Scalable Density-Based Subspace Clustering

Emmanuel Müller [○] Ira Assent [◇] Stephan Günnemann [●] Thomas Seidl [●]

[○]Karlsruhe Institute of Technology, Germany
emmanuel.mueller@kit.edu

[◇]Aarhus University, Denmark
ira@cs.au.dk

[●]RWTH Aachen University, Germany
{guennemann, seidl}@cs.rwth-aachen.de

ABSTRACT

For knowledge discovery in high dimensional databases, subspace clustering detects clusters in arbitrary subspace projections. Scalability is a crucial issue, as the number of possible projections is exponential in the number of dimensions. We propose a scalable density-based subspace clustering method that *steers* mining to few selected subspace clusters. Our novel steering technique reduces subspace processing by identifying and clustering promising subspaces and their combinations directly. Thereby, it narrows down the search space while maintaining accuracy. Thorough experiments on real and synthetic databases show that steering is efficient and scalable, with high quality results. For future work, our steering paradigm for density-based subspace clustering opens research potential for speeding up other subspace clustering approaches as well.

Categories and Subject Descriptors: H.2.8 Database management: Database applications [Data mining]

General Terms: Algorithms, Performance, Experimentation

Keywords: data mining, high dimensional data, scalability, density-based clustering, subspace clustering

1. INTRODUCTION

Clustering is an established technique for knowledge discovery in databases. Objects are grouped into clusters based on mutual similarity. In high dimensional databases, however, traditional clustering methods fail to identify meaningful groups in all attributes due to the "curse of dimensionality" [9]. In many applications, such as gene expression analysis, environmental surveillance or customer segmentation, one measures for each object a huge variety of attributes. Unfortunately, due to the scattered high dimensional data spaces clusters can not be detected using all available attributes. Groups of objects are hidden in subsets of the attributes, e.g. genes (objects) forming functional groups w.r.t. a specific set of medications (attributes). These application demands have led to research on subspace and projected clustering for more than a decade. Clusters are mined in subspaces of the high dimensional space. Unlike dimensionality reduction methods, subspace clustering determines relevant subspace projections per cluster [27, 25, 19].

As major drawback of existing subspace clustering algorithms, we observe their poor scalability w.r.t. increasing dimensionality. To search for clusters in any combination of the attributes, most subspace clustering algorithms borrow the apriori principle from association rule mining [4], as suggested in the first subspace clustering algorithm CLIQUE [3]. Using a monotonicity property, these approaches proceed in a step-by-step scheme from x-dimensional to $x+1$-dimensional subspaces and prune the search space.

This approach is commonly used in many subspace clustering algorithms [3, 29, 14, 22, 21, 7]. While this step-by-step processing avoids naive search of all possible subspaces, it has to generate exponentially many lower dimensional projections of each high dimensional subspace cluster. For a d-dimensional database, the search space can only be reduced from 2^d (naive search) to 2^k (apriori step-by-step search) where k is the maximal dimensionality of any hidden cluster. As a consequence, scalability with dimensionality is poor. This is also observed in our evaluation study [25], which shows that none of these algorithms scales.

Especially, for the density-based clustering paradigm, e.g., known from DBSCAN [11], and extended to subspace clustering e.g. in [14, 5], runtime performance is not acceptable for high dimensional data. Density-based approaches compute the neighborhood around each object in each subspace under study. Thus, the excessive database access results in poor scalability w.r.t. both dimensionality and database size. For example, runtimes of 7 days have been reported for 50-dimensional databases, and exponential increase w.r.t. maximal cluster dimensionality [14].

Furthermore, it has been shown that the exponential number of lower dimensional projections does not only hinder scalable processing, but also results in a tremendous amount of redundant subspace clusters that overwhelm the user. Recent models have been proposed to select subspace clusters that provide non-redundant result sets [5, 21, 7, 23]. However, it has been proven that selecting an optimal set of subspace clusters is an NP-hard problem [23]. Clearly, generation of all redundant clusters before optimizing the result set is an infeasible processing.

In this work, we take a novel approach to enhance the search space processing. We steer our subspace clustering directly to interesting subspace clusters, and exclude exces-

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

sive processing of many redundant cluster candidates. Thus, our novel processing scheme is scalable to high dimensional data. We reduce the computational overhead by excluding the vast majority of subspaces. We make use of the very effect crippling previous approaches: Any high dimensional subspace cluster shows up in many low dimensional projections. By mining only some of them, we gather enough information to "jump" directly to the more interesting high dimensional subspace clusters without processing the subspaces in-between.

A core contribution of our work is that we actively *steer* the algorithm to find the most interesting subspace clusters. Our steering includes efficient information gathering for the density-based paradigm, but more importantly, we propose a novel processing scheme that requires databases access only in very few subspace regions. It combines intermediate results to progressively identify new regions for subspace clustering in a best-first manner. This means that database scans for density computations are avoided entirely for very many subspace projections. Overall, we thus realize a scalable density-based subspace clustering processing.

Our contributions include:

- General processing scheme for subspace clustering that is scalable with increasing number of dimensions
- Our novel *steering* technique that allows best-first style selection of subspace clusters
- An instantiation of our steering for density-based clustering, extendible to other clustering paradigms

Additionally, we show empirically that our best-first selection of subspace clusters enables a scalable but also high quality subspace clustering. Based on recent subspace clustering models [5, 23], it ensure high quality results. Overall, our novel processing scheme results in enhanced runtime behavior and high quality subspace clustering.

This paper is structured as follows: we review related work in the following section (Section 2). We begin our discussion in Section 3 by providing some preliminaries and reviewing the scalability challenges faced in subspace clustering. We introduce our novel steering technique in Section 4, where we discuss information gathering (Section 4.1) and selective clustering of interesting subspace regions (Section 4.2), before analyzing the runtime complexity (Section 4.3). We demonstrate superior runtime and quality in the experiments (Section 5), before concluding.

2. RELATED WORK

Subspace clustering approaches are summarized in two surveys [27, 19]. Apriori based algorithms going through the search space step-by-step are the most commonly used ones. Both breadth-first [3, 26, 14, 22] and depth-first [29, 7] exhaustive processing of subspace dimensions do not scale w.r.t. the dimensionality and the result becomes overwhelmingly large as confirmed by our comparison study [25]. Similar scalability drawbacks of apriori processing have been observed also in related areas such as frequent itemset mining [4, 32]. However, for subspace clustering apriori based algorithms are still state-of-the-art.

Projected clustering iteratively partitions the database into clusters in subspace projections [1, 2, 30]. This process can be accelerated using randomization [28], or specialized data structures [31]. As opposed to subspace clustering, however, these partitioning approaches only determine a single projection for each object. Hence, they are unable to detect overlapping subspace clusters [19, 25].

