*, we now discuss further about the philosophy behind the *ADCO* function and how it compares to existing measures.

Our belief is that the user's goal of clustering will drive the type of clustering similarity measure used.

The membership based measures that we have discussed have a partition based philosophy. This corresponds to a user clustering goal where the aim is to derive partitions of objects (satisfying various measures). So each clustering is represented as a partition of objects and two clusterings are judged to be similar if their partition representations are similar.

The *ADCO* similarity function takes a different approach, in line with a more "data mining", or prediction model based philosophy to clustering. Consider the following quote from [31] that helps explain the spirit of this philosophy:

> A clustering is a hypothesis to suggest (or explain) groupings in the data. ... It becomes a model for the data and can potentially constitute a mechanism to classify unseen instances of the data.

Here, the user views a clustering as a set of prediction (decision) functions or a hypothesis generator. Clustering is performed to group objects. The clusters then correspond to groups sharing common factors (features) of the data. For each cluster (group), one can summarise it by inferring rules, in order to suggest specialized models. Under this philosophy, *two clusterings are considered to be similar if they are likely to give rise to similar types of prediction models*. Measuring this kind of similarity clearly requires knowledge about the feature space, since it is the basis for expressing and developing prediction models.

Such a philosophy to clustering is also in line with the *data recovery approach to clustering* [32], in which one first uses the observed data to form clusters, and then uses these clusters as a basis to recover the unobserved data.

We next provide a brief toy example to further illustrate the difference between clustering comparison using a partition based philosophy and clustering comparison using a prediction model based philosophy.

Example 1 *Consider the dataset*

Object ID	Name
1	*Bob*
2	*Bob*
3	*Alice*
4	*Alice*

which has four instances (objects) and a single attribute Name. *Assume that* Object ID *here is just listed for convenience and should not be regarded as an underlying attribute (feature) of the data.*

Let clustering $C = \{c_1, c_2\}$, *where* $c_1 = \{1, 3\}$ *and* $c_2 = \{2, 4\}$. *Let clustering* $C' = \{c'_1, c'_2\}$, *where* $c'_1 = \{2, 3\}$ *and* $c'_2 = \{1, 4\}$. *If using the ADCO measure, then* $ADCO(C, C') = 1$, *because the two clusterings are indistinguishable when just using the* Name *attribute to describe points. i.e. Both* C *and* C' *are considered to be equivalent in terms of the type of prediction model(s) that could be derived from the clusterings.*

On the other hand, if using a membership (partition) based similarity measure (e.g. the Rand Index), then $sim(C, C') = 0.33$. *This indicates that the two clusterings are rather dissimilar, since the mixes of objects (object IDs) in each cluster vary between the clusterings.*

In general, there there a number of factors for the user to consider, in deciding what type of clustering similarity measure to choose:

- *Flexibility:* Can the measure be used to compare clusterings of different datasets ? Can it be used to compare clusterings which have been reduced or summarized in some way (e.g. by sampling points) ?
- *Soundness as a Partition Measure:* Given two clusterings (partitions) C and C' of a collection of objects $\mathcal{O}$, is it true that $(Sim(C, C') = 1) \Leftrightarrow (C = C')$?
- *Complexity:* What is the runtime complexity of the measure (in terms of the numbers of points N, of clusters K, of attributes R and of bins Q) ? How does it scale for large datasets ?

Table 2 Comparison of *ADCO* with Membership-based Measures

	ADCO Measure	Membership Based Measures
Philosophy	Data mining	Partitioning
Flexibility	High	Low
Soundness	No	Yes
Complexity	$O(NRQ) + O(K_{min}^3)$	$O(N^2K^2)$
Objective Function Potential	Good	Unclear

– *Use as an Objective Function:* Can the similarity measure be used as the basis for an objective function within the problem of alternate clustering ?

In Table 2, we classify *ADCO* and the membership based measures according to these factors. We see that, relatively speaking, advantages of *ADCO* are 1) High flexibility, 2) Good complexity in terms of N, the number of data points (discussed further in Section 5) and 3) clear ability to be used as an objective function in alternate clustering. A disadvantage of *ADCO* is that it is not sound as a partition measure.

Observe that there is an inherent tension, between a measure possessing partition soundness and a measure possessing flexibility. Achieving flexibility means that properties of the feature space must be used, which in turn means that a more general definition is required for what it means for two clusterings to be "equal" (i.e. $C = C'$ doesn't just mean that C and C' are equal partitions).

3.5 Relationship to Cosine Similarity and the Handling Bias in Dataset Sizes

We next discuss the relationship of *ADCO* to cosine similarity and also consider how to use *ADCO* appropriately when comparing clusterings taken from datasets of different sizes.

Cosine similarity is a widely used similarity measure in information retrieval for finding a relevant set of documents given a query. It is calculated by the scalar product between two vectors (i.e. representing a query and a document) normalized by the magnitudes of the vectors. Given two vectors $C1$ and $C2$, the cosine similarity is given as

$$\begin{aligned} COSINESIM(C1, C2) &= \frac{C1 \cdot C2}{|C1||C2|} \\ &= cos(x) \quad \text{(where } x \text{ is the angle between } C1 \text{ and } C2) \end{aligned}$$

Thus, this measure evaluates the cosine of the angle between two vectors. i.e. The extent to which the vectors are pointing in the same direction (regardless of their magnitudes).

Let us compare this against our expression for *ADCO* from equation 7. That can be written as $ADCO(C1, C2) = (|C1|/|C2|) \times cos(x)$ (Assuming $|C2| \geq |C1|$). So mathematically speaking $ADCO(C1, C2) = (|C1|/|C2|) \times COSINESIM(C1, C2)$ and since $(|C1|/|C2|) \leq 1$, we have $ADCO(C1, C2) \leq COSINE(C1, C2)$. It can be seen from this that *ADCO* computes both the (cosine of) the angle between $C1$ and $C2$, as well as the ratio of their magnitudes. In contrast, $COSINESIM$ only computes the (cosine of) the angle. From a user perspective, the value $|C_i|$, which is the magnitude of clustering C_i, measures the degree of imbalance of C_i. If C_i has some highly dense

regions (e.g. its instances are densely concentrated into a few components of the density profile), then $|C_i|$ will be very high. Conversely, if the instances are near uniformly spread across clusters and bins, then $|C_i|$ will be low. Thus, the ratio $|C1|/|C2|$ might be viewed as a kind of balance ratio between the two clusterings. It assesses the degree to which they differ in terms of their overall density concentrations.

Example 1 Consider the following simple example illustrating the difference between $ADCO$ and $COSINESIM$. We have three clusterings, with their density profiles using 1 cluster, 1 attribute and 10 bins.

$C1 = (800, 10, 10, 10, 10, 10, 10, 10, 10, 10)$
$C2 = (700, 40, 0, 0, 0, 60, 40, 0, 0, 0, 60)$
$C3 = (400, 50, 50, 50, 50, 50, 50, 50, 50, 50)$

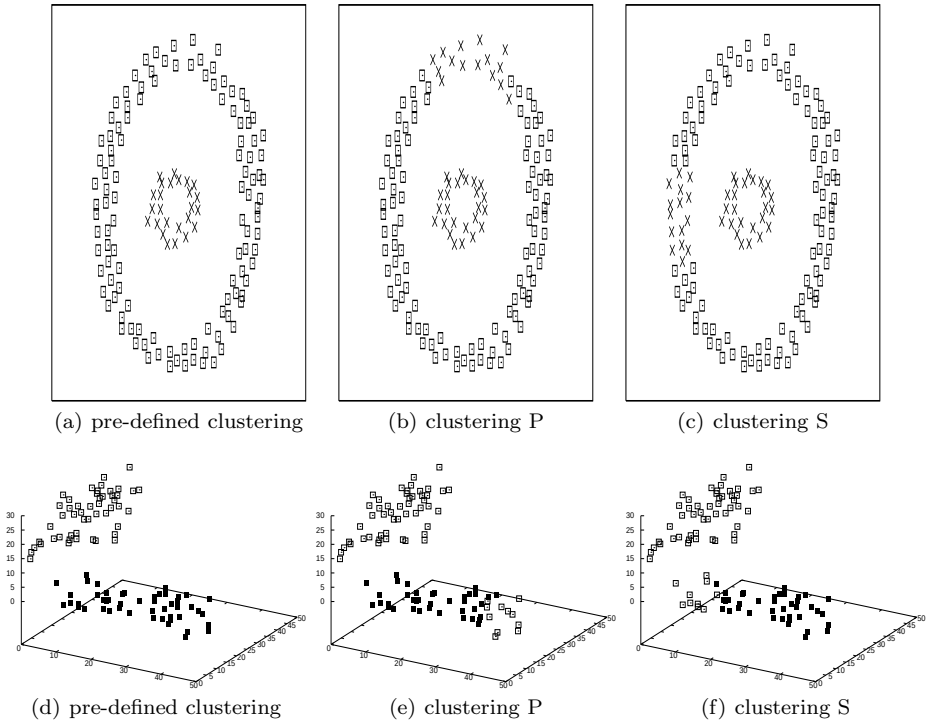
We have $ADCO(C1, C2) = 0.88$ and $ADCO(C1, C3) = 0.51$. So $ADCO$ recognises a significant difference in the two similarities. This seems reasonable and intuitive, based on the large difference between the count for the first density dimension in $C1$ versus its count in $C3$. In contrast, $COSINESIM(C1, C2) = 0.99$ and $COSINESIM(C1, C3) = 0.94$. So although there is the same ordering between similarities, $COSINESIM$ is not very sensitive to the large difference in density concentrations between $C1$ and $C3$. □

Another difference between $ADCO$ and $COSINESIM$ is the use of permutations. $ADCO$ computes a dot product based on finding a vector permutation to produce a good alignment between the different sets of clusters. The $COSINESIM$ measure does not use any permutations, since it was not originally developed with clustering comparison in mind. However one might envisage extending $COSINESIM$ to also allow permutations.

We next note a possible pitfall in using the $ADCO$ measure. We have stated that $ADCO$ may be used for comparing clusterings for different datasets. If each clustering contains the same number of instances, then there is no difficulty. However, if one clustering contains many more instances than the other, then the comparison may suffer from bias.

For example, if we compare a clustering $C1$ with density profile $(1, 1)$ against a clustering $C2$ with density profile $(100, 100)$, then the $ADCO$ value will be 0.01. Although this is understandable from the containment perspective we discussed in Section 3.2 "There is 1% of clustering $C2$ in clustering $C1$", it may not be a fair measure of comparison, due to the bias in magnitudes between the clusterings. Instead, the user may only wish to know about how similar $C1$ and $C2$ are in terms of the directions of their density profile vectors. To address this difficulty, there are a couple of natural possibilities. If $C1$ and $C2$ contain different numbers of instances, we can either

- Use a measure like $COSINESIM$ to measure the similarity between $C1$ and $C2$. This is insensitive to the magnitudes of $C1$ and $C2$. Or
- For the clustering which has more instances (say $C1$), perform stratified sampling across its clusters, to yield a new clustering $C1'$ which has the same number of instances as $C2$. Then compute $ADCO(C1', C2)$.



(a) pre-defined clustering (b) clustering P (c) clustering S

(d) pre-defined clustering (e) clustering P (f) clustering S

Fig. 5 Two groups of clusterings on synthetic data sets

4 Experiments to Evaluate the *ADCO* Measure

We carried out an experimental analysis for evaluating the behavior of *ADCO* on both synthetic and real world data sets. We compared *ADCO* against four existing clustering comparison measures : Rand index [33], Jaccard index [7], clustering error [11] and variation of information [34] (described in section 2). We emphasize here that the standard experimental methodology for validating such comparison metrics is to apply them to a set of clusterings and accompany the returned quantitative results with visual 2D projections of the data sets [7,33,35]. While such visual aids cannot qualitatively authenticate the comparison, they do offer a general 'feel' of the similarity between clusterings and they allow the reader to make an intuitive judgement about how well the measures performed.

All comparison measures, except variation of information, define the similarity to be between 0 and 1, where a higher value indicates higher similarity between clusterings; for variation of information, a higher value signifies higher dissimilarity.

4.1 Synthetic data sets

Three synthetic data sets are shown in Figures 1 and 5; each data set is associated with three clusterings: a pre-defined clustering plus two others (clusterings P and S). The number of data points clustered differently in P and S compared to the pre-

Table 3 Comparing clusterings in figures 1 and 5

Comparisons	ADCO	Rand index	Jaccard index	Clustering error metric	Variation of information
figures 1(a) vs. 1(c)	**0.84**	0.83	0.59	0.83	0.16
figures 1(a) vs. 1(b)	**0.53**	0.83	0.59	0.83	0.16
figures 5(a) vs. 5(c)	**0.94**	0.56	0.55	0.68	0.15
figures 5(a) vs. 5(b)	**0.67**	0.56	0.55	0.68	0.15
figures 5(d) vs. 5(f)	**0.92**	0.82	0.85	0.90	0.52
figures 5(d) vs. 5(e)	**0.90**	0.82	0.85	0.90	0.52

Table 4 Characteristics of data sets used in experiments

data sets	instances	classes	dimensions
Credit	1000	2	10
Diabetes	768	2	8
Eucalyptus	736	5	19
Glass	798	7	10
Ionosphere	351	2	34
Vehicle	946	4	18

defined clustering is the same. As argued earlier in section 1.1, in Figure 1, the pair of clusterings in 1(a) and 1(c) are more similar than the pair 1(a) and 1(b). Similarly in Figure 5, 5(a) and 5(c) are more similar than are 5(a) and 5(b). Lastly, clustering 5(f) is more similar to 5(d) than 5(e) is to 5(d). Table 3 shows the values returned by the similarity measures. We note that *ADCO* is the only measure that can recognise the variation in similarity in each of those pairs; all the other measures fail to distinguish the clustering comparisons, returning the same value for each of those pairs.

