Robust, Complete, and Efficient Correlation Clustering[*]

Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, Arthur Zimek

Department Institute for Informatics
Ludwig-Maximilians Universität München
http://www.dbs.ifi.lmu.de
{achtert,boehm,kriegel,kroegerp,zimek}@dbs.ifi.lmu.de

Abstract

Correlation clustering aims at the detection of data points that appear as hyperplanes in the data space and, thus, exhibit common correlations between different subsets of features. Recently proposed methods for correlation clustering usually suffer from several severe drawbacks including poor robustness against noise or parameter settings, incomplete results (i.e. missed clusters), poor usability due to complex input parameters, and poor scalability. In this paper, we propose the novel correlation clustering algorithm COPAC (COrrelation PArtition Clustering) that aims at improved robustness, completeness, usability, and efficiency. Our experimental evaluation empirically shows that COPAC is superior over existing state-of-the-art correlation clustering methods in terms of runtime, accuracy, and completeness of the results.

1 Introduction

Correlation clusters [5] appear as lines, planes, or, generally speaking, hyperplanes of arbitrary dimensionality $d_i < d$ in the data space, exhibiting a relatively high density of data points compared to the surrounding space. Correlation clustering algorithms group the data sets into subsets called correlation clusters such that the objects in the same correlation cluster are all associated to the same hyperplane of arbitrary dimensionality. For sake of brevity, if we have a correlation cluster associated to a λ-dimensional hyperplane, we will speak of a λ-dimensional correlation cluster. We will refer to the dimensionality of a hyperplane associated to a correlation cluster as correlation dimensionality. Of course, in applying correlation clustering one must be aware that, although linear correlation among features may indicate linear dependencies, the detected correlations can also be caused by features not comprised in the data set or they may even occur coincidentally.

Algorithms for correlation clustering integrate the concepts of clustering and correlation detection in a sophisticated way. The first approach that is exclusively designed to detect correlation clusters is ORCLUS [2] that integrates PCA into k-means clustering. The algorithm 4C [5] that integrates PCA into a density-based clustering algorithm shows superior effectivity over ORCLUS.

However, existing correlation clustering methods have several severe problems. The most important problem is that the correlation dimensionality of the detected correlation clusters heavily depends on a user-defined input parameter. ORCLUS generates results such that the correlation dimensionality of the respective correlation clusters corresponds to a user-provided parameter l. 4C limits the correlation dimension of the detected correlation clusters to the user-defined parameter λ. In fact, 4C tends to uncover correlation clusters of a correlation dimensionality that is rather near to λ. As a consequence, both methods may produce incomplete results, i.e. both are not able to find all correlation clusters of different correlation dimensionality during a single run, especially if the correlation dimensionalities of different correlation clusters vary considerably. A second drawback related to the first problem is the poor usability of the exsiting methods because they require the user to specify parameters that are usually hard to determine, e.g. the number of clusters, or the "thickness" of the correlation hyperplane. Further limitations of existing work include a weak robustness against noisy data and a poor scalability for large databases.

The most straightforward possibility to overcome the first limitation is to apply one of the existing algorithms multiple times (in fact $O(d)$ times, where d is the dimensionality of the feature space). Obviously, this is not a reasonable solution due to the considerable high computational cost for one single run. In this paper, we propose the novel correlation clustering algorithm COPAC (COrrelation PArtition Clustering) that simul-

[*] Partly supported by the German Ministry for Education, Science, Research and Technology (BMBF) under grant no. 031U212F within the BFAM project.

taneously searches for correlation clusters of arbitrary dimensionality. Let us point out that COPAC does not require the user to specify the number of clusters or any parameter regarding the correlation dimensionality beforehand. In order to further enhance the usability of COPAC, we will discuss the effect of the input parameters of COPAC. In addition, our experimental evaluation shows that COPAC is superior to ORCLUS and 4C in terms of runtime and produces significantly more accurate and complete results.

2 Related Work

A correlation cluster is a set of points where one or more features depend on other features or a (linear) combination of several features. As discussed above, most traditional clustering algorithms are not designed to detect correlation clusters.