Heuristics for selecting subspaces prior to clustering have been proposed [10, 15]. However, these heuristics are limited due to the fact that clusters usually only exist in some regions of a subspace, which are hidden by noise in the remaining regions. Other approaches use density estimation of regions in subspaces [18, 24]. While this improves efficiency, due to the limited information on just one or two dimensions, quality is relatively poor.

Indexing approaches for subspace clustering face the challenge that data needs to be searched w.r.t. arbitrary subsets of the attributes. Consequently, only inverted lists for single dimensions [14], or grid cell count structures for depth-first mining [6, 7] have been proposed for subspace clustering. In general, subspace clustering could benefit from novel index structure development [17, 20]. However, the exponential search space would still be the main problem. Thus, we focus on reducing the search space as the major challenge to be solved for scalable subspace clustering.

3. PRELIMINARIES AND SCALABILITY CHALLENGES

We begin by introducing notations and basic definitions. Given a database DB of objects $o = (o_1, \dots o_d) \in DB$ with d dimensions, traditional clustering aims at detecting groups of objects in all dimensions DIM. Subspace clustering searches for clusters in arbitrary subspace projections $S \subseteq DIM$. In general, a subspace cluster is described by a tuple $C = (O, S)$ with $O \subseteq DB$ being the clustered objects in a subspace S. For density-based subspace clustering, a cluster (O, S) is defined by a set of dense objects in subspace S [14, 5].

DEFINITION 1. ***Density-Based Subspace Cluster***
A subspace cluster $C = (O, S)$ is a set of objects $O \subseteq DB$ which form a dense region in a subspace $S \subseteq DIM$, i.e.:

(1) $\forall o \in O : density_S(o) \geq MinPts$
$density_S(o) = |N_\varepsilon(o)| = |\{p \in DB \mid dist_S(o,p) \leq \varepsilon\}|$

(2) $\forall o, p \in O : \exists q_1 \dots q_n : \; o = q_1 \wedge p = q_n$
$\wedge \forall i : q_{i-1} \in N_\varepsilon(q_i) \wedge q_i$ *fulfills* (1)

(3) $\forall o, p \in DB$ *fulfilling* (1) $\wedge$ (2) $: o \in O \Leftrightarrow p \in O$

Density is assessed by counting the objects in the ε-neighborhood w.r.t. the distance in the subspace projection only, i.e. for the Euclidean Distance in subspace S one uses the distande $dist_S(o,p) = \sqrt{\sum_{i \in S}(o_i - p_i)^2}$. According to the density-based paradigm [11], (1) the objects O form a dense region, (2) they are density-connected and (3) O is a maximal set of objects. As distances are measured in a subspace S, each object is part of at most one cluster in one subspace, but it may be part of multiple clusters in different subspaces.

In a naive computation, one would run density-based clustering on each of the 2^d possible subspaces. This does not scale, and thus, subspace clustering algorithms focus on pruning the exponential search space. Existing work using apriori density-based subspace clustering [3, 26, 29, 14, 7] relies on a density monotonicity property:

DEFINITION 2. ***Density Monotonicity***
If (O, S) is a dense subspace cluster, then for all $S' \subseteq S$ the projection (O, S') is dense as well.

This property has been proven for the density-based paradigm [14]. It is used for pruning by negation: if the lower dimensional projection is not a subspace cluster, then the higher dimensional projection is discarded. This excludes several of the sparse high dimensional subspaces, however, the majority of low dimensional subspaces has to be processed. For a density-based cluster in a k-dimensional subspace, existing techniques have to perform costly database scans in 2^k subspaces. This fact is at the core of the scalability issues faced by these approaches: each subspace cluster is processed redundantly in all its lower dimensional projections, creating a tremendous amount of redundant clusters. This results in exponential runtime w.r.t. number of dimensions.

A second scalability issue arises from density assessment for all objects in each clustered subspace. Naively, checking their ε-neighborhoods has quadratic complexity in the number of objects $O(|DB|^2)$. For a single (full) space, index structures may be used to reduce the complexity to $O(|DB| \log |DB|)$ as in [11]. However, in subspace clustering, one would need index support for any subspace projection. Few index structures have been proposed for subspace projections [17, 20], and only one-dimensional index structures with intersection operations (inverted lists) are used for subspace clustering [14]. Thus, subspace clustering could benefit from novel index structure development, which could be used to support our steering approach as well.

Obviously, state-of-the-art apriori approaches are not scalable neither with the number of objects in the database nor with the number of dimensions. It is our main goal to reduce the number of processed subspaces, and thus, the costly database access.

4. SCALABLE SUBSPACE CLUSTERING

In our work, we tackle the scalability challenges described above. We exploit the monotonicity property of density-based subspace clusters to exclude redundant low dimensional projection. We base on our non-redundant subspace clustering model [23]. It selects only the most interesting subspace clusters $R = \{C_1, \dots C_n\} \subseteq ALL$ where ALL is the set of all valid density-based subspace clusters according to Def. 1. The goal is to cluster as many objects as possible with only few high dimensional subspace clusters.

In contrast to previous approaches, we directly *steer* to high dimensional subspace clusters. The core idea is that we do not need to search all low dimensional subspaces in order to work our way up to the high dimensional subspaces. Instead, we use the monotonicity property that any high dimensional subspace cluster is reflected in many low dimensional projections (cf. Def. 2). While this redundancy is the drawback for apriori-based approaches that process all of these 2^k subspaces, we use this effect to combine low dimensional information for direct "jumps" to high dimensional subspaces. In Figure 1 this general principle is illustrated: We start with some initial information about the subspace clusters in the data. This initial information (*jump sources*) comprises low dimensional subspace regions which might be projections of the same high dimensional clusters. Their combination leads us directly to higher dimensional subspaces (*jump targets*, green regions in Fig. 1). Substantial runtime improvements are realized since subspaces besides the jump sources and the jump targets do not need to be analyzed. Furthermore, our steering requires only a single database scan for initialization and additional database access for the density-based refinement of each resulting subspace cluster. Overall, this enables the scalability of our processing in both the size and the dimensionality of the database.