4.2 Real World data sets

We used six real world data sets (credit, diabetes, eucalyptus, glass, ionosphere and vehicle) taken from the UCI machine learning repository [36]; their characteristics are shown in Table 4.1. Each data set came with pre-defined class labels, which can be used as the pre-defined clustering. We used three different clustering algorithms to cluster these data sets, after first removing the class labels: K-Means, EM and average linkage (henceforth referred to as AL). Together, these algorithms span several different approaches to clustering, representing partitional, model-based and hierarchical techniques respectively. We then compared the clusterings discovered by these algorithms to the pre-defined clustering, using our five clustering comparison measures. The results of these comparisons are given in Table 5, with accompanying Figures 6, 7, 8, 9, 10 and 11, which show 2-D projections of each of the three clusterings of each data set. The features of these projections were selected manually by human judgement. Note that for some data sets, portions of the data were removed when plotting the figures, for readability.

Consider Table 5, which shows the results of all clustering comparisons for these data sets. Looking first at the results for data set 'Credit', the *ADCO* values imply that both the K-Means and the EM algorithms generated clusterings which were highly similar to the pre-defined clustering, whereas the result of the AL algorithm was rather

less similar to the pre-defined clustering. Looking at a 2-D visualization of this data set in Figure 6, we can indeed verify that the output of Figure 6(d) (generated by AL) is significantly different compared to the pre-defined clustering in 6(a) since one of its clusters (cluster 2) has many more points than the other, while 6(b) and 6(c) are considerably more similar to 6(a). Thus, the *ADCO* values seem to provide an intuitive picture of the relative similarities and dissimilarities. On the other hand, the values of the other measures fail to recognize these aspects. In fact, they provide a rather different picture, implying that the AL clustering is closer to the pre-defined clustering than the EM clustering.

Looking next in Table 5 at the data set 'diabetes', the *ADCO* values indicate that the comparison between the pre-defined clustering and that of EM is higher than the comparison between AL and the pre-defined clustering. This is intuitively reasonable, looking again at the 2-D projections in Figure 7 (the AL clustering there has clusters that are not as well separated as the pre-defined). However, the Rand and Jaccard measures are not able to distinguish the two comparisons and return similar values for both. This type of anomaly was made by all the membership based measures at least once in all data sets (by CE for 'eucalyptus', by JI for 'glass', by CE for 'vehicle'). Figures 8, 9, 10 and 11 give 2-D projections for these other data sets - glass, ionosphere and vehicle. Broadly speaking, the *ADCO* values correspond with intuition about two clusterings being similar when the shapes and distributions of their clusters are similar. In contrast, the values of the three membership based measures are often difficult to interpret, since they cannot recognize differences in point feature distributions, and they are only able to recognize differences in point memberships.

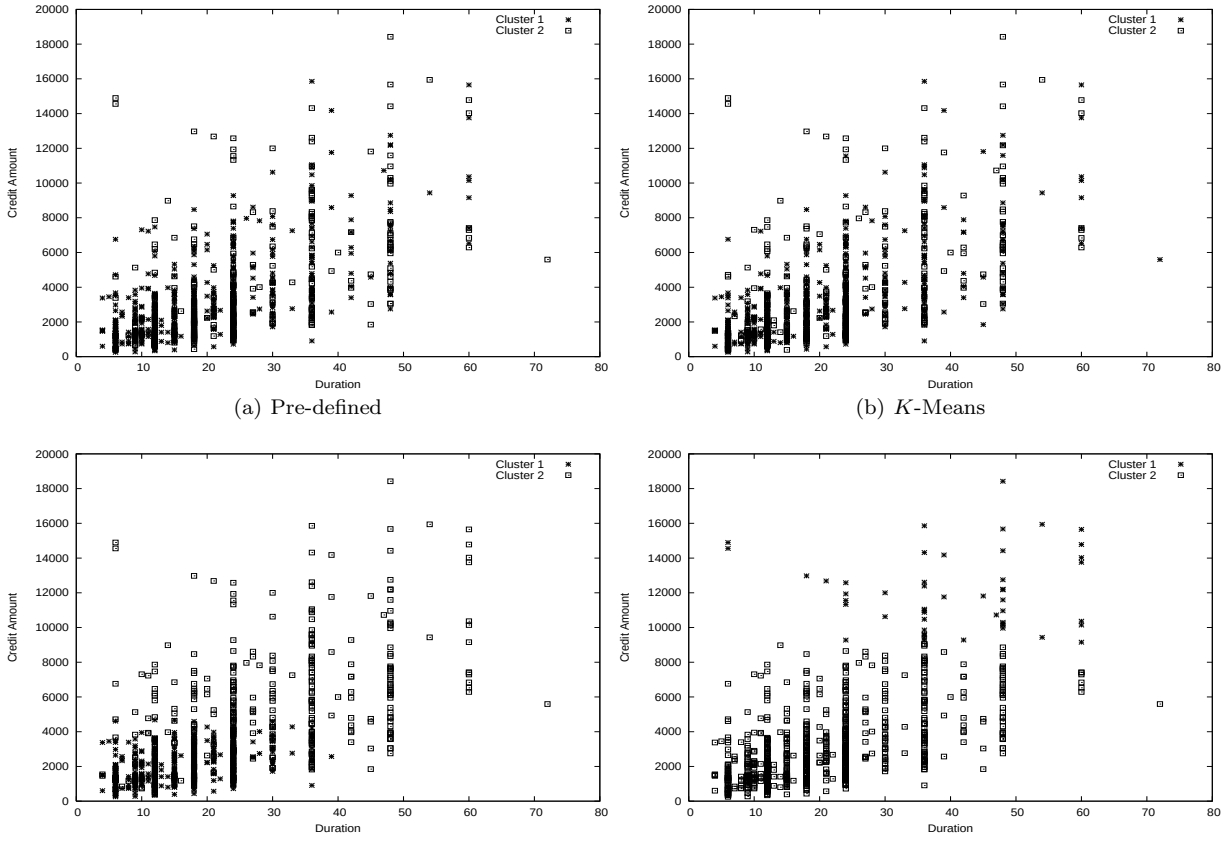
4.3 Stream Clustering Analysis

As mentioned in section 1.1, clustering similarity measures can be very useful in scenarios where data changes over time and clusterings at different time periods need to be compared to analyze the characteristics of data evolution. However, existing similarity measure are not applicable in this type of situation, since they require both clusterings to be over exactly the same set of data points. We tested the power of *ADCO* in this kind of scenario on the 'KDD network intrusion detection' data set, taken from the UCI repository [36].

This data set contains approximately 300,000 objects, 41 attributes and 38 classes (intrusion types). Treating it as a data stream, we took four contiguous snapshots, each over a different time period. Each snapshot contained 1000 points, which were clustered according to the different intrusion types (i.e. one clustering for each snapshot, with each cluster corresponding to an intrusion type). We ensured there were no common points between any of the snapshots. The mix of different intrusion types for each clustering contained is provided in Table 6, where the proportion of points belonging to each type is shown as a percentage. The snapshots were extracted in such a way that some shared similar distributions of intrusion types, while others were different.

Our objective was to test the following intuition: The *ADCO* value should be relatively high if the two clusterings contain similar distributions of intrusion types. On the other hand, if the two clusterings contain very different mixes of intrusion types, then the *ADCO* value should be relatively low.

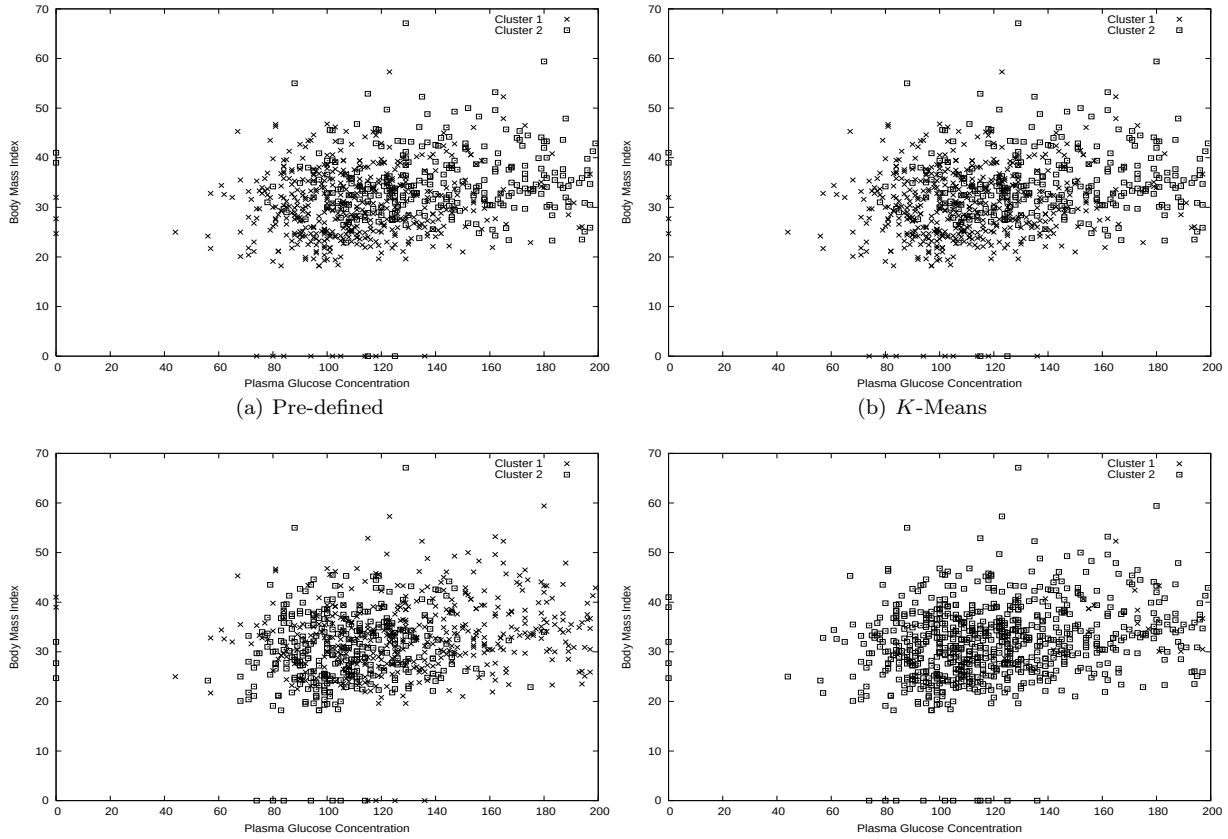
The results of our comparisons are presented in Table 7.



