Relevant Subspace Clustering: Mining the Most Interesting Non-Redundant Concepts in High Dimensional Data

Emmanuel Müller* Ira Assent† Stephan Günnemann* Ralph Krieger* Thomas Seidl*

*RWTH Aachen University, Germany
{mueller, guennemann, krieger, seidl}@cs.rwth-aachen.de

†Aalborg University, Denmark
ira@cs.aau.dk

***Abstract*—Subspace clustering aims at detecting clusters in any subspace projection of a high dimensional space. As the number of possible subspace projections is exponential in the number of dimensions, the result is often tremendously large. Recent approaches fail to reduce results to relevant subspace clusters. Their results are typically highly redundant, i.e. many clusters are detected multiple times in several projections.**

In this work, we propose a novel model for relevant subspace clustering (*RESCU*). We present a global optimization which detects the most interesting non-redundant subspace clusters. We prove that computation of this model is NP-hard. For *RESCU*, we propose an approximative solution that shows high accuracy with respect to our relevance model. Thorough experiments on synthetic and real world data show that *RESCU* successfully reduces the result to manageable sizes. It reliably achieves top clustering quality while competing approaches show greatly varying performance.

***Keywords*-data mining; high dimensional data; subspace clustering; redundancy removal; global optimization;**

I. Introduction

In many recent applications like sensor networks, customer segmentation or gene expression analysis, objects are described by very many attributes. As collecting and storing data is cheap, users tend to record everything they can. Analysis of such high dimensional data has become a major challenge. Traditional cluster analysis groups similar objects while separating dissimilar ones [12]. However, for many attributes, i.e. for high dimensional data, meaningful clusters no longer exist (cf. "curse of dimensionality") [7]. Dimensionality reduction techniques such as principle components analysis (PCA), do not provide a solution to this problem, as they reduce all objects to a single projection [13]. However, patterns occur in multiple projections of the data.

Subspace clustering detects clusters in arbitrary projections by automatically determining a set of relevant dimensions for each cluster [22], [17]. Thus, one is able to detect objects as part of various clusters in different subspaces. In bioinformatics, for example, genome data analysis clusters genes (objects) which show similar expression levels in a subset of experimental medical treatments (attributes). Such similarities might indicate functional relationships. Each gene might appear in multiple roles (subspace clusters). As a consequence, subspace clusters might overlap in the sense that they share objects. Recent research has seen a number of approaches using different definitions of what constitutes a subspace cluster [2], [14], [23], [16]. As summarized in a recent evaluation study [21], their common problem is that the output generated is typically huge.

Subspace clustering allows multiple clusters per object, but has to cope with detection of exponentially many subspace clusters in arbitrary projections. Many of these clusters do not provide any further information, as more or less the same object groups are detected in multiple projections of the data. Such redundant subspace clusters should be removed and only the most interesting ones, which provide novel knowledge about the data should be reported.

A related approach, projected clustering assigns each object to a single projection [1], [19]. This strict partitioning of the data into projected clusters can be regarded as extreme redundancy elimination. Projected clustering results in a manageable number of clusters, but is not able to detect overlapping clusters.

Some recent approaches towards modeling and removing redundant subspace clusters have been proposed. They define non-redundant and possibly overlapping subspace clusters, but only with a local scope. In [3], [5], a subspace cluster is redundant if it shares a certain fraction of objects with another one. Retaining only maximal subspace clusters, i.e. the highest dimensional one of two, results in a clear increase in clustering quality. This is due to the fact that maximal clusters tend to contain less noise and thus represent the inherent data structure more faithfully. This definition of redundancy, however, is limited in two respects. First, redundancy is based only on a pairwise comparison of clusters. And second, the redundancy check incorporates only the fraction of jointly detected objects [5]. We call this a local redundancy definition, as only local properties like object count and comparison of two subspace clusters is used. Consequently, a redundant subspace cluster that is covered by a combination of high dimensional subspace clusters is still reported as non-redundant. In contrast, we aim at a global redundancy check including a more flexible interestingness definition comparing each cluster with the overall set of detected subspace clusters.

A recent approach aims at extracting non-redundant axis-parallel subspace regions [18]. It defines non-redundant results as the minimal subset of subspace clusters to approximately compute the support of any other subspace cluster (assuming uniform distribution otherwise). This statistical approach, however, is based on the assumption of uniform distribution inside a cluster. Moreover the redundancy model is limited to the fixed cluster definition. As we show in our evaluations all of these approaches fail to detect all and only the hidden concepts, i.e. clusters, in a high dimensional database.

Our goal is to derive a novel model for non-redundant subspace clusters that takes a global look at overlapping clusters. The aim is to find all and only relevant concepts by optimizing the overall clustering result. The main contributions of our work are:

- Detection of all interesting clusters, but without the overwhelming result size of subspace clustering.
- Detection of only non-redundant clusters, but without the strict partitioning of projected clustering.

To achieve these objectives, we introduce a new global relevance model for subspace clustering. We combine a novel interestingness function for subspace clusters with a novel coverage criterion for an overall redundancy removal. By including the most interesting clusters and excluding redundant clusters, our result set contains all and only relevant subspace clusters. Any object can be part of multiple clusters, allowing overlapping subspace clusters. It is desirable to report as few clusters as possible to reduce redundancy, yet cover as many interesting concepts as possible. Furthermore, our relevance model is neither restricted to one cluster definition nor based on a fixed interestingness rating. These two aspects are typically application dependent and can be adapted by the instantiation of our model. Considering computation complexity, we prove that our relevant subspace clustering is NP-hard. Thus, for an efficient computation of our relevance model we propose an approximative algorithm generating possible cluster candidates in best-first order according to their relevance.

