EDSC: Efficient Density-Based Subspace Clustering

Ira Assent * Ralph Krieger Emmanuel Müller Thomas Seidl
Data management and data exploration group
RWTH Aachen University, Germany
{assent,krieger,mueller,seidl}@cs.rwth-aachen.de

ABSTRACT

Subspace clustering mines clusters hidden in subspaces of high-dimensional data sets. Density-based approaches have been shown to successfully mine clusters of arbitrary shape even in the presence of noise in full space clustering. Exhaustive search of all density-based subspace clusters, however, results in infeasible runtimes for large high-dimensional data sets. This is due to the exponential number of possible subspace projections in addition to the high computational cost of density-based clustering.

In this paper, we propose lossless efficient detection of density-based subspace clusters. In our EDSC (efficient density-based subspace clustering) algorithm we reduce the high computational cost of density-based subspace clustering by a complete multistep filter-and-refine algorithm. Our first hypercube filter step avoids exhaustive search of all regions in all subspaces by enclosing potentially density-based clusters in hypercubes. Our second filter step provides additional pruning based on a density monotonicity property. In the final refinement step, the exact unbiased density-based subspace clustering result is detected. As we prove that pruning is lossless in both filter steps, we guarantee completeness of the result.

In thorough experiments on synthetic and real world data sets, we demonstrate substantial efficiency gains. Our lossless EDSC approach outperforms existing density-based subspace clustering algorithms by orders of magnitude.

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

General Terms: Management

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

1. INTRODUCTION

Clustering is a data mining task for summarizing data such that similar objects are grouped together while dissimilar ones are separated. In high-dimensional data, clusters are typically hidden by irrelevant attributes and do not show across the full space. As relevance of attributes is not globally uniform for all clusters, global dimensionality reduction approaches are not adequate.

Subspace clustering detects clusters and their locally relevant attribute projections [3]. The idea is to mine clusters in all subspaces of the high-dimensional data space. As the number of possible subspaces is exponential in the number of dimensions, this is a computationally challenging task.

For efficient subspace clustering, grid-based discretization of the space or other lossy approximations have been proposed [3, 16, 18]. While these algorithms show good runtimes, they lose clusters which are cut apart by the grid or missed by the approximation. Apriori subspace clustering algorithms prune based on a monotonicity assumption of subspace clusters with respect to the dimensionality [3, 10]. However, the incurred dimensionality bias leads to loss of high dimensional subspace clusters [4]. Consequently, approximate, grid-based, and biased algorithms all fail to detect all subspace clusters.

Density-based approaches define clusters as dense areas separated by sparsely populated areas and have been shown to successfully mine arbitrarily shaped clusters even in the presence of noise [7, 10]. Their runtimes, however, are significantly higher due to repeated neighborhood density computations, making them infeasible for many practical applications [10, 11].

In this work, we propose a concept for overcoming the existing trade-off between accuracy and efficiency. We present a density-based subspace clustering algorithm EDSC (efficient density-based subspace clustering) in a multistep filter-and-refine architecture. Substantial efficiency gains are achieved by two novel filter steps. The first step reduces the search space by efficiently mining candidate subspace clusters in enclosing hypercubes. This hypercube filter reduces the number of database scans for density computations and, via our proven hypercube monotonicity, allows for effective and lossless pruning of hypercubes in many irrelevant subspace projections. For the second step, a novel density filter that further prunes subspace cluster candidates without loss of completeness, we prove a monotonicity property for unbiased density approaches. Both filter steps significantly reduce the number of subspace cluster candidates without loss of results. The final refinement step ensures that the exact density-based subspace clustering result is found, yet with a substantially lower runtime. Our multistep approach

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

* now with Department of Computer Science, Aalborg University, Denmark

thus efficiently detects density-based subspace clusters even in large high-dimensional data spaces.

Summarizing, EDSC shows two core characteristics:

- **Accuracy**: density-based subspace clustering with guaranteed lossless pruning
- **Efficiency**: efficient subspace clustering via a novel multistep filter-and-refine algorithm

2. RELATED WORK

In the literature, several different clustering paradigms exist. Partitioning algorithms are limited to the detection of convex clusters where the number of clusters is known in advance [14, 13]. Density-based algorithms such as DBSCAN are capable of detecting arbitrarily shaped clusters and have proven to work remarkably well in noisy settings [7, 8]. For any clustering paradigm, full space clustering algorithms do not scale to high-dimensional spaces. They suffer from the "curse of dimensionality", i.e. distances grow increasingly similar with increasing dimensionality and meaningful clusters can no longer be detected [5].

Dimensionality reduction techniques like PCA (principle components analysis) aim at discarding irrelevant dimensions [9]. In many practical applications, however, no globally irrelevant dimensions exist, but only locally irrelevant dimensions for each cluster are observed. Consequently, research has focused on clustering in subspace projections. Projected clustering computes a partition into projected clusterings [2, 1, 15]. However, overlapping clusters in different projections cannot be detected. ENCLUS, RIS and FIRES search for subspaces which might potentially contain clusters [6, 11, 12]. Clusters are mined in a second step using any traditional full space clustering algorithm. As there is no connection between these two steps, they suffer from costly repeated cluster computations in unnecessary projections and loss of subspace clusters in other projections.

Subspace clustering mines clusters directly in arbitrary, possibly overlapping, subspaces. As the number of subspaces is exponential in the number of dimensions, existing algorithms trade-off efficiency for accuracy. CLIQUE uses a grid to discretize the search space [3]. Grids greatly reduce the computational complexity, yet clusters which spread across cells are missed and results are sensitive to the position of the grid. SCHISM extends CLIQUE using a variable threshold adapted to the dimensionality of subspaces [18]. Heuristics and a grid-based discretization are used for pruning. Consequently, clusters are lost as well as in any existing grid-based discretization.

SUBCLU extends DBSCAN to subspace clustering to overcome grid-based cluster loss [10]. Runtimes are better than for naive re-runs of DBSCAN, yet still not feasible for practical applications. Moreover, the assumption of density monotonicity still results in losing of subspace clusters in higher dimensions. DUSC overcomes this dimensionality bias by adapting density assessment to the dimensionality of the subspace [4]. Consequently, with its unbiased density model it is capable of detecting arbitrarily shaped subspace clusters of any dimensionality. As density monotonicity does not hold for unbiased density, it requires novel techniques for efficient computation.

3. DENSITY-BASED SUBSPACE CLUSTERS

For notational convenience, we assume the following terminology: we mine a database $\mathbf{DB}$ with N objects $O = (o_1 \dots o_d)$ in d attributes $\mathbf{A_i} \in [0, \mathbf{v}]$. A $|\mathbf{S}|$-dimensional subspace is denoted by $\mathbf{S} = \{k_1, ..., k_{|\mathbf{S}|}\} \subseteq \mathbf{D} = \{1, ..., d\}$. A projection of O to $\mathbf{S}$ is $O^{\mathbf{S}} = (o_{k_1}, ..., o_{k_{|\mathbf{S}|}})$. The distance between two objects O and P in subspace $\mathbf{S}$ is calculated by restricting the calculation to the dimensions of the subspace: $dist^{\mathbf{S}}(O, P) = \sqrt{\sum_{i \in \mathbf{S}} (o_i - p_i)^2}$. Our notations are summarized in Table 1.