Existing correlation clustering algorithms are usually partitioning-based (e.g. the k-means style ORCLUS [2], CURLER [14] and DIC [7]) or density-based (e.g. 4C [5]). However, the methods of both clustering paradigms suffer from (i) *weak robustness* (i.e. high sensitivity against noise and parameter settings), (ii) *incompleteness* (i.e. important clusters may be missed), (iii) *poor usability* (i.e. the input parameters are hard to determine), and (iv) *bad efficiency* (i.e. high runtimes w.r.t. database size and dimensionality). In this paper, we propose COPAC to overcome these problems by improved robustness, completeness, usability, and efficiency.

Subspace clustering algorithms that detect clusters in axis parallel projections of the original data set [3, 1, 12, 8, 4] are not able to capture local data correlations and find clusters of correlated objects since the principal axes of correlated data are arbitrarily oriented.

Pattern-based clustering algorithms [16, 15, 9, 11] limit themselves to a very special form of correlation where all attributes are positively correlated. Negative correlations or correlations where one attribute is determined by two or more other attributes cannot be detected.

3 COPAC

The general idea of COPAC is as follows: We assign a *local* correlation dimensionality to each object of the database representing the dimensionality of the correlation cluster this point fits in best. We partition the database objects according to their local correlation dimensionality in a first step. Database objects of different local correlation dimensionality can obviously not form a common correlation cluster. Therefore, in a second step, we apply a novel correlation clustering algorithm to each of the partitions in order to compute correlation clusters of different correlation dimensionalities. In the following, we describe both steps in more detail. Thereby, we assume $\mathcal{D}$ to be a database of n feature vectors in a d-dimensional feature space, i.e. $\mathcal{D} \subseteq \mathbb{R}^d$.

3.1 Local Correlation Partitioning In general, one way to formalize the concept of correlation clusters is to use PCA. Formally, let $\mathcal{C}$ be a correlation cluster, i.e. $\mathcal{C} \subseteq \mathcal{D}$, and $\bar{X}$ denote the centroid of all points in $\mathcal{C}$. The $d \times d$ *covariance matrix* $\mathbf{\Sigma}_\mathcal{C}$ of $\mathcal{C}$ is defined as:

$$\mathbf{\Sigma}_\mathcal{C} = \frac{1}{|\mathcal{C}|} \cdot \sum_{X \in \mathcal{C}} (X - \bar{X}) \cdot (X - \bar{X})^\mathbf{T}.$$

Since the covariance matrix $\mathbf{\Sigma}_\mathcal{C}$ of $\mathcal{C}$ is a positive semi-definite square matrix, it can be decomposed into the *Eigenvalue matrix* $\mathbf{E}_\mathcal{C}$ of $\mathbf{\Sigma}_\mathcal{C}$ and the *Eigenvector matrix* $\mathbf{V}_\mathcal{C}$ of $\mathbf{\Sigma}_\mathcal{C}$ such that $\mathbf{\Sigma}_\mathcal{C} = \mathbf{V}_\mathcal{C} \cdot \mathbf{E}_\mathcal{C} \cdot \mathbf{V}_\mathcal{C}^\mathbf{T}$. The Eigenvalue matrix $\mathbf{E}_\mathcal{C}$ is a diagonal matrix storing the d non-negative Eigenvalues of $\mathbf{\Sigma}_\mathcal{C}$ in decreasing order. The Eigenvector matrix $\mathbf{V}_\mathcal{C}$ is an orthonormal matrix with the corresponding d Eigenvectors of $\mathbf{\Sigma}_\mathcal{C}$.

The correlation dimensionality of $\mathcal{C}$ is the number of dimensions of the (arbitrarily oriented) subspace which is spanned by the major axes in $\mathbf{V}_\mathcal{C}$. If, for instance, the points in $\mathcal{C}$ are located near by a common line, the correlation dimensionality of $\mathcal{C}$ will be 1. One basic idea is to assign a local correlation dimensionality to all points in $\mathcal{C}$. This local correlation dimensionality of the points in $\mathcal{C}$ should be equal to the correlation dimensionality in $\mathcal{C}$. In the first step of COPAC we partition the objects of the database according to their local correlation dimensionality. The task is to determine the local correlation dimensionality for all points $p \in \mathcal{D}$ before computing the correlation clusters. In fact, we assume that the cluster in which any $P \in \mathcal{D}$ fits best can be observed from the local neighborhood of P. Thus, we compute the local correlation dimensionality of P from the k-nearest neighbors of P, denoted by $\mathcal{N}_P$.