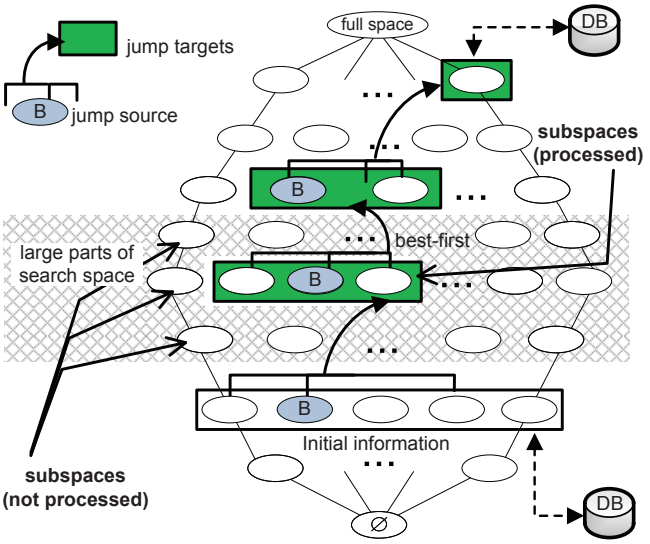


Figure 1: Multiple steered jumps

To reach high dimensional subspaces via jumps, information from the very low dimensional spaces, however, might not be sufficiently accurate. For example, 2-dimensional regions typically do not provide enough insight to jump directly to 20-dimensional regions. Thus, we introduce *multiple jumps* with successive information gathering. As illustrated in Figure 1, our technique jumps based on the initial information to some promising targets; these targets provide us with novel information that can be used for further and higher dimensional jumps in later steps. Thus, the jump targets from previous steps serve as jump sources in later steps. Using this intermediate information leads to better steering of our approach, enhanced jumps, and thus, to better clustering quality. Let us note that we still only use a few jump sources and never generate the full set of potential subspace regions in all the intermediate subspace projections. Thus, we avoid many subspaces altogether, i.e. multiple jumps tackle the exponential search space problem. Our main contribution for scalability and high quality jumps is the choice of the jump targets (cf. Sec. 4.2). However, let us discuss the underlying information gathering first.

4.1 Information Gathering for Steering

Steering requires information about the most promising subspace regions. This information is gathered during multiple jumps and is used to choose the next jump targets. Let us first describe the instantiation of information gathering for the density-based clustering paradigm before we discuss its general properties, which might be of interest for extensions to other paradigms as well.

Density-Based Instantiation. For density-based clustering it is essential to provide an information gathering that ensures scalability w.r.t. the number of objects in the database. Thus, we try to use as few database scans as possible to gather information on promising dense regions. We base on our previous approaches using hyperrectangle approximation of density-based subspace clusters and density estimation techniques [6, 24]. In a nutshell, these techniques do not compute the actual density by costly database scans but they base on object counts in hyperrectangles.

Hyperrectangles are obtained through discretization of the search space. Discretization is a well-known method for speeding up subspace clustering within a given subspace [3, 26, 29, 6]. We define hyperrectangles as follows:

DEFINITION 3. ***Hyperrectangle***
A hyperrectangle H in subspace $S = \{d_1, \ldots, d_s\}$ is defined as a tuple $[(d_1, low_1 - up_1), \ldots, (d_s, low_s - up_s)]$ of lower and upper bound interval identifiers $low_i - up_i$ in dimension d_i.

We write e.g. $H = [(1,4),(2,3-5),(3,6)]$ if upper and lower bounds in a dimension are identical, i.e., short for $H = [(1,4-4),(2,3-5),(3,6-6)]$, a hyperrectangle in interval four in dimension one, intervals three to five in dimension two, and interval six in dimension three.

So far, hyperrectangles have been used only in apriori-based processing schemes to process subspace regions from low dimensional to high dimensional subspaces. As all potentially dense regions are processed, these techniques are not able to tackle the exponential search space. We use hyperrectangles as an information source for steering to the most promising subspaces instead.

General Information Gathering. In general, any clustering model could benefit from our steering. The only required property is the underlying monotonicity of interesting regions, i.e., high dimensional subspace clusters are reflected in low dimensions. We gather this low dimensional information, which allows us to steer our processing to the desired high dimensional subspace clusters.

Our main contribution is to change the essential part of subspace clustering algorithms, the processing scheme. In contrast to previous methods we do not process subspaces step-by-step in breadth-first or depth-first order through all lower dimensional projections. Instead, we gather information about processed hyperrectangles in a priority queue and use this as jump information to go directly to the most promising subspaces. The priority queue simply stores the gathered information about dense subspace regions. In our case, it is a sorted list of dense hyperrectangle. The important part is the sorting function in the priority queue, and the selection of the most interesting subspace region as a jump target. Each jump provides a higher dimensional hyperrectangle to be stored in the priority queue. We discuss the jump in the following section.

The general idea of information gathering in our priority queue is based on simple object counts using hyperrectangles as in all of the traditional grid-based subspace clustering definitions [3, 26, 29] or efficiency improvements for density-based subspace clustering [6, 24]. The significant improvement is the novel processing of these hypercubes by a steered best-first search. All other methods use the density monotonicity property (cf. Def. 2) for step-by-step processing in breadth-first or depth-first order. Our algorithm uses these ideas for the efficient information gathering, but the scalability w.r.t. increasing number of dimensions is achieved by the steered jumps. This novel processing is the essential step in our algorithm.

Algorithm 1 Scalable Subspace Cluster Search

```
1  R = ∅
2  pqueue = DB.initializeEstimation()
3  while ∃ candidate (O, S) with uncovered objects do
4      candidate = pqueue.getFirst()
5      if candidate not yet processed then
6          targets = steeredJumps(pqueue)
7          pqueue.addAll(targets)
8      else
9          R = R ∪ {DB.refine(candidate)}
10         pqueue.resort(R)
```

A generalized overview of our approach is given in Algorithm 1. First, we initialize our information in the priority queue by density estimation. This gives us the basis for selecting the best candidate from the priority queue. If a candidate is selected for the first time, it is used for steered jumps (cf. Sec. 4.2). We perform multiple jumps to higher dimensional subspaces and add these to our information basis. Else, if the candidate has already been used as jump source before, it is refined (additional database access). We insert it into the result set R and resort the queue according to our sorting function, which depends on the current result set R. The algorithm continues until the database is represented (covered) entirely, except for possible noise.

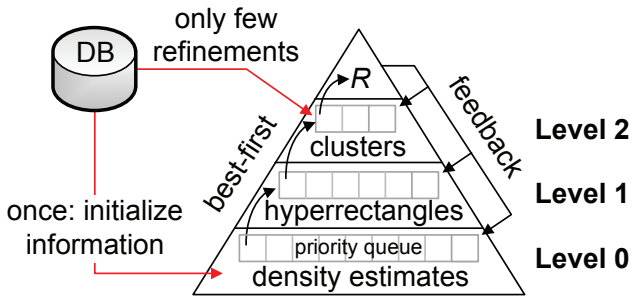


Figure 2: Priority queues for steering

The general processing scheme using our novel steering procedure (cf. Algorithm 1) is instantiated for density-based subspace clustering as follows. We split up the priority queue into three levels for multiple density granularities. Thus, while steering through subspaces, we are able to gather information about density estimates (Level 0), hyperrectangles (Level 1) up to density-based clusters (Level 2) in specific subspace regions.