N	number of data objects
$\mathbf{D} = \{1, ..., d\}$	dimensions of full space
$\mathbf{S} = \{k_1, ..., k_{\|\mathbf{S}\|}\}$	subspace with $\|\mathbf{S}\|$ dimensions
$\mathbf{A_i} \in [0, \mathbf{v}]$	dimension with value range $0 \dots \mathbf{v}$
$O^{\mathbf{S}} = (o_{k_1}, ..., o_{k_{\|\mathbf{S}\|}})$	object in subspace S
N_ε^S	neighborhood with radius ε
$minPoints$	minimum number of objects
$expDen(\mathbf{s})$	expected density for dimensionality $\mathbf{s}$
F	density factor threshold

Table 1: Notations

In density-based clustering, *clusters* are defined as dense areas separated by sparsely populated areas [7]. Density is evaluated for every object. Objects are dense *core objects* if their neighborhood, specified by an ε-range around the object, contains more than the threshold $minPoints$ many objects. Chains of objects which are *density-connected* form the actual clusters. This is illustrated in Figure 1: e.g. the objects to the right form a cluster C_4, as each of them contains many objects within its neighborhood (depicted as a circle centered at one of the objects). Dense objects are connected to clusters to detect arbitrarily shaped clusters even in noisy settings:

DEFINITION 1. ***Density and Connectivity.***

An object O is ***dense*** *if at least $minPoints$ objects are in its ε-neighborhood $N_\varepsilon(O) = \{P \in \mathbf{DB} \mid dist(O, P) \leq \varepsilon\}$:*

$$O \text{ dense : } \Leftrightarrow |N_\varepsilon(O)| \geq minPoints$$

Two dense objects O and P are ***density-connected*** *if there is a chain of dense neighboring objects between them:*

$$\exists\, Q_1 \dots Q_n \in \mathbf{DB} : Q_1 = P, Q_n = O, \forall\, i : \ dense(Q_i) \wedge$$

$$\wedge\ \forall\ i = 1 \dots n-1 : dist(Q_i, Q_{i+1}) \leq \varepsilon.$$

Density-connected objects thus are elements of chains of objects which are mutually included in one another's neighborhoods. SUBCLU extends this model to subspace clustering in a straightforward manner. In each subspace, clusters must exceed the $minPoints$ threshold within the ε neighborhood. In the SCHISM, RIS or DUSC approaches [11, 18, 4], the ensuing dimensionality bias is discussed: with increasing dimensionality, distances grow and typical densities drop. Figure 1 illustrates this effect: the 2d representation of the data shows a larger spread than the 1d projections at the left and bottom. In general in high dimensional subspaces, the expected density is smaller than in low dimensional spaces. Consequently, large thresholds are almost never exceeded in high dimensional spaces, resulting in cluster loss. On the other hand, small thresholds that find high dimensional clusters produce tremendous amounts of trivial low dimensional subspace clusters. Thus, density should be normalized by the expected density of the dimensionality [4]. The expected density is simply the average number of objects in

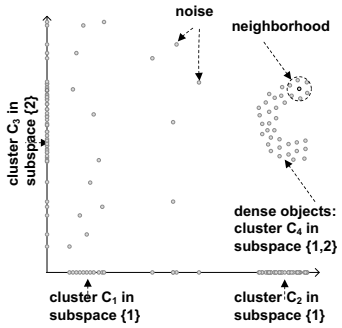


Figure 1: Density based subspace clustering

the ε-neighborhood w.r.t. the dimensionality. This can be computed as the ratio of the volume of the ε-sphere to the volume of the subspace. For details please refer to [4].

DEFINITION 2. ***Unbiased Density.***

*An object O is **dense** in subspace $\mathbf{S}$ of dimensionality $\mathbf{s}$ if its ε-neighborhood in $\mathbf{S}$, $N_\varepsilon^{\mathbf{S}}(O) = \{P \in \mathbf{DB} \mid dist^{\mathbf{S}}(O,P) \leq \varepsilon\}$, contains more than minPoints objects and exceeds the expected density in this subspace by a factor F:*

$$O \text{ dense in } \mathbf{S} :\Leftrightarrow |N_\varepsilon^{\mathbf{S}}(O)| \geq max\{minPoints, F \cdot expDen(\mathbf{s})\}$$

Density in subspaces is thus normalized with respect to the dimensionality simply by adapting the threshold to the expected density.

The density measure may be further refined to assign more less weight to objects further away from the center of the neighborhood range. For a monotonously decreasing weighting function $W : \mathbb{R} \to \mathbb{R}$, $density(O) = \sum_{P \in N_\varepsilon^{\mathbf{S}}((O)} W\left(dist^{\mathbf{S}}(O,P)\right)$ weights the objects within the neighborhood according to their distance from the object O. DUSC uses Epanechnikov kernel, whereas the SUBCLU approach corresponds to the Rectangle kernel [4, 10]. Our algorithm works for any of these weighting functions.

Density-based subspace clusters are defined as maximal sets of objects that are density-connected via mutual inclusion within one another's neighborhood in the subspace projection. Moreover, they should be non-redundant, i.e. if the cluster appears in some dimensionality, we exclude its lower dimensional projections (e.g. in Figure 1, keep C_1 and C_4, but not C_2 and C_3, which are covered by C_4) [4]:

DEFINITION 3. ***Subspace Cluster.***

A set of objects $C \subseteq \mathbf{DB}$ in subspace $\mathbf{S} \subseteq \mathbf{D}$ with $|S| > 1$ and $|C| > minSize$ is a subspace cluster if:

- *C is **density-connected**: $\forall O, P \in C$: $O^{\mathbf{S}}, P^{\mathbf{S}}$ density-connected w.r.t unbiased density (Def 2).*
- *C is **maximal**: $\forall O, P \in \mathbf{DB}$: if $O \in C$ and $O^{\mathbf{S}}, P^{\mathbf{S}}$ density-connected then $P \in C$.*
- *C is **non-redundant**: there is no higher-dimensional cluster containing points in C.*

Thus, subspace clusters are sets of density-connected objects that are maximal, i.e. all density-connected objects are grouped together. They should be non-redundant, i.e. a subspace cluster that is repeated in a lower-dimensional projection is not considered to be novel information. Additionally, we exclude trivial one-dimensional clusters.

4. EFFICIENT MULTISTEP APPROACH

Detecting all subspace clusters in any subspace is a highly complex task. The number of possible subspaces is exponential in the number of attributes. Naively searching all of these subspaces is hence not feasible in most applications and pruning the search space is crucial. In this section, we propose our efficient and lossless density-based subspace clustering algorithm EDSC (efficient density-based subspace clustering).

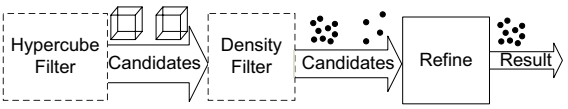


Figure 2: Multistep EDSC algorithm

4.1 Multistep architecture

We propose a multistep filter-and-refine architecture as illustrated in Figure 2, for efficient subspace clustering. Instead of running a complex subspace clustering algorithm on the entire database, we restrict the search space in a chain of filter steps. Each filter step reduces the search space to a successively smaller set of candidates, i.e. sets of objects that are potential density-based subspace clusters. EDSC uses two novel filters for both complete and selective pruning to ensure that the overall algorithm is accurate and efficient. Completeness means that the filters do not produce any false dismissals, i.e. we guarantee that all subspace clusters are included in the candidate sets, and that the result set of the EDSC algorithm contains all subspace clusters. Selective pruning means that the algorithm achieves a large reduction in the search space, i.e. many irrelevant sets of objects and their higher-dimensional projections can be excluded from further consideration for substantial speed-up. The combination of both filters in EDSC is illustrated in Figure 3: from the fact that a candidate fails one of the two filter tests, we infer that it is not a subspace cluster in any higher dimensional subspace.