Definition 1. (local correlation dimensionality) *Let $\alpha \in]0,1[$, $P \in \mathcal{D}$, and $\mathcal{N}_P$ denote the set of k-nearest neighbors of P. Then the* local correlation dimensionality *λ_P of the point P is the smallest number of Eigenvalues e_i in the Eigenvalue matrix $\mathbf{E}_{\mathcal{N}_P}$ explaining a portion of at least α of the total variance, i.e.*

$$\lambda_P = \min_{r \in \{1,\ldots,d\}} \left\{ r \;\middle|\; \frac{\sum_{i=1}^{r} e_i}{\sum_{i=1}^{d} e_i} \geq \alpha \right\}$$

Let us note that $\mathbf{E}_{\mathcal{N}_P}$ is the Eigenvalue matrix of $\mathbf{\Sigma}_{\mathcal{N}_P}$ which is the covariance matrix of $\mathcal{N}_P$. Typically, values for α are chosen between 0.8 and 0.9. For example, $\alpha = 0.85$ denotes that the obtained principal components explain 85% of the total variance.

Based on Definition 1, the first step of COPAC partitions the database objects according to their local correlation dimensionality. All points P sharing a common local correlation dimensionality λ_P are assigned to a partition $\mathcal{D}_{\lambda_P}$ of the database $\mathcal{D}$. This results in a set of d disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_d$ of $\mathcal{D}$, where $\mathcal{D}_i$ contains all points exhibiting a correlation dimensionality of i. Of course, some partitions may remain empty. In terms of correlation clustering, $\mathcal{D}_d$ contains only noise, because if $\lambda_P = d$, then there is no linear dependency of features found among the neighbors of P.

Let us note that by means of this first step of COPAC one does not only yield an appropriate correlation dimensionality for each point in advance, but presumably also a considerable reduction of the number of data points n, that are to be processed by each single run of a correlation clustering algorithm, on average $\frac{n}{d}$. In fact, COPAC processes each data object only once during the second step when determining the correlation clusters.

3.2 Determination of Correlation Clusters

Once we have partitioned the database objects into partitions $\mathcal{D}_1, \ldots, \mathcal{D}_d$ according to their local correlation dimensionality in $\mathcal{D}$, we can extract the correlation clusters from each of the partitions. Since each point $P \in \mathcal{D}$ can only be part of a correlation cluster of dimensionality λ_P, we can run the correlation cluster extraction on each partition $\mathcal{D}_1, \ldots, \mathcal{D}_{d-1}$ separately. As discussed above, the points in $\mathcal{D}_d$ are noise because there is no linear dependency among a set of features in the local neighborhood of these points. In the following, $\lambda_\mathcal{C}$ denotes the correlation dimensionality of cluster $\mathcal{C}$. Note that the number of attributes actually involved in linear dependencies within cluster $\mathcal{C}$ is not $\lambda_\mathcal{C}$, but $d - \lambda_\mathcal{C}$.

To detect the correlation clusters in a given partition $\mathcal{D}_i$, we propose to use a method which is similar to 4C but extends this algorithm in two aspects: First, in contrast to 4C (and also to ORCLUS), we can restrict our method to find only correlation clusters of a given correlation dimensionality, because the points in $\mathcal{D}_i$ can only be part of a correlation cluster with correlation dimensionality i. Second, 4C limits itself to "connected" correlation clusters, i.e. points that share a common hyperplane but are located significantly far apart are not assigned to the same cluster. Our method overcomes this limitation.

