Projected Clustering for Huge Data Sets in MapReduce

Sergej Fries
Department Computer Science 9
RWTH Aachen University
D-52056 Aachen, Germany
fries@cs.rwth-aachen.de

Stephan Wels
Department Computer Science 9
RWTH Aachen University
D-52056 Aachen, Germany
stephan.wels@rwth-aachen.de

Thomas Seidl
Department Computer Science 9
RWTH Aachen University
D-52056 Aachen, Germany
seidl@cs.rwth-aachen.de

ABSTRACT
Fast growing data sets with a very high number of attributes become a common situation in social, industry and scientific areas. A meaningful analysis of these data sets requires sophisticated data mining techniques as projected clustering that are able to deal with such complex data.
In this work, we investigate solutions for extending the state-of-the-art projected clustering algorithm P3C for large data sets in high-dimensional spaces. We show that the original model of the P3C algorithm is not suitable to deal with huge data sets. Therefore, we propose the necessary changes of the underlying clustering model and then present an efficient MapReduce-based implementation - our novel P3C$^+$-MR algorithm. The effectiveness of the proposed changes on large data sets and the efficiency of the P3C$^+$-MR algorithm are comprehensively evaluated on synthetic and real-world data sets. Additionally, we propose the P3C$^+$-MR-Light algorithm, a simplified version of P3C$^+$-MR that shows extraordinary good results in terms of runtime and result quality on large data sets. In the end, we compare our solutions to existing approaches.

Categories and Subject Descriptors
D.1.3 [**Concurrent Programming**]: MapReduce; H.2.8 [**Database Application**]: Data Mining; I.5.3 [**Clustering**]: Projected Clustering

Keywords
MapReduce, Projected Clustering, Huge data

1. INTRODUCTION
Unsupervised techniques, also called clustering techniques, are an important part in the area of data mining. The task of clustering is the grouping of objects by their similarity to each other, such that similar objects are located in the same group (cluster) and dissimilar objects are located in different clusters. Clustering techniques are used in several areas of science, industry and society and play an important role for data preprocessing, exploration and hypotheses generation. In the last decades a large number of clustering approaches were developed. Expectation Maximization (EM) algorithm, DBSCAN and k-means algorithms are only a very small subset of existing approaches that are used in different applications. Most of the traditional approaches consider all attributes of the data for determining an appropriate grouping of objects. However, as recent developments show, clustering becomes less and less meaningful with a growing number of dimensions due to the so called *curse of dimensionality*. The curse of dimensionality states that with a growing number of dimensions the distances between objects become more and more alike, such that no meaningful grouping is possible anymore. Additionally, in data sets with a large number of dimensions the problem of irrelevant or noise dimensions occurs. Noise dimensions lead to an unwanted scatter of the data points and hide the real cluster patterns of the data. Different approaches for this problem have been investigated among other things *projected* and *subspace* clustering algorithms. There, one assumes that only a subset of dimensions is relevant for a cluster and the remaining dimensions are noise. The subspace clustering approaches further allow points to be in multiple clusters at the same time, while projected clustering approaches require a unique assignment of points to clusters. The task of subspace and projected clustering algorithms is to identify the clusters and their subspaces. The interested reader will find a survey in [1].

Both, projected and subspace clustering are sophisticated data analysis techniques for high dimensional spaces but they are often connected with high computational and I/O costs, since the search in all subspaces has an exponential time complexity in the number of dimensions and often requires large number of iterations over all data. This reason hinders the development of efficient algorithms for large data sets. A broad analysis of the existing algorithms, however, showed that the P3C algorithm [2] possesses a parallelization-friendly structure and is well suitable for processing large data volumes. Therefore, in this work we present P3C$^+$-MR, the first exact solution for MapReduce-based projected clustering. To the best of our knowledge, at this point of time, there are no other exact scalable implementations for projected clustering techniques and our approach is the first solution of this kind. We will also point out different problems, which occur when trying to perform P3C on huge data sets and present our solution - the P3C$^+$-MR-Light algorithm.