The refinement step additionally ensures that there are also no false positives, i.e. only those candidates are actually reported as results that constitute subspace clusters.

Hypercube filter. Existing density-based subspace clustering algorithms suffer from the fact that for each subspace and for each object, density has to be computed. This requires repeated neighborhood scans of the database. Our first filter step thus determines hypercubes that surround each potential subspace cluster, thus reducing the neighborhood evaluation to significantly fewer objects. The algorithm then processes only these candidate in higher-dimensional projections and thus can efficiently reduce the exponential search space.

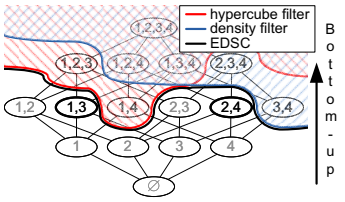


Figure 3: Pruning the search space

Density filter. In the second filter, we build on the hypercube restriction from the first step. The basic idea is to determine for any candidate hypercube not only if it potentially contains a density-based subspace cluster but also if any of its higher-dimensional projections may contain one. For unbiased density (cf. Def. 2), monotonicity does not hold and thus cannot be used for lossless pruning (see Section 4.3 for details). We determine a new monotonicity property that allows for lossless pruning even with respect to this unbiased density criterion. This new property, which determines a "minimum" density over all subspaces, ensures that we may safely prune a candidate hypercube and all its higher-dimensional projections, without loss of completeness. Our *weak density* filtering precedes the final refinement step.

Refinement. Subspace clusters which fulfill the "minimum" density criterion (pass the *weak density* filter) are now iteratively processed in higher-dimensional subspaces to finally determine the exact result according to the unbiased variable density definition. The refinement step thus removes any remaining false alarms as it uses the stricter unbiased density.

4.2 Hypercube filter

To avoid repeated scans of the entire database to determine the neighborhood of objects, we enclose potential subspace cluster regions in hypercubes.

4.2.1 Building hypercubes

With the novel density-conserving grid we guarantee to enclose cluster regions and thus achieve a lossless and efficient pruning. The density-conserving grid is devised to meet exactly the definition of density-based subspace clusters. Defined as sets of density-connected objects w.r.t. their ε-neighborhoods, subspace clusters might spread across multiple grid cells. To detect these clusters, we introduce novel connectivity borders of ε width. Subspace clusters which are density-connected across cells can be detected along the connectivity border between cells. In the algorithm, these cells are merged until completely enclosing hypercubes for each subspace cluster are found.

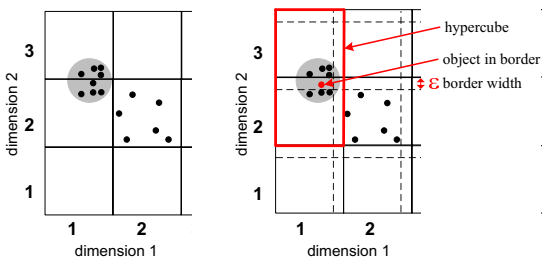


Figure 4: Traditional vs. density-conserving grid

Figure 4 illustrates the idea: on the left side, we show a traditional grid structure. Traditional grids loose clusters that spread across several cells. For example, by simply comparing the count of objects in each cell with an exemplary threshold of 5 objects, the dark shaded cluster at the left hand side would be missed, and only the objects in cell $(2, 2)$ would be reported as a cluster. Our novel density-conserving grid, illustrated at the right side, has additional borders of the size of the neighborhood, ε. Checking these borders, our EDSC algorithm detects all potential spreads of a cluster from one cell to another and merges the corresponding cells to build an enclosing hypercube. Thus, the dark shaded cluster is successfully detected.

Formally, our density-conserving grid is a partitioning of the data space into regular grid cells, plus novel connectivity borders that are exactly the size of ε-neighborhoods that are used in the definition of density-connected objects (Def. 3). Each cell is identified by the indices of the cell intervals in each dimension. Borders are additionally marked for later processing in our algorithm:

DEFINITION 4. ***Density-conserving grid.***

A ***density-conserving grid*** *is a regular grid with connectivity borders:*

- *A* ***regular grid*** *is a partition of the attribute range* $\mathbf{v}$ *into* $g = \lceil \frac{\mathbf{v}}{w} \rceil$ *intervals* p_i *of equal width* w *and* $p_i = [w \cdot (i-1), w \cdot i),\ i = 1 \ldots g$.
- ***Connectivity-borders*** *are intervals of* ε*-width at the upper border of each cell in each dimension* $b_i = [w \cdot i - \varepsilon, w \cdot i),\ i = 1, \ldots, g$
- *A* s*-dimensional* ***subspace cell*** $\mathbf{C}_{\alpha_1,\ldots,\alpha_d}$ *is given by an index vector* $\alpha_1, \ldots, \alpha_d$ *of the corresponding intervals* $p_{\alpha_j},\ \alpha_j \in \{1, \ldots, g, *\}$, *where stars denote the unconstrained dimensions of the cell.*
- *A border* $\mathbf{B}_{\alpha_1,\ldots,\overline{\alpha_k},\ldots,\alpha_d}$ *in dimension* k *is given by the index vector* $\alpha_1, \ldots, \overline{\alpha_k}, \ldots, \alpha_d$ *of its cell* $\mathbf{C}_{\alpha_1,\ldots,\alpha_d}$, *where the dash denotes the border dimension* k.

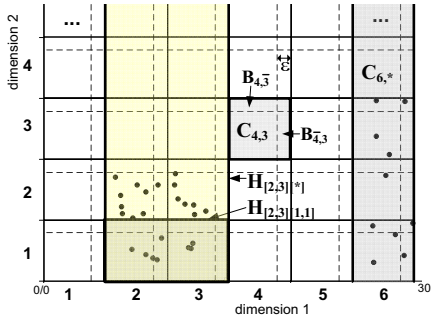


Figure 5: Density-conserving grid and hypercubes

In Figure 5 we illustrate our approach by an example of a two-dimensional space where each attribute range $\mathbf{v}$ is 0 to 30, and the grid resolution g is set to 6. For example, cell $\mathbf{C_{4,3}}$ contains two connectivity borders $\mathbf{B}_{\overline{4},\mathbf{3}}$ and $\mathbf{B}_{\mathbf{4},\overline{3}}$, one in each dimension. In the example, $\mathbf{C_{4,3}}$ is a 2-dimensional cell which is restricted in interval 4 for dimension 1 and in interval 3 for dimension 2. $\mathbf{C}_{\mathbf{6},*}$ is a 1-dimensional cell restricted in interval 6 for dimension 1 and not constrained in dimension 2. In general $(d - |\mathbf{S}|)$ stars denotes the unconstrained dimensions of the cell. Hypercubes consist of merged cells, given by interval ranges per dimension. $\mathbf{H}_{[\mathbf{2,3}][\mathbf{1,1}]}$ is a 2-dimensional example hypercube. $|\mathbf{H}_{[a_1,b_1]\ldots[a_d,b_d]}|$ denotes the number of objects in a hypercube. For example, $\mathbf{H}_{[\mathbf{2,3}][*]}$ contains 24 objects (all objects contained in the 2nd and 3rd interval), while its constrained projection to interval one in dimension one $|\mathbf{H}_{[\mathbf{2,3}][\mathbf{1,1}]}|$ contains 8 objects.

