In Proc. 5th IEEE International Conference on Data Mining (ICDM), Houston, TX, 2005

A Generic Framework for Efficient Subspace Clustering of High-Dimensional Data

Hans-Peter Kriegel, Peer Kröger, Matthias Renz, Sebastian Wurst
Institute for Computer Science, University of Munich
{kriegel,kroegerp,renz,wurst}@dbs.ifi.lmu.de

Abstract

Subspace clustering *has been investigated extensively since traditional clustering algorithms often fail to detect meaningful clusters in high-dimensional data spaces. Many recently proposed subspace clustering methods suffer from two severe problems: First, the algorithms typically scale exponentially with the data dimensionality and/or the subspace dimensionality of the clusters. Second, for performance reasons, many algorithms use a global density threshold for clustering, which is quite questionable since clusters in subspaces of significantly different dimensionality will most likely exhibt significantly varying densities. In this paper, we propose a generic framework to overcome these limitations. Our framework is based on an efficient filter-refinement architecture that scales at most quadratic w.r.t. the data dimensionality and the dimensionality of the subspace clusters. It can be applied to any clustering notions including notions that are based on a local density threshold. A broad experimental evaluation on synthetic and real-world data empirically shows that our method achieves a significant gain of runtime and quality in comparison to state-of-the-art subspace clustering algorithms.*

1 Introduction

One of the primary data mining tasks is clustering which aims at partitioning the data objects into groups (clusters) of similar objects. A lot of work has been done in the area of clustering (see e.g. [10] for an overview). Many real-world data sets consist of very high dimensional feature spaces. In such high-dimensional feature spaces features may be irrelevant for clustering. In addition, different subgroups of features may be irrelevant w.r.t. to varying clusters and different clusters in varying subspaces may overlap. Thus, clusters can no longer be found in the full-dimensional space. As a consequence, traditional clustering algorithms usually have scalability and/or efficiency problems. Usually, global dimensionality reduction techniques such as PCA cannot be applied to these data sets because they cannot account for local trends of overlapping clusters in the data.

A prominent application for these problems is the analysis of gene expression data. Gene expression data contains the expression level of thousands of genes, indicating how *active* it is, according to a set of samples. A common task is to find clusters of functionally related genes, having a similar expression level. Genes may exhibit several different functionalities that may be needed in different sets of samples. Thus, the genes may be clustered differently in varying subsets of samples. If the samples are patients, another task is clustering the patients into homogeneous groups according to specific phenotypes such as gender, age or diseases. Since different genes are responsible for different phenotypes, we face the problem that the patients can be clustered differently in varying subsets of genes.

To cope with the problem of local feature irrelevance, the procedure of feature selection deciding about the relevance of different features has to be linked with the clustering process more closely. In recent years, several methods have been proposed that couple the analysis of feature relevance, i.e. variance along an attribute, with the process of clustering. These existing methods can be distinguished into: (1) Algorithms that compute a discrete partitioning of the data objects, i.e. each object is uniquely assigned to one cluster; an arbitrary subset of attributes may be relevant for the cluster. (2) Algorithms that automatically detect all clusters in all subspaces of the original feature space. Clusters in different subspaces may overlap.

Obviously, algorithms that allow overlapping clusters generate more information than algorithms that compute a unique assignment because they uncover the information that objects may be clustered differently in varying subspaces. As discussed above, our

motivating applications, call for solutions that explicitly allow overlapping clusters. Thus, in this paper, we focus on overlapping clusters. Throughout the rest of the paper, when we speak of subspace clustering, we mean the task of finding overlapping clusters.

In this paper, we present FIRES (FIlter REfinement Subspace clustering), a general framework for efficient subspace clustering. It is generic in such a way that it works with all kinds of clustering notions. It starts with 1D clusters that can be constructed with a clustering method of choice and merges these 1D clusters to generate approximations of subspace clusters. An optional refinement step can compute the true subspace clusters, again using any clustering notion of choice and a variable cluster criterion. Thus, FIRES overcomes two severe limitations of existing subspace clustering methods as we will discuss later (cf. Section 2).

The rest of this paper is organized as follows. We discuss related work and point out our contributions in Section 2. The generic FIRES framework is described in Section 3. Section 4 presents a broad experimental evaluation of FIRES. Section 5 concludes the paper.