Figure 2 illustrates the priority queue at three levels: Level 0 at the bottom consists of rough density estimates of subspace regions. For efficient but accurate density estimation, we use density estimation based on 2-dimensional histograms [24]. While the density estimates are very efficiently computable, they give only a rough approximation of potential subspace cluster regions. In order to steer our algorithm effectively through the search space, we use a much tighter approximation of subspace clusters through hyperrectangles

on the next level (Level 1). With our novel steering we do best-first refinement of cluster candidates on each level, the most promising dense regions move up to the next levels. At the top, on Level 2 density-based subspace clusters are selected for the final result.

4.2 Steering: Jumping to Promising Subspaces

In this section, we describe how we steer search to the most promising subspace regions. We give details on the steered jumps based on the gathered hyperrectangles and define a preference function for the best-first search. We note the following jump terminologies (cf. Fig. 1):

- The *jump source* is a hyperrectangle B; this candidate is the basis / starting point of a jump and the most promising hyperrectangle detected so far.
- The *jump information* is a set $\mathcal{I}$ of already generated cluster candidates / hyperrectangles; it helps to steer the jump to promising regions and consists of the remaining entries in the priority queue.
- A *jump target* T is a hyperrectangle selected for a jump; it is an extension of the jump source B to new hyperrectangles in higher dimensionalities. We discuss selection below.

In a running example for this section, assume a jump source $B = [(\mathbf{1},\mathbf{2}),(\mathbf{2},\mathbf{2}),(\mathbf{4},\mathbf{2})]$ that consists of the second discretized interval in dimensions one, two and four. With the jump information, assume we identified dimensions and intervals $H = [(\mathbf{5},\mathbf{2}),(\mathbf{7},\mathbf{2}),(\mathbf{6},\mathbf{2}),(\mathbf{11},\mathbf{2})]$ as a meaningful extension for B. The hyperrectangles B and H are combined to jump target T to form a new candidate in a higher dimensional subspace, i.e. the jump target covers all dimensions and intervals of both B and H. Formally, we get the new higher dimensional hyperrectangle T with $Dim(T) = Dim(B) \cup Dim(H)$, $\forall d_i \in Dim(B) : Int(B,d_i) = Int(T,d_i)$ and $\forall d_i \in Dim(H) : Int(H,d_i) = Int(T,d_i)$. With $Dim(H)$ being the dimensions contained in H, and $Int(H,d_i)$ as the corresponding intervals. In our example, the jump source could be extended using the jump information by simply forming the union of the two: $B \cup H = [(\mathbf{1},\mathbf{2}),(\mathbf{2},\mathbf{2}),(\mathbf{4},\mathbf{2}),(\mathbf{5},\mathbf{2}),(\mathbf{7},\mathbf{2}),(\mathbf{6},\mathbf{2}),(\mathbf{11},\mathbf{2})]$. Our goal, of course, is the combination of the jump source with the best hyperrectangles in the jump information, in the sense that most promising jump targets are identified. In the following we discuss the actual selection of jump targets.

Determining jump targets. To find jump targets, i.e., promising high dimensional regions, we need to identify dimensions and intervals from the currently available jump information, for possible combination with the jump source. Obviously, dimensions already included in the jump source do not lead to new jump targets, and neither do combinations of different intervals in the same subspace. Such combinations are therefore not valid for the purpose of steering to promising regions.

We refer to valid combinations of hyperrectangles in the jump information as *jump indicators*. A jump indicator can be imagined as a derived hyperrectangle that does not share any dimension with the jump source. Also, several jump indicators can be combined with the jump source to obtain the jump target of a certain dimensionality. Formally, jump indicators are:

Definition 4. ***Jump indicator***
A jump indicator P of length k is a hyperrectangle, which does not share any common dimensions with the jump source B such that $(Dim(P) \cap Dim(B) = \emptyset)$ and $|Dim(P)| = k$.

In our example, the jump target is derived out of a ranking of indicators $(\mathbf{5},\mathbf{2}),(\mathbf{7},\mathbf{2}),(\mathbf{6},\mathbf{2}),(\mathbf{11},\mathbf{2}),\ldots$ in combination with the jump source. In general, we rank jump indicators by their potential impact, i.e. how "helpful" they are in leading to subspace clusters for our result set. Deriving the impact is discussed later on.

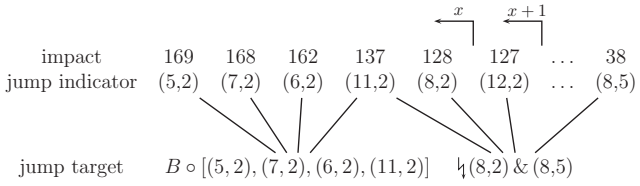


Figure 3: Generating jump targets

In the final step, we combine these ranked jump indicators with the jump source to jump targets. In the following, we illustrate this process assuming $k = 1$ (one-dimensional indicator) for simplicity of presentation. In Figure 3, we give an example ranked list of possible jump indicators (dimension and interval tuples in the center row of the figure) and their impact values (top row of the figure). The combination of four one-dimensional jump indicators with our jump source yields a jump target with the dimensionality of $B = [(1,2),(2,2),(4,2)]$ plus four: $[(5,2),(7,2),(6,2),(11,2)]$. Please note, that some combinations are not valid, since they would result in disjoint intervals in the same dimension. For example, in the lower right in the figure, intervals $(8,2)$ and $(8,5)$ cannot be combined with the source at the same time. By construction of the hyperrectangles, there cannot be a subspace cluster in dimension eight existing in both interval two and interval five.

Thus, we construct jump targets by combining smaller k-dimensional indicators. These potential jump targets are ranked, such that we always process the most promising combinations first. We include jump indicators up to a position x in the ranking (cf. top right in Fig. 3). Jump targets are constructed only from these jump indicators. Once those jump targets have been generated, the position of x is incremented by one, and new jump targets are generated up to the new value of x (arrow for $x+1$ in the figure).

In order to limit the number of cluster candidates that are processed, we generate only a fixed number of jump targets. Parameter *generatedClusters* sets the number of subspace clusters that are generated from a given jump source. To show that only few cluster candidates have to be generated, we study this parameter experimentally in Section 5.