This density-conserving grid is now used to efficiently construct hypercubes that enclose potential density-based subspace clusters. Thus, we avoid repeated clustering on individual objects in each subspace.

4.2.2 Filtering using hypercubes

The hypercube filter on this density-conserving grid works as follows: all grid cells are accessed exactly once, by processing them in lexicographic order on their interval indices (e.g. in Figure 5, $\mathbf{C_{1,1}}, \mathbf{C_{1,2}}, \ldots, \mathbf{C_{6,5}}, \mathbf{C_{6,6}}$). For each non-empty cell, its borders are checked to see whether a subspace cluster might spread across the cell borders. As the size of the border is exactly the size of the neighborhood in the density-connectivity definition, any such spread is guaranteed to be detected by checking all borders of the cell. Only objects in the borders could connect a cluster from one cell to the next, otherwise the density-connectivity property is not fulfilled. If the borders are empty, the hypercube is maximal and encloses the entire potential subspace cluster. We denote it as MICH (*maximal induced cluster hypercube*) and continue with the next filter step. If the borders of the cell are not empty, we repeatedly merge hypercubes (details in Section 4.4.2) and check borders until the MICH is maximal. As our hypercube filter checks all cells and all connectivity borders between them, EDSC is guaranteed to find all MICHs, i.e. all potential subspace clusters.

THEOREM 1. ***Completeness of hypercube filter***

Each subspace cluster C is enclosed by a MICH.

PROOF.
By Definition 3, C is a set of maximal density-connected objects $Q_1, \ldots, Q_n$.
Case 1: All density-connected objects are within one cell. Thus an index vector $\alpha_1, \ldots, \alpha_d$ exists such that all objects are in the corresponding grid cell $\forall i \in \{1, \ldots, n\} : Q_i \in \mathbf{C}_{\alpha_1,\ldots,\alpha_\mathbf{d}}$. The cell $\mathbf{C}_{\alpha_1,\ldots,\alpha_\mathbf{d}}$ forms a MICH and encloses all objects in C.
Case 2: All density-connected objects are within two neighboring cells. Thus there are two directly density-connected objects Q_i and Q_{i+1} positioned in two neighboring grid cells. Furthermore, at least one dimension l exists, in which these cells differ, w.l.o.g. $\beta_l = \alpha_l + 1$. In our grid cell notation, we have:

Q_i	$\in$	$\mathbf{C}_{\alpha_1,\ldots,\alpha_\mathbf{d}}$	left cell
Q_{i+1}	$\in$	$\mathbf{C}_{\beta_1,\ldots,\beta_\mathbf{d}}$	right cell
p_{α_l}	$=$	$[w \cdot (\alpha_l - 1), w \cdot \alpha_l)$	left interval
b_{α_l}	$=$	$[w \cdot \alpha_l - \varepsilon, w \cdot \alpha_l)$	border interval
p_{β_l}	$=$	$[w \cdot \alpha_l, w \cdot (\alpha_l + 1))$	right interval

The necessary condition for merging in EDSC is that there is an object in the connectivity-border.
Therefore, we proof $Q_i \in \mathbf{B}_{\alpha_1,\ldots,\overline{\alpha_l},\ldots,\alpha_\mathbf{d}}$ by contradiction: Assume $Q_i \notin \mathbf{B}_{\alpha_1,\ldots,\overline{\alpha_l},\ldots,\alpha_\mathbf{d}}$ and hence the distance of Q_i to the upper limit of the grid cell intervall p_{α_l} is greater than ε. Hence the overall distance of Q_i to any object in the interval p_{α_l+1} is also larger than ε. Formally, $dist(Q_i, Q_{i+1}) \geq dist(Q_i^{\{l\}}, w \cdot \alpha_l) > \varepsilon \Rightarrow Q_i \notin N_\varepsilon(Q_{i+1})$. This is a contradiction to the density-connectedness of Q_i and Q_{i+1}.

As Q_i is in the connectivity-border $\mathbf{B}_{\alpha_1,\ldots,\overline{\alpha_l},\ldots,\alpha_\mathbf{d}}$ of cell $\mathbf{C}_{\alpha_1,\ldots,\alpha_\mathbf{d}}$, the merge is detected when processing $\mathbf{C}_{\alpha_1,\ldots,\alpha_\mathbf{d}}$. EDSC will therefore merge the cells $\mathbf{C}_{\alpha_1,\ldots,\alpha_\mathbf{d}}$ and $\mathbf{C}_{\beta_1,\ldots,\beta_\mathbf{d}}$ to an enclosing MICH.
Case 3: For density-connected objects within more than two neighboring cells, the argument of Case 2 for directly connected neighboring Q_i, Q_{i+1} can be inductively extended to all objects in the density-connected set. Therefore, the entire density-connected subspace cluster $Q_1, \ldots, Q_n$ is detected through border checks, and the enclosing MICH M_C is built. □

For good runtime performance it is important to prune hypercubes that cannot contain subspace clusters. We propose pruning those hypercubes that do not fulfill minimum size requirements for density-based subspace clusters. As we search for clusters with at least $minSize$ objects, monotonicity with respect to the number of objects allows for safe pruning. We exploit monotonicity on the number of objects in hypercubes of different subspaces. Monotonicity means that as we restrict a hypercube in more dimensions the number of objects cannot increase. Therefore, a hypercube which does not contain enough objects in a subspace $\mathbf{S}$ does not contain enough objects in any higher-dimensional subspace $\mathbf{T}$ ($\mathbf{S} \subseteq \mathbf{T} \subseteq \mathbf{D}$), either. Thus, we may safely prune this hypercube from further consideration without losing any subspace cluster.