(a) Pre-defined (b) K-Means

(c) EM (d) Average Linkage

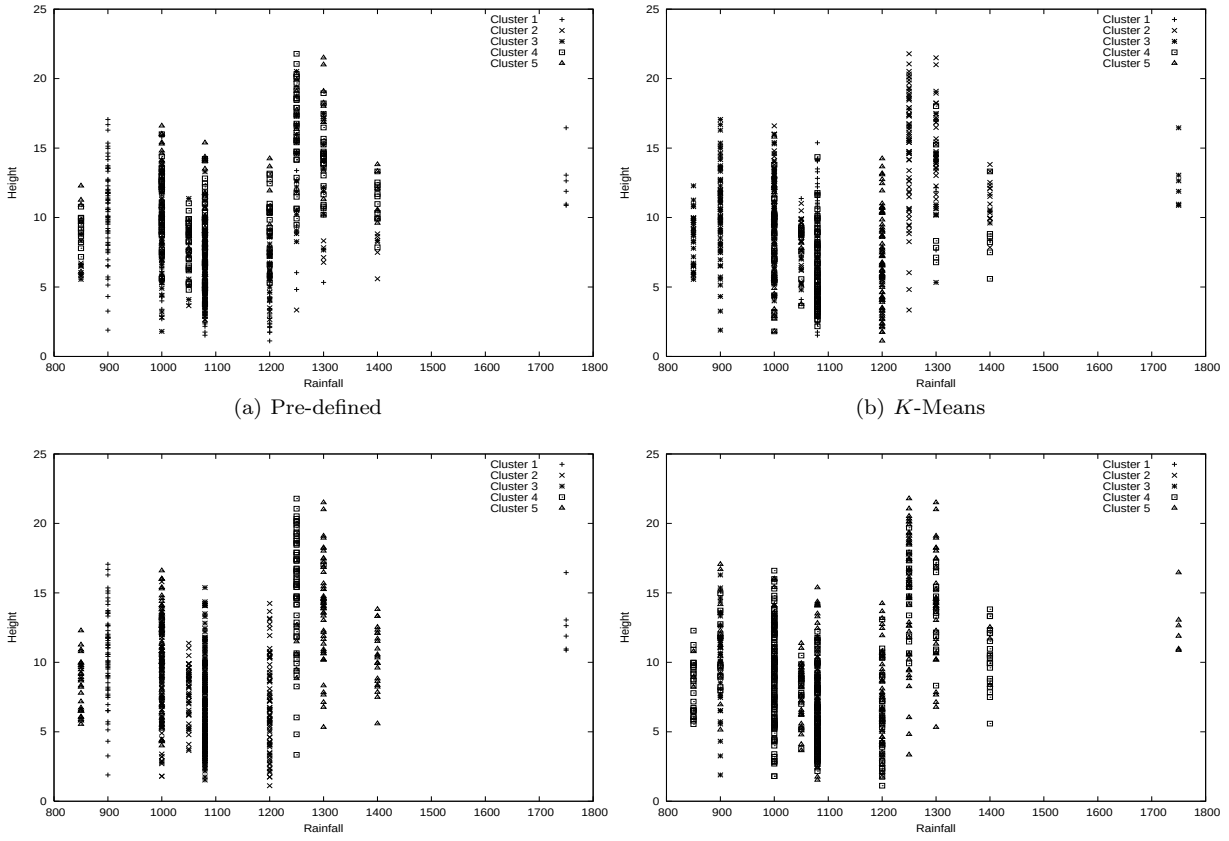
Fig. 6 Clusterings of 'credit' from K-Means, EM and Average Linkage algorithms and its pre-defined clustering. Data is projected onto attributes: 'duration' and 'credit amount'.



(a) Pre-defined (b) K-Means

(c) EM (d) Average Linkage

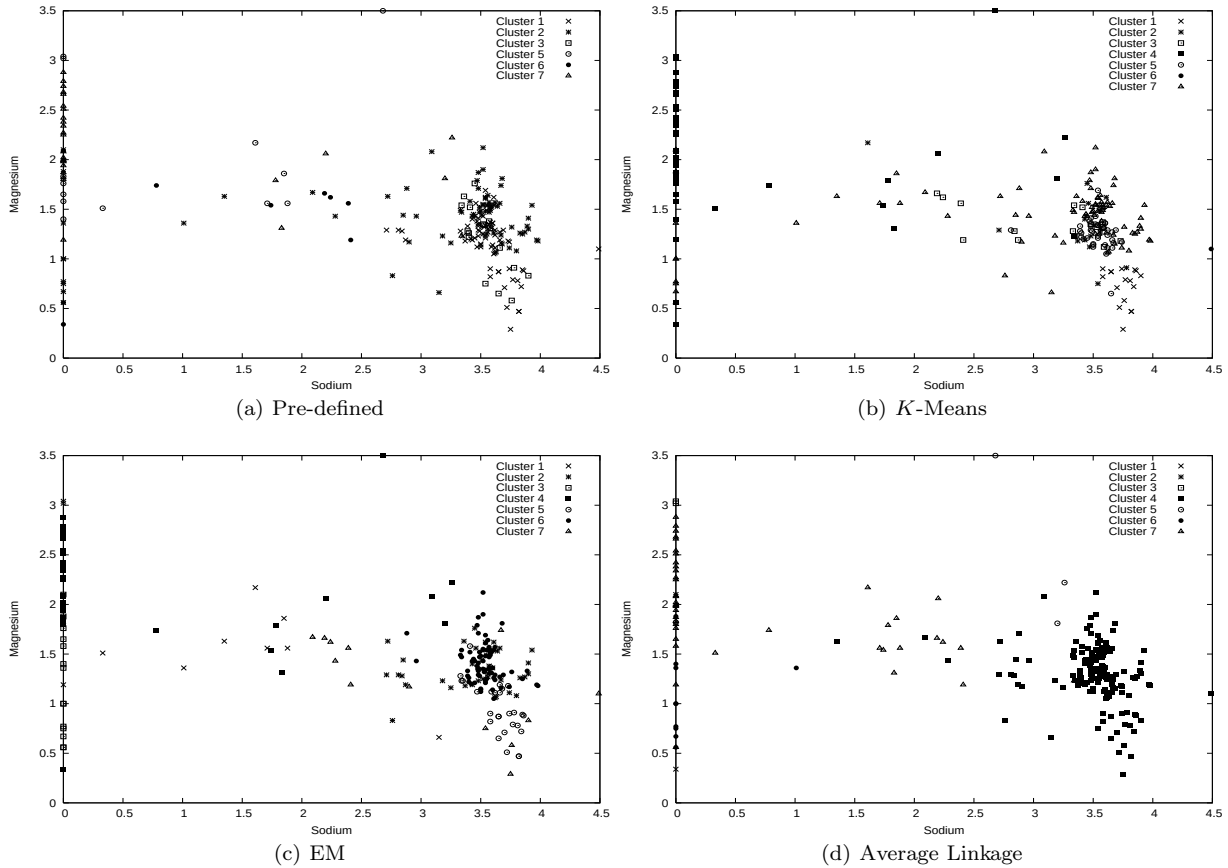
Fig. 7 Clusterings of 'diabetes' from K-Means, EM and Average Linkage algorithms and its pre-defined clustering. Data is projected onto attributes: 'plasma glucose concentration' and 'body mass index'.



(a) Pre-defined (b) K-Means

(c) EM (d) Average Linkage

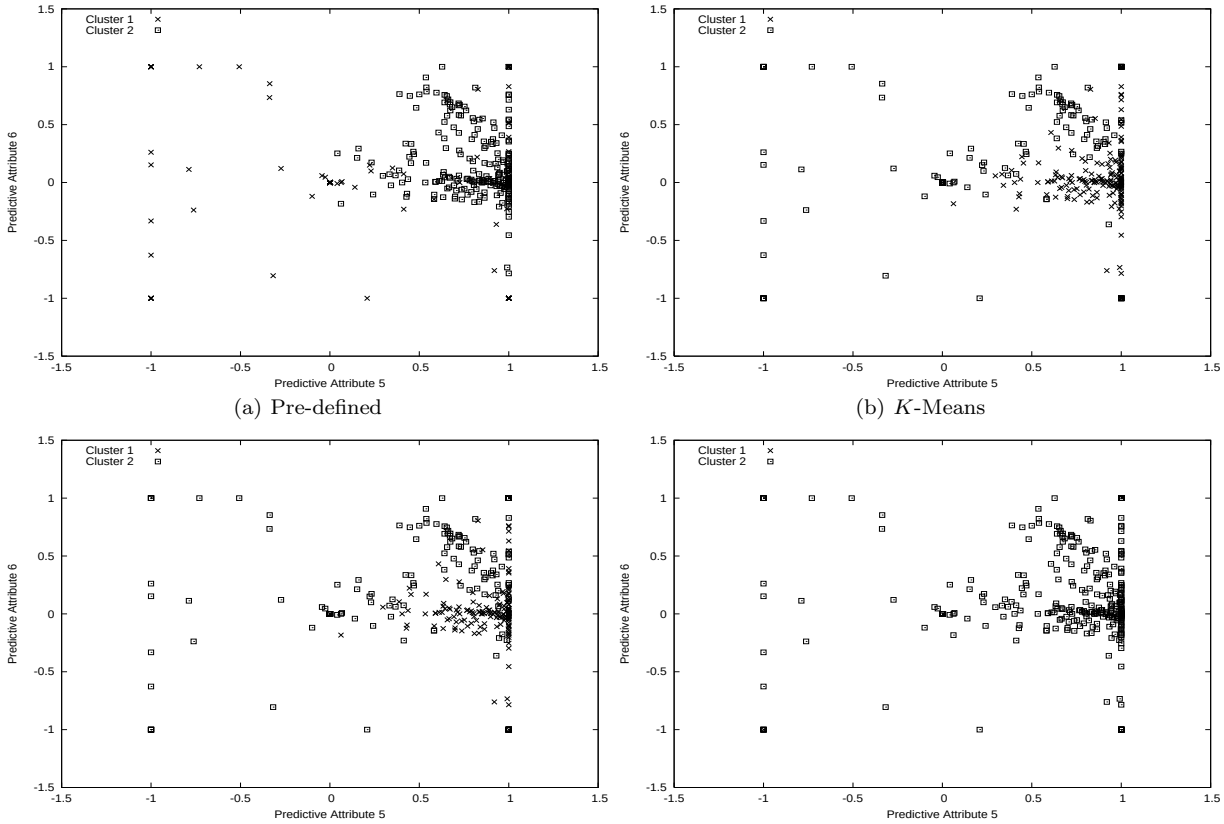
Fig. 8 Clusterings of 'eucalyptus' from K-Means, EM and Average Linkage algorithms and its pre-defined clustering. Data is projected onto attributes: 'rainfall' and 'height'.



(a) Pre-defined (b) K-Means

(c) EM (d) Average Linkage

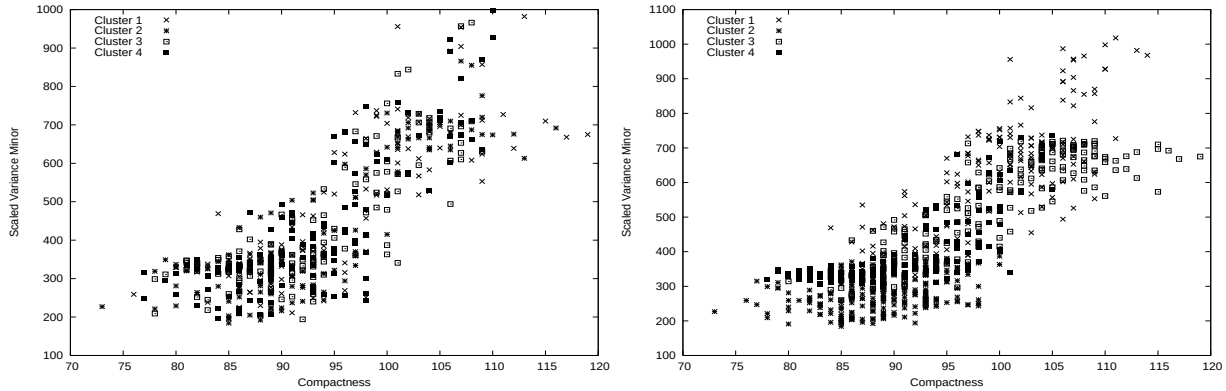
Fig. 9 Clusterings of 'glass' from K-Means, EM and Average Linkage algorithms and its pre-defined clustering. Data is projected onto attributes: 'refractive index' and 'sodium'.



(a) Pre-defined (b) K-Means

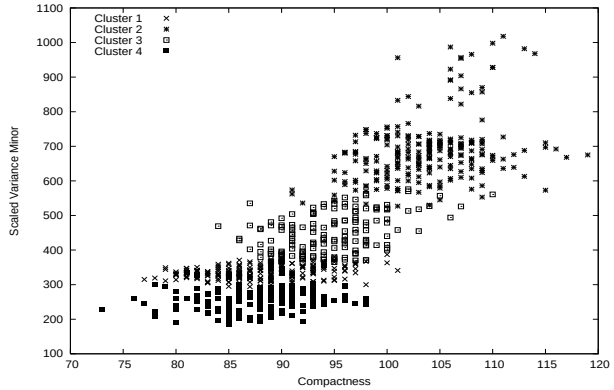
(c) EM (d) Average Linkage

Fig. 10 Clusterings of 'ionosphere' from K-Means, EM and Average Linkage algorithms and its pre-defined clustering. Data is projected onto attributes: 'predictive attribute 5' and 'predictive attribute 6'.

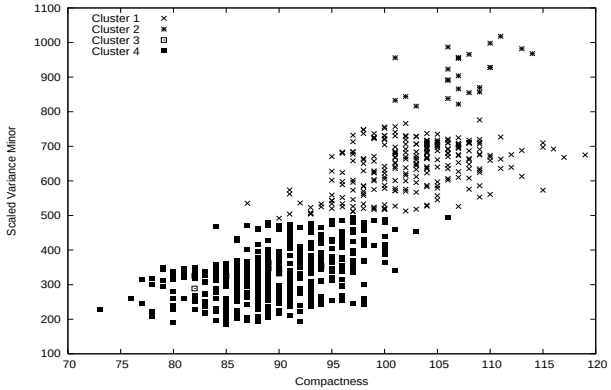


(a) Pre-defined

(b) K-Means



(c) EM



(d) Average Linkage

Fig. 11 Clusterings of 'vehicle' from K-Means, EM and Average Linkage algorithms and its pre-defined clustering. Data is projected onto attributes: 'compactness' and 'scaled variance minor'.