The main contributions of our work are:

1. Adaptation of the P3C clustering model for large data sets that results in our novel P3C$^+$ algorithm

2. Two efficient MapReduce-based implementations P3C$^+$-MR

and P3C$^+$-MR-Light of P3C$^+$

3. Evaluation of proposed solution and comparison to an existing approximate sampling-based solution [3]

Following sections are organized as follows: in Section 2 we give a short review of some well-known projected clustering algorithms, explain our decision to choose P3C and briefly introduce the MapReduce framework. In the following Sections 3 and 4 we describe the P3C algorithm in details, explain the statistical and algorithmic issues that make P3C not effective for large data sets and propose the necessary changes of the clustering model to deal with this problem that results in our P3C$^+$ algorithm. In Sections 5 and 6, we present P3C$^+$-MR and P3C$^+$-MR-Light, the MapReduce implementations of the P3C$^+$ and provide experimental evaluation in Section 7. Finally, Section 8 concludes this work.

2. RELATED WORK

In the past decade, a large number of projected clustering algorithms were proposed. Similar to full space clustering approaches these techniques are based on different assumption about the data that lead to different clustering models.
PROCLUS[4] algorithm is a k-medoid-like approach that searches for such subspaces of clusters, which minimize the dimension-wise standard deviations of distances of points in the neighborhood of the medoids. Similar to k-medoid, the algorithm starts with a set of k medoids and iteratively improves the result by exchanging low-quality medoids by better ones as long as the quality of the clustering is improving.
DOC[5] is a density based approach that defines a cluster as a set of dense points in a hyperrectangle. Intuitively, an optimal projected cluster in DOC is a cluster with as many objects as possible and as large dimensionality as possible. With this goal in mind, DOC performs Monte Carlo simulations for calculating an approximate solution that is shown to be a 2-approximate solution.
While k-means-based approaches are relatively easily expressible as a MapReduce program, density based techniques that consider the object neighborhood are in general much more complicated. They require similarity join techniques as MR-SimJoin [6] or MR-DSJ [7] with high data replication ratios and high communication overhead.
Approaches including STATPC[8] or P3C[2] are a further class of clustering techniques that rely on statistical interpretation of the clusters. As well in P3C as in STATPC a cluster is a set of points and dimensions in a hyperrectangle that have a significantly larger support (number of objects) than a comparable hyperrectangle where the points follow the uniform distribution. Both approaches employ algorithms that provide an approximate solution of the desired clustering.
An own class of approaches are generative models. Those, usually parametric methods, consider a clustering to be generated by a stochastic process and the main task is to learn the parameters of the underlying distributions. The learning procedure for non-trivial models by means of variational Bayes or other optimization techniques often requires a large number of iterations over all data and therefore produce a high I/O workload.

An important practical issue for clustering algorithms is the number of parameters and how intuitive those are. The PROCLUS algorithm requires two parameters k - the number of clusters and l - the average number of dimensions. The DOC algorithm relies on two user-defined parameters α and β that describe the relative proportions of objects in a cluster C in order to define C as optimal. Both P3C and STATPC rely on statistical tests and require one or three confidence levels - a Poisson threshold or α_0, α_K, α_H, respectively. In DOC, P3C and STATPC the number of clusters is determined automatically based on the provided parameter values.

Despite the significant growth of data volume and data dimensionality, there are almost no projected or subspace clustering algorithms that are able to deal with large amounts of data.
pMAFIA[9] is a parallel version of the MAFIA subspace clustering algorithm, which however is not applicable for projected clustering. To our best knowledge BoW [3] is the only work that addresses the problem of efficient projected clustering for large data sets in MapReduce. BoW is a general framework for parallelizing clustering algorithms, whose result sets are defined by means of hyperrectangles. The idea of the algorithm is to distribute the calculations to multiple machines by splitting the data into small subsets and executing the desired algorithm on these data blocks. In the end phase the partial results of all subsets are combined by merging intersecting hyperrectangles to larger hyperrectangles. The authors propose different strategies as well for sampling as also for the calculation part, which can either reduce the number of computations or reduce the I/O overhead. However, in both cases BoW remains an approximate algorithm and as we will show in the experimental section the quality of the algorithm may become very low.