Impact of jump indicators. Obviously, the crucial part in determining jump targets lies in the ranking of the jump indicators based on their impact. As mentioned above, we know that any high dimensional subspace cluster is likely to be reflected in many different low dimensional projections. We use this observation to calculate the impact. If a jump indicator P is part of several currently known subspace regions, this suggests that it is a projection of a higher dimensional cluster. The first aspect to calculate the im-

pact is based on the similarities of the jump source B and any hyperrectangle H_i in the jump information. If the cluster similarity $cs(B, H_i)$ (introduced below and formalized in Definition 6) is high and P is included in H_i, the impact should also be high.

As a second aspect, we want to steer the search to novel dense regions that are not yet represented to avoid generating redundant results. Thus, we propose a function *cp* (cluster preference) to measure representation of new information. Cluster preference is the basis for our best-first search, and provides flexible means to be instantiated also by other user specific interestingness measures. In this work, we restrict ourself to pure coverage of objects as basic measure. Subspace clusters which contain objects which are already represented (covered) by the result set contribute little or no new knowledge to the result. By prioritizing hyperrectangles that are not yet covered, we favor those jump targets which are likely to lead to the desired subspace clusters.

Overall, we define the impact of a jump indicator as the product of the cluster similarity (cf. Def. 6) and the cluster preference (cf. Def. 7) values. Thus, the impact takes the similarity between the jump source and the hyperrectangles into account to identify promising higher dimensional regions, while giving preference to those that contain not yet covered objects. We present the two functions *cs* and *cp* in more detail later on.

DEFINITION 5. ***Impact of jump indicators***
Let P be a jump indicator, B the jump source and $\mathcal{I}$ the jump information. The impact of P is the cluster similarity and preference of all hyperrectangles $H_i \in \mathcal{I}$ in which P is contained.

$$\sum_{H_i \in \mathcal{I},\ P^{in}\ H_i} cs(B, H_i) \cdot cp(H_i)$$

An obvious question to ask is how to efficiently obtain the jump indicators and their impact. We propose an efficient way of generating all possible jump indicators along with their impact. We transform the problem of combining hyperrectangles (given by their intervals in the respective dimensions) into a problem of (weighted) itemset mining [12, 13]. A k-dimensional jump indicator can be mapped to a k-itemset, i.e. a set of k items that occur in a transaction. The (weighted) support, i.e. the number of occurrences of this itemset, corresponds to the impact. In itemset mining, efficient algorithms exist [13], which we use to compute the interestingness, and the top-m-ranked results.

An example is given in Fig. 4: at the top, the jump source is given as a set of tuples of dimensions and interval identifiers. To determine the desired jump indicators, the source can be combined with the remaining hyperrectangles in the first column. The second column notes dimensions and intervals which could be used to extend the source. To compute the impact w.r.t. a jump indicator, e.g. [(5, 2)], we sum up the products $cs \cdot cp$ as the (weighted) support of all transactions that contain the jump indicator. So, the first transaction contributes to the support, but not the second, etc. Note that this procedure can easily be extended to ranges of intervals, e.g. an entry $(1, 3-5)$ would be mapped to items $(1, 3), (1, 4), (1, 5)$.

Cluster similarity and cluster preference. Calculation of similarities between subspace clusters is an important aspect for the impact of jump indicators. The higher

jump source: $[(\mathbf{1}, \mathbf{2}), (\mathbf{2}, \mathbf{2}), (\mathbf{4}, \mathbf{2})]$, jump indicator $[(5, 2)]$

jump information (hyperrectangle H_i)	transactions (potential extension)	$cs \cdot cp$ w.r.t. H_i	support for $[(5, 2)]$
$[(\mathbf{1}, \mathbf{2}), (\mathbf{2}, \mathbf{2}), (5, 2)]$	$\{(5, 2)\}$	$\mathbf{2}$	$+2$
$[(\mathbf{2}, \mathbf{2}), (\mathbf{4}, \mathbf{2}), (7, 2)]$	$\{(7, 2)\}$	$\mathbf{2}$	
$[(\mathbf{1}, \mathbf{2}), (5, 2), (6, 2)]$	$\{(5, 2), (6, 2)\}$	$\mathbf{1}$	$+1$
$[(\mathbf{1}, \mathbf{2}), (5, 2), (7, 2)]$	$\{(5, 2), (7, 2)\}$	$\mathbf{1}$	$+1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Figure 4: Impact of jump indicators

the similarity $cs(B, H_i)$ of a hyperrectangle H_i, the more interesting this hyperrectangle is with respect to the jump source B. A high similarity indicates that both hyperrectangles might be a projection of the same high dimensional subspace cluster. Thus, this jump indicator should be prioritized. The goal in determining the cluster similarity *cs* is that the more dimensions are shared by the jump source and another hyperrectangle, the more similar they should be. Also, we need to take into account in which intervals they overlap. Clearly, if the intervals are disjoint, the two hyperrectangles do not represent the same higher dimensional subspace cluster.

Thus, only overlapping hyperrectangles have positive similarity values. We define the number of overlapping intervals of two hyperrectangles H_1 and H_2 in dimension d as

$$Overlap(H_1, H_2, d) = \begin{cases} |Int(H_1, d) \cap Int(H_2, d)| & (*) \\ 0 & \text{else} \end{cases}$$

$$(*)\ d \in Dim(H_1) \cap Dim(H_2)$$

Beside just determining the cardinality of the intersection of the dimensions, e.g. $|Dim(B) \cap Dim(H_i)|$, our overall definition of the cluster similarity takes also the portion of overlap into account:

DEFINITION 6. ***Cluster Similarity***
For a given jump source B, and the hyperrectangles H_i available as jump information, let $|Dim(B) \cap Dim(H_i)| = k$. We define the cluster similarity cs as follows:

$$cs(B, H_i) = \begin{cases} k \cdot \sqrt[k]{\prod_{d \in Dim(B) \cap Dim(H_i)} \frac{Overlap(B, H_i, d)}{|Int(B, d)|}} & k > 0 \\ 0 & else \end{cases}$$

Thus, similarity is the average overlap between the jump source and the jump information and therefore it reflects the degree to which promising new hyperrectangles can be generated through their combination.

In the following we give an extended example on the calculation of our cluster similarity function *cs*. First of all, we require for two similar hyperrectangles that their corresponding intervals overlap. In Figure 5, two hyperrectangles are depicted in their two-dimensional projections. Both of them contain dimensions 1 and 2, but since they do not share intervals in dimension 2, they do not cover the same area in the search space. Thus, they do not represent the same high dimensional cluster, and we cannot infer meaningful jump indicators based on these two hyperrectangles. Consequently, the similarity denotes a value of 0 (cf. Def. 6).