Table 5 Dissimilarity values measured by *ADCO*, Rand index (RI), Jaccard index (JI), Clustering error (CE) and variation of information (VI) when clusterings from K-Means, EM and average linkage (AL) algorithms are compared against the pre-defined clusterings

data set	measures	K-means	EM	Average Linkage
Credit	*ADCO*	0.97	0.93	0.76
	RI	0.70	0.53	0.59
	JI	0.77	0.68	0.71
	CE	0.82	0.63	0.71
	VI	0.91	1.24	0.80
diabetes	*ADCO*	1	0.83	0.71
	RI	1	0.57	0.55
	JI	1	0.70	0.69
	CE	1	0.69	0.66
	VI	0	1.11	0.75
Eucalyptus	*ADCO*	0.82	0.63	0.51
	RI	0.74	0.67	0.53
	JI	0.79	0.75	0.68
	CE	0.50	0.36	0.31
	VI	2.46	2.62	2.27
Glass	*ADCO*	0.87	0.79	0.53
	RI	0.86	0.72	0.64
	JI	0.88	0.78	0.73
	CE	0.68	0.48	0.50
	VI	1.13	1.99	1.17
Ionosphere	*ADCO*	0.91	0.87	0.69
	RI	0.61	0.63	0.54
	JI	0.72	0.73	0.68
	CE	0.74	0.75	0.64
	VI	1.12	0.99	0.67
Vehicle	*ADCO*	0.47	0.40	0.25
	RI	0.62	0.62	0.49
	JI	0.73	0.72	0.66
	CE	0.21	0.21	0.19
	VI	2.54	2.52	1.95

In order to interpret the validity of these results, we first need to understand some background about what is known about these attacks from the intrusion detection literature [37–39].

- The 'neptune', 'smurf', 'waremaster' and 'mailbomb' attacks are similar kinds of attacks, all being denial of service attacks, which abuse a legitimate feature of the operating system. The 'neptune' attack creates excessive connections, 'waremaster' transfers excessive amounts of data using anonymous ftp, 'smurf' floods the host with excessive echo request packets and 'mailbomb' overflows the system mailqueue by sending excessive emails. They generally can be characterized by the action of excessive packets being sent to a specific port on a given host, to abuse or overflow a system resource.
- The 'mscan' attack is a different type of attack, called a surveillance or probing attack, which involves lots of requests to a range of ports on a given host, searching for known vulnerabilities. These requests all occur within a short period of time and originate from some outside machine.

We now discuss a selection of the results from Table 7, which should be read in conjunction with the snapshot information in Table 6.

Table 6 Four snapshots from the KDD network intrusion detection data set. Each snapshot had 1000 points and was taken from a different time period. Snapshots have no common points and each had a dominant intrusion type.

Snapshot	Included intrusion types and their percentage
KDD-A	'normal (70%)', 'snmpgetattack (10%)', 'named (0.2%)', 'xlock (0.5%)', 'smurf (18%)', 'neptune (0.2%)'
KDD-B	'normal (83%)', 'snmpgetattack (13%)', 'warezmaster (3.6%)', 'xterm (0.4%)'
KDD-C	'smurf (27%)', 'mscan (50%)', 'warezmaster (11%)', 'normal (12%)', 'buffer-overflow (0.01%)', 'neptune (0.01%)'
KDD-D	'normal (2.3%)', 'neptune (97%)', 'snmpgetattack (0.7%)'
KDD-E	'normal (2.4%)', 'snmpgetattack (0.3%)', 'warezmaster (97.3%)'
KDD-F	'normal (0.1%)', 'snmpgetattack (18%)', 'mailbomb (97.4%)', 'portsweep (0.1%)', 'warezmaster (0.6%)'

- *KDD-A versus KDD-B*: These two snapshots had similar proportions of attack types 'normal' and 'snmpgetattack', which accounted for a large fraction of the data in the snapshot. So it is reasonable that they have a high *ADCO* similarity value.
- *KDD-C versus KDD-D*: These two snapshots did not share any intrusion types, except a small percentage of 'normal'. Moreover, the dominant intrusion type in *C* was 'mscan', which is quite a different class of attack from the dominant intrusion type in D, which was 'neptune'. So it is reasonable that they have low *ADCO* similarity (0.418).
- *KDD-A versus KDD-D*: Snapshot A contained a very large percentage of 'normal' and a moderate percentage of 'smurf' whereas D contained mostly 'neptune', so it is reasonable that the *ADCO* similarity is not high (0.566). The fact that the similarity isn't as low as the C versus D comparison (which was 0.418), is because there is still a fair degree of similarity between the 'smurf' cluster in D and the 'neptune' cluster in D. Recall that 'neptune' and 'smurf' are similar types of attacks.
- *KDD-D versus KDD-E*: These shared a small percentage of 'normal' in each, but the dominant intrusion type in D was 'neptune' and the dominant type in E was 'warezmaster'. Although these are different attacks, they have very similar behaviour, being both denial of service attacks. It is therefore reasonable that the *ADCO* similarity is relatively high (0.736).
- *KDD-D versus KDD-F*: These shared a small percentage of each, but the dominant intrusion type in D was 'neptune' and the dominant type in F was 'mailbomb'. Again, although these are different attacks, they have very similar behaviour, being both denial of service attacks. It is therefore reasonable that the *ADCO* similarity is high (0.998).

So overall, the *ADCO* behaviour is quite reasonable, returning high and low similarity values which can explained with reference to the semantics of the dataset.

Table 7 Result of comparing different KDD subsets to each other.

	KDD-B	KDD-C	KDD-D	KDD-E	KDD-F
KDD-A	0.852	0.627	0.566	0.682	0.566
KDD-B		0.499	0.665	0.813	0.665
KDD-C			0.418	0.394	0.418
KDD-D				0.736	0.998
KDD-E					0.759

4.4 Discretization Pre-processing for *ADCO*

It was briefly mentioned in Section 3.1 that computing the *ADCO* measure relies upon a discretization pre-processing step being performed for every continuous attribute [2]. We now discuss and analyse this aspect in more detail.

ADCO requires the range of each attribute to be discretized, so that density profile vectors can be created describing the density of each attribute for sub-ranges. The nature of these density profiles in turn influences the result of the similarity calculation between clustering vectors. i.e. The (absolute) similarity of two clusterings may change, according to how the density profile is represented. This is both an advantage and a disadvantage. The advantage of allowing different discretizations is that *ADCO* is more flexible and can be adapted to different datasets and feature spaces. The disadvantage is that non-expert users may require some guidance on default discretization settings.

The process of discretization is a well studied pre-processing technique in data mining and machine learning [22, 23, 21, 25, 26]. It is particularly popular for use with algorithms that require discrete or symbolic data, such as association rule mining, frequent or sequential pattern discovery and logical learning algorithms. As with many tasks in machine learning, there is no universally 'correct' method and the discretization choice will depend on domain knowledge about the feature space. This is analogous to other activities in clustering, such as feature selection, choosing the number of clusters and choosing a distance function.

Throughout the experiments in this paper, we mainly focus on equi-width binning as the default method to discretize each attribute, because it is both conceptually simple and also very efficient to compute. Figure 12 shows the effect of varying the number of equi-width bins from 1-100 for the six datasets previously considered. Observe that the *ADCO* value asymptotes as $Q \rightarrow \infty$, since in the limit, points cannot be separated any further by using finer grained bins. Also observe that across the six datasets, the relative ordering between the curves is maintained as the number of bins is increased (except only for Figure 12 d)). This is important, since clustering comparison is often done in a context where several clusterings are being compared against a gold standard, or pre-defined clustering. In this situation, it is the relative similarities which are of importance, not their absolute values.

In addition to examining the varying behavior of *ADCO* for a single discretization algorithm, it is also interesting to compare the *ADCO* values across different algorithms. One popular alternative to equi-width binning is equi-frequency binning, where all bins formed are required to contain the same number of points (have the same density). Another popular alternative is to use class based discretization, such

[2] Note that if any nominal attributes exist in the feature space, they need not be discretized and density profiles can be directly constructed for each nominal value.

Table 8 The table lists the *ADCO* values between the pre-defined clustering and the clusterings from K-means, EM and Average Linkage algorithms, when *ADCO* implements entropy-based discretization technique as discussed in [40].

Data Set	K-means	EM	Average Linkage
Credit	0.99	0.96	0.70
Diabetes	0.91	0.93	0.91
Eucalyptus	0.89	0.71	0.68
Glass	0.82	0.87	0.43
Ionosphere	0.86	0.92	0.59
Vehicle	0.72	0.62	0.26

Table 9 The table lists the *ADCO* values between the pre-defined clustering and the clusterings from K-means, EM and Average Linkage algorithms, when *ADCO* implements equal-frequency discretization technique (with 10 bins)

Data Set	K-means	EM	Average Linkage
Credit	0.98	0.96	0.71
Diabetes	0.79	0.88	0.79
Eucalyptus	0.74	0.54	0.49
Glass	0.71	0.72	0.45
Ionosphere	0.63	0.91	0.63
Vehicle	0.61	0.51	0.26

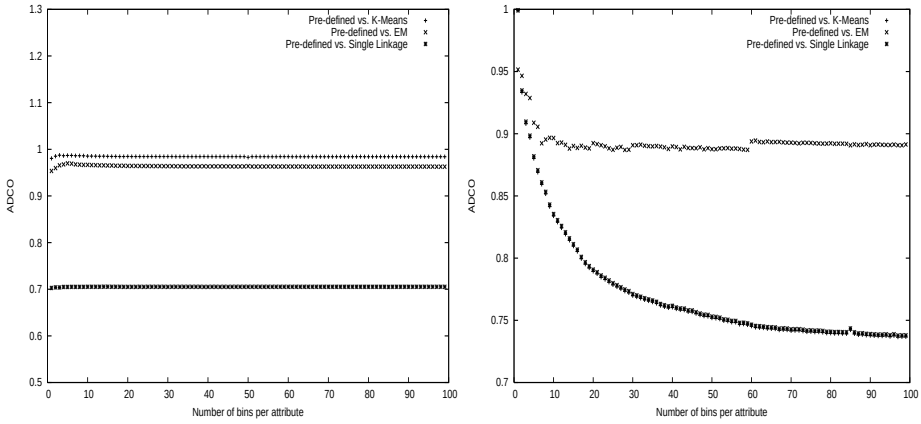
as 'Minimum Description Length (MDL)' discretization [40]. This technique requires a class label to be associated with each point. For our purposes, we can choose one of the two clusterings being compared as a reference clustering and then associate a class with each of its clusters, so that the class label for a point indicates which cluster the point belongs to.

Table 9 shows the *ADCO* values for our 6 datasets when using equi-frequency binning with 10 bins and Table 8 shows the *ADCO* values when using the entropy technique, where the number of bins is automatically chosen for each attribute. Comparing Tables 8, 9 and Table 5 (which uses equi-length discretization), some differences in absolute values are apparent. Significantly though, the three discretization choices yield consistent behavior with regard to the Credit, Glass and Vehicle datasets, all reporting that the Average Linkage Clustering is substantially different to the pre-defined clustering.

Another way to assess the influence of discretization on *ADCO* behavior is to compare its behavior for an alternate clustering task, when the *ADCO* measure is used as an objective function under a given discretization. We discuss results for this experiment in Section 7.2, after the MAXIMUS alternate clustering approach is presented.