2 Related Work and Contributions

The complexity of the exhaustive search for all subspace clusters is $O(2^d)$, where d is the data dimensionality. Most subspace clustering algorithms work bottom-up similar to the *Apriori* algorithm for finding frequent itemsets, starting with 1D clusters and merging these clusters to compute clusters of higher dimensionality. To reduce the search space, the algorithms usually rely on a downward closure property of density enabling the *Apriori*-like search strategy. [13] provides a review of subspace clustering algorithms.

CLIQUE [2], the pioneering approach to subspace clustering uses a grid-based clustering notion. The data space is partitioned by an axis-parallel grid into equi-sized units of width ξ. Only units which contain at least τ points are considered as dense. Dense units satisfy the downward closure property. A cluster is defined as a maximal set of adjacent dense units. Obviously, the accuracy and the efficiency of CLIQUE heavily depends on the granularity and the positioning of the grid. On the other hand, a higher grid granularity results in higher runtimes. Modifications of CLIQUE include ENCLUS [5] and MAFIA [12].

SUBCLU [11] uses the DBSCAN cluster model of density-connected sets [8]. It is shown that density-connected sets satisfy the downward closure property. Compared to the grid-based approaches SUBCLU achieves a better clustering quality but requires a higher runtime.

Another recent approach called DOC [14] uses a random seed of points to guide a greedy search for subspace clusters. DOC measures the density of subspace clusters using hypercubes of fixed width w and thus has similar problems like CLIQUE.

Recently, some feature selection techniques have been proposed, that elaborate the clustering quality of subspaces [6, 3]. Most of these methods cannot use an *Apriori* style approach but use random or greedy search methods instead, producing incomplete results or suffering from high runtimes.

Pattern-based clustering [16] aims at grouping points in clusters that exhibit a similar trend in a subset of attributes rather than points that are dense in some subspace. This approach is too general for our focus and usually suffers from high runtimes.

As discussed above, projected clustering algorithms that do not allow overlapping clusters, e.g. PROCLUS [1] and PreDeCon [4], are also not considered here.

Our Contributions. The existing subspace clustering algorithms usually have the following two drawbacks. First, all methods usually scale exponentially with the number of features and/or the dimensionality of the subspace clusters [13]. Second, most subspace clustering approaches use a global density threshold for performance reasons. Usually, the global density threshold provides the downward closure property ensuring the application of the *Apriori* style algorithm. However, it is quite questionable that a global density threshold is applicable on clusters in subspaces of considerably different dimensionality, since density naturally decreases with increasing dimensionality. In addition, the density of clusters in a common subspace may significantly vary.

Here, we propose a framework for efficient subspace clustering that (i) scales at most quadratic with the data dimensionality and the subspace dimensionality, and (ii) allows to use any sophisticated clustering model, e.g. one that accounts for the subspace dimensionality and for local point density.

3 Efficient Subspace Clustering

In the following, we assume that $\mathcal{D}$ is a database of n points in a d-dimensional feature space, i.e. $\mathcal{D} \subseteq \mathbb{R}^d$.

3.1 General Idea

Subspace clustering is a complex problem due to the high complexity of the search and the output space. The time consuming step is to identify the relevant attributes of the subspace cluster. We claim that it

is desirable to generate subspace clusters of maximum dimensionality. Since each cluster in a subspace S is also hidden in each projection of S, this redundancy can be encapsulated in the subspace cluster of maximal dimensionality. This reduces the search space and especially the output space significantly. Our key idea is to use a filter-refinement architecture to speed up the subspace finding process, i.e. we efficiently compute approximations of the subspace clusters which can be refined to generate the true subspace clusters and are of maximum dimensionality.

Our framework FIRES (FIlter REfinement Subspace clustering) consists of the following three steps:

1. Preclustering: First, all 1D clusters called *base-clusters* are computed. This is similar to existing subspace clustering approaches and can be done using any clustering algorithm of choice.

2. Generation of Subspace Cluster Approximations: In a second step, the base-clusters are merged to find maximal-dimensional subspace cluster approximations. However, we do not want to merge them in an *Apriori* style but use an algorithm that scales at most quadratic w.r.t. the number of dimensions.

3. Postprocessing of Subspace Clusters: A third step can be applied to refine the cluster approximations retrieved after the second step.

The key benefits of our method are (1) the dramatically reduced runtime and (2) the possibility to apply a clustering model that can detect clusters of different densities to Step 1 and 3.