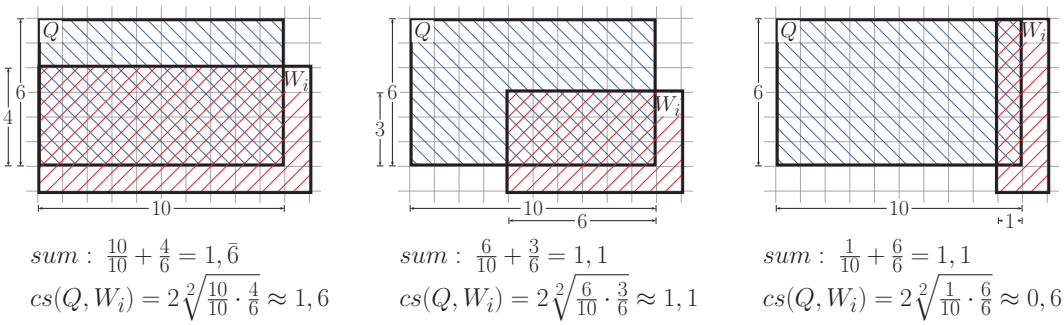


Figure 6: Similarity computation

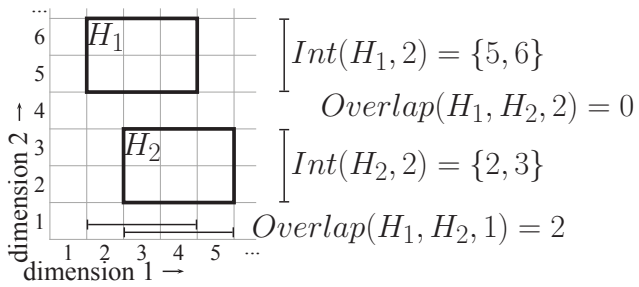


Figure 5: Hyperrectangles in the same subspace with no overlap (disjoint intervals in dimension 2)

As a second aspect, the extent of overlap between two hyperrectangles has to be considered. Hyperrectangles which overlap more (e.g. Fig. 6 (a) more than (b)), are much more likely to be reflections of the same higher dimensional subspace cluster, so they should be considered more similar with respect to the function *cs*. Consequently, the impact of the corresponding jump indicators is also higher.

To incorporate also the extent we do not simply determine the number of shared dimensions between hyperrectangles, but we consider the overlap with respect to the intervals in the jump source $Overlap(B, H_i, d)/|Int(B, d)|$. Thereby, we much better reflect the degree to which the two hyperrectangles are thought to reflect the same higher dimensional subspace cluster. The sum over these overlaps

$$\sum_{d \in Dim(B) \cap Dim(H_i)} (Overlap(B, H_i, d)/|Int(B, d)|)$$

, however, conceals small overlaps in one dimension if there is a larger overlap in another. We therefore propose using a product of the overlaps. Consider the example in Figure 6: taking the sum of overlaps yields $6/10 + 3/6 = 1.1$ in (b) and also $1/10 + 6/6 = 1.1$ in (c), even though the overlap is considerably larger in (b). Taking the product of the overlapping intervals is e.g. in (c) $1/10 \cdot 6/6 = 1/10$, and in (b) a higher value of $6/10 \cdot 3/6 = 3/10$.

The number of dimensions, however, is underrated: e.g. an overlap of 8/10 per dimension yields $(8/10)^2$ for two dimensions, yet$(8/10)^3$ for three dimensions. To make sure that we still favor those situations where more dimensions overlap, we introduce a normalization factor which intuitively gives us the "average" overlap per dimension. Considering the product, e.g. $1/10 \cdot 6/6 = 1/10$, as a volume in the k-dimensional space, we determine the side length of a hypercube of the same volume, i.e. we compute the kth root of the volume. In the example in (c), this hypercube has a side length of $\sqrt[2]{1/10} \approx 0,32$, whereas (b) has a value of $0,55$. Multiplying this value with the number of overlapping dimensions k is our Definition 6 of the cluster similarity.

For the second aspect, the cluster preference function *cp* should steer the jump to regions not yet covered by other clusters. Thus, w.r.t. the currently known subspace clusters available, R and its covered objects $Cov(R)$, we denote the objects in H that are already contained in R as $|Obj(H) \cap Cov(R)|$ and the number of objects in the hyperrectangle itself as $|Obj(H)|$. We define the preference function such that those hyperrectangles are preferred which contain many objects that are not yet included in other subspace clusters:

DEFINITION 7. ***Cluster preference***
Let R be the current set of already detected clusters. Then we define the cluster preference function cp for a hyperrectangle H as the number of objects in H that are not yet covered:

$$cp(H) = |Obj(H)| - |Obj(H) \cap Cov(R)|$$

Both cluster preference and cluster similarity are combined to steer our best-first search according to Def. 5.

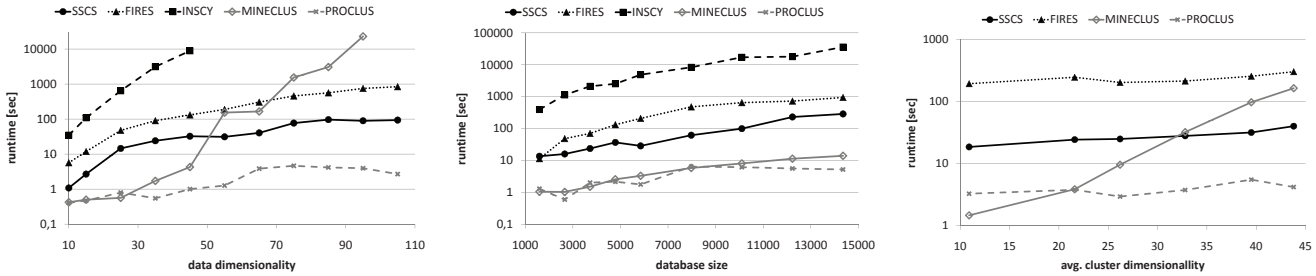
4.3 Complexity Analysis

Our novel jump technique using intermediate jump sources is the key for scalable processing. Let us discuss best and worst case scenarios for the computational complexity of this approach compared with the exponential complexity of traditional step-by-step algorithms. Their best case is our worst case: The multi-jump processing degenerates to a step-by-step processing if we perform jumps with only a single dimension increment. Even approximated density regions cannot reduce expensive database scans in this scenario as all subspaces regions are investigated. Thus, our worst case scenario is computationally equivalent to the best case processing for traditional apriori based methods with a complexity of

$$O(2^k \cdot |DB|^2)$$

where k is the maximal subspace cluster dimensionality, and $|DB|$ denotes the size of the database.