For clustering we need to define a distance measure that assesses the distance between two given points evaluating how well they share a common hyperplane. The Euclidean distance between the respective points is assessed only to measure the deviation orthogonal to the common hyperplane. To yield such a distance measure we first define a distance between two points with respect to one of the points. This distance is based on a distance matrix for each point P that is derived by an adaptation of the Eigenvalues of the covariance matrix of the local neighborhood of P:

Definition 2. (correlation distance matrix) *Let $P \in \mathcal{D}$, λ_P the local correlation dimensionality of P, and $\mathbf{V}_P$, $\mathbf{E}_P$ the corresponding Eigenvectors and Eigenvalues of the point P based on the local neighborhood of P, i.e. $\mathcal{N}_P$. An adapted Eigenvalue matrix $\hat{\mathbf{E}}_P$ with entries $\hat{e}_i \in \{0,1\}$, $(i = 1, \ldots, d)$ is derived according to the following rule:*

$$\hat{e}_i = \begin{cases} 0 & \textit{if} \quad i \leq \lambda_P \\ 1 & \textit{if} \quad i > \lambda_P \end{cases}$$

The matrix $\hat{\mathbf{M}}_P = \mathbf{V}_P \hat{\mathbf{E}}_P \mathbf{V}_P^\mathbf{T}$ is called the correlation distance matrix *of P.*

Using the *correlation distance matrix of P* one can easily derive a distance measure that assesses the distance between P and another point Q w.r.t. P:

Definition 3. (correlation distance measure) *Let $P, Q \in \mathcal{D}$. The* correlation distance measure *between P and Q w.r.t. point P is given by:*

$$cdist_P(P,Q) = \sqrt{(P-Q) \cdot \hat{\mathbf{M}}_P \cdot (P-Q)^\mathbf{T}}.$$

Basically, the *correlation distance measure* w.r.t. a point P is a weighted distance where the weights are based on the local neighborhood of P. The weights are constructed to take into account only distances along the Eigenvectors that correspond to small Eigenvalues, while distances along the λ_P first Eigenvectors of $\mathcal{N}_P$ are neglected. Thus, assessing the distance between P and Q using the *correlation distance measure* of P will in general not yield the same result as using the *correlation distance measure* w.r.t. Q, i.e. $cdist_P(P,Q) \neq cdist_Q(Q,P)$. Therefore, given the *correlation distance measures* for both points, P and Q, we define a distance function (i.e. a distance measure that fulfills symmetry and reflexivity) as follows:

Definition 4. (correlation distance) *Let $P, Q \in \mathcal{D}$. The* correlation distance *between P and Q is given by:*

$$cdist(P,Q) = \max\{cdist_P(P,Q), cdist_Q(Q,P)\}$$

Having defined a suitable distance function for correlation clustering, we can now integrate these concepts into GDBSCAN [13] which is a generalization of the well-known DBSCAN clustering algorithm [6]. For that purpose, we need to define a generalized neighborhood of an object O, denoted by $\mathcal{N}_{NPred}(O)$, given by $\mathcal{N}_{NPred}(O) = \{P \mid NPred(O,P)\}$, where $NPred(O,P)$ is a predicate on O and P that has to be reflexive and symmetric. In addition, to decide whether or not object O is inside a cluster (core point), a generalized minimum weight of $\mathcal{N}_{NPred}(O)$ must be defined, denoted by $MinWeight(\mathcal{N}_{NPred}(O))$.

Intuitively, we define the predicate $NPred(P,Q)$ analogously to the ε-neighborhood, using the correlation distance from Definition 4.

Definition 5. (neighborhood predicate) *Let $P, Q \in \mathcal{D}$ and $\varepsilon \in \mathbb{R}^+$. The* neighborhood predicate *of P and Q is given by: $NPred(P,Q) \Leftrightarrow cdist(P,Q) \leq \varepsilon$.*

The neighborhood predicate $NPred(P,Q)$ is reflexive and symmetric, since it is based on the reflexive and symmetric correlation distance as defined in Def. 4.

The second issue is to define the minimum weight $MinWeight$ on the neighborhood. Intuitively, if *MinWeight* of the neighborhood of a point P is true, P is considered as core point by the run of GDBSCAN. Analogously to traditional clustering, we require that a point P finds at least μ points in its ε-neighborhood using the correlation distance as distance function.

Definition 6. (minimum weight) *Let $\mathcal{N}_{NPred}(P)$ be the neighborhood of P based on the neighborhood predicate as defined in Definition 5 and $\mu \in \mathbb{N}^+$. The minimum weight of $\mathcal{N}_{NPred}(P)$ such that $P \in \mathcal{D}$ is a core point is given by:*

$$MinWeight(\mathcal{N}_{NPred}(P)) \Leftrightarrow |\mathcal{N}_{NPred}(P)| \geq \mu.$$

In summary, COPAC partitions the database points according to their local correlation dimensionality in step 1 and applies GDBSCAN with *NPred* and *MinWeight* as defined in Definitions 5 and 6, respectively, to each partition separately in step 2.