THEOREM 2. ***Hypercube monotonicity.***

For any two subspaces with $\mathbf{S} \subseteq \mathbf{T}$, and any hypercube $\mathbf{H}$ identified by interval indices $[a_1, b_1) \ldots [a_d, b_d)$ the number of objects in $\mathbf{T}$ is bound by the number of objects in $\mathbf{S}$:

$$|\mathbf{H}^{\mathbf{S}}_{[a_1,b_1)\ldots[a_d,b_d)}| \geq |\mathbf{H}^{\mathbf{T}}_{[a_1,b_1)\ldots[a_d,b_d)}|$$

PROOF.
$|\mathbf{H}^{\mathbf{S}}_{[a_1,b_1)\ldots[a_d,b_d)}|$
$= |\{(o_1, \ldots, o_d) \in \mathbf{DB} \mid \forall i \in \mathbf{S} : o_i \in [w \cdot a_i, w \cdot b_i)\}|$
$\geq |\{(o_1, \ldots, o_d) \in \mathbf{DB} \mid \forall i \in \mathbf{T} : o_i \in [w \cdot a_i, w \cdot b_i)\}|$
$= |\mathbf{H}^{\mathbf{T}}_{[a_1,b_1)\ldots[a_d,b_d)}|$
As $\mathbf{S} \subseteq \mathbf{T}$ holds, their are more constraints on objects in subspace $\mathbf{T}$. And therefore in the hypercube $\mathbf{H}^{\mathbf{T}}_{[a_1,b_1)\ldots[a_d,b_d)}$ there are at most as many objects as in $\mathbf{H}^{\mathbf{S}}_{[a_1,b_1)\ldots[a_d,b_d)}$. □

As we have proven in Theorem 1, each density-based subspace cluster is enclosed by a MICH. Thus our EDSC algorithm may safely prune sparse hypercubes and all its higher-dimensional projections. Any MICH $\mathbf{H}_{[a_1,b_1)\ldots[a_d,b_d)}$ is checked in subspace $\mathbf{S}$ to see if $|\mathbf{H}^{\mathbf{S}}_{[a_1,b_1)\ldots[a_d,b_d)}| \geq minSize$. If this is the case, we continue with this candidate. Else, we have detected a spare region and we know for sure that we can not only discard $\mathbf{H}^{\mathbf{S}}_{[a_1,b_1)\ldots[a_d,b_d)}$, but also all higher-dimensional projections $\mathbf{H}^{\mathbf{T}}_{[a_1,b_1)\ldots[a_d,b_d)}$ with $\mathbf{S} \subseteq \mathbf{T}$, since for them $|\mathbf{H}^{\mathbf{T}}_{[a_1,b_1)\ldots[a_d,b_d)}| \geq minSize$ cannot hold either.

4.3 Density filter

MICHs are conservative approximations of potential subspace clusters, i.e. all subspace clusters are enclosed by one MICH, but not all MICHs contain subspace clusters. To efficiently detect those sets of objects that are not dense and thus do not form subspace clusters, we devise our density filter. It prunes those subspace clusters that do not fulfill minimum density in the current and also all higher-dimensional subspaces. As we prove completeness also for our second filter, overall lossless pruning is ensured.

4.3.1 Density monotonicity

In fixed threshold subspace clustering (more objects in the neighborhood than $minPoints$), density is monotonous with

respect to the number of dimensions which is used for pruning as in SUBCLU [10]. Those sets of objects that are not dense in a certain subspace, may not be dense in any higher-dimensional subspace, either, and are consequently pruned. In principle, we could use the same pruning idea. However, as shown in our previous work, this fixed threshold leads to dimensionality bias, as the expected density decreases rapidly with increasing dimensionality [4]. Thus, high dimensional subspace clusters are almost never detected. We therefore use unbiased density in Definition 2 (density larger than F times the expected density). While this definition allows for detection of clusters in arbitrary subspaces, it is no longer monotonous, as the expected density decreases with growing dimensionality. For completeness, we therefore propose using a weaker criterion for pruning based on monotonicity of the number of objects in the neighborhood with increasing dimensionality.

THEOREM 3. ***Density monotonicity.***

For any two subspaces with $\mathbf{S} \subseteq \mathbf{T}$*, and any object* O *the number of objects in the neighborhood in subspace* $\mathbf{T}$ *is bound by the number of objects in subspace* $\mathbf{S}$*:*

$$|N_\varepsilon^{\mathbf{S}}(O)| \geq |N_\varepsilon^{\mathbf{T}}(O)|$$

PROOF.
The proof is straightforward from the definition of the neighborhood. It uses the fact that with more dimensions in $\mathbf{T}$, the distances of objects are larger than in $\mathbf{S}$:

$$\begin{aligned} dist^{\mathbf{S}}(O,P) &= \sqrt{\textstyle\sum_{i\in\mathbf{S}}(o_i-p_i)^2} \\ &\leq \sqrt{\textstyle\sum_{i\in\mathbf{T}}(o_i-p_i)^2} = dist^{\mathbf{T}}(O,P) \\ |N_\varepsilon^{\mathbf{S}}(O)| &= |\{P\in\mathbf{DB} \mid dist^{\mathbf{S}}(O,P)\leq\varepsilon\}| \geq \\ &\quad |\{P\in\mathbf{DB} \mid dist^{\mathbf{T}}(O,P)\leq\varepsilon\}| = |N_\varepsilon^{\mathbf{T}}(O)| \end{aligned}$$

□

Pruning with this criterion for unbiased density measures (cf. Def. 2) would violate completeness. The expected density decreases with increasing dimensionality. If we were to discard a set of objects that does not exceed the $F \cdot expDen(\mathbf{s})$ threshold for a dimensionality $\mathbf{s}$, it can still exceed the lower threshold $F \cdot expDen(\mathbf{s}')$ for $\mathbf{s}' \geq \mathbf{s}$. Pruning based on $\mathbf{s}$ alone would therefore lose clusters. To allow for lossless pruning even for algorithms with unbiased density measurements, our new "weak density" criterion is based on the lowest possible expected density in the highest dimensionality d.

DEFINITION 5. ***Weak Density.***

An object $O \in \mathbf{DB}$ *is weak dense in* $\mathbf{S}$ *if:*

$$|N_\varepsilon^{\mathbf{S}}(O)| \geq max\{minPoints, F \cdot expDen(d)\}$$

Thus, density in the subspace with the lowest expected density leads to a complete filter:

THEOREM 4. ***Completeness of density filter***

For all subspaces $\mathbf{T}$ *with* $\mathbf{S} \subseteq \mathbf{T} \subseteq \mathbf{D}$ *and* $|\mathbf{T}| = t$ *holds:*

If an object is not ***weak dense*** *in* $\mathbf{S}$
it is not ***dense*** *in* $\mathbf{T}$.

PROOF.

$$|N_\varepsilon^{\mathbf{S}}(O)| < max\{minPoints, F \cdot expDen(d)\} \quad (1)$$
$$\Rightarrow |N_\varepsilon^{\mathbf{T}}(O)| < max\{minPoints, F \cdot expDen(d)\} \quad (2)$$
$$\Rightarrow |N_\varepsilon^{\mathbf{T}}(O)| < max\{minPoints, F \cdot expDen(t)\} \quad (3)$$

Using monotonicity property (Theorem 3) the left side of inequality $(1 \rightarrow 2)$ is monotonically decreasing with increasing dimensionality. As the expected density ($expDen$) is calculated as the ratio of the volume of the ε-sphere to the volume of the subspace, $expDen$ is monotonously decreasing with increasing dimensionality. Hence, the subspace of the highest dimensionality d has the lowest density: $expDen(d) \leq expDen(t)$. Thus the maximum on the right side of inequality (2) is less than or equal to the maximum in inequality (3). □

Following the weak density theorem, an object can be pruned if it violates the weak density condition, as it is not dense in any higher-dimensional subspace. Hence we check any candidate that passed the hypercube filter according to the weak density criterion in the density filter. If it is no subspace cluster with respect to both $minPoints$ and $F \cdot expDen(d)$, we may safely discard all its objects, as we know for sure that they do not constitute a subspace cluster with respect to unbiased density $F \cdot expDen(t)$ and $minPoints$ in any dimensionality t.