MapReduce: MapReduce is an established, error-tolerant framework for parallel computing and a programming paradigm developed by Google Corp. in 2004. Its open-source implementation Hadoop found a wide spread in the industry and is successfully used at many internet based world player companies including Facebook, Twitter and Yahoo. Inspired by functional programming, a MapReduce program is composed of two functions 'map' and 'reduce' that are executed consecutively. Usually, the map function is responsible for reading the data in form of (key,value)-pairs from a storage system (in Hadoop this is the HDFS) and their preprocessing. The input pairs are in general arbitrarily distributed on different computing nodes such that each record is processed separately, independent from the other data records. The output of the map-phase are intermediate (key,value) pairs that are sorted according to their keys and pairs with equal keys are sent to the same computing node, a *'Reducer'*. Each reducer executes a 'reduce' function on incoming data and usually aggregates the results and that way combines the information of different records with equal keys. The generated records of the reduce phase are then stored on the storage system. An introduction to MapReduce is given in [10].

The reason for our choice of P3C algorithm is the sound statistical model, algorithm structure that allows for an efficient MapReduce-based solution, good quality shown in the evaluation of different projected and subspace clustering algorithms [11], and as stated in the original work [2] simple and stable parameter setting.

3. ORIGINAL P3C ALGORITHM

In this section, we provide necessary definitions and describe the original P3C algorithm. Then, in the following Section 4, we will identify the problems of this solution in the large data scenario and present our novel P3C$^+$ algorithm.

3.1 Definitions and notation

For the sake of self-containment, we introduce notation and definitions, which are based on the original P3C definitions. Let $D = \{x_{ij} | i \in [1, n], j \in [1, d]\}$ be a d-dimensional set of points of cardinality n and $A = \{a_1, \dots, a_d\}$ be a set of attributes of the data set D. Similar to original work, we assume without loss of generality each attribute to be normalized on the [0,1] range.

DEFINITION 1 (INTERVAL, SUPPORT, SUPPORT SET). *An* interval *on the attribute a_j in the range $i_l \leq x \leq i_u$ is denoted by $I_{a_j} = [i_l, i_u]$ and the width of this interval by $width(I) = i_u - i_l$. The* support set *of I, denoted by $SuppSet(I_{a_j})$, represents the set of objects x_{ij} located in the interval I_{a_j} in dimension a_j, i.e., $\{x_{ij} \in D | i_l \leq x_{ij} \leq i_u \land j = a_j\}$. The* support *$Supp(I_{a_j})$ of I_{a_j} is the cardinality of its support set $SuppSet(I_{a_j})$.*

DEFINITION 2 (P-SIGNATURE). *A* p-signature *$\boldsymbol{S}$ is a set of p intervals $\boldsymbol{S} = \{I_{a_{i_1}}, \ldots, I_{a_{i_p}}\}$ with disjunct attributes $a_{i_j} \in [1, d]$. The* support set *of $\boldsymbol{S}$ is the intersection of the support sets of the containing intervals, i.e., $SuppSet(\boldsymbol{S}) = \{x \in D | x \in \bigcap_{I \in \boldsymbol{S}} I\}$. The support of $\boldsymbol{S}$ is the cardinality of $SuppSet(\boldsymbol{S})$. The set of attributes of a p-signature $\boldsymbol{S}$ is denoted by $Attr(\boldsymbol{S})$.*

DEFINITION 3 (PROJECTED CLUSTER). *A* projected cluster *is a tuple $C = (X_i, Y_i)$ consisting of a set of points X_i and a set of* relevant *attributes Y_i. An attribute is called relevant if it follows a non-uniform distribution.*

DEFINITION 4 (TRUE P-SIGNATURE). *A* true p-signature *U of a projected cluster $C = (X_i, Y_i)$ is a p-signature $\boldsymbol{S}$ that consists of smallest intervals $I_{a_{j_k}}, j_k \in Y_i$ which contain all points X_i in the attributes Y_i.*

With these definitions we proceed with description of the P3C algorithm.

3.2 P3C algorithm description

In this section, we briefly introduce the clustering model and the basic algorithmic design of the P3C algorithm.

3.2.1 Clustering model

Let $\hat{\mathcal{C}} = \{C_1, \ldots, C_k\}$ be the set of hidden projected clusters in the data set D. The desired clustering result is the set of true signatures $\hat{\mathcal{U}}$ with:

$$\forall_{C \in \hat{\mathcal{C}}} \exists!_{U \in \hat{\mathcal{U}}} : U \text{ is the true signature of C}$$
$$\land \forall_{U \in \hat{\mathcal{U}}} \exists!_{C \in \hat{\mathcal{C}}} : U \text{ is the true signature of C}$$

where $\exists!$ stands for 'exists exactly one'.
The P3C algorithm approximates $\hat{\mathcal{U}}$ by generating and refining a set of so-called *cluster cores* (Definition 5).