3.3 Parameter Estimation. COPAC has three input parameters. The parameter $k \in \mathbb{N}^+$ specifies the number of points considered to compute the neighborhood $\mathcal{N}_P$ of a point $P \in \mathcal{D}$. From this neighborhood, the $d \times d$ covariance matrix $\mathbf{\Sigma}_P$ and, thus, the correlation dimensionality λ_P of P is computed. As discussed above, k should not be too small in order to produce a stable covariance matrix. On the other hand, it should not be too high in order to reflect only the local correlation. Otherwise, noise points could also destabilize the correlation matrix and. It turned out that setting $k = 3 \cdot d$ was robust in all our tests throughout all our experiments. In general, setting $3 \cdot d \leq k$ seems to be a reasonable suggestion. The parameter $\mu \in \mathbb{N}^+$ specifies the minimum number of points in a cluster and, therefore, is quite intuitive. Obviously, $\mu \leq k$ should hold. The parameter $\varepsilon \in \mathbb{R}^+$ is used to specify the neighborhood predicate and can be choosen as proposed in [6] and [13]. Let us note that we have in fact a fourth parameter α to compute the correlation dimensionality λ_P of a point $P \in \mathcal{D}$. As discussed above, this parameter is very robust in the range between $0.8 \leq \alpha \leq 0.9$. Thus, we choose $\alpha = 0.85$ throughout all our experiments.

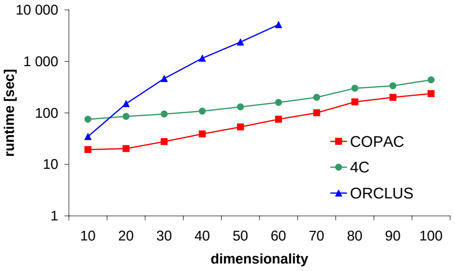


Figure 1: Runtime vs. data dimensionality.

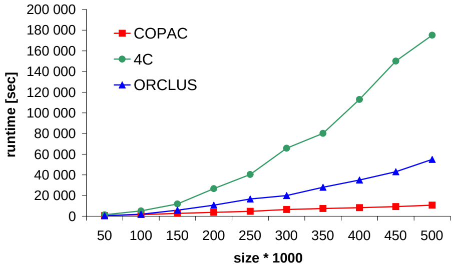


Figure 2: Runtime vs. data set size.

4 Evaluation

4.1 Efficiency In our first experiment we compared the runtime of COPAC, 4C, and ORCLUS w.r.t. the dimensionality d of the data set. To achieve a fair comparison, the parameters for all competitors were choosen such that the quality of the resulting clusterings was optimized. As it can be seen in Fig. 1, COPAC gains a significant speed-up over ORCLUS and 4C (note the logarithmic scale of the runtime).

In our second experiment we evaluated the impact of the size of the data set on the runtime of COPAC, 4C, and ORCLUS. Again the parameters for all competitors were set in order to yield the best clustering. Figure 2 illustrates the runtime of COPAC, 4C, and ORCLUS w.r.t. the data set size. Again, COPAC clearly outperforms ORCLUS and 4C in terms of efficiency.

Table 1: COPAC clustering on breast cancer data.

c ID	λ	# B	# M
1	2	108	0
2	3	12	0
3	4	100	0
4	5	46	0
5	5	0	113
6	6	0	126
noise		50	2

4.2 Robustness, Completeness, and Usability

To demonstrate the robustness, completeness, and usability of COPAC in comparison to ORCLUS and 4C, we synthesized a data set $\mathcal{D} \in \mathbb{R}^3$ with two clusters of correlation dimensionality 2, three clusters of correlation dimensionality 1, and some points of noise. The data set is depicted in Fig. 3(a). The clusters partially intersect making the separation of the clusters a highly complex task.