This concludes the discussion of the two filters in our multistep EDSC approach and the completeness proof. In the next section, we discuss the overall algorithm as illustrated in Figure 2 and provide more detailed information on efficient handling and merging of hypercubes.

4.4 The EDSC algorithm

Figure 6 outlines the complete algorithm that consist of hypercube filter, density filter and refinement (Fig. 2).

4.4.1 Algorithm overview

Recall that each MICH encloses a potential density-based cluster (a subspace cluster candidate). Our algorithm processes each such MICH in all subspace projections. Thus, the **hypercube filter** (Line 1-12) recursively calls the other filter steps of our multistep algorithm for each MICH. Using the hypercube monotonicity property in Theorem 2 a MICH and all its higher-dimensional subspace projections can be pruned if the MICH does not contain at least $minSize$ objects (Line 14). If a MICH is a cluster candidate, it is subsequently analyzed by the **density filter** of the EDSC algorithm (Line 15) based on the weak density in Theorem 3. If the hypercube does not contain any weak density connected subspace cluster the corresponding hypercube and all its higher-dimensional hypercube projections can be pruned. Additionally, redundant clusters are discarded according to the last part of the subspace cluster definition in Def. 3 (Line 17). The **refinement step** removes any false alarms: the method *ExpDensityScan* performs the actual subspace clustering w.r.t. the expected density of the subspaces.

4.4.2 Efficient hypercube computation

In this section, we give algorithmic details and an illustrative example on how to efficiently create enclosing hypercubes by merging of grid cells.

The algorithm starts on cells that are merged until an enclosing hypercube for each subspace cluster is found. For efficiency reasons we process cells in lexicographical order to ensure that each cell has to be processed only once. We mark future merges w.r.t lexicographic order as *induced merges*; when these marked cells are actually processed, *performed merges* combine the cells. Lexicographically greater cells are processed in the future, hence we denote them as *future*

```
1  for k = d_first ... d do                                   /* for each dimension and */
2      for i = 1 ... g do                                     /* each cell in each dimension */
3          C = restrictCell(C, [k, i]);                       /* restrict C in dimension k */
4          if restrictions(C) < 2 then                        /* no testing for one-dimensional cells */
5              EDSC(C, k + 1) ;                               /* continue EDSC recursion */
6          else
7              if restrictions(C) == 2 then
8                  I = testBorder(C) ;                        /* test all borders */
9              else if restrictions(C) > 2 then
10                 I = testNewBorder([k, i]);                 /* test only new border */
11             induceMerge(C, I) ;                            /* for future neighbors of C */
12             C = performMerges(C) ;                         /* for past neighbors of C */
13             if induceCounter(C)==0 then                    /* MICH complete */
14                 if objectCounter(C) > minSize then         /* 1st filter step */
15                     if WeakDensityScan(C) then             /* 2nd filter step */
16                         EDSC(C,k+1);                       /* further restriction */
17                         if isNonRedundant(C) then
18                             ExpDensityScan(C)              /* refinement step */
19                         else
20                             pruneCluster(C) ;              /* prune C */
21                     else
22                         prune(C) ;    /* prune C and all of its higher dim.  projections */
23                 else
24                     prune(C) ;        /* prune C and all of its higher dim.  projections */
```

Figure 6: EDSC algorithm: EDSC(C, d_{first})

cells, and lexicographically smaller cells as *past cells*. An enclosing hypercube is found if no more induced merges and performed merges are necessary from the current cell.

If a density-connected subspace cluster stretches from one cell into another, the connectivity border contains at least one object. Otherwise, as we set the border width exactly to the neighborhood range ε, objects in one cell cannot be in the ε-neighborhood of the other (see Definition 1). When a cell is processed, the connectivity borders of the cell are checked. For each border which contains an object a merge is induced into the corresponding adjacent cell. If both connectivity borders contain an object a cluster might also extend into the diagonal cell, i.e. adjacent to the intersection of both borders, ensuring an induced merge to this diagonal cell.

When processing a cell, first merges into future cells are induced and then merges with past cells are performed. A merge is always performed with a past cell which induced a merge into the cell. During a merge process the number of objects (the *objectCounter*) and the number of merges induced into future cells (the *induceCounter*) are aggregated. We use the concept of induced merges to indicate whether an enclosing hypercube for a density-connected region is found: if all induced merges of a hypercube are performed ($induceCounter = 0$) a MICH is found.

Checking connectivity borders guarantees that the grid does not cut density-based clusters apart and that each subspace cluster is enclosed by a MICH (see Theorem 1).

4.4.3 Merge example

Figure 7 illustrates the merge steps performed to identify a MICH enclosing the example cluster. We start by processing each cell of dimensionality two, beginning with cell $\mathbf{C_{1,1}}$. The number of objects is determined, and if the cell is not empty, its connectivity borders are tested. Cell $\mathbf{C_{1,1}}$ is empty and hence no merge is induced. Next, cell $\mathbf{C_{2,1}}$ is processed. As both connectivity borders contain an object, cell $\mathbf{C_{2,1}}$ induces a merge into $\mathbf{C_{3,1}}$, $\mathbf{C_{2,2}}$ and into $\mathbf{C_{3,2}}$. Next, cell $\mathbf{C_{3,1}}$ performs the merge with $\mathbf{C_{2,1}}$, creating $\mathbf{H_{[2,3][1,1]}}$. When $\mathbf{C_{2,2}}$ is processed, it is merged with $\mathbf{C_{1,2}}$ to $\mathbf{H_{[1,2][2,2]}}$. After this, the next merge induced into $\mathbf{C_{2,2}}$ is performed (second part of Figure 7). This merge step creates the hypercube $\mathbf{H_{[1,3][1,2]}}$. Finally, cell $\mathbf{C_{3,2}}$ is processed. As $\mathbf{C_{3,2}}$ was already merged with both cells, only the *induceCounter* is decremented. After this step, no more merges are necessary and $\mathbf{H_{[1,3][1,2]}}$ corresponds to a MICH.

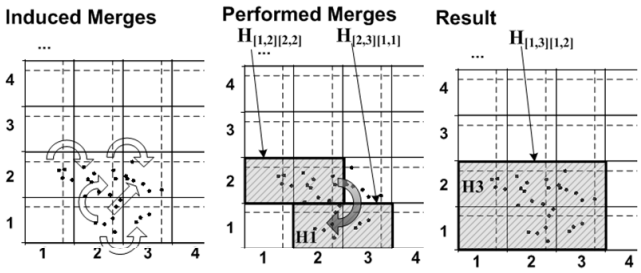


Figure 7: Induced and performed merges

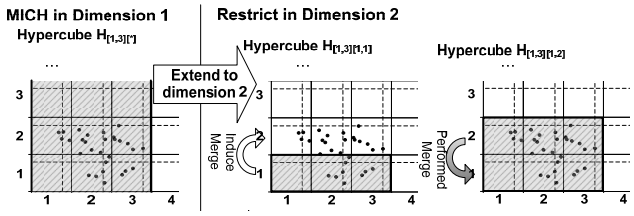


Figure 8: Extending a MICH to next dimension

After a MICH is found the EDSC algorithm checks if the hypercube contains more than $minSize$ objects. If the region does not contain enough objects, the hypercube and all higher-dimensional projections of that hypercube are pruned (monotonicity property, Theorem 2). Thus this **hypercube filter** step based on merges in the density-conserving grid reduces the number of time consuming database scans for density-connected clusters and to prune the search space.