However, in general we do not perform step-by-step processing. Our steering excludes large parts of the search space, and thus, significantly reduces the complexity. In



(a) data dimensionality (b) size of the database (c) avg. cluster dimensionality

Figure 7: Scalability w.r.t. dimensionality and size of the database

the best case, our multi-jump technique only investigates the high dimensional density-based clusters that are output as the result R. In addition to a single database scan for initialization, our method directly jumps to the clustered subspace regions and has a best case complexity of:

$$O((1 + |R|) \cdot |DB|^2)$$

i.e. the complexity is independent of the dimensionality, and depends only on the output size. Even though the complexity of our approach remains quadratic in the size of the database, using density estimation and discretization greatly reduces the overall computational cost as intermediate jumps are performed without additional database access. This is confirmed in the experiments (cf. Sec. 5).

The steering based on these intermediate jumps is an important aspect for high quality results. A single jump from low dimensional to high dimensional subspaces is likely to miss parts of the result R due to limited information in the initial jump source. Using multiple jumps, our steering with intermediate jump sources achieves high clustering quality (as validated in the experiments in Sec. 5 as well). Considering the complexity, the intermediate jumps increase the computational cost by processing a few extra subspace regions. They greatly boost quality as they choose promising jump targets.

As we will show in the following section, our subspace processing based on this jump selection efficiently detects high quality clusters and is scalable to large and high dimensional databases.

5. EXPERIMENTS

We compare our scalable subspace cluster search (SSCS) with recent density-based (INSCY [7] and FIRES [18]) and projected clustering (PROCLUS [1] and MINECLUS [31]) algorithms. As measures for clustering quality we use F1 and accuracy [25]. We use synthetic data for scalability evaluation as in [25], as well as benchmark data from the UCI archive [8]. Besides 16-dim. pendigits data, we use 32 and 48-dim. variants by interpolation of polylines. Furthermore, we use 50-dim. Face and 50-dim. Swedish Leafs data sets [16]. All experiments were run on the same machine with Intel Quad Core Opteron 2.3 GHz CPUs. Furthermore, we ensure comparability and repeatability of our experiments by running all experiments in our open-source framework. Supplementary material for evaluation is provided on our project website[1].

[1]http://dme.rwth-aachen.de/OpenSubspace/SSCS

5.1 Scalability on Synthetic Data

We begin with scalability analysis on synthetic data, increasing the number of dimensions, the most important parameter for performance of subspace clustering algorithms.

Data Dimensionality. As depicted in Fig. 7(a), apriori-based approaches do not scale with increasing dimensionality. For example, the step-by-step method INSCY shows exponential runtimes. As analyzed in [25], subspace clustering approaches such as CLIQUE [3] and SUBCLU [14] show even worse performance than INSCY. However, already INSCY did not finish for the 55-dimensional data within 48 hours. In contrast, approximative solutions like FIRES [18] scale slightly better, but provide only poor clustering quality (cf. Fig. 8). They miss relevant subspace clusters due to their rough approximation of subspace clusters. Without any steering logic, clustering amounts to a heuristic guess. In contrast, our approach steers in a principled fashion directly to subspace clusters, which provide best quality.

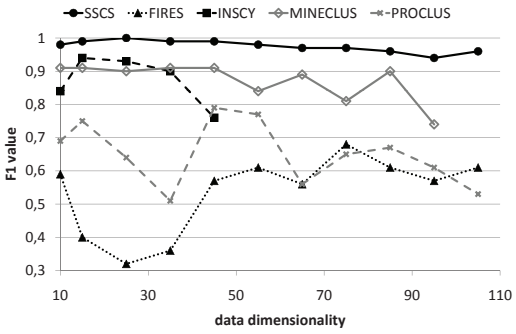


Figure 8: Quality on scalability experiment

As comparison baseline, we depict also projected clustering approaches, which are known to be scalable to high dimensional data [25]. In terms of runtime, approaches like MINECLUS [31] achieve good results for low dimensional data due to their randomization of the search (Fig. 7(a)). However, they show poor scalability. Only the simpler PROCLUS approach scales with almost linear increase of runtime, but shows poorer clustering quality (Fig. 8). Compared to PROCLUS, our approach achieves similar scalability of runtimes while SSCS provides constantly better quality results, not reached by any other subspace clustering algorithm.

Database Size. Figure 7(b) shows runtime of the algorithms with respect to increasing database size. We use

	SSCS			FIRES			INSCY		
	runtime	F1 value	accuracy	runtime	F1 value	accuracy	runtime	F1 value	accuracy
Pendigits16	118	0,63	0,87	150	0,30	0,43	3937	0,51	0,57
Pendigits32	565	0,62	0,86	433	0,23	0,40	X	X	X
Pendigits48	1571	0,60	0,85	6562	0,27	0,50	X	X	X
Face50	982	0,21	0,59	2924	0,17	0,32	X	X	X
Leaf50	934	0,24	0,55	31	0,12	0,32	X	X	X

Table 1: Summary of real world data experiments

20-dimensional databases for comparison with all density-based approaches, as INSCY does not run on most of our higher dimensional databases. For these data we observe best runtimes for the less effective projected clustering approaches as in the previous experiment. Our approach scales well with similar slope in runtime for increasing number of objects. Overall, the results show that subspace clustering is more affected by dimensionality than by database size. Nevertheless, due to our efficient steering to higher dimensional jump targets we achieve far better runtimes than density-based competitors.

Cluster Dimensionality. Figure 7(c) shows the effect of increasing cluster dimensionality. INSCY is not depicted, as for this experiment we have used a high dimensional dataset with 65 dimensions. PROCLUS shows best runtimes due to its simple clustering model. For density-based approaches, our approach clearly outperforms the approximative FIRES, but we also outperform the projected MINECLUS approach that is heavily affected by the increasing cluster dimensionality. Our algorithm scales much better to higher dimensional clusters.

Overall, Figures 7 and 8 show that our approach scales very well in terms of runtime with best clustering quality.

5.2 Evaluation on real world data

In the following experiments, we evaluate real world data. As projected clustering approaches cannot detect overlapping clusters and require the (unknown) number of hidden clusters, PROCLUS and MINECLUS are not comparable (cf. Sec. 2). Table 1 summmarizes runtime (in seconds) and cluster quality via F1 value and accuracy. In addition to the databases Pendigits16, Face50 and Leaf50, we vary the dimensionality of the pendigits data from 16 to 48. For these three datasets we observe similar scalability results as on the synthetic data. INSCY does not scale to more than the 16 dimensional pendigits dataset while the approximative FIRES approach has in most cases higher runtimes than our approach. We have highlighted in gray the lowest runtimes and highest cluster quality for each database. Our approach clearly outperforms existing density-based subspace clustering. Although FIRES achieved lower runtime in two cases, it shows significantly lower quality on all databases. Overall, we efficiently achieve high quality for real world data as well.