We summarize all used symbols of this work in Table 1.

3.2.2 Algorithmic design

Given the clustering model, we proceed with the algorithmic design of the P3C algorithm and describe how the desired solution is computed.
In order to determine the set of true signatures $\hat{\mathcal{U}}$, the P3C algorithm employs a filter and refinement strategy. It first computes a set of cluster cores $\hat{K}$ and refines them in the second clustering step. To determine $\hat{K}$, each attribute is discretized (*histogram building* step) and on each non-uniformly distributed attribute all relevant intervals ($\hat{I}$) are extracted. An interval is relevant, if it has a significantly high support. To select the relevant attributes, the P3C algorithm first applies the standard χ^2 test and determines the non-uniform attributes. Then, the bin with highest support is marked as relevant and removed from the histogram. The procedure is repeated as long as the remaining not marked bins are not uniformly distributed. That way, the set $\hat{I}$ of all potentially interesting intervals is computed.

D	data base	a	an attribute
d	data dimensionality	n	database size
I	interval	$\hat{I}$	set of intervals
S	p-signature	$\hat{\mathbf{S}}$	set of p-signatures
K	cluster core	$\hat{K}$	set of cluster cores
U	true p-signature	$\hat{\mathcal{U}}$	set of true p-signatures
C	hidden cluster	$\hat{\mathcal{C}}$	set of hidden clusters
Cl	found cluster	$\hat{\mathbf{Cl}}$	set of found clusters

Table 1: Used notation in this work

After merging adjacent marked bins, the generation of *cluster cores* begins. For that, the intervals of all attributes are iteratively combined to high dimensional p-signatures. In each iteration every created p-signature $\mathbf{S}_i$ is extended by an interval of an additional attribute such that its dimensionality grows by one. In order to decide whether the resulting (p+1)-signature $\mathbf{S}_{i+1}$ is a projection of a hidden cluster, it is checked whether the support of $\mathbf{S}_{i+1}$ is significantly larger than the expected support, formally

$$\forall_{I \in \mathbf{S}_{i+1} = \{I_1, .., I_{i+1}\}} Supp_{exp}(\mathbf{S}_{i+1} \setminus \{I\}, I) <_p Supp(\mathbf{S}_{i+1}) \tag{1}$$

where $x <_p y$ means that y is significantly larger than x according to the Poisson test. For this task, P3C employs the standard Poisson statistics. Only p-signatures that satisfy this test are extended by further attributes in the following iterations.
The expected support of a Signature $\mathbf{S} \cup \{I_a\}$ is calculated under the assumption that the support set of **S** is uniformly distributed on the attribute a as

$$Supp_{exp}(\mathbf{S}, I_a) = Supp(\mathbf{S}) \cdot width(I_a). \tag{2}$$

DEFINITION 5 (CLUSTER CORE). *Let K be a p-signature. K is a* cluster core *iff it contains (1.) only and (2.) all relevant intervals out of $\hat{I}$ that represent a hidden cluster, s.t.:*

1. *For any q-signature $\boldsymbol{Q} \subset K$, q in $\{1, .., p-1\}$ and any interval $I \in K \setminus \boldsymbol{Q}$, it holds that:*
$$Supp(\boldsymbol{Q} \cup \{I\}) >_p Supp_{exp}(\boldsymbol{Q}, I)$$
2. *For any interval $I \in \hat{I} \setminus K$, it holds that*
$$Supp(K \cup \{I\}) \not>_p Supp_{exp}(K, I)$$

The computed cluster cores $\hat{K}$ (i.e. the subset of generated p-signatures that apply to Definition 5) are considered to be approximations of projections of the real clusters $\hat{\mathcal{C}}$. To refine the cluster cores the *expectation maximization (EM) algorithm* is employed. The initial mean and covariance matrix parameters of the Gaussian components are determined from the calculated cluster cores. For each cluster core K_i, a Gaussian G_i is added to the initialization of the EM. By considering only dimensions that are relevant to at least one cluster core, the EM algorithm is executed in the lower dimensional sub space A_{rel}:

$$A_{rel} = \{a \in A | \exists_{K \in \hat{K}} : a \text{ is relevant for } K\} \quad (3)$$