3.2 Preclustering

Similar to existing subspace clustering approaches, FIRES first generates clusterings in each dimension of the data space. Thereby we assume that the clusterings of each dimension suffice to identify all higher-dimensional subspace clusters. For this preprocessing step, we can choose any clustering method, e.g. DBSCAN [8], k-means [10], SNN [7] or grid-based clustering. In the following we call the 1D clusters resulting from the pre-clustering *base-clusters*, denoted by C^1.

For the preclustering we propose a filter step which drops irrelevant base-clusters, that do not contain any vital subspace cluster information. Small base-clusters do not likely include significant subspace clusters, because they usually indicate a sparse area in the higher-dimensional space. For this reason we compute the average size s_{avg} of all base-clusters and skip those base-clusters whose size is smaller than 25% of s_{avg}. The threshold of 25% is empirically determined and has shown best results in our experiments.

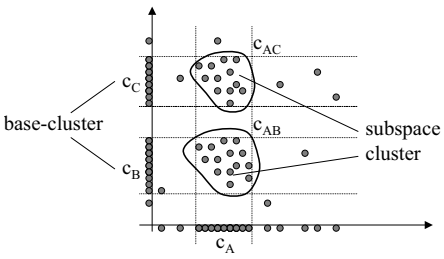


Figure 1. Overlapping subspace clusters

3.3 Generation of Subspace Cluster Approximations

Approximations of maximal-dimensional subspace clusters are determined by suitably merging the base-clusters derived from our preprocessing step. Out of all merge possibilities, whose number is exponential in the number of base-clusters, we have to find the most promising merge candidates by searching for all base-clusters which contain nearly the same objects. In the following, we call c_{app} *subspace cluster approximation* of a subspace cluster c_{sub}, iff c_{app} is built by merging all base-clusters sharing objects with c_{sub}. Those objects which are shared by a set S_B of base-clusters must be close in all dimensions of the base-clusters in S_B, and thus, must be close in the subspace spanned by S_B. Consequently, a good indication for the existence of a subspace cluster is if the intersection between base-clusters is large. In the following, we denote base-clusters to be similar if they share a sufficiently high number of objects. The *similarity between base-clusters* $c_1, c_2 \in C^1$ is defined as $sim(c_1, c_2) = |c_1 \cap c_2|$.

In contrast to functions commonly used to measure the similarity between sets, such as the Jaccard-Distance, our similarity function is insensitive to noise. This is necessary because the base-clusters usually contain, besides true subspace-cluster objects, a lot of noise in higher-dimensional space.

Before starting with the merge phase, we have to detect promising merge candidates, possibly hidden in the set of base-clusters. Different subspace clusters may overlap in one or more dimensions, thus one base-cluster can include objects of multiple subspace clusters. In our approach, we try to merge similar base-clusters. The example depicted in Figure 1 shows two subspace clusters c_{AB} and c_{AC}, indicated by the base-clusters c_A, c_B and c_C. The base-cluster c_A is similar to the base-cluster c_B and to c_C. However c_B and c_C are obviously not similar. In order to avoid that we merge all three base-clusters c_B, c_C, and c_A, we have to split c_A into two separate instances, such that

one instance mainly contains the objects of c_{AB} and the other contains the objects of c_{AC}. More generally, each base-cluster which includes more than one overlapping subspace cluster, should be split into multiple base-clusters in such a way, that each of them contains at most one of the subspace clusters. In the example of Figure 1 the base-cluster c_A is similar to two base-clusters which are from the same dimension. In this special case, we can easily detect that c_A indicates two subspace clusters, since two base-clusters of the same dimension must be disjunctive. However, it may happen that one base-cluster of dimension D includes separate subspace clusters which may differ in all dimensions except for D. In our approach, we apply simple heuristics for the decision when to split base-clusters. First, we search for the most-similar-cluster of a split candidate, which is defined as follows:

Definition 1 (Most Similar Cluster) *Let $c \in C^1$; the* most similar base-cluster $MSC(c) \in C^1$ *of c ($c \neq MSC(c)$) fulfills the following constraint:*
$\forall c_p \in C^1 : sim(c, MSC(c)) \geq sim(c, c_p)$.