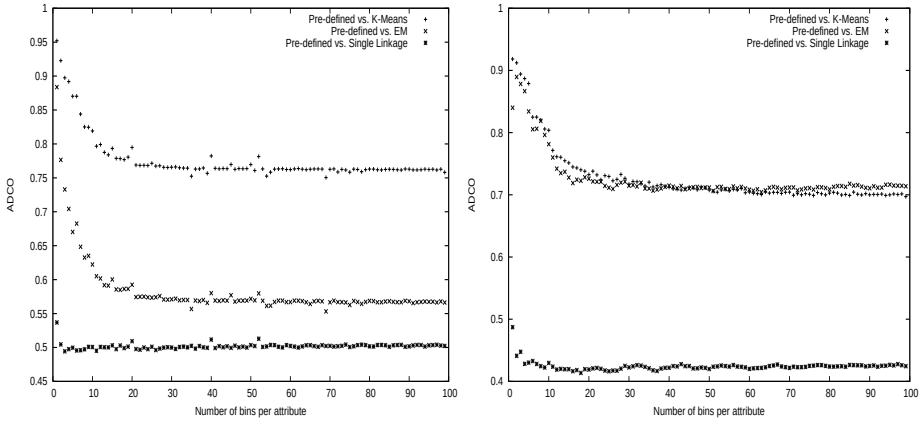
5 Complexity of *ADCO*

The runtime complexity of *ADCO* depends on the number of objects, attributes, bins and clusters. Given N objects, K_{min} clusters, R attributes and Q bins per attribute, *ADCO* requires $O(NRQ)$ operations for finding the density profile for each clustering. An additional $O(K_{min}^3)$ operations are required when calculating permutations for the scalar product using the Hungarian algorithm. The computation of the normalizing factor takes $O(NRQ)$ operations, since permutation is not necessary. So, the overall



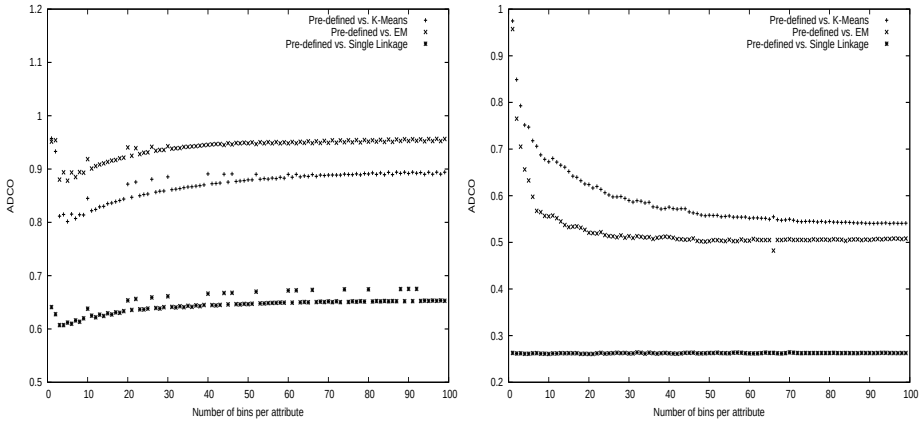
(a) Credit

(b) Diabetes



(c) Eucalyptus

(d) Glass



(e) Ionosphere

(f) Vehicle

Fig. 12 Changes in *ADCO* values as the number of (equi-width) bins per attribute increases from 1 to 100

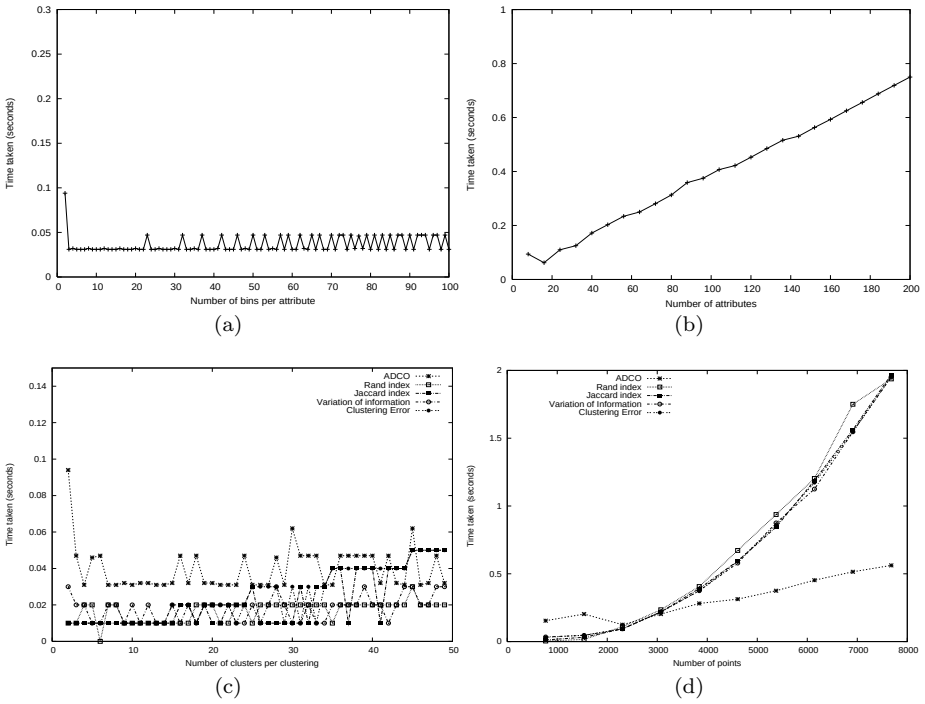


Fig. 13 Computing time of *ADCO* when the number of bins (13(a)), attributes (13(b)) and clusters (13(c)) and instances (13(d)) are changed.

complexity of *ADCO* is $O(NRQ) + O(K^3_{min})$. On the other hand, other comparison measures discussed in section 2 typically require $O(N^2K^2)$ operations, since they need to examine point-pairs in cluster pairs.

We experimentally tested the runtime scalability of *ADCO* by separately increasing the number of bins, attributes, clusters and instances. The results are given in Figure 13, where the two different clusterings being compared were generated from the 'diabetes' data set. The data set includes 768 instances, 8 attributes and 2 clusters and the number of bins per attribute was set to 10. When varying one of these parameters, all other values remained constant. When changing the number of clusters and instances, we also compared *ADCO*'s performance against other measures.

In Figure 13, we observe that increasing the value of Q has little impact on the running time of *ADCO*. The reason is: as Q increases (causing the range of values each bin can take to get narrower), the number of points per attribute-bin region would also decrease. In fact, the density of each region would be either 0 (i.e. empty) or equal to the total number of points sharing exactly the same attribute values and will remain constant as Q increases infinitely.

The complexity of *ADCO* is $O(K^3_{min})$ in the number of clusters, since the cluster order of the second clustering C' needs to be permuted to determine the best matching between the two clusterings. In practice though, despite the cubic worst case complexity, the Hungarian algorithm is a highly efficient heuristic, as seen by the curve of Figure 13(c). We can also see that other measures behaved quite similarly when changing the number of clusters. The occasional fluctuations in time were small differences in test

initialization times. Figure 13(b) shows that the performance time does increase as the size of the feature space grows, yet the overall amount of time taken is still quite low. When tested with bigger data sets (e.g. 'splice' with 3190 instances and 62 attributes), similar observations were made. Finally, when increasing the number of instances, we see in Figure 13(d) that other measures are much more impacted than *ADCO*, since they have $O(N^2)$ complexity in the number of instances.

6 An Application of *ADCO* to Multiple Alternate Clustering Generation

In this section, we show how *ADCO* can be used to help discover multiple alternate clustering solutions, given an initial pre-defined clustering. The key idea is that the *ADCO* measure can be used an objective function for this task, which can be encoded as an integer linear program. Alternate clustering algorithms [41,3,42] are frequently utilized in exploratory data analysis, where users wish to gain a deeper understanding of their data by retrieving several clusterings. Our method is embodied in an algorithm we call MAXIMUS (**MAX**imized D**I**ssimilarity in **MU**ltiple Clustering**S**), which employs *ADCO* as its clustering similarity objective function to generate multiple alternate solutions.

Although a number of alternate clustering algorithms exist [4,43,41,42,16], many of them do not emphasize the uniqueness of each solution compared to others. Therefore, clusterings generated by these techniques are often redundant. Moreover, previous methods introduced in [4,43,41] are limited to generating only a single alternate clustering and are somewhat inefficient in their runtime performance. On the other hand, MAXIMUS is able to generate a user-specified number of alternate clusterings, while maximizing the overall dissimilarity and quality, which are two core requirements in alternate clustering algorithms [41]. We define these requirements below.

Definition 3 ***Dissimilarity & Overall Dissimilarity*** *: Let C'_u and C'_v be two clusterings of D. The dissimilarity between them is determined by a function $Diss(C'_u, C'_v)$, where $0 \leq Diss(C'_u, C'_v) \leq 1$. Larger values of $Diss(C'_u, C'_v)$ indicate higher dissimilarity. Let $\mathscr{C}' = \{C'_1, C'_2, .., C'_M\}$ be a set of M clusterings. The overall (average) dissimilarity $\mathscr{C}'$ is defined as $\mathcal{OD}_{\mathscr{C}'} = \frac{\sum_{u=1}^{M}\sum_{v=2}^{M} Diss(C'_u, C'_v)}{M(M-1)/2}$, which is the average pairwise dissimilarity between all clusterings in $\mathscr{C}'$.*

Definition 4 ***Quality & Overall Quality*** *: The quality of a clustering C'_u is a function $Qual(C'_u)$, where $0 \leq Qual(C'_u) \leq 1$. Larger values of $Qual(C'_u)$ indicate higher quality. The overall quality $\mathcal{OQ}_{\mathscr{C}'}$ of a clustering set $\mathscr{C}' = \{C'_1, C'_2, .., C'_M\}$ is the average of the quality values of all the clusterings in $\mathscr{C}'$: $\mathcal{OQ}_{\mathscr{C}'} = \frac{\sum_{u=1}^{M} Qual(C'_u)}{M}$.*

With these definitions, the target problem of MAXIMUS can be roughly stated as follows.

Problem definition *: Given a data set D and an existing clustering C'_p, generate an alternate clustering set $\mathscr{C}'$ containing M alternate clusterings $\mathscr{C}' = \{C'_1, C'_2, .., C'_M\}$, such that $\mathcal{OQ}_{\mathscr{C}'}$ and $\mathcal{OD}_{\mathscr{C}'}$ are simultaneously high.*

6.1 MAXIMUS Algorithm Description

The MAXIMUS algorithm generates one clustering at a time, in three stages. Initially, it calculates the maximum dissimilarity between any currently available clusterings and a potential target alternate solution, by forming an integer programming model (we refer to this as IP hereafter). The objective of this IP model is to minimize the scalar product between density profiles between the known clusterings and the unknown target alternate clustering C'_u.

A solution of the IP model yields the number of points that should be distributed to each cluster of C'_u in each attribute-bin region (i, j). This distribution information is then utilized as constraints to guide the clustering of the points in each region (i, j), via a constraint-based K-means style algorithm. Compared to more traditional constraint-based techniques [6, 44, 45], which utilize instance-based constraints, we implement a new type of constraint, called a *'distribution constraint'*, which specifies the number of points that should be assigned to different regions of the feature space. The output of this stage is a set of 'localized' clusterings, which in the final step are combined to create an overall consensus clustering. In the following sections, we describe each of these stages in more detail. We first present the process of generating one alternate clustering given the pre-defined clustering C'_p and in section 6.5 we explain how further alternate clusterings can be created.

6.2 Maximizing Dissimilarity via Integer Programming

Recall equation 2, in which the maximum scalar product between density profiles of two clusterings was used. In order to create a new clustering C'_u with maximal dissimilarity to an existing (known) clustering C'_p, we require the value of $sim(C'_p, C'_u)$ to be minimum. Since the density profile of C'_p is known, each density value in the vector $V_{C'_p}$ is some *constant value*, while the density values in $V_{C'_u}$ are unknown (integer) variables. Based on equation 2, we can formulate a minimization objective as

$$\min \left[\max_{\rho} \sum_{k=1}^{K_{min}} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_{C'_p}(k, i, j) \times dens_{C'_u}(\rho(k), i, j) \right] \tag{9}$$

where the values of $dens_{C'_p}(k, i, j)$ are constants and $dens_{C'_u}(k, i, j)$ are variables. This objective function then needs to be restricted according to the following two constraints.

Attribute-Bin Density Constraint : The attribute-bin density constraint ensures that the solutions of the IP model, which correspond to the number of points distributed to each cluster of C'_u in each (i, j) region, is limited by the total number of points located in that area. More formally, let $dens(i, j)$ denote the total number of points located in (i, j) and let $dens_{C'_u}(k, i, j)$ refer to the number of points in (i, j) that belong to the kth cluster of C'_u. The value of $dens(i, j)$ is known from the data set and must satisfy $dens(i, j) = \sum_{k=1}^{K} dens_{C'_u}(k, i, j)$. In the IP model, this constraint is represented as a matrix of the form $Ax = b$, where A and b are defined as:

$$A = \begin{bmatrix} dens_{C'_u}(1,1,1) & dens_{C'_u}(2,1,1) & \dots & dens_{C'_u}(K,1,1) \\ dens_{C'_u}(1,1,2) & dens_{C'_u}(2,1,2) & \dots & dens_{C'_u}(K,1,2) \\ \vdots & \vdots & \ddots & \vdots \\ dens_{C'_u}(1,1,Q) & dens_{C'_u}(2,1,Q) & \dots & dens_{C'_u}(K,1,Q) \\ dens_{C'_u}(1,2,1) & dens_{C'_u}(2,2,1) & \dots & dens_{C'_u}(K,2,1) \\ \vdots & \vdots & \ddots & \vdots \\ dens_{C'_u}(1,R,Q) & dens_{C'_u}(2,R,Q) & \dots & dens_{C'_u}(K,R,Q) \end{bmatrix}$$

$$b^T = \big[\, dens(1,1), dens(1,2), .., dens(1,Q), dens(2,1), .., dens(R,Q) \,\big] \tag{10}$$

Permutation Inequality Constraint : In equation 9, we must choose a specific permutation ρ_s of the possible cluster orderings in C'_u, that returns the maximum scalar product value when clusters of C'_u are paired up with clusters of C'_p. Suppose that $\mathscr{P} = \{\rho_1, \rho_2, .., \rho_L\}$ is the set of possible permutations (say L of them) that is possible for C'_u. Since the values of density profile $dens_{C'_u}$ are unknown, we can set ρ_1 for example, as the permutation that gives the maximum scalar product value compared to other permutations using the following constraint:

$$V_{C'_p} \cdot \rho_1(V_{C'_u}) > V_{C'_p} \cdot \rho_j(V_{C'_u}), \text{ for all } j \text{ s.t. } (2 \leq j \leq L) \tag{11}$$

Using the objective function in equation 9 and the constraints in equations 10 and 11, an IP solver can be used to find the unknown variables, corresponding to the density value of each attribute-bin region. In practice, we used the LP Solve package version 5.5.1, which is available from [46]. Note that some attribute-bin regions may be empty and they are discarded when determining the density profiles of clusterings.