II. Relevant Subspace Clustering

In this section, we introduce our relevant subspace clustering definition. Most existing approaches use only local properties, i.e. objects O and dimensions S, to define if $C = (O, S)$ is a subspace cluster or not. We take a global view, considering the theoretic set of possible subspace clusters $IN = \{C_1 \dots C_n\}$ in totality for our relevant clustering definition. As pre-computation of IN is too expensive we propose an approximative algorithm using a novel on-demand cluster generation derived from our relevance model in Section III. Most of the clusters in IN do not provide any knowledge for the user, thus, our relevance model defines which of these clusters to output (cf. Fig. 1). We aim at reducing the output to only the *relevant* subspace clustering $M \subseteq IN$. The remaining clusters overwhelm the user, thus hinder the analysis and are removed.

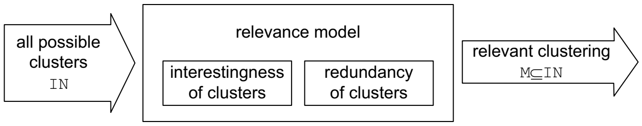


Figure 1. Components of a relevance model

Two aspects are important for relevance. One is the *interestingness* of a cluster itself. The interestingness evaluates a cluster locally via its properties like dimensionality or size. For example, clusters in only one dimension might not be interesting. Interestingness is often user or application dependent and therefore it should be adaptable. A detailed discussion of the interestingness is given in Section II-A.

Another aspect is the *redundancy* of clusters. Redundancy means that this cluster does not contribute to new knowledge with respect to other clusters (for example if another cluster with similar properties exists). Consequently, a redundant cluster should not be outputted. The redundancy is discussed in Section II-B.

Please note that the two aspects are not entirely independent. Given the choice between two clusters with similar properties, the less interesting one should be marked as redundant. Both aspects constitute our relevance model: A relevant cluster is interesting but not redundant. The overall relevance model is described in Section II-C.

In Section II-D, we prove important complexity results and conclude in Section II-E with an instantiation of our model.

A. Interestingness of a cluster

In this section, we quantify the interestingness of a cluster. Figure 2 gives an example of clusters in dimensions 1 and 2 (left), and 3 and 4 (right), respectively. Objects in both illustrations are represented using the same symbol. For example, assume that we deem clusters with more dimensions more interesting. The 2d (two dimensional) clusters C_1 and C_2 are then favored over the 1d clusters $C_3 - C_6$. The number of objects, the diameter or the density of a cluster are other typical choices of interestingness.

Interestingness is not handled explicitly in existing projected and subspace clustering algorithms. In projected clustering algorithms, overlapping clusters are not detected due to the partitioning. In Figure 2, the most interesting clusters C_1 and C_2 are not found. They both contain some objects that occur in the other cluster as well, and are therefore mutually exclusive, even though the objects occur in dimensions 1 and 2, or 3 and 4, respectively. As a consequence, potentially interesting clusters are missed by the occurrence of *other* interesting clusters. Another problem is the handling of outliers only in a post-processing step. Interestingness of the clusters is calculated before removal of outliers and hence misleading values can occur.

Subspace clustering algorithms realize a very limited interestingness calculation. Each set of points and dimensions that fulfills the cluster definition (e.g. exceeding a density threshold) is equally interesting. It is a binary decision "cluster" or "no cluster".

And finally, the calculation of the interestingness in both models is usually fixed and cannot be modified by the user.

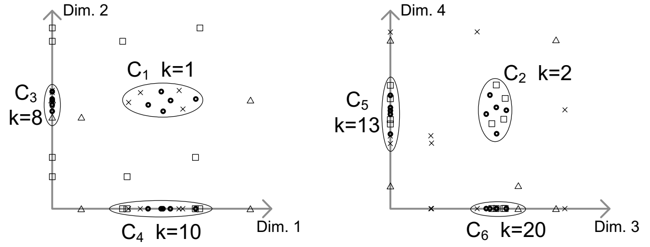


Figure 2. Interestingness by cost values

Our model overcomes these problems. We assign an interestingness value to each cluster. It is a local rating, so that other clusters do not influence this measure. Furthermore, our cluster definition (cf. Sec. II-E) accounts for outliers so that misleading values do not occur.

We model the (un-)interestingness via a cost function k. Small values denote interesting clusters. For example, in Figure 2 the 1d cluster C_4 might get a cost value of $k = 10$ while the more interesting 2d cluster C_1 might get a lower cost value of $k = 1$.

For any cluster C, given by the set of its objects O and the corresponding dimensions S, we define the cost function as follows:

Definition 1: Cost function
Let $\mathcal{P}(DB)$ be the power set of all database objects and $\mathcal{P}(Dim)$ the power set of the dimensions. A cost function

$$k : \mathcal{P}(DB) \times \mathcal{P}(Dim) \to \mathbb{R}$$

assigns cost $k(O, S)$ to the subspace cluster $C = (O, S)$.

By defining a cost function, we can account for several aspects of a subspace cluster, like the dimensionality or the density. Changes to the cost function yield an easy adaptation of the model. This flexibility is not achieved by other approaches. An instantiation of the function is presented in Section II-E. By first calculating the interesting values individually and allowing overlap we find higher quality clusterings. In Figure 2 we may select both C_1 and C_2.

For computational reasons, we assume cost functions which assign a strictly positive value to all subspace clusters, i.e. given a set $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$ of clusters, the function fulfills $k(O_i, S_i) > 0$ for all $(O_i, S_i) \in M$. To mine the most interesting clusters we minimize the overall cost. However, to also maximize the number of objects covered by a clustering, we additionally take coverage into account. We formalize coverage as follows:

Definition 2: Coverage of a clustering
Given a clustering $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$, the coverage of M is defined as follows: $Cov(M) = \bigcup_{i=1}^{n} O_i$

The coverage of a clustering M is the union of the objects in all selected clusters. We now define the overall relative cost of a clustering as the sum of the individual cost values normalized by the number of covered objects.

Definition 3: Overall relative cost of a clustering
Let $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$ be a clustering and k a cost function. We define the overall relative cost of M as:

$$RK(M) = \frac{K(M)}{|Cov(M)|} \text{ with } K(M) = \sum_{(O_i,S_i) \in M} k(O_i, S_i)$$

The smaller the overall relative cost $RK(M)$, the more interesting is the clustering M per covered object. We achieve a high coverage and at the same time small total cost.