Let the base-cluster c_S be the split candidate and let c_T be the most similar cluster of c_S, i.e. $c_T = MSC(c_S)$. If the number of objects shared by both clusters c_S and c_T is very high, then the shared objects probably indicate a subspace cluster. If there is another subspace cluster hidden in c_S, the number of the remaining objects $o \in (c_S - c_T)$ must also be large. We simply split c_S, iff both, the number of objects shared by c_S and c_T and the number of the remaining objects in $c_S - c_T$ exceed a specific threshold. Let s_{avg} be again the average size of all base-clusters, then we split c_S into $c_{S1} = (c_S \cap c_T)$ and $c_{S2} = (c_S - c_T)$, iff $|c_S \cap c_T| \geq \frac{2}{3} s_{avg} \wedge |c_S - c_T| \geq \frac{2}{3} s_{avg}$.

After detecting all base-clusters, we have to identify the most promising merge candidates. Our similarity function does not suffice to identify them, because it penalizes small clusters and prefers large clusters. In addition, our goal is not to merge only two base-clusters, but to consider a set of base clusters to be merged simultaneously. Thus, we need a relative similarity measure to identify suitable merge candidates.

Definition 2 (k-Most-Similar-Clusters) *Let $c \in C^1$. We call $MSC_k(c) \subseteq C^1$ the* k-most-similar-clusters, *iff it is the smallest subset of C^1 that contains at least k base-clusters, where the following condition holds: $\forall c_p \in MSC_k(c), \forall c_q \in C^1 - MSC_k(c) :$ $sim(c, c_p) > sim(c, c_q)$.*

The k-most-similar-clusters of c_A contain those k base-clusters, that share the most objects with c_A.

In our approach, we try to merge those base-clusters which are all mostly similar to each other, i.e. that share many of their k-most-similar-clusters. This avoids that a set of well fitting base-clusters will be destroyed if we add a merge candidate which is similar to one of the base-clusters but does not fit well to the entire set. Thus, in order to obtain base-cluster sets with homogeneous similarity between their elements, we prefer to merge base-clusters where many of their k-most-similar-clusters are equal.

Definition 3 (Best-Merge-Candidates) *Let $c \in C^1$ and $k, \mu \in \mathbb{N}^+$ ($\mu \leq k$). The* best-merge-candidates *$BMC(c)$ of c are defined by the set:*
$BMC(c) := \{x \in C^1 | \ |MSC_k(c) \cap MSC_k(x)| \geq \mu\}$.

Merging all best-merge-candidates should suffice to filter out unfitting merge candidates and seems very promising to find high quality subspace clusters. However, due to the limiting parameter k this method may be too restrictive, and thus, important cluster information may be lost. We by-pass this problem by allowing to merge best-merge-candidate sets. Two base-clusters are merged, if they share at least one base-cluster which fulfills the properties of a best-merge-cluster, which is defined in the following:

Definition 4 (Best-Merge-Cluster) *Let $c \in C^1$ and $minClu \in \mathbb{N}^+$, then c is called* best-merge-cluster, *iff $|BMC(c)| \geq minClu$.*

The generation of subspace cluster approximations proceeds as follows: We first generate all best-merge-clusters. Then we group all pairs of best-merge-clusters c_A and c_B, iff $c_A \in BMC(c_B)$ and *vice versa*. Finally, we add all best-merge-candidates to these groups. The resulting set of base-clusters form our subspace cluster approximations.

3.4 Postprocessing

Significant subspace clusters should achieve a good trade-off between dimensionality and cluster size. However, the information of the base-clusters does not suffice to generate correct and complete subspace clusters. Therefore, we require postprocessing steps in order to achieve good results. We propose two steps which are subsequently applied to all subspace-cluster approximations: (1) "Pruning" improves the quality of the mergeable-cluster-set by identifying and removing "meaningless" base-clusters. (2) "Refinement" removes noise and completes the subspace clusters.

Pruning. There may be clusters in the subspace cluster approximations which do not contain relevant information of the corresponding subspace cluster. In order to eliminate these irrelevant clusters, we need a notion of the quality of a cluster C w.r.t. its dimensionality $dim(C)$ and the number of objects shared by all its base clusters $size(C) = |\bigcap_{i=1..n} c_i \in C|$. We define the *quality of a subspace cluster* C as $score(C) = f(size(C)) \cdot dim(C)$, where $f : \mathbb{N} \to \mathbb{R}_0^+$ denotes a weighting function for the cluster size. Obviously the size of the corresponding subspace cluster should have less weight than its dimensionality. In our experiments, we achieved the best results with $f(x) = \sqrt{x}$.