6.3 Generating Localized Clusterings using Distribution Constraints

The integer solutions of the IP model correspond to the number of points that should be assigned to each cluster c'_k of the alternate clustering C'_u in each attribute-bin region (i, j). These distributions ensure that C'_u will have maximal dissimilarity when compared to C'_p. The integer solutions themselves, however, do not indicate the *physical* point-to-cluster memberships. That is, we do not know which points are assigned to which clusters. Furthermore, while the dissimilarity between two clusterings is maximized, we have not yet considered the additional requirement that C'_u also needs to be of high quality. Therefore, in order to assign cluster labels to data objects, while still maintaining high dissimilarity, we treat the solutions of the IP model as 'distribution constraints', to be imposed on clusters in each attribute-bin region. These constraints specify the number of points that can be assigned to a particular cluster. It is defined as follows.

Definition 5 *A **Distribution Constraint** has the form $n_{c'_k}(i,j) = dens_{C'_u}(k,i,j)$, which specifies the total number of points to be assigned to the cluster c'_k in the target alternate clustering $C'_u = \{c'_1, c'_2, .., c'_K\}$ for the attribute-bin region (i,j).*

For MAXIMUS, each constraint $n_{c'_k}(i,j)$ is derived from the solutions of the IP model. As mentioned above, this is quite different from the instance-based 'must-link' and 'cannot-link' constraints discussed in [6], which explicitly declare a set of points that must be together or separated.

Using these constraints, we distribute points to clusters in each attribute-bin region separately (for a total of RQ regions) via what we call a "distribution-constraint K-means algorithm". The objective is to assign points to their closest cluster centroid (according to Euclidean distance) whilst also satisfying the distribution constraints. The Euclidean distance function allows us to refine the clustering C'_u in terms of quality while maintaining the maximum dissimilarity achieved through the IP model. The output of this step is a set of RQ 'localized' clusterings (i.e. one clustering for each attribute-bin region)[3].

Algorithm 2 describes this process. For each attribute-bin region (i,j), we find a subset $D_{(i,j)}$ of data set D (line 3), which is the set of all points belonging to (i,j) and we generate K random, initial centroids $W_{(i,j)}$ (line 4). We then generate a 2D distance matrix between points in $D_{(i,j)}$ and the centroids in $W_{(i,j)}$. Each cell in the matrix is, therefore the distance between a point and a cluster centroid. This matrix is computed so that points which are closest to the centroids take the priority for the assignment according to the distribution constraints. The partitioning step continues by finding a point x_{min} and cluster centroid w_{min} pair, which has the minimum distance in the distance matrix (lines 8 to 15).

If the distribution constraint for (i,j) restricts the assignment of x_{min} to c'_{min} (whose centroid is w_{min}), this indicates that the cluster has reached its *density limit* and therefore its distribution constraint can no longer be satisfied. Therefore, we specify the distance between x_{min} and w_{min} in the distance matrix as *null* (line 19), so that no more points would be assigned to this cluster. The assignment of points to clusters concludes when all points have their corresponding cluster labels, at which point the centroids of clusters are recalculated before repeating the assignment process. Once a localized clustering is generated in (i,j), we add this to the localized clustering set $LC_{C'_u}$ (line 26). Finally, each localized clustering is guaranteed to converge, since the sum of points specified by the distribution constraints in each attribute-bin region is equal to the total number of points in that region (i.e. $\sum_{k=1}^{K} n_{c'_k}(i,j) = dens(i,j) = |D_{(i,j)}|$), which ensures that all points in (i,j) are assigned to clusters.

6.4 Consensus Clustering

Each localized clustering from the previous stage is for a region of the data set D and every point in D occurs in exactly R regions. Hence each point will have R labels, one for each localized clustering. Depending on how partitioning was conducted, the labels for a point may be different. So, in this final step, we perform a consensus process in which all localized clusterings are combined via a majority voting technique [47], to generate a final alternate clustering C'_k, so that each point has exactly one cluster label. Majority voting is a technique that is well known for creating a robust and reliable clustering [17,48,49]. This technique observes the cluster labels assigned to each point in each of the localized clusterings and evaluates their consistency across all clusterings.

[3] In practice, the total number of local clusterings generated is less than RQ, since some attribute-bin regions are vacant.

Algorithm 2 Distribution-constraint K-Means algorithm

Require: density profile vector of alternate clustering C'_u, $V_{C'_u}$, acquired from the IP solutions
Ensure: a set of localized clusterings $LC_{C'_u}$ is returned

for $i = 1$ to R {build a localized clustering $C'_{(i,j)}$ from all attribute-bin regions} **do**
  for $j = 1$ to Q **do**
    $D_{(i,j)} = findPoints(i, j)$ {returns all points located in (i, j)}
    $W_{(i,j)} = \{w_1, w_2, .., w_K\} = randomCentroids(K, (i, j))$
    $distanceMatrix = findDistance(D_{(i,j)}, W_{(i,j)})$ {A distance matrix containing distances between all points in $D_{(i,j)}$ and all centroids in $W_{(i,j)}$}
    $converged = false$, $c'_{min} = null$, $minDistance = null$, $x_{min} = null$
    while $converged = false$ **do**
      for $k = 1$ to $|D_{(i,j)}|$ {find a point and a centroid sharing minimum distance} **do**
        for $m = 1$ to K **do**
          $distance = findDistance(x_k, w_m)$
          if $distance < minDistance$ **then**
            $minDistance = distance$, $c'_{min} = c'_m$, $x_{min} = x_k$
          end if
        end for
      end for
      if $|c'_{min}| < dens_{c'_{min}}(i, j)$ **then**
        $c'_{min} = c'_{min} \cup x_{min}$ {add x_{min} to c'_{min} if constraint is satisfied.}
      else
        $distanceMatrix(c'_{min}, x_{min}) = null$ {if constraint fails, set the distance to $null$}
      end if
    end while
    $centroid_{new} = generateCentroids(K)$ {recalculate centroids after assigning points}
    if $test(converged) = true$ **then**
      exit
    end if
    $LC_{C'_u} = LC_{C'_u} \cup C'_{(i,j)}$
  end for
end for

Since the cluster labels have the same meaning across all the localized clusters, the label for each point can be determined by just choosing the one which occurs the most times. When two labels share the same number of votes, we have randomly selected one label. Once this process finishes, we are left with a single, final alternate clustering C'.

6.5 Extracting Multiple Clusterings

We now describe how further alternate clustering solutions can be generated. Specifically, suppose we are looking for the M-th solution, given a current set of alternate clusterings $\mathscr{C}' = \{C'_1, C'_2, .., C_{M-1}\}$. The minimization function we used earlier needs to change to reflect the more complex overall dissimilarity definition, so that the new alternate clustering will have maximized pairwise dissimilarity compared to all of the currently available clusterings in $\mathscr{C}'$.

$$\min \left[\max_{\rho} \sum_{h=1}^{M-1} \sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=1}^{Q} dens_{C'_h}(k, i, j) \times dens_{C'_u}(\rho(k), i, j) \right] \tag{12}$$

The above equation calculates the lowest average similarity between the target clustering C' and all known clusterings in $\mathcal{C}'$. In conjunction with this objective function, the attribute-bin density constraint used earlier remains the same, but we must

modify the permutation constraint in equation 11, to include all the known values of previously found clusterings.

$$A = \begin{bmatrix} dens_{C'_1}(1,1,1) & dens_{C'_1}(2,1,1) & \dots & dens_{C'_1}(K,1,1) \\ dens_{C'_1}(1,1,2) & dens_{C'_1}(2,1,2) & \dots & dens_{C'_1}(K,1,2) \\ \vdots & \vdots & \ddots & \vdots \\ dens_{C'_1}(1,1,Q) & dens_{C'_1}(2,1,Q) & \dots & dens_{C'_1}(K,1,Q) \\ dens_{C'_1}(1,2,1) & dens_{C'_1}(2,2,1) & \dots & dens_{C'_1}(K,2,1) \\ \vdots & \vdots & \ddots & \vdots \\ dens_{C'_1}(1,R,Q) & dens_{C'_1}(2,R,Q) & \dots & dens_{C'_1}(K,R,Q) \\ \vdots & \vdots & \ddots & \vdots \\ dens_{C'_{M-1}}(1,R,Q) & dens_{C'_{M-1}}(2,R,Q) & \dots & dens_{C'_{M-1}}(K,R,Q) \end{bmatrix}$$

$$b^T = \left[dens(1,1), dens(1,2), .., dens(1,Q), dens(2,1), .., dens(R,Q) \right] \tag{13}$$

The minimization objective and the permutation constraint must be updated each time a new alternate clustering is added to the set.

The total number of variables created in MAXIMUS is $MKRQ$, where RQ is the number of attribute-bin regions, K is the number of clusters for each alternate clusterings and M is the total number of solutions to be found. Generating each localized clustering takes $O(IK(dens(i,j))$ operations, where I is the number of iterations required and $dens(i,j)$ is the total number of points in each region. Therefore, it takes $O(RQIK(dens(i,j))$ to generate all the localized clusterings. Finally the consensus clustering will take $O(NRQ)$ to perform its majority voting and to determine the final cluster labels for the points.

7 Experiments to Evaluate the MAXIMUS Algorithm

For our experimental analysis, we chose 13 real world data sets and compared the output of MAXIMUS against existing single and multiple alternate clustering algorithms. We validated the output alternate clustering(s) in terms of dissimilarity using the popular Jaccard index [7] (JI) due to its simplicity and robustness compared to the Rand index (defined in Table 1). The quality was evaluated using the generalized Dunn index [50] as defined in equation 14 below.

$$GDI(C) = \frac{min_{i \neq j} \left\{ \delta(c_i, c_j) \right\}}{max_{1 \leq l \leq k} \left\{ \Delta(c_l) \right\}}, \tag{14}$$

where δ is the inter-cluster distance and Δ is the intra-cluster distance; GDI is a reliable measure as tested in [51].

The two measures are also combined to give an overall quality-dissimilarity score as described in [41] and given as follows.

$$DQ(C, C') = \frac{2JI(C, C')GDI(C')}{JI(C, C') + GDI(C')} \tag{15}$$

For all three measures - JI, GDI and DQ - higher values indicate better results.

7.1 Comparing Against Single Alternate Clustering Algorithms

Thirteen real world data sets taken from the UCI repository [36] were used. These data sets all have pre-defined class labels which can be used to form pre-defined clusterings. We compared MAXIMUS against CIB [4], CondEns [43] and COALA [41], which are algorithms that generate only a single alternate clustering with respect to a pre-defined clustering. We describe each of these algorithms below and then present results of experiments comparing their dissimilarity, quality, overall DQ-Measure and time taken.

Conditional Information Bottleneck was introduced in [4] and is based on the notion of *information bottleneck* (IB) [52]. The general idea of the IB method is that given two variables (i.e. X representing objects, Y representing the features), the goal is to keep the shared information between these two variables maximum (mutual information), while one variable is compressed through another variable. CIB extends this by introducing another variable (i.e. Z representing the pre-defined class labels) where the new objective is to find the optimal assignment of X to C, while preserving as much information about Y conditioned on the information provided by the Z. The above concept is embodied in a conditional information bottleneck algorithm [4], which takes a sequential clustering approach. Here, K clusters are first randomly formed and each data object is moved around clusters in order to maximize the overall conditional mutual information, which effectively factors out the known structure Z. CIB requires an initial parameter for the number of iterations to refine the resultant clustering and we set this to 5 in our experiments.

CondEns extends the concept of CIB and utilizes the cluster ensembles, which can be generated by any clustering algorithm (e.g. K-Means, EM, Average-Linkage). It consists of three steps. In the first stage, given a pre-defined clustering Z, *local clusterings* are generated for each of the clusters in Z. The second stage extends the local solutions by assigning each instance to the possible clusters of the global clustering solution, which depends on the specific base clustering method employed. Finally, the conditional mutual information equation is used for combining these clusterings to generate a final consensus clustering, which is different to the pre-defined structure. We used the K-means algorithm as its base clustering algorithm in our experiments.

COALA is a hierarchical algorithm, which achieves clustering dissimilarity through a set of cannot-link constraints established between all pairs of points belonging to the same clusters in the pre-defined clustering. In each cluster merge step, a pair of closest clusters and a pair of closest clusters that satisfy the constraints (i.e. any pair of points in two clusters do not belong to the same clusters in the input clustering) are compared against ω threshold, which effectively controls the trade-off between dissimilarity and quality. This value was set to 0.6 in our experiments.