Additionally our EDSC algorithm analyzes only one MICH at a time by successively extending the MICH in all dimensions. When a MICH is extended to a new dimension it is iteratively restricted to each grid cell. For example a two dimensional hypercube embedded in a four dimensional space $\mathbf{H}_{[1,3][1,2][*][*]}$ can be extended into dimensions three and four. We first extended it in dimension three and restrict it to cell one ($\mathbf{H}_{[1,3][1,2][1,1][*]}$). Afterwards the merge procedure is applied recursively. Assume the merge step mines the MICH $\mathbf{H}_{[1,3][1,2][1,2][*]}$. This MICH is next restricted in dimension four: $\mathbf{H}_{[1,3][1,2][1,2][1,1]}$. After this step the next hypercube ($\mathbf{H}_{[1,3][1,2][*][1,1]}$) is analyzed and processed accordingly.

For simplicity, we demonstrate the extension step using a one-dimensional MICH $\mathbf{H}_{[1,3][*]}$ (see Figure 8). After extending the hypercube $\mathbf{H}_{[1,3][*]}$ to dimension two and restricting it to the first grid cell $\mathbf{H}_{[1,3][1,1]}$ the merge procedure is applied again. As the corresponding connectivity border is not empty, a merge is induced and performed directly afterwards, resulting in hypercube $\mathbf{H}_{[1,3][1,2]}$. $\mathbf{H}_{[1,3][1,2]}$ again corresponds to a MICH in subspace $\mathbf{S} = \{1, 2\}$ as all of its connectivity borders are empty.

This concludes the discussion of our EDSC algorithm. In the following, we demonstrate the efficiency and accuracy of our approach in the experimental evaluation.

5. EXPERIMENTS

We evaluate the efficiency of the EDSC algorithm as the first lossless but also efficient approach to density-based subspace clustering. The most recent non-approximate density-based subspace clustering algorithm SUBCLU, which extends DBSCAN to subspace clustering, is used for comparison [10]. Additionally, we contrast our approach with SCHISM, a recent efficient but approximative grid-based algorithm. Experiments were run on Pentium 4 machines with 2.4 Ghz and 1 GB main memory. To evaluate scalability we use synthetic data sets. Further on we show the performance of EDSC on three real world data sets.

5.1 Scalability

Setup and data sets. Based on properties of real world data sets we generate synthetic data for scalability experiments. We extend a method proposed in SUBCLU [10] to generate density-based clusters in arbitrary subspaces. Given the subspace and the number of objects for each cluster, dense regions separated by noisy regions are created. Objects can belong to multiple subspace clusters, just as in most real world data sets. We generate data of different dimensionalities and hide subspace clusters with a maximal dimensionality of 80% of the data dimensionality. Four subspace clusters are hidden in the synthetic data with two of them overlapping in 10% of their objects.

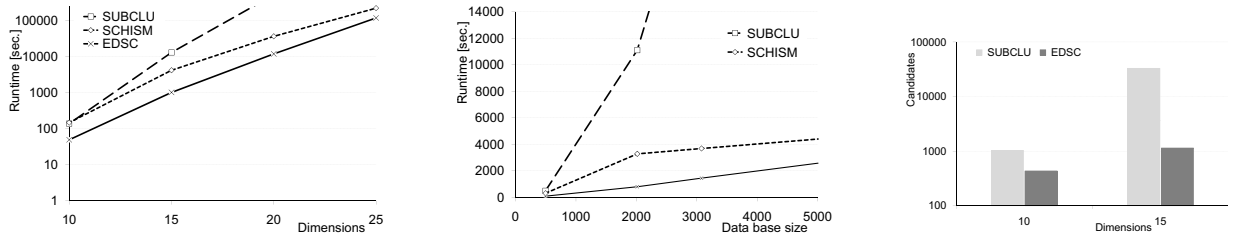
Scalability w.r.t. dimensionality. As the number of possible subspace clusters depends exponentially on the dimensionality of the subspace, scalability in term of dimensionality is crucial for any subspace clustering algorithm. As depicted in Figure 9(a), SUBCLU does not scale w.r.t. dimensionality. For the 20-dimensional data set the algorithm did not even finish after six days. The reason is the tremendous amount of 1 and 2-dimensional subspaces that have to be investigated with density-based clustering before processing any higher-dimensional subspace. SCHISM as a grid-based approach has better runtimes, however, it is only an approximative technique as it loses density-based clusters due to its traditional grid structure. The EDSC algorithm overcomes the scalability problems of SUBCLU due to the efficient filter steps. And in contrast to SCHISM, the EDSC approach is lossless because of its novel density-conserving grid.

Scalability w.r.t. data base size. In the next experiment we generate 15-dimensional data sets with different approximate 500-5000 objects. As we can see in Figure 9(b), EDSC and SCHISM scale well w.r.t. the number of objects. SUBCLU also does not scale with increasing number of objects because of the time consuming data base scans needed for density-based clustering. In contrast, EDSC uses in its first filter step the novel density-conserving grid to avoid these scans and thus is nearly not influenced by the data base size.

Selectivity. To illustrate the efficiency of the hypercube filter we analyze its selectivity compared to the number of density-based scans in SUBCLU. Recall that density-based clustering has quadratic complexity w.r.t. the number of objects. Thus avoiding density-based scans contributes significantly to the performance gains of EDSC. Figure 9(c) shows the number of required density-based clustering computations for the data from our first experiment with varying dimensionality. The highest dimensionality shown is 15-dimensional because SUBCLU does not scale to higher-dimensional data. We see that indeed the main drawback of SUBCLU is the large number of regions that have to be clustered. Multistep filtering in EDSC substantially lowers the number of density-based clustering computations. Many regions that have to be processed by SUBCLU are pruned by EDSC. Pruning is possible without any data base scans in the first filter step, only based on the information of the grid cells and the novel border elements in the new density-conserving grid.

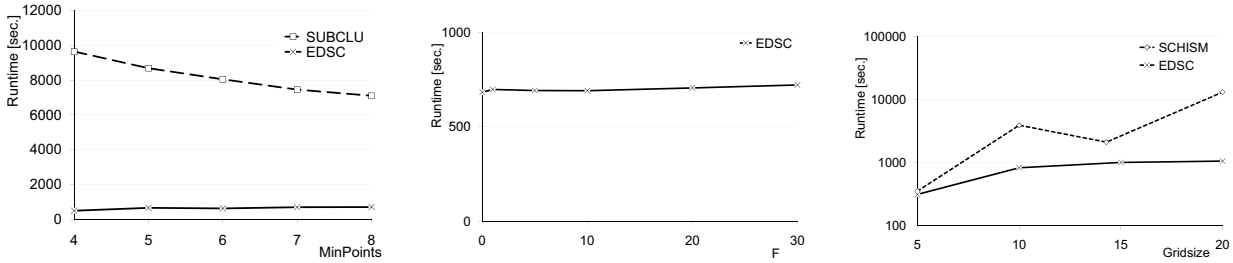
5.2 Parameterization

Detecting clusters in arbitrary subspaces may require tedious parameter tuning in some algorithms. We evaluate parameter setting for SUBCLU, SCHISM and EDSC to study their parameter sensitivity and demonstrate that the EDSC algorithm is quite robust in terms of parameter settings.