The pruning removes a merge candidate $c \in C^1$ of a subspace-cluster approximation C if the following condition holds:

$$score(C - \{c\}) > score(C) \wedge$$

$$\forall c_p \in C : c_p \neq c \Rightarrow score(C - \{c_p\}) \leq score(C - \{c\}).$$

After removing the merge-candidate c, we continue the top-down pruning with the remaining merge candidates, until we achieve the best score. The score-function avoids that large high-dimensional subspace clusters degenerate to low-dimensional clusters.

Refinement. The subspace cluster c_{sub} which is approximated by c_{app} can be built from the base-clusters $c_1, .., c_n$ merged in c_{app}. Two building methods can be distinguished, the union $c_{sub} = c_1 \cup .. \cup c_n$ and the intersection $c_{sub} = c_1 \cap .. \cap c_n$. Obviously, both variants do not yield the true subspace cluster c_{sub}. Due to the fact that the base-clusters contain a lot of noise in a higher dimensional subspace, the intersection variant seems to produce more accurate approximations than the union variant. However the intersection merge has significant draw-backs because the detected subspace cluster would be too strict, thus many promising candidates would be lost. Furthermore, parts of the true subspace cluster can be outside of the approximation which would lead to uncomplete results.

We achieve better subspace cluster results when applying an additional refinement step. First, we use the union of all base-clusters in order to avoid that potential cluster candidates are lost or become uncomplete. Then, we cluster again the merged set of objects in the corresponding subspace. Thereby, we can apply any clustering algorithm, e.g. DBSCAN, SNN-clustering, k-means, grid-based clustering or even hierarchical clustering. We propose the use of SNN to detect clusters of different density, or the use of DBSCAN. Thereby the density threshold is adjusted to the subspace dimensionality, i.e. the $minPts$-value is kept constant and we adjust the ε-range as follows: $\varepsilon = \varepsilon_1 n / \sqrt[d]{n}$, where ε_1 denotes the ε-value in the 1D subspace.

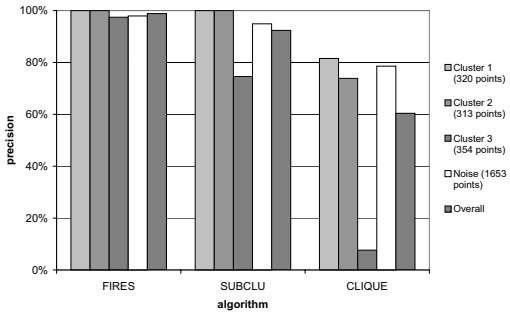


Figure 2. Comparison of accuracy.

4 Evaluation

We evaluated FIRES in comparison with CLIQUE and SUBCLU. All tests were run on a LINUX workstation featuring a 2.4 GHz CPU and 3.6 GB RAM.

4.1 Accuracy

To measure the effectiveness of FIRES, SUBCLU and CLIQUE, we measure the ratio of points found in each cluster/noise and the points that should be in the according cluster/noise. We do not consider the redundant projections of all subspace clusters generated by the *Apriori* style of SUBCLU and CLIQUE but only concentrate on the true clusters hidden by the data generator. We apply DBSCAN to generate the base-clusters using a parameter setting as suggested in [8] and as refinement method with paramter settings for ε and $minpts$ as proposed in Section 3.4. Figure 2 illustrates results of FIRES in comparison to SUBCLU, and CLIQUE applied on a synthetic dataset containing three clusters of significantly varaying dimensionality and density. For each method, the result with the best parameter setting is presented. It can be observed that FIRES clearly outperfoms SUBCLU and CLIQUE w.r.t. accuracy. Only FIRES finds the third cluster having the lowest density in a 10D subspace with a satisfying precision. Both, CLIQUE and SUBCLU, miss out major parts of this cluster.

In Figure 3 we evaluate the impact of different algorithms to generate the base-clusters on the final accuracy. We compare DBSCAN, k-means, and a CLIQUE-like approach, which partitions each dimension into intervals of equal size. An interval is considered as dense if it contains more than $minPts$ points. Neighboring

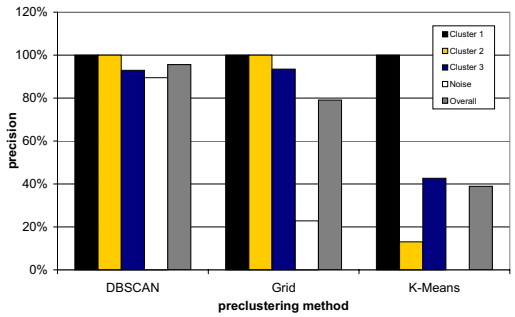


Figure 3. Comparison of preclustering.

dense cells are linked to form base-clusters. The parameters for each method are optimized to achieve a fair comparison. Again, the used data set contains three clusters in subspaces of different dimensionality and exhibits significantly different densities. Obviously, using DBSCAN as preclustering method yields the best results. Using k-means, cluster 2 cannot be detected. The grid-based approach has problems with cluster 3, which has the highest subspace dimensionality.