B. Redundancy of a cluster

The second aspect of relevance is non-redundancy. A large cluster C and all of its lower dimensional projections could be assigned low cost values if interestingness is based on size. Selecting all projections along with C based on interestingness alone leads to a poor overall result. One gets very many redundant clusters, while C would be sufficient.

We therefore take a global view for redundancy elimination and compare a cluster with other clusters. While the interestingness is a local measure based on the cluster itself, the redundancy takes other clusters into account.

Existing projected and subspace clustering algorithms do not address redundancy handling adequately. Projected clustering simply forces results to be non-redundant by assigning each object to a single cluster at the cost of missing overlapping clusters. Subspace clustering algorithms, in contrast, either use no or a mere local approach to check the redundancy. Such an approach compares only two clusters [5]. If the clusters cover nearly the same objects, one of them is redundant. The problem of this local approach is illustrated in Figure 3. Obviously, in both subfigures the cluster C_2 is redundant because it is induced by the other clusters C_1, resp. C_{1a}, C_{1b}. A local approach could identify the redundancy in the left figure. Cluster C_2 is redundant, as it covers C_1 and only a few additional objects. In the right figure, the fraction of points shared by C_{1a} and C_2 as well as by C_{1b} and C_2 is small, and the cluster C_2 is misleadingly classified as non-redundant. This mistake is the result of the local view on redundancy, i.e. for each check only a pairwise comparison of clusters is performed.

We use a global view for the redundancy checks, i.e. we use all clusters at the same time to judge the redundancy

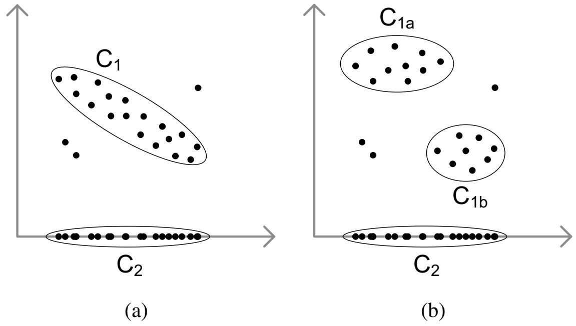


Figure 3. Local and global redundant clusters

of another cluster. This approach results in more accurate decisions.

As one can see from the above example, the redundancy of a cluster is linked to the coverage of objects. If a set of clusters shares many objects with a cluster C, C is a redundant cluster. In other words: A cluster is redundant if it does not cover many new objects. The cluster C_2 in Figure 3(b) is redundant, because with respect to the two other clusters only a few new objects are covered. The same holds for the cluster C_2 in Figure 3(a). The fact that we consider all clusters for the redundancy checks yields a global redundancy model. Thereby we identify in both subfigures the cluster C_2 as redundant.

Basic Set-Cover approach: If we select the *minimal* number of clusters such that all objects are covered, we can realize a global redundancy model. The output of clusters that only contain already covered objects is prevented. This check is with respect to all other clusters in the result set. At the same time we identify multiple overlapping concepts in the data, because all objects have to be covered.

This novel way of subspace clustering can be considered as an instance of the Set-Cover problem [11]. Given several finite sets, Set-Cover seeks for the minimal number of sets that cover the whole population. In the clustering context, we want to find the minimal number of clusters, such that all objects are covered.

However the direct application of Set-Cover to our task is not possible.

Problem 1: Simply choosing the minimal number of clusters means that usually low dimensional clusters, which tend to contain more objects, are preferred over high dimensional clusters. This preference usually conflicts with the user's notion of interestingness. Instead of choosing the minimal number of clusters, we determine the clustering with minimal relative cost (cf. Def. 3). This setup takes the desired interestingness notion into account. In Figure 3(b), for example, we would choose the two 2d clusters instead of the one 1d cluster. We thus use an extension of the Set-Cover problem to the Weighted-Set-Cover problem [9].

Problem 2: Covering all objects by clusters is not always a meaningful solution, as some databases contain outliers that do not fit to any concept. The Set-Cover problem enforces a complete cover of all objects and potentially finds no solution in this case. Or, all objects can be covered but only by uninteresting clusters. For example, if the outliers are only contained in the 1d clusters the Set-Cover problem enforces choosing these uninteresting ones. An example is in Figure 3(b), as one has to select C_2 to get a complete coverage. We therefore propose a generalization of the Set-Cover problem to cope with outliers.

Gain-based redundancy: We solve these problems by a gain-based extension of the Set-Cover problem. The basic idea is to measure the gain of a new cluster if we add it to a known clustering. In other words, we have to answer the question: Is it worthwhile to take the "more complex" clustering?

Let us consider Figure 4 and assume the clusters C_1 and C_2 to be selected. Intuitively, the cluster C_3 is redundant with respect to the selected ones, so its gain should be small. Two important aspects contribute to this fact. First, the cluster C_3 covers only a few new objects, i.e. many objects are already contained in other clusters. These new objects, covered by the cluster $C_3 = (O, S)$, can be calculated via the residual set $O \backslash Cov(\{C_1, C_2\})$. And second, the cost of the cluster is very high, i.e. the cluster is not interesting. High cost $k(O, S)$ should correspond to a low gain. Overall we take the ratio of these two measures. Cluster gain thus is the additional coverage in relation to its cost.

Definition 4: Cluster gain
Given a cluster $C = (O, S)$, a clustering M and a cost function k for $M \cup \{C\}$. The cluster gain of C with respect to M is:

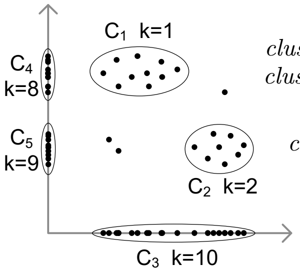
$$clus\_gain(C, M) = \frac{|O \backslash Cov(M)|}{k(O, S)}$$

Usually, high dimensional clusters are favored so that these clusters should get low cost values. Nevertheless, by changing the cost function we can also change the cluster gains and hence the preferred clusters.