The results in Figure 14 and Table 10 show that MAXIMUS is overall, a high performing algorithm compared to others in terms of dissimilarity, quality, the overall DQ-Measure and time taken. Looking at Figure 14(a) and the overall DQ-Measure, we see that MAXIMUS is best in 7 out of 13 datasets. For individual comparisons on the DQ measure, MAXIMUS has 12 wins and 1 loss against CIB, 9 wins and 4 losses against CondEns, 7 wins and 6 losses against COALA. Behind MAXIMUS, COALA is the next best algorithm in terms of performance on the overall DQ-Measure. However we can see in Table 10 that COALA is generally at least 100 times slower than MAXIMUS, making it impractical for large datasets.

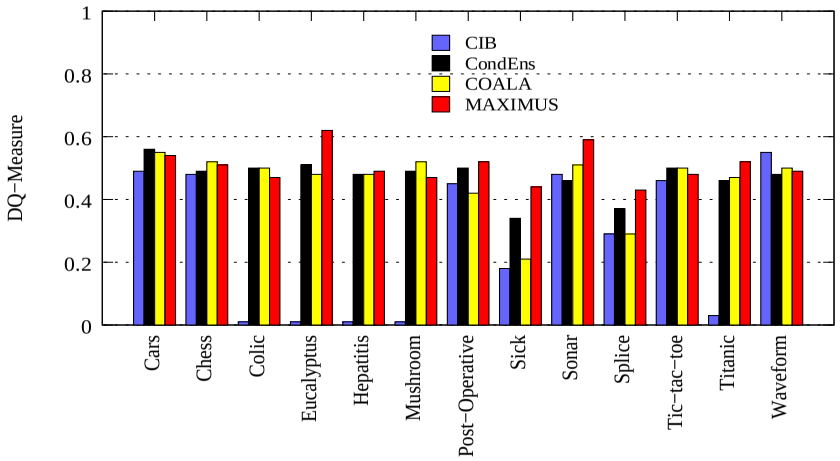
Looking again at Figure 14, while MAXIMUS sometimes created lower quality clusterings compared to the other techniques for some data sets, it compensated by generating highly dissimilar clusterings. In contrast, CIB and CondEns generally performed poorly compared to COALA and MAXIMUS. From our experiments, CIB performed the worst overall since it could not find any new alternate clusterings for 'eucalyptus' and 'hepatitis' (indicated by 0 dissimilarity and DQ-Measure values). Moreover, CIB was quite slow. Although CondEns was overall the fastest technique, in comparison to MAXIMUS, it failed to create high quality and highly dissimilar solutions for all data sets. We also found the solutions of CondEns inconsistent, perhaps because it is based on a K-means algorithm and has a random initialization.

For our second experiment, we tested the performance of MAXIMUS against the MetaClusterer [42], to evaluate its ability to discover multiple alternate clusterings. Note that the COALA, CondEns and CIB algorithms cannot generate multiple alternate clusterings. MetaClusterer takes a sampling-based approach to finding multiple clusterings. It also defines a 'clustering distance' metric so that a comparative analysis can be performed before presenting users with the final set of solutions. The technique consists of two stages, where a large number of qualitatively different clusterings (called base-level clusterings) are first generated using two approaches: 1) repeated K-means method and 2) attribute-weighting via Zipf distribution law. Since the standard K-means algorithm selects random initial centroids, it is possible to create a number of clusterings by ensuring that each execution of the algorithm is initiated with different centroids. Once these base clusterings are generated, they are considered as 'data objects' and supplied to a hierarchical algorithm. The clusterings are then merged iteratively using a clustering distance function called 'Cluster Difference' (which is a slight variation of Rand index and Jaccard index) until there remains a single clustering. The resultant hierarchy of clusterings then provides a 'meta level' view of clusterings.

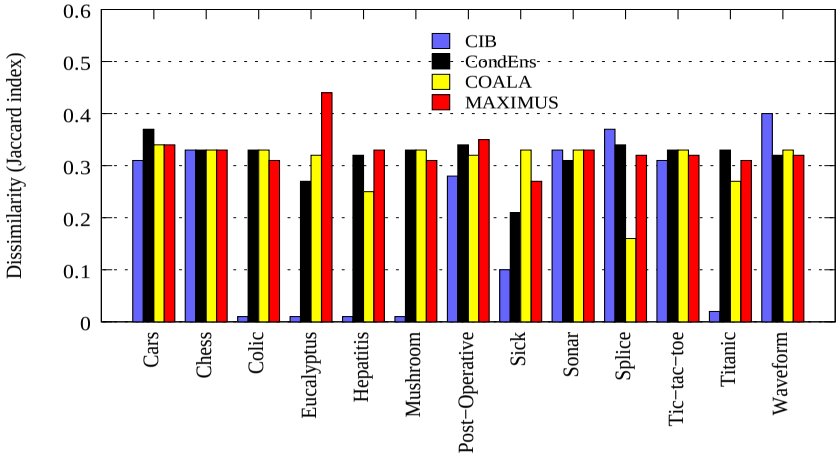
For each data set, we generated two alternate clusterings to supplement the given, pre-defined clustering. Overall scores for quality, dissimilarity and DQ value were then computed. The results are shown in Figure 15. Firstly, the values of the DQ-Measure demonstrate that MAXIMUS performs far better than MetaClusterer for all data sets except 'eucalyptus'. Indeed, the clusterings generated for 'chess', 'sonar' and 'splice' were relatively poor for MetaClusterer, whereas MAXIMUS was able to form rather better sets of clustering solutions. When comparing dissimilarity and quality, MAXIMUS again scored better GDI and JI values. Although there were a few data sets where MetaClusterer scored better dissimilarity or quality, we found that apart from 'chess' and 'eucalyptus' data sets, it never scored better than MAXIMUS in both criteria. Finally, the performance of MetaClusterer was inconsistent compared to MAXIMUS. For 'sonar' and 'splice', the alternate clusterings generated were at times exactly the same as one of the existing clusterings.

Overall then, we can see that MAXIMUS is generally a high performing algorithm and an important new tool for alternate clustering. We can see from the experiments it has the following desirable properties:

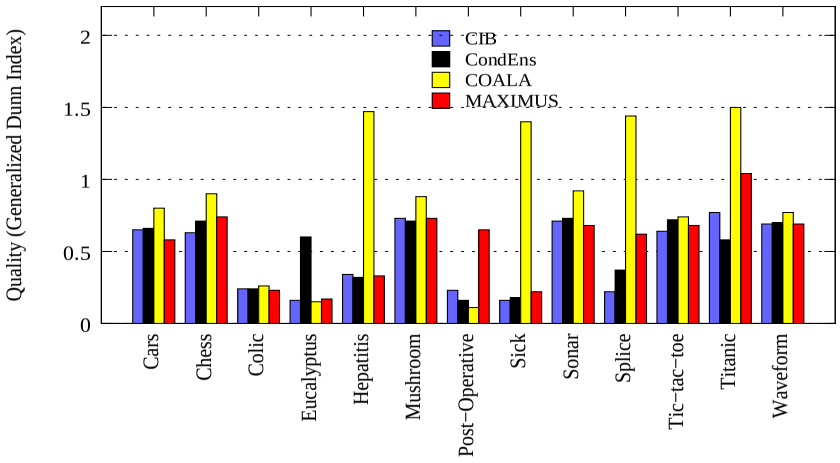
- It is one of the best two algorithms for finding a single alternate clustering (COALA being the other), in terms of overall performance (DQ-Measure).
- It is considerably faster than COALA, usually around 100 times.
- It performs considerably better than the only other algorithm that exists for finding multiple alternate clusterings (MetaClusterer).



(a) DQ-Measure comparison

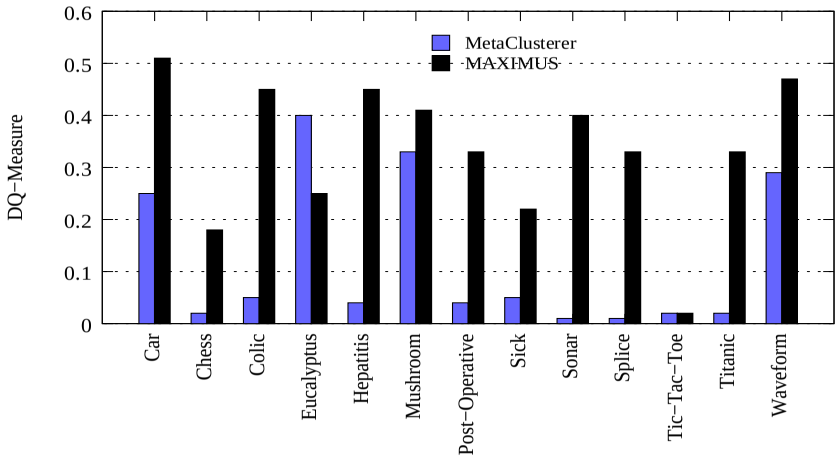


(b) Dissimilarity comparison

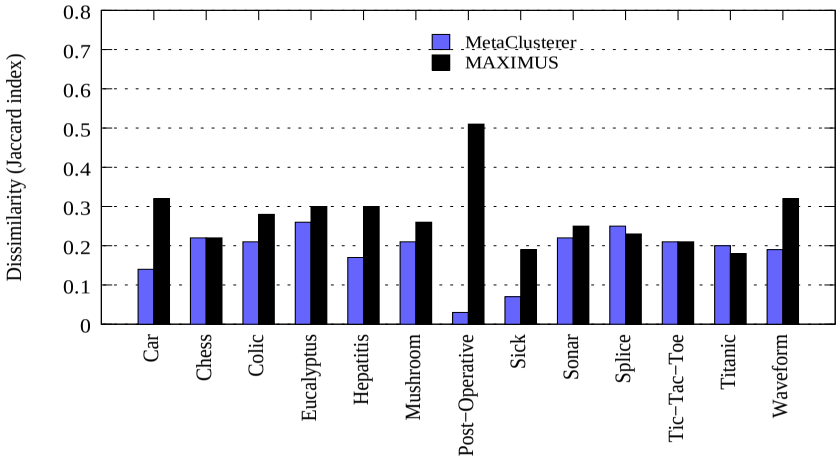


(c) Quality comparison

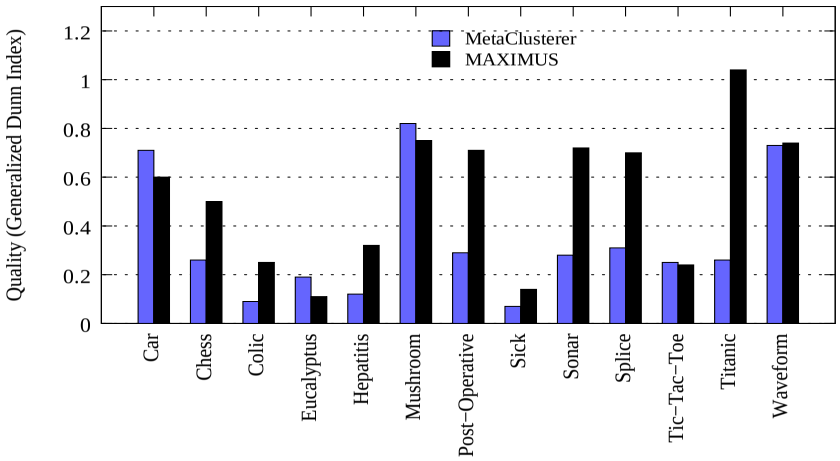
Fig. 14 Comparison of CIB, CondEns, COALA and MAXIMUS using 13 data sets in terms of dissimilarity, quality and DQ-Measure.



(a) DQ-Measure comparison



(b) Dissimilarity comparison



(c) Quality comparison

Fig. 15 Comparison of MetaClusterer and MAXIMUS using 13 data sets in terms of dissimilarity, quality and DQ-Measure.

Table 10 Time taken (in seconds) to find alternate clusterings. The first four columns compare MAXIMUS with other single alternate clustering algorithms (CIB, CondEns and COALA) for generating one alternate clustering. The last two columns, on the other hand, are relevant to times taken for extracting multiple clusterings for MetaClusterer and MAXIMUS.

data sets	CIB	CondEns	COALA	MAXIMUS	Meta Clusterer	MAXIMUS
Cars	173.035	1.45	1140.28	3.11	19.7	6.15
Chess	2598.81	3.75	8131.94	20.49	80.28	49.76
Colic	368	2	3.82	0.66	35.34	0.98
Hepatitis	1.3	0.39	0.25	0.41	30.12	0.58
Mushroom	14406.95	11.94	89975.26	269.94	257.45	154.58
Post Operative	0.33	0.22	0.08	0.38	13.56	0.49
Sick	1865.61	2.87	41772.09	26.18	80.64	53.99
Splice	1923.78	5.19	6067.45	50.69	118.45	94.05
Tic-tac-toe	19.41	0.45	292.92	0.75	17.39	1.11
Titanic	45.67	1.15	16555.43	1.52	21.26	3.06
Waveform	3443.95	6.92	207868.82	47.44	191.21	122.89

Table 11 Comparing the DQ-Measures of MAXIMUS on seven data sets with three different discretization methods applied.