We evaluated the impact of parameterization by changing one of the three parameters k, μ and $minClu$ at a time, while the other two are fixed at an optimized value. The results are illustrated in Figure 4. The parameter μ is not critical as far as it is not set too low. If it is set too low, some clusters are still found with high precision but some of the clusters are not found at all. Roughly the same can be said about the parameter k. If k is reduced, the precision drops slightly, whereas precision drops significantly if k is increased. In our experiments, we observed that we achieve good results if μ not differs significantly from k. The third parameter $minClu$ turned out to be stable throughout the range of tested settings, i.e. the precision of FIRES is very high at any tested $minClu$ value. Accuracy only decreases very slightly when we increase the value for $minClu$. However, we made the observation, that reducing $minClu$ increases the number of generated subspace clusters because we get a larger number of best-merge-clusters.

We found out that DBSCAN with self-adapting parameters is slightly more accurate than SNN (detailed results are not shown due to space limitations). Nevertheless, if the densities of clusters in subspaces of equal dimensionality differ significantly, SNN achieves better results than DBSCAN. However, SNN clustering requires considerably a higher runtime than DBSCAN, and thus, we decided to use DBSCAN in our remaining tests. The next experiment compared the effectiveness of the results using and leaving out certain aspects of our postprocessing step. We compared the results of using both steps (pruning and refinement), using only one of the two steps, and using none of them. The results (not shown due to space limitations) suggest that applying no postprocessing decreases the accuracy of FIRES considerably. Clearly the best results can be achieved when using both steps of postprocessing. The reason for this is that the pruning step removes base-clusters — i.e. dimensions — from the cluster approximation that reduce the quality of the subspace cluster. Without any dimensions that do not belong to the cluster, the refinement finds only those points within the approximation that really belong to the cluster.

4.2 Scalability

We compared the scalabilty of FIRES, CLIQUE and SUBCLU on synthetic data sets.

For evaluating the scalability w.r.t. the number of points in the dataset (cf. Figure 5(a)) we use 25D datasets each contains one 5D subspace cluster. We analyse the time FIRES needed for base-clusters generation and the time needed for the merge-step. Obviously, FIRES scales slightly superlinear w.r.t. the number of points in the dataset, more than CLIQUE but less than SUBCLU. This is due to the use of DBSCAN as method for preclustering and refinement. The runtime of CLIQUE scales linear because of the rather simple but efficient grid-based clustering model. However, the computation of subspaces is obviously the bottleneck for the overall runtime. Since FIRES scales better in comparison to CLIQUE w.r.t. data dimensionality, on higher dimensional datasets, the number of points may become a less critical parameter. Due to the linear scalability of the merge step of FIRES, one can use a simpler, e.g. grid-based, clustering model rather than DBSCAN to speed-up FIRES.

We investigate the scalability w.r.t. the data dimensionality using datasets of 1,000 points each containing one 5D subspace cluster. As it can be seen from Figure 5(b), the runtimes of both, CLIQUE and SUBCLU, increase clearly superlinear, whereas the runtime of FIRES increases only linear. Our experiments show that FIRES is the only method that can efficiently be applied to a dataset with more than 1,000 dimensions.

The impact of the highest dimensionality of a subspace cluster on the runtime is evaluated using datasets of 1,000 50D points. From Figure 5(c) we observe FIRES again clearly outperfoms SUBCLU and CLIQUE.