We identify a cluster as redundant (with respect to a given clustering) if its cluster gain is too small. In Figure 4 the cluster gain of C_3 (with respect to $\{C_1, C_2\}$) is only 0.3 as C_3 covers 3 new objects and $k(C_3) = 10$. Instead, C_2 has a higher cluster gain of 4 (with respect to $\{C_1\}$). Consistent with this idea, a clustering M is redundancy-free if *all* clusters from M exceed a minimum cluster gain. The gain has to be measured with respect to the remaining clusters from M so that each cluster adds sufficient information to the overall clustering.

Definition 5: Redundancy-free clustering
Given a clustering $M \subseteq IN$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. The clustering M is redundancy-free, iff

$$\forall C \in M : clus\_gain(C, M \backslash \{C\}) > \Delta$$



$$clus\_gain(C_3, \{C_1, C_2\}) = \frac{3}{10} = 0.3$$
$$clus\_gain(C_4, \{C_1, C_2\}) = \frac{1}{8} = 0.13$$
$$clus\_gain(C_2, \{C_1\}) = \frac{8}{2} = 4$$
$$clus\_gain(C_1, \{C_2\}) = \frac{10}{1} = 10$$
$$\frac{K(\{C_1,C_2\})}{|Cov(\{C_1,C_2\})|} = \frac{3}{18} = 0.17$$
$$\frac{K(\{C_4,C_5\})}{|Cov(\{C_4,C_5\})|} = \frac{17}{21} = 0.81$$

Figure 4. Relevant clustering

In this way we achieve a global view for the redundancy checks. Each cluster C has to compete with all remaining cluster $M \backslash \{C\}$ in the result set at the same time.

As one can see, the clustering $\{C_1, C_2\}$ in Figure 4 is redundancy-free while the clusterings $\{C_1, C_2, C_3\}$ or $\{C_1, C_2, C_4\}$ contain redundant information (assuming $\Delta = 0.5$). Definition 5 gives us the basis to avoid redundancy in our clustering. However, if we remove clusters from a redundancy-free clustering, this property still holds for the smaller clustering. The removal of information, i.e. clusters, cannot generate redundancy. This means that also the empty clustering is redundancy-free. That is obviously not desired by the user. To obtain a relevant clustering, it should be redundancy-free but at the same time cover as many objects as possible.

C. Overall relevance of a clustering

To enforce the selection of clusters we introduce the property of concept-covering. A clustering M does *not* satisfy this property as long as clusters exist, which have a high gain and are not in M. M is not the result because further interesting concepts can be covered. A clustering is concept-covering, and hence the coverage sufficient, if all remaining clusters have a small cluster gain. We formalize this by:

Definition 6: Concept-covering clustering
Given a clustering $M \subseteq IN$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. The clustering M is concept-covering, iff

$$\forall C \in IN \backslash M : clus\_gain(C, M) \leq \Delta$$

Intuitively, a clustering is concept-covering if adding *any* new cluster *always* results in redundancy. A concept-covering clustering in Figure 4 is $\{C_1, C_2\}$. Clustering $\{C_1\}$ is not concept-covering because we could add C_2 without introducing redundancy.

The property of concept-covering clusterings is a relaxation of complete coverage. It solves the problem of enforcing a complete coverage as in the Set-Cover problem (cf. Problem 2).

RESCU: Relevant subspace clustering: Our RESCU model demands a relevant clustering to be redundancy-free and concept-covering. However, as the example in Figure 4 illustrates, several clusterings could fulfill both properties, e.g. $\{C_1, C_2\}$ or $\{C_4, C_5\}$. To select the most interesting clustering, we additionally compare their relative cost (cf. Def. 3). Our relative cost function (cost per covered object) ensures finding an optimal clustering with both interesting clusters and high coverage. The relative cost for the clustering $\{C_1, C_2\}$ in Figure 4 is just 0.17, but 0.81 for $\{C_4, C_5\}$. This is formalized by:

Definition 7: Relevant subspace clustering (RESCU)
Given a clustering $M \subseteq IN$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. M is relevant, iff

- *M is redundancy-free and concept-covering (Def. 5 & 6) AND*
- *M has minimal relative cost, i.e. $RK(M) \leq RK(N)$ for all redundancy-free and concept-covering clusterings $N \subseteq IN$*

The relevant clustering in our previous example is thus $\{C_1, C_2\}$. Also for the example in Figure 2 the relevant clustering is $\{C_1, C_2\}$ even though the two clusters share some objects. This illustrates that handling of overlapping clusters is possible in our model.

The flexibility of our model additionally provides the possibility of classifying other clusterings as relevant by changing the interestingness criterion. If we adapt the cost function so that C_1 and C_2 in Fig. 4 get higher cost values, the clustering $\{C_4, C_5\}$ could become relevant. Thus our model enables the user to control the output as desired.

D. Complexity results

Calculating a RESCU clustering is a NP-hard problem. We prove this by giving a polynomial reduction of Weighted-Set-Cover problem, which is a NP-complete problem, to our RESCU model, i.e. Weighted-Set-Cover $\leq_p RESCU$. The Weighted-Set-Cover problem seeks those non-empty sets that together have minimal weights and fully cover all objects. For this reduction we need to map the input of the Weighted-Set-Cover problem to an input of RESCU and show that the resulting relevant subspace clustering is a valid solution for the Weighted-Set-Cover problem.

Theorem 1: Computing RESCU (Def. 7) is NP-hard.

Proof:
We show that Weighted-Set-Cover $\leq_p RESCU$.
Input mapping: We map the input of the Weighted-Set-Cover (objects, sets, weight function) to an input for RESCU (database DB, possible clusters IN, cost function k). We further map the assumption of the Set-Cover (complete coverage exists) to ($\Delta = 0$) in RESCU. As RESCU is a more general problem it finds a clustering even in databases containing noise, where complete coverage is meaningless. We now have to show that relevant subspace clustering (cf. Def. 7) corresponds to a solution of the Weighted-Set-Cover problem.