Data Set	Equi-width 10 bin	MDL-Discretized (variable bin)	Equal Frequency 10 bin
Colic	0.47	0.46	0.48
Hepatitis	0.49	0.49	0.5
Mushroom	0.47	0.45	0.47
Post-Operative	0.52	0.51	0.53
Sick	0.44	0.48	0.49
Splice	0.43	0.44	0.43
Waveform	0.49	0.5	0.56

7.2 Influence of Discretization Method on MAXIMUS Performance

Recall that in Section 4.4, we discussed how choice of discretization can influence the *ADCO* similarity value. We now report in Table 11 how the DQ measure obtained by MAXIMUS varies for the three discretization techniques discussed.

We can see from this table that the DQ measures for all three discretizations are quite similar, with perhaps marginally better DQ performance resulting from the use of equal-frequency discretization in four of the datasets. This suggests that discretization methods do not have a strong impact on the performance of MAXIMUS.

7.3 Alternate Clustering Using Membership-Based Measures ?

Lastly, a possible question to consider is whether the other membership-based clustering measures, such as the Rand or Jaccard Index, could be employed instead of *ADCO* as an objective function for alternate clustering generation. i.e Could they be encoded by constraints whose solution corresponds to a desirable alternate clustering ? We leave this as an open problem. However, there appear to be two major difficulties. Firstly,

each data point would need to be represented as a variable, meaning a huge number of variables would be required for large datasets. Secondly, it appears that the natural encoding would lead to a non-linear integer program, which is considerably more difficult to solve than an integer linear program. This is because membership based techniques require counting of the number of pairs of points occurring in the same or different clusters. The natural constraint to encode membership of pair is nonlinear (a product of variables): $p_1^c \times p_2^c$, where p_i^c is a boolean variable indicating membership of point p_i in cluster c.

8 Future Work - Extensions of *ADCO*

We now provide some brief discussion about possible enhancements or extensions of the *ADCO* measure.

More Complex Density Profiles: As described, the *ADCO* measure uses univariate profiles of each attribute, to form a density profile vector for a clustering. One might envisage the construction of more complex and detailed profiles to extend *ADCO*. In particular, consider all possible pairs of features and for each pair, represent their joint density as a 2 dimensional grid and record this in the density profile. Such an idea offers the potential advantage of capturing extra semantics of the feature space to more accurately represent a clustering. However, it has the disadvantage of making the computation of *ADCO* more complex, increasing from $O(NRQ) + O(k^3)$ to $O(NR^2Q^2) + O(K^3)$. Also, density profile vectors need to have more dimensions and as a consequence, would rapidly become much sparser as the number of bins increases.

Implicit Consideration of Non-linear Feature Spaces: The definition of the *ADCO* measure in equation 6 uses a dot product operator in both the numerator and the denominator. This opens the door to implicit consideration of non-linear features spaces, via the use of the well known 'kernel trick', for transforming a linear algorithm into a non linear algorithm, without explicitly needing to enumerate the non-linear space. Namely, instead of computing the dot product $V_C \cdot \rho(V_{C'})$, one replaces it by $\phi(V_C, \rho(V_{C'}))$, where ϕ is a kernel function. Any of the well known kernel functions might be used, such as polynomial or RBF kernel. In the case of the polynomial kernel with degree 2, the kernel function would be

$$\phi(V_C, \rho(V_{C'}) = (V_C \cdot \rho(V_{C'}) + 1)^2$$

which implicitly transforms a feature space (x, y) into the non linear feature space $(1, \sqrt{2}x, \sqrt{2}y, x^2, y^2, \sqrt{2}xy)$. The use of such a kernelized *ADCO* could be useful in situations where the clusterings and/or dataset exhibit highly non linear structure.

More Flexible Matching: Similar to the description in [15], an explicit one-to-one mapping between clusters of the two clusterings might be 'softened' so that even the unmatched cluster pairs may contribute toward the similarity. One possible approach would be to determine every possible pairwise mapping and assign appropriate weights so all pairs contribute to the final value. Alternatively, one could merge similar clusters together, to obtain an equal number of clusters in each clustering.

9 Summary and Conclusions

We have introduced a new density-based clustering similarity measure called *ADCO*, which addresses some important limitations of existing methods.

In particular, *ADCO* adopts a flexible approach to clustering comparison, by considering aspects of the feature space. This allows clustering similarity comparison to follow a data mining style philosophy, whereby similarity is judged according to properties of the clustering as a 'predictor' or a 'hypothesis'. It also means that one can for the first time, compare clusterings derived from different datasets.

We also established the usefulness of *ADCO* for the alternate clustering generation problem. By formulating alternate clustering as an integer programming problem, *ADCO* can be used as an objective function that drives the discovery of a dissimilar solution. We embodied this approach in a new alternate clustering algorithm called MAXIMUS, which can deliver multiple, high quality alternate clusterings, with good efficiency.

A Appendix

Proof of Theorem 1: Metric properties 1, 2 and 3 follow straightforwardly from the non-negativity, symmetry and identity of indiscernibles of *ADCO*. For property 4, we we need to prove that $D'_{ADCO}(C1, C2) \leq D'_{ADCO}(C1, C3)+D'_{ADCO}(C3, C2)$. Noting that $ADCO(C_i, C_j) \leq 1$ for any C_i and C_j, we can derive as follows:

$$\begin{aligned}
ADCO(C1, C3) + ADCO(C3, C2) &\leq 1 + 1 \\
ADCO(C1, C3) + ADCO(C3, C2) &\leq 2 + ADCO(C1, C2) \\
2 + ADCO(C1, C3) + ADCO(C3, C2) &\leq 2 + 2 + ADCO(C1, C2) \\
2 - ADCO(C1, C2) &\leq 2 - ADCO(C1, C3) + 2 - ADCO(C3, C2) \\
D'_{ADCO}(C1, C2) &\leq D'_{ADCO}(C1, C3) + D'_{ADCO}(C3, C2)
\end{aligned}$$

□

Acknowledgements This work was partially supported by National ICT Australia (NICTA). NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Bae, E., Bailey, J., Dong, G.: Clustering similarity comparison using density profiles. In: Australian Joint Conference on Artificial Intelligence, pp. 342–351 (2006)
2. Theodoridis, S., Koutroumbas, K.: Pattern Recognition. Academic Press (1999)
3. Gondek, D., Hofmann, T.: Non-redundant data clustering. In: International Conference on Data Mining, pp. 75–82 (2004)
4. Gondek, D., Hofmann, T.: Conditional information bottleneck clustering. In: International Conference on Data Mining, pp. 36–42 (2003)
5. Rand, W.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association pp. 846–850 (1971)
6. Davidson, I.: Clustering with constraints: Feasibility issues and the k-means algorithm. In: SIAM International Conference on Data Mining (2005)

7. Hamers, L., Hemeryck, Y., Herweyers, G., Janssen, M., Keters, H., Rousseau, R., Vanhoutte, A.: Similarity measures in scientometric research: the Jaccard index versus Salton's cosine formula. Information Processing and Management **25**(3), 315–318 (1989)
8. Wallace, D.L.: Comment. Journal of the American Statistical Association **78**(383), 569–576 (1983)
9. Hubert, L., Arabie, P.: Comparing partitions. Journal of classification **2**(1), 193–218 (1985)
10. Aggarwal, C.C.: A framework for diagnosing changes in evolving data streams. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 575–586 (2003)
11. Meilă, M.: Comparing clusterings: an axiomatic view. In: Proceedings of the 22nd international conference on Machine learning, pp. 577–584 (2005)
12. Larsen, B., Aone, C.: Fast and effective text mining using linear time document clustering. In: Proceedings of the Conference on Knowledge Discovery and Data Mining, pp. 16–22 (1999)
13. Meila, M.: Comparing clusterings, Technical Report, Dept. of Statistics, University of Washington (2002)
14. Fred, A., Jain, A.: Robust data clustering. In: Proceedings of Conference on Computer Vision and Pattern Recognition, pp. 128–133 (2003)
15. Meila, M.: Comparing clusterings - an axiomatic view. In: International Conference on Machine Learning (2005)
16. Fred, A.L.N., Jain, A.K.: Combining multiple clusterings using evidence accumulation. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(6), 835–850 (2005)
17. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. Journal on Machine Learning **3**, 583–617 (2003)
18. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: Application in vlsi domain. In: Design Automation Conference, p. 526 (1997)
19. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 81–92 (2003)
20. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistic Quarterly **2**, 83–97 (1955)
21. Ratanamahatana, C.: Cloni : Clustering of square root of n interval discretization. Data Mining IV, Info. and Comm. Tech **29** (2003)
22. Chmielewski, M.R., Grzymala-busse, J.W.: Global discretization of continuous attributes as preprocessing for machine learning. In: International Journal of Approximate Reasoning, pp. 294–301 (1996)
23. Richeldi, M., Rossotto, M.: Class-driven statistical discretization of continuous attributes (extended abstract). In: Proceedings of the 8th European Conference on Machine Learning, pp. 335–338. Springer-Verlag, London, UK (1995)
24. Bacardit, J., i Guiu, J.M.G.: Analysis and improvements of the adaptive discretization intervals knowledge representation. In: GECCO (2), pp. 726–738 (2004)
25. Torgo, L., Soares, C.: Dynamic discretization of continuous attributes. In: Proceedings of the Sixth Ibero-American Conference on AI, pp. 160–169 (1998)
26. Yang, Y., Webb, G.I.: Discretization for naive-bayes learning: managing discretization bias and variance. Machine Learning **74**(1), 39–74 (2009)
27. Ekman, G.: A direct method for multidimensional ratio scaling. Psychometrika **28**(1), 33–41 (1963)
28. Gregson, R.A.M.: Psychometrics of Similarity. Academic Press (1975)
29. Borg, I., Groenen, P.: Modern Multidimensional Scaling: Theory and Applications. Springer (1997)
30. Gower, J.C., Legendre., P.: Metric and dissimilarity properties of dissimilarity coefficients. Journal of Classification **3**, 5–48 (1986)
31. Estivill-Castro, V.: Why so many clustering algorithms: a position paper. SIGKDD Explor. Newsl. **4**(1), 65–75 (2002)
32. Mirkin, B.: Clustering for Data Mining: A Data Recovery Approach. Chapman and Hall/CRC (2005)
33. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association **66**, 622–626 (1971)
34. Meila, M.: Comparing clusterings - technical report (2003). URL citeseer.ist.psu.edu/meila02comparing.html

35. Zhou, D., Li, J., Zha, H.: A new mallows distance based metric for comparing clusterings. In: Proceedings of the 22nd international conference on Machine learning, pp. 1028–1035 (2005)
36. Repository, U.: (2008). Http://archive.ics.uci.edu/ml
37. Kendall, K.: A database of computer attacks for the evaluation of intrusion detection systems. Masters Thesis, Massachusetts Institute of Technology (1999)
38. Sung, A.H., Mukkamala, S.: Identifying important features for intrusion detection using support vector machines and neural networks. In: Proceedings of the Symposium on Applications and the Internet (SAINT), pp. 209–217 (2003)
39. Streilein, W.W., Cunningham, R.K., Webster, S.E.: Improved detection of low-profile probe and denial-of-service attacks. In: Proceedings of Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection (2001)
40. Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning **8**, 87–102 (1992)
41. Bae, E., Bailey, J.: Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In: International Conference on Data Mining, pp. 53–62 (2006)
42. Caruana, R., Elhawary, M., Nguyen, N., Smith, C.: Meta clustering. In: International Conference on Data Mining, pp. 107–118 (2006)
43. Gondek, D.: Non-redundant data clustering. In: International Conference on Data Mining, pp. 75–82 (2004)
44. Davidson, I., Ravi, S.: Identifying and generating easy sets of constraints for clustering. In: Conference on Artificial Intelligence (2006)
45. Davidson, I.: Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In: Pacific Asia Conference on Knowledge Discovery, pp. 59–70 (2005)
46. Mixed Integer Linear Programming (MILP) Solver: (2007). Http://lpsolve.sourceforge.net
47. Topchy, A.P., Law, M.H.C., Jain, A.K., Fred, A.L.: Analysis of consensus partition in cluster ensemble. In: Proceedings of the Fourth IEEE International Conference on Data Mining, pp. 225–232 (2004)
48. Topchy, A., Jain, A.K.: Clustering ensembles : Models of consensus and weak partitions. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(12), 1866–1881 (2005)
49. Topchy, A., Martin, H., Law, C., Jain, A., Fred, A.: Analysis of consensus partition in cluster ensemble. In: International Conference on Data Mining, pp. 225–232 (2004)
50. Dunn, J.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. In: Journal of Cybernetics, pp. 32–57 (1974)
51. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: International Conference on Knowledge Discovery and Data Mining, pp. 16–22 (1999)
52. Tishby, N., Pereira, F., Bialek, W.: The information bottleneck method. Allerton Conference on Communication, Control and Computing pp. 368–377 (1999)