Last, we evaluate the runtime of FIRES using different preclustering. We compare DBSCAN, k-means and the grid-based approach on a 10D dataset, one 5D subspace cluster and a varying number of points in the

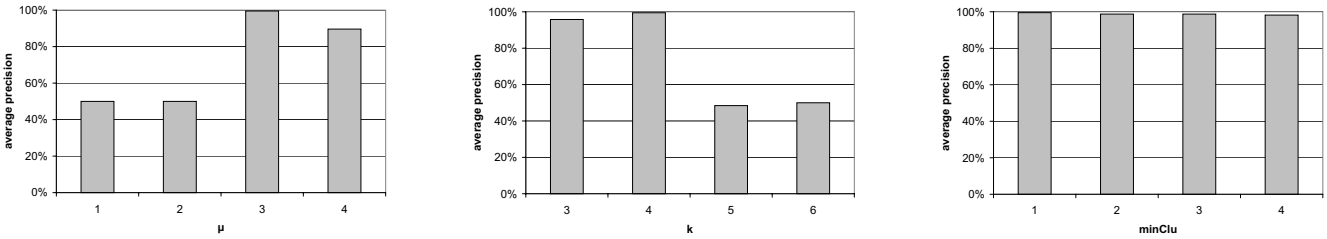
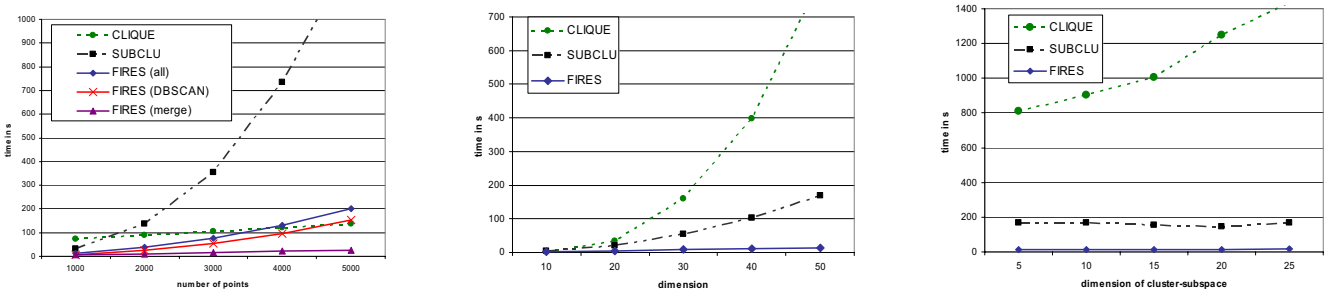


Figure 4. Impact of parameterization on the accuracy of FIRES.



(a) Number of points (b) Data dimensionality (c) Subspace dimensionality

Figure 5. Scalabilty of FIRES, SUBCLU, and CLIQUE.

datasets. The results (not shown due to space limitations) illustrate that the runtime of FIRES using any of these algorithms is still approximately quadratic w.r.t. to the number of points in the dataset, but FIRES with k-means as well as with the grid-based approach is clearly faster than with DBSCAN. This experiment shows that the runtime can be decreased significantly if applying simpler but more efficient clustering methods for preclustering and/or postprocessing.

4.3 Real-World Data

We applied FIRES to two real-world gene expression datasets. The first data set called "Bio1" [15] contains the expression level of about 4,000 genes measured at 24 different time slots during yeast mitotic cell cycle. The task is to find clusters of functionally related genes. The second dataset called "Bio2" [9] contains the expression levels of 72 patients suffering acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). For each patient, approximately 7100 genes are measured. The task is to find groups of patients with homogeneous phenotypes. The patients are labeled according to their leukemia type. For some of them information on additional phenotypes such as gender is provided.

Some sample clusters FIRES found on the Bio1 dataset are visualized in Figure 6. Biological criteria for interesting and meaningful clusters are genes that have similar function, participate in a common pathway, and/or build larger complexes. Based on these criteria, we evaluated the clusters FIRES generates using the public yeast genome database SGD[1]. As it can be seen, in both datasets FIRES uncovers subspace clusters that fulfill at least one of the biological criteria and thus are biologically meaningful and relevant. On the first dataset, one cluster contains the genes CKB1 and CKA2, both subunits of the protein kinase CK2. In the same cluster, DFR1 and ARO1 are two genes participating in the chorismate pathway. A second cluster contains four genes with the same biological function (YIP1, SED5, BFR2, and SEC21). A third cluster contains three genes that are part of the large ribosomal complex (RPL12B, RPL14A, and RPL16A). CLIQUE and SUBCLU could not reproduce these clusters.