RESCU generates a valid solution for Set-Cover:

(1) Every chosen set contributes at least one object to the overall coverage:
A relevant clustering M is non-redundant (Def. 5):
$\Leftrightarrow \forall\, C \in M : clus\_gain(C, M\backslash\{C\}) = \frac{|O\backslash Cov(M\backslash\{C\})|}{k(O,S)} > 0$
Thus, all sets contribute at least one object
$\forall\, C \in M : O = Cov(\{C\}) \not\subseteq Cov(M\backslash\{C\})$.

(2) For a Weighted-Set-Cover all objects have to be covered by at least one set:
A relevant clustering M fulfills the concept-covering property (Def. 6):
$\Leftrightarrow \forall\, C \in IN\backslash M : clus\_gain(C, M) = \frac{|O\backslash Cov(M)|}{k(O,S)} \leq 0$
$\Leftrightarrow \nexists\, C \in IN\backslash M :\ |O\backslash Cov(M)| > 0$
Thus, all objects are covered: $Cov(M) = DB$.

(3) The sum of weights is minimal for the chosen sets:
A relevant clustering M has minimal relative costs (Def. 7):
For all non-redundant and concept-covering clusterings $N \subseteq IN$: $RK(M) \leq RK(N) \Leftrightarrow \frac{K(M)}{|Cov(M)|} \leq \frac{K(N)}{|Cov(N)|}$
From $Cov(M) = Cov(N) = DB$ we have: $K(M)$ is minimal.

(1) $\wedge$ (2) $\wedge$ (3) $\Rightarrow$ M is a valid Set-Cover solution.
RESCU is NP-hard. ■

As we have proven, RESCU is a generalization of the Set-Cover problem and thus NP-hard. Furthermore, it has two advantages for detection of relevant clusterings. First, we can handle outliers so that we can find a solution even if a complete coverage is not possible. And second, RESCU incorporates clustering properties and thus mines the most relevant concepts instead of simply covering the data.

E. Instantiation of the model

By the flexibility of our model we can handle any definition of subspace clusters (as the input of the model) and cost functions (to rate the clusters). For a practical evaluation we instantiate these two aspects.

Definition of subspace clusters.
We use density-based clustering because it detects clusters of arbitrary shape and size even in noisy data [10]. The idea is to define clusters as dense areas separated by sparsely populated areas. In this way outliers are not part of the clusters within the same subspace and misleading ratings are prevented. An object is considered dense if its neighborhood, i.e. an ε-distance region around it, is sufficiently populated. We follow the definition from [14] with the modification, that we adjust the ε-range according to the subspace dimensionality. Thereby we account for increasing distances between the objects in higher dimensional spaces, similar to recent approaches [16], [3]. The value of ε in a subspace with dimensionality d is

$$\varepsilon = \left[\frac{4\cdot n}{3\cdot\Gamma(1.5)}\right]^{\frac{1}{5}} \cdot \varepsilon_1 \cdot \left[\frac{d+2}{4\cdot n} \cdot \Gamma(\frac{d}{2}+1)\right]^{\frac{1}{5}}$$

where ε_1 denotes the ε-range in the 1d subspace, n the database size and Γ the gamma function.

Definition of cost functions.
The cost function used in our experiments is $k(O,S) = \frac{1}{|S|^\beta}$ with $\beta \geq 0$. Higher dimensional clusters get lower cost values and are therefore more interesting than lower dimensional clusters. Preference for higher dimensional clusters is also used in [5]. Variation of β influences how much different subspace dimensionalities affect the cost value. A very high β-value implies that the result set mainly contains high dimensional clusters, whereas a low value tends to lead to lower dimensional clusters. If all cost values are nearly identical low dimensional clusters are usually preferred because these clusters generally cover more objects and so their cluster gain is higher.
We implemented further cost functions that reflect other interestingness objectives. For example, the density of the subspace clusters can be taken into account. Due to space limitations these functions are not included in this work.

III. Approximative Algorithm

There are two major challenges for efficient computation of relevant subspace clustering. First, as we have shown in Section II-D, global optimization of the clustering is a NP-hard problem. And second, the assumed input set IN is too large and its enumeration may be difficult or even impossible. Thus, we assume that there exists no efficient and especially no scalable solution. However, we provide an approximative solution tackling both of these challenges based on three main contributions:

- Greedy selection of clusters.
- Relevance update for concept-covering.
- On-demand generation and ranking of subspace clusters according to their cluster gain.

A naive approach would compute all possible subspace clusters (IN is exponential w.r.t. number of dimensions) and then choose an optimal subset of clusters (exponential in the number of results). We use an approach that generates promising clusters on-demand and ranks them according to their relevance information. Thus we avoid an expensive pre-computation of all possible subspace clusters. With our greedy approach we relax the optimization by choosing in each step the most promising cluster available. Please note that a new cluster changes the overall coverage of the data, and it changes the relevance of remaining clusters. Updating the relevance is thus essential for concept-covering.

Greedy Selection of Clusters:
Our greedy approach iteratively includes the best cluster so far. Such an approximation idea is known for other NP-hard problems like Set-Cover [11], [9]. For our novel relevant subspace clustering, this basic idea of greedy processing leads to efficient computation but also high quality subspace clustering results by choosing in each step the most relevant cluster according to our cluster gain definition.