The predefined clusters in the synthetic data set have been separated clearly by COPAC (cf. Fig. 3). The parameters were choosen as suggested above, in particular $\mu = 50$, $k = 50$, and $\varepsilon = 0.004$. Neither ORCLUS nor 4C were able to find the clusters equally well, although we test a broad range of parameter settings. Best results for ORCLUS were reported setting $l = 2$ and $k = 5$ (cf. Fig. 4). For 4C we found the best results with $\varepsilon = 0.1$, $\mu = 50$, $\lambda = 2$, and $\delta = 0.25$ (cf. Fig. 4).

In summary, COPAC shows better robustness against noise and parameter settings than ORCLUS and 4C. Due to the suggestions presented above, the parameter choice for COPAC was very easy and results in a complete detection of clusters. For both ORCLUS and 4C, we needed several runs with different parameters in order to produce the best but still not optimal (i.e. complete) results.

4.3 Results on Real-world Data

Wisconsin Breast Cancer data. The Wisconsin Breast Cancer database [10] consists of 699 patients suffering from two types of breast cancer, benign ("B") and malignant ("M"). Each patient is represented by a 9-dimensional vector of specific biomedical features. COPAC detected six pure correlation clusters in this data set, the results are summarized in Table 1.

Gene expression data. This data set was donated by our project partners studying apoptosis (a genetically controlled pathway of cell death) in human tumor cells. The data set contains the expression level of 4610 genes at five different time slots after initiating the apoptosis pathway. COPAC found two different, biologically relevant clusters of functionally related genes (cf. Table 2). The first cluster contains negatively correlated genes related to the mitochondrion, especially to the mitochondrion membrane indicating that the volume of the energy metabolism (which is located in mitochondria) is decreasing during cell death. The second cluster that contains several genes that are related to the tumor necrosis factor (TNF) and several interleukin and interleukin receptor genes. Interleukins play a key role in the human immune system, especially in cancer response. The strong negative correlation indicates the response of the cells to necrosis again with decreasing activity while cell death proceeds.

Table 2: COPAC clustering on expression data.

cID	sample gene names	description
1	NDUFB10, MTRF1, TIMM17A, CPS1, NM44, COX10, FIBP, TRAP1, MTERF, HK1, HADHA, ASAH2, CPS1, CA5A, BNI3PL, TOM34, ME2	proteins located in and/or related to mitochondrial membran
2	TNFRSF6, TNFRSF11A, TNFRSF7, TNFRSF1B, TNFRSF5, TNFRSF1A, TRAF5, TRAF2, TNFSF12	proteins related to tumor necrosis factor (TNF)
	IL1A, IL1B, IL2, IL6, IL10, IL18, IL24, IL1RN, IL2RG, IL4R, IL6R, IL7R, IL10RA, IL10RB, IL12A, IL12RB2, IL15RA, IL22R	interleukins or their receptors activating immune response

5 Conclusions

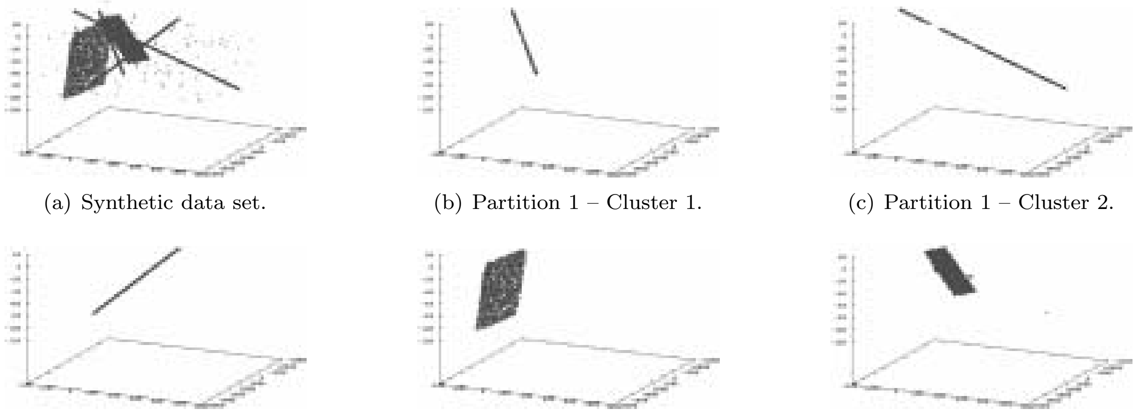
In this paper, we have proposed COPAC, a novel algorithm for correlation clustering that does not require any presumptions concerning e.g. the number of clusters or their dimensionality. The complete cluster information is retrieved by one single run of COPAC, while other algorithms usually need several runs to detect all clusters. The superiority of COPAC has been shown both theoretically as well as experimentally: COPAC clearly outperforms ORCLUS and 4C in terms of robustness, completeness, usability, and effectivity on synthetic and real world data sets.