5.3 Jump Parameterization

Our multi-jump approach uses gathered information based on low dimensional subspace cluster candidates to determine jumps (cf. Fig. 1). We evaluate the amount of knowledge to extract for multi-jumps. The generated jump targets for each jump ensure a high quality clustering result. We varied the number of generated candidates from 1 to 200 and observe a drop of quality for very low numbers (Fig. 9). However, by generating at least 30 candidates our approach reliably shows high clustering quality. Overall, our steering requires only very few cluster candidates. In contrast to previous approaches, it overcomes the exhaustive generation of cluster candidates by intelligent jumps.

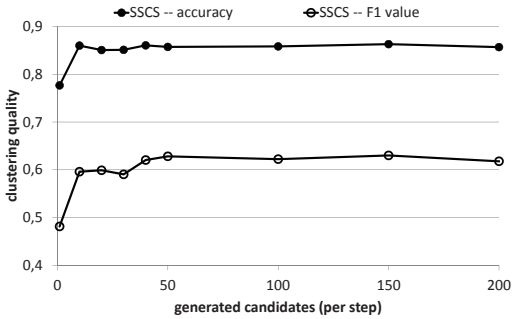


Figure 9: Amount of extracted knowledge

6. CONCLUSION

We introduce a scalable density-based subspace clustering technique. Our main contribution for scalability and high quality is a principled steering of the algorithm in a best-first manner. Information is gathered on promising subspace regions and progressively obtained results are used for multiple jumps that efficiently lead to good subspace clustering results. Our method thereby solves issues faced by apriori-based subspace clustering because of the exponential search space problem and repeated database scans. Overall, our general steering proposes a novel processing scheme for subspace clustering. It ensures scalability w.r.t. increasing number of dimensions, and thus, shows significant runtime improvements compared to the currently used apriori processing scheme.

We propose an instantiation of our steering scheme to the density-based clustering paradigm. It has shown to be a challenging subspace clustering paradigm as it requires quadratic runtime for each of the exponential many subspace projection considered by step-by-step processing schemes. Our steering enables the first density-based subspace clustering technique, which scales to high dimensional data. In our experiments, we demonstrate that our technique is efficient, scales better than existing subspace and projected clustering algorithms, and provides high quality results.

7. FUTURE WORK

For future work, our general steering scheme opens research potential for speeding up other subspace clustering approaches as well. Let us sketch the basic properties for such an instantiation: Steering requires the traditional mono-

tonicity property, as proven for most subspace clustering approaches. This is the only assumption made for the steered jumps in our work. Thus, it is widely applicable with only two hot spots that have to be instantiated in our general processing scheme.

First, information gathering has to be instantiated according to the underlying cluster definition. This can be done similarly to apriori-based techniques. It has to ensure efficient access to required cluster properties without expensive database access, and a sorted list of cluster candidates.

Second, combination of these candidates to higher dimensional clusters is the main idea of steering. As general properties, we proposed cluster preference and cluster similarity functions for this steered jumps. Both should be adapted to the underlying objective function. Given instantiations for these two functions, our steering chooses the best-first order for processing novel subspace clusters.

Acknowledgment

This work has been supported in part by the UMIC Research Centre, RWTH Aachen University, Germany; and by the Danish Council for Strategic Research, grant 10-092316.

8. REFERENCES

[1] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, and J. Park. Fast algorithms for projected clustering. In *SIGMOD*, pages 61–72, 1999.

[2] C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD*, pages 70–81, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, pages 94–105, 1998.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, pages 487–499, 1994.

[5] I. Assent, R. Krieger, E. Müller, and T. Seidl. DUSC: Dimensionality unbiased subspace clustering. In *ICDM*, pages 409–414, 2007.

[6] I. Assent, R. Krieger, E. Müller, and T. Seidl. EDSC: Efficient density-based subspace clustering. In *CIKM*, pages 1093–1102, 2008.

[7] I. Assent, R. Krieger, E. Müller, and T. Seidl. INSCY: Indexing subspace clusters with in-process-removal of redundancy. In *ICDM*, pages 719–724, 2008.

[8] A. Asuncion and D. Newman. UCI machine learning repository: http://www.ics.uci.edu/˜mlearn/mlrepository.html, 2007.

[9] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful. In *IDBT*, pages 217–235, 1999.

[10] C.-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *KDD*, pages 84–93, 1999.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *KDD*, pages 226–231, 1996.

[12] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[13] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000.

[14] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *SDM*, pages 246–257, 2004.

[15] K. Kailing, H.-P. Kriegel, P. Kröger, and S. Wanka. Ranking interesting subspaces for clustering high dimensional data. In *PKDD*, pages 241–252, 2003.

[16] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering homepage, 2006.

[17] H. Kriegel, P. Kroger, M. Schubert, and Z. Zhu. Efficient query processing in arbitrary subspaces using vector approximations. In *SSDBM*, pages 184–190, 2006.

[18] H.-P. Kriegel, P. Kröger, M. Renz, and S. Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *ICDM*, pages 250–257, 2005.

[19] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 3(1), 2009.

[20] X. Lian and L. Chen. Similarity Search in Arbitrary Subspaces Under Lp-Norm. In *ICDE*, pages 317–326. IEEE Computer Society, 2008.

[21] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *KDD*, pages 533–541, 2008.

[22] G. Moise, J. Sander, and M. Ester. P3C: A robust projected clustering algorithm. In *ICDM*, pages 414–425, 2006.

[23] E. Müller, I. Assent, S. Günnemann, R. Krieger, and T. Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *ICDM*, pages 377–386, 2009.

[24] E. Müller, I. Assent, R. Krieger, S. Günnemann, and T. Seidl. DensEst: Density estimation for data mining in high dimensional spaces. In *SDM*, pages 173–184, 2009.

[25] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. *PVLDB*, 2(1):1270–1281, 2009.

[26] H. Nagesh, S. Goil, and A. Choudhary. Adaptive grids for clustering massive data sets. In *SDM*, 2001.

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

[28] C. Procopiuc, M. Jones, P. Agarwal, and T. Murali. A monte carlo algorithm for fast projective clustering. In *SIGMOD*, pages 418–427, 2002.

[29] K. Sequeira and M. Zaki. SCHISM: A new approach for interesting subspace mining. In *ICDM*, pages 186–193, 2004.

[30] D. Witten and R. Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726, 2010.

[31] M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In *ICDM*, pages 689–692, 2003.

[32] F. Zhu, X. Yan, J. Han, P. Yu, and H. Cheng. Mining Colossal Frequent Patterns by Core Pattern Fusion. In *ICDE*, pages 706–715, 2007.