By iteratively picking clusters, we relax two parts of Definition 7. First, instead of checking the global redundancy

(Def. 5) we compute an approximative solution, as in each step i the relevance of a new cluster C_i is checked only w.r.t. clusters $M_i = \{C_1 \ldots C_{i-1}\}$ already chosen in the previous $i-1$ steps. The cluster C_i is non-redundant w.r.t. M_i. Secondly, we choose the cluster with the highest $clus\_gain$ in order to have the most interesting clusters in the result and to approximate the minimal relative cost of the relevant clustering:

Definition 8: Relaxation of relevant clustering
C_i is inserted into the current result set M_i iff

(1) C_i is a non-redundant cluster w.r.t. M_i:
$clus\_gain(C_i, M_i) > \Delta$

(2) C_i is the most interesting cluster in step i:
$\forall C \in IN \backslash M_i : clus\_gain(C_i, M_i) \geq clus\_gain(C, M_i)$

Our relaxation yields an efficient processing. It computes a chain of most relevant clusters C_i yielding an approximative solution for our RESCU model. It is a best-first method as in each step the cluster with the highest gain is selected. The greedy processing terminates if no more relevant clusters are available in the residual set of clusters according to the concept-covering property (cf. Def. 6).

Relevance Update:
The set of resulting clusters directly influences the cluster gain of the next cluster to be chosen. We have to update the cluster gain (Def. 4) of all candidates $C \in IN \backslash M_{i+1}$, each time we insert a cluster C_i into the result set $M_{i+1} = M_i \cup \{C_i\}$. As the new cluster C_i changes the overall coverage of objects $Cov(M_{i+1}) \supseteq Cov(M_i)$, we have to adjust the relevance of all remaining cluster candidates. Our relevance update decreases the gain of redundant clusters that are already covered by C_i. Consequently, other concepts not yet covered have a relatively higher likelihood of being chosen in the following iteration.

Example: In our example given in Figure 4 we first choose C_1 as the first relevant subspace cluster with maximal cluster gain $clus\_gain(C_1, \{\}) = 10$. Intuitively, our algorithm chooses according to the cluster gain definition, which prefers higher dimensional clusters that contain objects not yet covered by other relevant clusters. Thus, in the second step we choose C_2 with the currently highest gain $clus\_gain(C_2, \{C_1\}) = 4$, a 2d-cluster with not yet covered objects. Choosing $\{C_1, C_2\}$ forces a relevance update as now most of the objects in C_3, C_4, C_5 are already covered. Thus, the next most relevant cluster is C_3 with an updated relevance of only 0.3 as it contributes only 3 objects to the overall clustering. Thus, assuming $\Delta = 0.5$, the algorithm has detected all and only the relevant clusters as any remaining cluster has a lower cluster gain.

On-Demand Generation and Ranking:
For our greedy processing we maintain an up-to-date ranked list of subspace clusters with a high cluster gain. However, it would be computationally too expensive to compute all possible clusters IN and then sort them according to their cluster gain. In contrast to such an exhaustive generation of all possible clusters, our approach computes candidates on-demand and reduces computation to only the most promising regions. As most of the exponentially many clusters in IN are not interesting or they are redundant w.r.t. the final result set, they do not affect the global optimization. Hence, our on-demand candidate generation computes only the most promising cluster candidates according to their cluster gain.

For on-demand generation of these regions we use our recent density estimation technique which combines a given set of lower dimensional cluster candidates (initially 2d clusters) to possibly interesting higher dimensional patterns [20]. These new cluster candidates are inserted into our ranking based on their cluster gain. Due to relevance updates in later steps, the positions of these new cluster candidates might be rearranged. The currently most relevant candidates are at the top of the ranking.

Furthermore, as density-based clustering is a computationally expensive task we compute density estimation based on discretized grid cells to approximate the true densities [4]. Efficiency is ensured as we only perform our density-based clustering on the top ranked candidates. Thus, by choosing the top candidate from the ranking, we focus in each step on the most promising cluster either to generate new candidates or to refine an approximative cluster.

For inclusion in the clustering result according to greedy processing we select the top ranked cluster that has been refined and used for new candidate generation.

Using this on-demand ranking in our greedy approach, we ensure efficiency by generation of only a small set of promising subspace cluster candidates as needed. Furthermore, in each step we select the most promising (top ranked) region for processing.

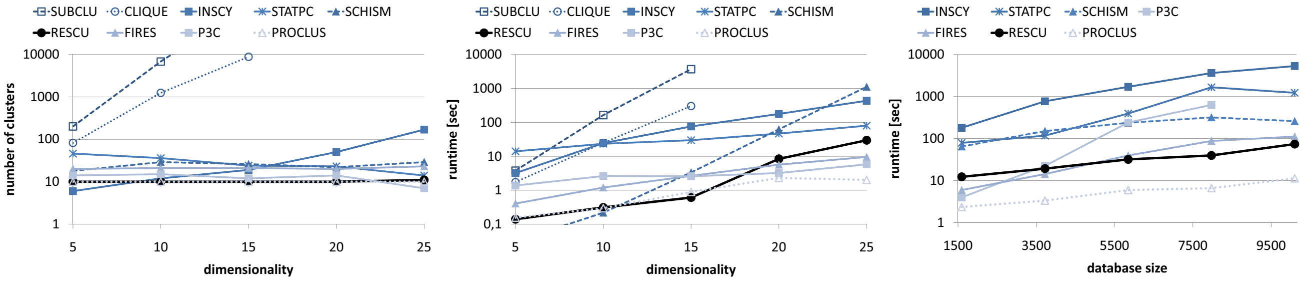
IV. Experiments

We compare RESCU with recent representatives of different high dimensional clustering paradigms: Grid-based subspace clustering CLIQUE [2] and its extension SCHISM [23]; Density-based algorithms SUBCLU [14], INSCY [5] and the approximative FIRES [16]; Projected clustering PROCLUS [1] and statistical P3C [19] and StatPC [18].

Note that we have optimized parameters for each algorithm on each data set. Furthermore, we provide supplementary material[1] (executables, exact parameter settings and data sets used in our evaluation) for repeatability and comparison. For a fair comparison of the competing approaches we use a recent evaluation framework [21]. All implementations are in Java and experiments were run on Intel Core 2 Duo computers with 3 GHz and 2 GB main memory.