Given a set of k cluster cores, the resulting Gaussian components $\tilde{G}_i$ of the EM algorithm are converted into a set of k projected clusters $\hat{\mathbf{Cl}} = \{\mathbf{Cl}_1, .., \mathbf{Cl}_k\}$:

$$\begin{aligned} &\mathbf{Cl}_i = (X_i, Y_i), \text{ with} \\ &x \in X_i \iff i = argmax_{i \in \{1..k\}}(p(x|\tilde{G}_i)) \\ &a \in Y_i \iff a \text{ relevant for } K_i \end{aligned}$$

Since the EM algorithm assigns all points, both true members and outliers, to the clusters, outlier detection has to be applied to remove the outliers. The P3C algorithm applies standard multivariate outlier detection techniques [12] (outlier detection step). The employed technique first computes for each member x of cluster $\mathbf{Cl}_i$ the Mahalanobis distance d_{Mah} in A_{rel} based on the mean and covariance matrix of G_i. The points for which d_{Mah} is larger than the critical value of the χ^2 distribution with $|A_{rel}|$ degrees of freedom at a confidence level of $\alpha = 0.001$ are then considered as outliers. An additional *attribute inspection* step is executed to find relevant attributes of clusters $\mathbf{Cl}_i$, which were missed in the cluster core generation step. Similar to the relevant interval detection step in the beginning, the additional relevant attributes for cluster $\mathbf{Cl}_i$ are determined by building a histogram for the members of $\mathbf{Cl}_i$ and finding attributes A_i that are not uniformly distributed. The correct intervals of the output signatures are determined in the last *interval tightening* step. For each projected cluster $\mathbf{Cl}_i$, a signature $\mathbf{S}_i^{output}$ is provided with:

$$\begin{aligned} \mathbf{S}_i^{output} = &\{I_a = (i_{l,a} i_{u,a}) | a \in A_i\} \\ &\wedge i_{l,a} = min_{x \in \mathbf{Cl}_i}(x_a) \wedge i_{u,a} = max_{x \in \mathbf{Cl}_i}(x_a) \end{aligned}$$

The final result of the P3C algorithm is the set of output signatures $\hat{\mathbf{S}}^{output} = \{\mathbf{S}_i^{output} | K_i \in \hat{K}\}$.

4. BOOSTING P3C MODEL FOR BIG DATA

In this section, we discuss the reason why the original P3C algorithm is not applicable in the large scale scenario. We will reason why the employed statistical significance test is not sufficient when dealing with huge data sets and propose a statistically well-founded extension. We will also introduce several algorithmic improvements that considerably increase the quality of the clustering results. We start with a discussion of statistical issues in Section 4.1 and continue with algorithmic improvements in Section 4.2.

4.1 Statistical issues when dealing with Big Data

The P3C algorithm makes use of different statistical techniques for determining the overall clustering result. In this section we discuss the statistical issues of Sturge's rule and the Poisson test in the cluster generation step for large data sets and introduce our solutions.

4.1.1 Sturge's Rule

To determine the optimal number of bins in every dimension the P3C algorithm uses the Sturge's rule

$$\text{number bins} = \lceil 1 + \log_2 n \rceil \, .$$

However, as was shown [13], Sturge's rule tends to oversmooth the histograms, i.e., it proposes a too small number of bins for a good approximation of the underlying distribution. Too smooth histograms may on one hand make the identification of relevant bins more difficult in the histogram building processing and at the same time make the overall clustering result more imprecise. Additionally, as stated in [14] Sturge's rule is not well-founded and produces similar results to well-founded approaches for small sample sizes only. Alternative heuristics including the Freedman-Diaconis rule provides much more reliable results and should be used instead. According to the Freedman-Diaconis rule, the bin size is determined by:

$$\textit{bin size} = 2IQR(x)n^{-\frac{1}{3}},$$

where n is the sample size and $IQR(x)$ is the interquartile range of the data x. Since the determination of the precise IQR value is a data and computationally intensive task, we assume in our work that every dimension follows the uniform data distribution such that $IQR(x) = \frac{1}{2}$. Despite this simplification, the utilization of Freedman-Diaconis rule leads to much more accurate approximations of the given data distributions for large data sets and therefore to more exact clustering results.