The results of FIRES on the Bio2 dataset were also very interesting. The data set is problematic since it has only a very few number of points (72) but a very high dimensionality (7070). In general, the 72 patients may be clustered rather differently, depending on dif-

[1] `http://www.yeastgenome.org/`

ORF	Gene	Annotation
Cluster 1		
YGL019W	CKB1	subunit of CK2
YOR061W	CKA2	subunit of CK2
YOR236W	DFR1	chorismate pathway
YDR127W	ARO1	chorismate pathway
Cluster 2		
YGR172C	YIP1	ER to Golgi transport
YLR026C	SED5	ER to Golgi transport
YDR299W	BFR2	ER to Golgi transport
YNL287W	SEC21	ER to Golgi transport
Cluster 3		
YDR418W	RPL12B	part of ribosome
YIL133C	RPL16A	part of ribosome
YKL006W	RPL14A	part of ribosome

Figure 6. Results on Bio1.

ferent phenotypes. FIRES generated a lot of subspace clusters of varying dimensionality that may represent partitionings according such certain phenotypes. Unfortunately, most clusters could not be evaluated since more information on the patients is not known. However, three clusters found by FIRES provide very interesting insights. One cluster consists of 16 patients 15 of which suffer from ALL. In fact, all these ALL patients are marked as B-cell ALL patients. No T-cell ALL person is in that cluster. A second cluster contained 25 patients, 18 male and only three female. The gender of the remaining four patients is not annotated. A third cluster contained 27 of 32 ALL (of both, B-cell and T-cell type) patients. Let us point out that the response time of FIRES on this very high dimensional dataset was quite low with around 15 seconds. Both SUBCLU and CLIQUE produced memory overflows due to a high number of candidate subspaces.

5 Conclusions

In this paper, we proposed the new filter-refinement subspace clustering algorithm FIRES, which overcomes both problems of existing subspace clustering methods including inadequate scalabilty w.r.t. data dimensionality and/or subspace dimensionality and the use of a global density threshold. FIRES efficiently computes maximum dimensional cluster approximations from 1D clusters that can be refined to obtain the true clusters. Any clustering notion can be incooperated into FIRES in order to account for clusters of varying density. A thorough experimental evaluation has shown that the effectiveness of FIRES is significantly better than that of well-known algorithms such as SUBCLU and CLIQUE. In addition, FIRES clearly outperforms SUBCLU and CLIQUE in terms of scalability and runtime w.r.t. data dimensionality and subspace dimensionality. We demonstrated the usability of FIRES finding interesting and biologically meaningfull clusters in several different gene expression datasets.

References

[1] C. C. Aggarwal and C. Procopiuc. "Fast Algorithms for Projected Clustering". In *Proc. ACM SIGMOD*, 1999.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications". In *Proc. ACM SIGMOD*, 1998.

[3] C. Baumgartner, K. Kailing, H.-P. Kriegel, P. Kröger, and C. Plant. "Subspace Selection for Clustering High-Dimensional Data". In *Proc. ICDM*, 2004.

[4] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. "Density Connected Clustering with Local Subspace Preferences". In *Proc. ICDM*, 2004.

[5] C.-H. Cheng, A.-C. Fu, and Y. Zhang. "Entropy-Based Subspace Clustering for Mining Numerical Data". In *Proc. ACM SIGKDD*, 1999.

[6] M. Dash, K. Choi, P. Scheuermann, and H. Liu. "Feature Selection for Clustering – A Filter Solution". In *Proc. ICDM*, 2002.

[7] L. Ertöz, M. Steinbach, and V. Kumar. "Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data". In *Proc. SIAM Data Mining*, 2003.

[8] 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.

[9] T. R. Golub et al. "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring". *Sience*, 286:531–537, 1999.

[10] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2001.

[11] K. Kailing, H.-P. Kriegel, and P. Kröger. "Density-Connected Subspace Clustering for High-Dimensional Data". In *Proc. SIAM Data Mining*, 2004.

[12] H. Nagesh, S. Goil, and A. Choudhary. "Adaptive Grids for Clustering Massive Data Sets". In *Proc. SIAM Data Mining*, 2001.

[13] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: A review. *SIGKDD Explorations*, 6(1):90–105, 2004.

[14] C. M. Procopiuc, M. Jones, P. K. Agarwal, and M. T. M. "A Monte Carlo Algorithm for Fast Projective Clustering". In *Proc. ACM SIGMOD*, 2002.

[15] P. Spellman et al. "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization.". *Molecular Biolology of the Cell*, 9:3273–3297, 1998.

[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.