(a) Scalability vs. dimensions (b) Scalability vs. data base size (c) Selectivity vs. dimension

Figure 9: Scalability evaluation. Parameters: $gridSize = 10; minSize = 5\% \cdot |\mathbf{DB}|; F = 1; minPoints = 8; \varepsilon = 4$



(a) MinPoints vs. runtime (b) F parameter vs. runtime (c) Gridsize vs. runtime

Figure 10: Parameter evaluation. Data: dimensions $d = 15$; data base size $|\mathbf{DB}| = 1500$

Fixed density thresholds. As already mentioned SUBCLU is dimensionality biased due to a fixed density threshold $MinPoints$. For different dimensional subspaces the measured density is not comparable. To find high dimensional subspace clusters one has to use a lower value for $MinPoints$ in SUBCLU. In EDSC this parameter is robust because of the unbiased density approach as defined in Section 3 following the DUSC approach [4]. Figure 10(a) illustrates that with decreasing $MinPoints$ the runtime of SUBCLU increases. Finding high dimensional subspace clusters is virtually infeasible because already for $MinPoints = 8$ SUBCLU did not scale.

Variable density thresholds. As EDSC uses a variable threshold its density measure is comparable for clusters in different subspaces. In Figure 10(a) we see that the choice of $MinPoints$ has almost no effect on the runtime of the EDSC algorithm. As we can see in Figure 10(b) the second parameter F has almost constant runtimes, too. Using a unbiased density measure which automatically adapts to the dimensionality of the subspace, the variable density threshold is simply set with the fixed parameter F. F reflects the factor by which the variable expected density should be exceeded. Thus EDSC is easily parameterized by a constant and thus robust factor, yet adapts to the dimensionality of the subspace. Additionally the EDSC approach achieves a lossless pruning for this robust variable thresholds due to the second filter step based on the weak density criterion.

Gridsize. Grid-based algorithms like SCHISM suffer from sensitivity to the grid resolution. In Figure 10(c), runtime of SCHISM and EDSC for varying gridsizes are shown. The performance of SCHISM depends largely on the grid structure not only in terms of runtime as shown in Figure 10(c), but also in terms of accuracy (discussed later on, Fig. 12(b)). This is due to the sensitivity of SCHISM to grid cell resolution and positioning. EDSC is robust to positioning and resolution of the grid through its grid cell merges proposed in our novel density-conserving grid.

Weighting functions. EDSC uses a weighted density measure for density assessment in the neighborhood of an object (cf. Section 3). We show that its runtime is insensitive to the choice of kernels for weighting. Figure 11 illustrates the runtimes for SUBCLU and SCHISM compared to EDSC with the Rectangle (REC) $W(x) = 1$, Epanechnikov (EPA) $W(x) = 1 - x^2$ and Bisquare (BIS) $W(x) = (1 - x^2)^2$ kernels, respectively.

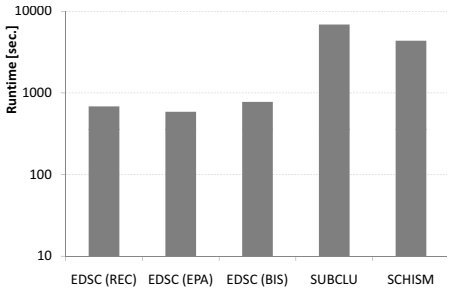
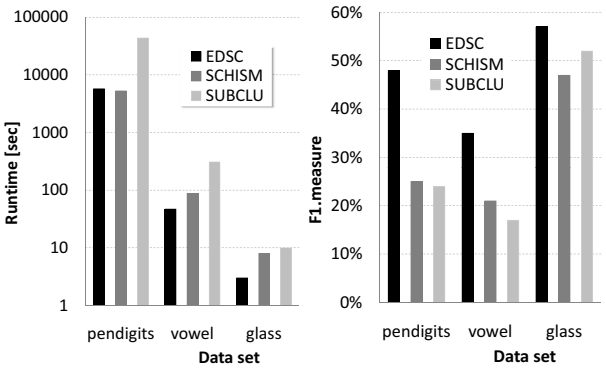


Figure 11: Three kernels vs. SUBCLU and SCHISM

5.3 Quality and runtimes on real world data.

Class labeled data (pendigits, vowel and glass) from UCI machine learning repository [17] are used as ground truth to evaluate the quality of subspace clustering as in other recent subspace clustering approaches [1, 18]. The data cover a variety of dimensionalities and data base sizes from 9 dimensions in glass, 10 in vowel up to 16 in pendigits, and



(a) Runtimes (b) Accuracy

Figure 12: Real world data

with 214 objects in glass, 990 in vowel and 7494 in pendigits, respectively.

Runtimes for all data sets are given in Figure 12(a), the corresponding accuracy measurements in Figure 12(b). Accuracy is determined as the F1-value that is commonly used in evaluation of classifiers and recently also for subspace or projected clustering as well [19, 15]. It is computed as the harmonic mean of recall ("are all clusters detected?") and precision ("are the clusters accurately detected?") values, respectively. The F1-value of the whole clustering is simply the average of all F1-values. The class label assigned to any detected subspace cluster is its most frequent class label.

From the F1-value results we can see that the enormous efficiency gains of the EDSC algorithm are achieved for an effective density-based model, namely DUSC [4] that outperforms competing approaches. Thus, EDSC is an algorithm that due to its efficient but also lossless multistep approach shows significantly better runtimes for a subspace clustering model of very high accuracy, in two datasets even better runtimes than the approximate SCHISM approach.

6. CONCLUSION

We introduced EDSC, an efficient density-based subspace clustering algorithm. EDSC uses a data base inspired multistep approach which allows powerful pruning of the search space by exploiting two monotonicity properties. In our first filter step EDSC efficiently prunes the search space without any density-based clustering. It is based on our novel density-conserving grid which ensures complete detection of density-connected regions by enclosing hypercubes. In our second filter step EDSC ensures lossless pruning for variable density thresholds by incorporating our weak density criterion. For both filter steps we prove lossless pruning and thus the completeness of our density-based subspace clustering. Overall EDSC achieves efficiency without jeopardizing accuracy, as our multistep algorithm is capable of losslessly detecting subspace clusters with respect to unbiased density and different kernel weighting functions. Our experiments on large and high-dimensional synthetic and real world data sets show that EDSC outperforms recent subspace clustering algorithms by orders of magnitude.

7. ACKNOWLEDGMENTS

This research was funded in part by the cluster of excellence on Ultra-high speed Mobile Information and Communication (UMIC) of the DFG (German Research Foundation grant EXC 89).

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

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

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

[7] 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.

[8] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, pages 58–65, 1998.

[9] I. Joliffe. *Principal Component Analysis*. Springer, New York, 1986.

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

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

[12] 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.

[13] S. Lauritzen. The EM algorithm for graphical association models with missing data. *Comp. Statistics & Data Analysis*, 19:191–201, 1995.

[14] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symp. Math. stat. & prob.*, pages 281–297, 1967.

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

[16] H. Nagesh, S. Goil, and A. Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. In *TR 9906-010, NWU*, 1999.

[17] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of MLDBs, 1998.

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

[19] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, USA, 2005.