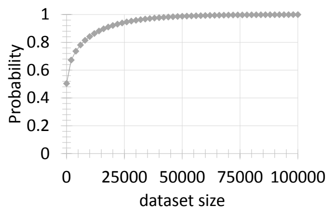


Figure 1: Probability to observe $101\% \cdot \mu$ objects in a hyper-rectangle for growing average size. For sufficiently large data sets each the probability is almost 100%.

4.1.2 Poisson test in cluster core generation step

The second issue that may lead to much more problematic cases is the employed hypothesis test for the determination of significantly large supports of constructed p-signatures in the cluster core generation step. By means of a Poisson distribution the actual supports of p-signatures are tested against the respective expected supports. The result of the computation is a p-value - a probability to observe the support as extreme as or more extreme than the actually observed support, if the test statistic really were distributed as it would be under the null hypothesis. However, this probability strongly depends on the database size under consideration. The p-values will monotonously decrease as the data set size grows with constant relative deviation in bin supports. To see this, remember that the Poisson distribution can be approximated by a Gaussian with $\mu = \lambda$ and $\sigma = \sqrt{\lambda}$. I.e., the standard deviation of the normal distribution grows with a square root of the average number of elements per bin. For a linearly growing average number of elements per bin, this means that the standard deviation grows much slower than the constant relative deviation and the Poisson test will consider even relatively small deviations from the average as significant as long as the data size is large enough. The simulations in Figure 1 support this theoretical result. For growing average number of elements μ the probability to observe at least $101\% \cdot \mu$ elements is depicted. As one can see, for sufficiently large data sets the probability becomes almost 100%. From the statistical hypothesis point of view, this result is expectable and desired. A growing number of points in the sample increases the power of the test, i.e, it increases the probability of the test to reject the null hypothesis, if the alternative hypothesis is true.

The problematic part in our context is that the Poisson test only determines the significance of a deviation but not the *strength of the effect* (A relative deviation of one percent might be significant, but not relevant for our cause). Therefore, the P3C algorithm will consider p-signatures with a relatively small support as relevant for further computation. And this on its hand results in a very large number of identified candidate clusters and in the end in a low quality of the clustering. We will show the described effects in the experimental section 7.

To tackle this problem we extend the cluster core generation step and employ a complementing class of test, called *effect size test* [15]. An effect size is a measure of the strength of a phenomenon that complements significance tests by providing the information how *large* an effect is. In the considered problem we define the strength of a phenomenon as the relative deviation of the actual support $Supp(\mathbf{S} \cup \{I_a\} = \mathbf{S}_{\cup I})$ of the p-signature $\mathbf{S}$ and interval I_a under consideration of the expected support (cf. Equation 2). The definition of the desired effect size corresponds to the *Cohen's d* statistics:

$$\text{Cohen's d}_{cc} = \frac{Supp(\mathbf{S}_{\cup I}) - Supp_{exp}(\mathbf{S}_{\cup I})}{\sigma}, \qquad (4)$$

where σ is the variance of expected number of objects such that the Cohen's d$_{cc}$ calculates the 'amount of variation' from $Supp(\mathbf{S}_{\cup I})$ to $Supp_{exp}(\mathbf{S}_{\cup I})$ ('cc' stands for cluster core). By defining the variance parameter $\sigma = Supp_{exp}(\mathbf{S}_{\cup I})$, the resulting value is the desired relative deviation.
We introduce a new parameter $\theta_{cc} > 0$ that controls the desired strength of the effect. Consequently, a p-signature is only interesting, if $Supp(\mathbf{S}_{\cup I})$ is significantly larger than $Supp_{exp}(\mathbf{S}_{\cup I})$ and the effect size is larger than or equal to the specified threshold: $\theta_{cc} \leq$ Cohen's d$_{cc}$.

In fact, the χ^2 possess similar properties as the Poisson test. For larger data sets the power of the test grows and the bins with a relative small deviation from the average bin support (small effect) will nevertheless be considered as relevant. Assuming a uniform distributed noise (as the P3C clustering model does), this, however, should not influence the clustering result in a negative way. For data sets, in which the above assumption does not hold and the noise shows a non-uniform distribution, a possible solution to tackle the problem is the employment of the complementary effect size tests.