References

[1] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proc. SIGMOD*, 1999.

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional space. In *Proc. SIGMOD*, 2000.

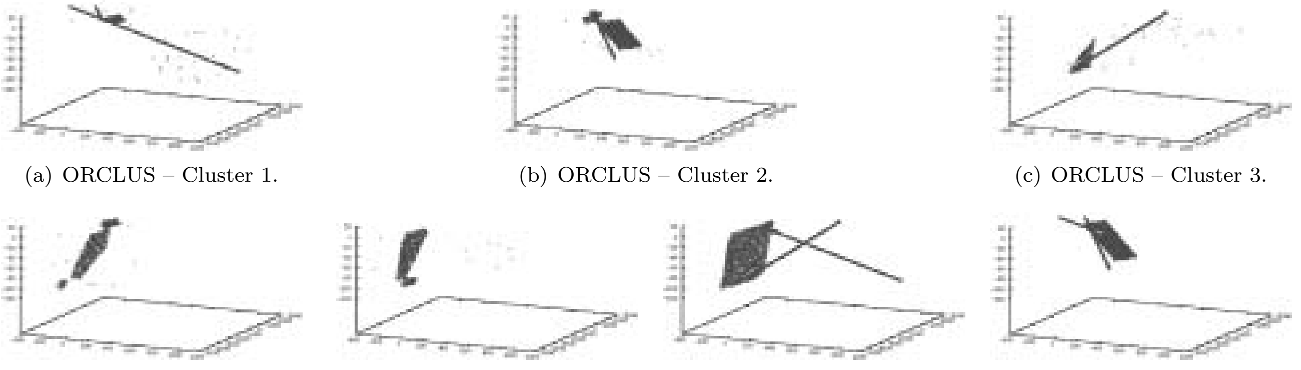
[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimen-



(a) Synthetic data set. (b) Partition 1 – Cluster 1. (c) Partition 1 – Cluster 2.

(d) Partition 1 – Cluster 3. (e) Partition 2 – Cluster 1. (f) Partition 2 – Cluster 2.

Figure 3: Synthetic data set: partitions and clustering with COPAC.



(a) ORCLUS – Cluster 1. (b) ORCLUS – Cluster 2. (c) ORCLUS – Cluster 3.

(d) ORCLUS – Cluster 4. (e) ORCLUS – Cluster 5. (f) 4C – Cluster 1. (g) 4C – Cluster 2.

Figure 4: Synthetic data set: clustering with ORCLUS and 4C.

sional data for data mining applications. In *Proc. SIGMOD*, 1998.

[4] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *Proc. ICDM*, 2004.

[5] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *Proc. SIGMOD*, 2004.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, 1996.

[7] A. Gionis, A. Hinneburg, S. Papadimitriou, and P. Tsaparas. Dimension induced clustering. In *Proc. KDD*, 2005.

[8] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proc. SDM*, 2004.

[9] J. Liu and W. Wang. OP-Cluster: Clustering by tendency in high dimensional spaces. In *Proc. ICDM*, 2003.

[10] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.

[11] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. MaPle: A fast algorithm for maximal pattern-based clustering. In *Proc. ICDM*, 2003.

[12] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proc. SIGMOD*, 2002.

[13] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2:169–194, 1998.

[14] A. K. H. Tung, X. Xu, and C. B. Ooi. CURLER: Finding and visualizing nonlinear correlated clusters. In *Proc. SIGMOD*, 2005.

[15] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proc. SIGMOD*, 2002.

[16] J. Yang, W. Wang, H. Wang, and P. S. Yu. Delta-Clusters: Capturing subspace correlation in a large data set. In *Proc. ICDE*, 2002.