Benchmark data from the UCI archive [6] (also used in the evaluation study in [21]) is used to study performance on

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



(a) Result size

(b) Runtime

(c) Runtime

Figure 5. Scalability w.r.t. dimensionality and size of the database

real data. In addition, we use 17-dim. features as extracted in [5] from sequence data in [15]. Besides the UCI 16-dim. pendigits data, we use 32 and 48-dim. variants by interpolation of the available polylines. As the true number of clusters is unknown for real data, accuracy of class labels as in e.g. [19], [5], [20] is measured. We employ synthetic data for validating that all generated clusters are identified and for scalability experiments. Following the method in [14], [5] for overlapping density-based clusters in arbitrary subspaces, we generate data of different dimensionalities and hide subspace clusters with a dimensionality of 50%, 60% and 80% of the data dimensionality.

We measure quality as the F1 value [19], [5], [20], [24], which among other things evaluates the cluster purity. It is computed as the harmonic mean of recall ("are all clusters detected?") and precision ("are the clusters accurately/purely detected?"). On real data, the class label of a detected subspace cluster is its most frequent label. The F1 value of the entire clustering is the average of all clusters' F1 values. Additionally, we provide the commonly used accuracy of classifiers (e.g. C4.5 decision tree) built on the detected patterns [8], [20]. A high accuracy indicates that the subspace clustering is a good generalization of the underlying data distribution. By providing both measures on many data sets, we provide a thorough analysis for real world data.

A. Scalability

Our comparative study begins with an analysis of the result size and runtime (Fig. 5(a) & 5(b)) with respect to the dimensionality on a synthetic data set with 10 hidden clusters.

Redundancy maintenance: First, we want to show the result size of the algorithms CLIQUE and SUBCLU, which do not provide redundancy removal. They produce overwhelming result sets that are several orders of magnitude larger than the hidden clusters. They suffer from the fact, that any cluster typically induces several very similar clusters in lower dimensional projections, especially as dimensionality increases. Our RESCU approach successfully detects only the 10 relevant subspace clusters.

Figure 5(b) illustrates the effect of size on the runtime of these approaches. (Please note the logarithmic scale.) Beyond 15 dimensions, the poor scalability of CLIQUE and SUBCLU due to their result size renders analysis infeasible on standard desktop PCs. In the following experiments, we do not include CLIQUE and SUBCLU due to their poor performance in terms of result size and runtimes. RESCU, as a representative for models with redundancy removal, clearly outperforms these algorithms.

In summary, redundancy removal is a key property to provide interpretable results and good scalability. Algorithms that maintain redundant clusters are not considered in the following experiments.

Redundancy removal: Compared to CLIQUE and SUBCLU the remaining algorithms scale to higher dimensional data sets and show significantly smaller result sizes in the range from 6 to 169 as depicted in Figure 5(a). Please note that comparison with some algorithms that require the number of clusters as an input, like the projected clustering approach PROCLUS, does not reflect their performance in real application scenarios where this information is typically not available. Most of the algorithms output more than 10 clusters, however, they where not able to detect all of the 10 hidden clusters. RESCU is able to find all hidden clusters. Overall only our RESCU relevance model is able to find the hidden and thus relevant clusters. We further investigate the quality of the clustering result on real world data sets in Sec. IV-B.

The runtime of RESCU (Fig. 5(b)) is comparable to that of the other approaches. It is less affected by the dimensionality as it computes only the relevant subspace clusters on-demand and excludes most irrelevant results. Our relevance ranking of cluster candidates and the greedy processing ensures an overall efficient computation. Although some algorithms have smaller runtime, RESCU is still efficient and, as we believe, this aspect is compensated by the higher clustering quality of RESCU.

Our next experiment in Figure 5(c) shows the runtime of the algorithms with respect to the database size. Our RESCU approach scales well whereas most of the competing algorithms show a greater increase in runtime. Overall the results from this experiment are comparable to the results in Fig. 5(b).

B. *Quality on real world data*

Our next experiment in Figure 6 evaluates F1 value and accuracy for the pendigits data. We vary the dimensionalities from 16 to 48. For F1 measure on the 16d data set we observe top quality results for RESCU, P3C and PROCLUS. Hence these algorithms find almost all and pure clusters. While P3C does not scale to the higher dimensional data sets, RESCU and PROCLUS reach again high qualities. Considering accuracy, our RESCU approach has the best quality results. SCHISM is second in accuracy, but has a significantly lower F1 value. In general, RESCU is the only approach that shows top results for both measures in all dimensionalities. Our novel model is able to detect the most interesting and non-redundant clusters in the data set.

It must be highlighted that the results of the two density-based approaches INSCY and FIRES cannot compete with RESCU. The high quality of RESCU is not only a result of the density-based instantiation of the subspace cluster definition but in particular due to our new relevance model.

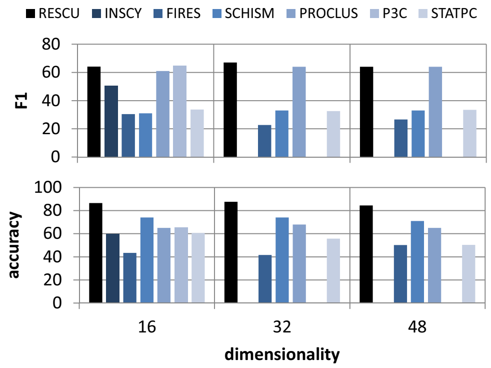


Figure 6. Quality on pendigits data (7494 objects; 16-48 dimensions)

In Figure 7 we show F1 and accuracy results for six further real world data sets. In addition to the absolute values we note the relative quality compared to the best measurement on each data set. Best 95% results are highlighted in gray. RESCU achieves top quality results for *all* data sets with respect to *both* measures. Competing approaches show highly varying performance. None of them achieves top quality allover. Although some of the approaches achieve slightly better results on some of the data sets, RESCU reliably shows top results on all data sets.