4.2 Improvement of the algorithmic design

In this section, we introduce the algorithmic changes of the original P3C algorithm that result in our novel P3C$^+$ algorithm. These algorithmic changes aim at improving the overall clustering quality and are motivated by problems that we discovered during our experimental evaluation of the original P3C algorithm on large data sets.

4.2.1 *Cluster Core Redundancy*

We extend Definition 5 of cluster cores by a *redundancy constraint*. A signature is *redundant*, if it describes only an intersection of hidden clusters $\hat{\mathcal{C}}_{sub} \subseteq \hat{\mathcal{C}}$ in a subspace of the united relevant intervals of $\hat{\mathcal{C}}_{sub}$. Such a redundant signature provides misleading information (presence of a cluster that doesn't exist) and should thus be removed from the set of cluster cores.

Figure 2 illustrates an example of a redundant signature. C_1 and C_2 consist of 50 data points each, where C_1 is clustered in the $\{a_1, a_3\}$ subspace and C_2 in $\{a_1, a_2\}$. Given $\hat{I} = \{I_1, .., I_4\}$ and $width(I_i) = 0.1, i \in \{1..4\}$, the cluster core generation step finds

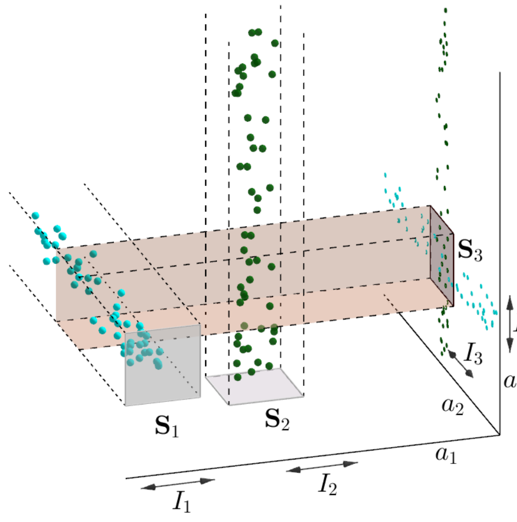


Figure 2: A data set, where the intersecting region of two hidden clusters C_1 and C_2 causes an additional (redundant) signature in the $\{a_2, a_3\}$ subspace.

the 3 depicted 2-signatures. Obviously, $\mathbf{S}_1$ and $\mathbf{S}_2$ show enough support to pass the Poisson test with

$$Supp(\mathbf{S}_i) = 50 >_p 1 = 100{\cdot}0.1{\cdot}0.1 = Supp_{exp}(\mathbf{S}_i), i \in \{1, 2\},$$

Assuming that both clusters are uniformly distributed in their respective irrelevant dimension, the support of $\mathbf{S}_3$ becomes $50 * 0.1 + 50 * 0.1 = 10$. On a confidence level $\alpha = 10^{-6}$, it holds that $1 <_p 10$, such that $\mathbf{S}_3$ also passes the Poisson test.
Obviously, signature $\mathbf{S}_3$ is redundant to $\mathbf{S}_1$ and $\mathbf{S}_2$ and should thus be removed from the set of cluster cores.

In order to identify redundant signatures, we exploit the lower ratio of $\frac{Supp}{Supp_{exp}}$ of a redundant signature, compared to the intersecting signatures it originates from. We define redundancy of a signature $\mathbf{S}$, given a set of signatures $\hat{\mathbf{S}}$ as follows:

$$\mathbf{S} \text{ redundant in } \hat{\mathbf{S}} \iff \mathbf{S} \subseteq \cup_{\mathbf{S}_i \in \hat{\mathbf{S}} \wedge \mathbf{S}_i >_r \mathbf{S}} \mathbf{S}_i, \qquad (5)$$

where the intuitive meaning of $\mathbf{S}_1 >_r \mathbf{S}_2$ is that observing $\mathbf{S}_1$ is more interesting than $\mathbf{S}_2$:

$$\mathbf{S}_1 >_r \mathbf{S}_2 \iff \frac{Supp(\mathbf{S}_1)}{Supp_{exp}(\mathbf{S}_1)} > \frac{Supp(\mathbf{S}_2)}{Supp_{exp}(\mathbf{S}_2)}. \qquad (6)$$