C. *Parametrization*

We also evaluate the approximation quality of RESCU compared to the full optimization model. As we have proven RESCU is NP-hard and thus we can compute an optimal solution only for very small settings. In this experiment we use datasets with 10 clusters. The cost and objects per cluster are randomly selected from $(0;1]$ and $[1;100]$, respectively. For each Δ value we generate at least 20 random datasets to calculate the optimal and approximative solution. Figure 8 gives the relative approximation quality compared to the optimal solution (in terms of cost) for different values of Δ. On average, the approximative solution shows only small differences to the optimal solution. The whiskers, corresponding to the 15% and 85% quantiles respectively, indicate that also rare cases yield good approximation qualities. The approximation quality is remarkably robust against the parameter Δ, the cluster gain threshold. Even for extremely rigid values close to zero, average approximation is close to optimal.

	Glass (214; 9)				*Vowel (990; 10)*				*Diabetes (768; 8)*			
	F1		*Accuracy*		*F1*		*Accuracy*		*F1*		*Accuracy*	
RESCU	60	100%	62	100%	44	100%	64	96%	71	100%	69	100%
INSCY	56	93%	54	87%	37	84%	67	100%	58	82%	65	94%
FIRES	30	50%	49	79%	10	23%	12	18%	33	46%	65	94%
SCHISM	45	75%	49	79%	24	55%	53	79%	69	97%	69	100%
PROCLUS	39	65%	54	87%	32	73%	30	45%	44	62%	65	94%
P3C	17	28%	39	63%	8	18%	16	24%	44	62%	65	94%
STATPC	19	32%	47	76%	17	39%	47	70%	39	55%	64	93%

	Shape (160; 17)				*Liver-Dis. (345; 6)*				*Breast (198; 33)*			
	F1		*Accuracy*		F1		*Accuracy*		*F1*		*Accuracy*	
RESCU	60	100%	75	100%	62	97%	61	98%	67	100%	76	97%
INSCY	56	93%	61	81%	62	97%	59	95%	65	97%	70	90%
FIRES	56	93%	62	83%	50	78%	53	85%	46	69%	75	96%
SCHISM	38	63%	59	79%	64	100%	58	94%	65	97%	71	91%
PROCLUS	60	100%	62	83%	46	72%	62	100%	47	70%	77	99%
P3C	39	65%	45	60%	36	56%	58	94%	63	94%	77	99%
STATPC	31	52%	62	83%	57	89%	58	94%	41	61%	78	100%

Figure 7. F1 and accuracy on real world data
[Captions: data set (size; dimensionality)]

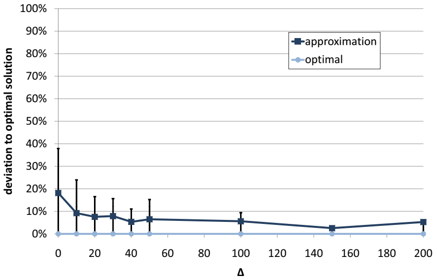


Figure 8. Approximation quality (Δ variation)

Next, we evaluate the flexibility of our RESCU model. For our cost function instantiation (cf. Sec. II-E), we vary the β parameter that controls interestingness as a trade-off between higher dimensional clusters and more objects per cluster. For the 32d pendigits data set, Figure 9 shows the median size (triangles) of the clusters and the number of high-D (≥ 12) clusters (squares) for the relevant clustering of a given β value. As we can see, low β values give strong preference to large and few clusters, whereas high values result in many clusters with less objects. Likewise, RESCU can be just as easily adapted using any other cost function that reflect the interestingness in a user's analysis.

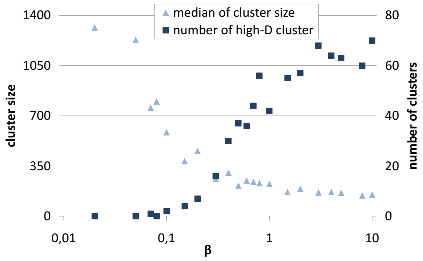


Figure 9. Effects on clustering by β variation

V. Conclusion

We introduce the *RESCU* (relevant subspace clustering) model for mining the most interesting non-redundant clusters in high dimensional data. Our novel model incorporates both non-redundancy and global interestingness via a new cluster gain definition. We prove that the computation of this model is NP-hard. For *RESCU*, we propose an approximative solution that shows high accuracy with respect to our relevance model. Thorough experiments demonstrate that *RESCU* reliably outperforms existing subspace and projected clustering algorithms while automatically reducing the output to all and only relevant clusters.

Acknowledgment

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).

References

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

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

[3] I. Assent, R. Krieger, E. Müller, and T. Seidl, "DUSC: Dimensionality unbiased subspace clustering," in *ICDM*, 2007, pp. 409–414.

[4] I. Assent, R. Krieger, E. Müller, and T. Seidl, "EDSC: Efficient density-based subspace clustering," in *CIKM*, 2008, pp. 1093–1102.

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

[6] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[7] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbors meaningful," in *IDBT*, 1999, pp. 217–235.

[8] B. Bringmann and A. Zimmermann, "The chosen few: On identifying valuable patterns," in *ICDM*, 2007, pp. 63–72.

[9] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

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

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP- Completeness*. San Francisco: W. H. Freeman, 1979.

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

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

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

[15] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos, "LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures," in *VLDB*, 2006, pp. 882–893.

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

[17] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *TKDD*, vol. 3, no. 1, 2009.

[18] 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*, 2008, pp. 533–541.

[19] G. Moise, J. Sander, and M. Ester, "P3C: A robust projected clustering algorithm." in *ICDM*, 2006, pp. 414–425.

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

[21] E. Müller, S. Günnemann, I. Assent, and T. Seidl, "Evaluating clustering in subspace projections of high dimensional data," *PVLDB*, vol. 2, no. 1, pp. 1270–1281, 2009.

[22] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *SIGKDD Explorations*, vol. 6, no. 1, pp. 90–105, 2004.

[23] K. Sequeira and M. Zaki, "SCHISM: A new approach for interesting subspace mining," in *ICDM*, 2004, pp. 186–193.

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