The expected support of a p-signature is calculated based on the assumption that the data is distributed uniformly on each attribute as:

$$Supp_{exp}(\mathbf{S} = \{I_1, .., I_p\}) = n \prod_{I_i \in \mathbf{S}} width(I_i). \qquad (7)$$

When we sort the 3 signatures found depicted in Figure 2, we get $\mathbf{S}_3 <_r \mathbf{S}_1$ and $\mathbf{S}_3 <_r \mathbf{S}_2$. With $\mathbf{S}_3 \subseteq (\mathbf{S}_1 \cup \mathbf{S}_2)$, the redundancy criterion 5 is met, such that the redundant signature $\mathbf{S}_3$ is successfully identified and deleted from the set of cluster cores.

4.2.2 *Outlier Detection*

The outlier detection step of the P3C algorithm removes objects from obtained clusters after the EM clustering step. Outlier detection is essential for determining tight borders of the computed clusters since every outlier objects extends the borders.
Similar to the original P3C algorithms, we employ standard multivariate outlier detection techniques [12] with some modifications described later on. In order to determine if a point x of a cluster **Cl** is an outlier or not, the Mahalanobis distance of x to the cluster mean $\mu_{\mathbf{Cl}}$ is compared to a critical value d_{crit} of the χ^2 distribution (for $\alpha = 0.001$) with $|A_{rel}|$ degrees of freedom. All objects with distances above d_{crit} are marked as outliers.

The Mahalanobis distance is based on two parameters μ (mean) and Σ (covariance matrix) that are estimated from the cluster members. The *naive* approach computes both parameters from all points of a cluster **Cl**, but as stated in [12] it suffers from the masking effect, i.e., the outlier objects itself influence the covariance and means and that way get masked. More robust *minimum volume ellipsoid estimator (MVE)* considers only points in a minimum volume ellipsoid covering half of the points of a cluster and that way improve the detection quality.
In this work we investigate both approaches, the naive and the MVE based outlier detectors, on huge data sets. However, due to computational complexity of calculating the exact MVE parameter estimators, we employ an approximate MVB (minimum volume ball) solution, i.e., instead of considering half of the points of a cluster in a minimum volume ellipsoid, we compute a minimal volume ball containing half of the points. The mean μ_{MVB} of the MVB then equals to dimension-wise medians of points of a cluster and the radius r_{MVB} equals to the median of the distances of all cluster points to μ_{MVB}.

4.2.3 Attribute Inspection

The attribute inspection is responsible for selecting relevant cluster attributes after the outlier detection step. This step corresponds to interval detection step in the original P3C with the difference that it is executed on points of a found cluster **Cl** without outlier objects and not on all points of the data set. I.e., for the members of a cluster the histograms are generated and relevant intervals identified. The original algorithms stops at this point and accepts all found relevant intervals. This is inconsistent with the cluster core generation approach, where relevant intervals are tested using the Poisson test. Our experiments showed that the additional interval proving step improves the overall clustering result. For each additional suggested interval I_{new} we perform the test described in Equation 1 Therefore, our P3C$^+$ is extended by an additional cluster support test in the cluster core detection step to which we refer as to *AI proving*.

5. P3C$^+$-MR IN MAPREDUCE

In this section, we provide a description of MapReduce jobs needed to implement the P3C$^+$-MR algorithm. Most parts of the algorithm are equal to a summation of statistics, which are calculated for each data object independently. This property of P3C$^+$-MR makes it fit naturally into the data parallel MapReduce framework. The desired statistic s is calculated for each object in the mapper phase and summarized in the reduce phase. We use the following shortcut notation:

$$s = \sum_{x_i} s(x_i) = \sum_{S \in Splits}^{Reduce} \sum_{x_i \in S}^{Map} s(x_i)$$

Since the calculations take about the same time for each object, perfect load balancing is achieved naturally.

Due to simplicity of some steps, we will include the implementation of non-trivial MapReduce jobs and only give a short textual description including the respective summation formula for remaining jobs.

5.1 Histogram building

In this work, we assume normalized data space in the range $[0, 1]$. Then the computing of histograms with known number of bins m is trivial. In the mapper phase the histograms for given data subset $Split$ are calculated and a single reducer combines the partial results to the overall histogram. The summation form for the support of bin i on dimension d is given by:

$$Supp(bin_{d,i}) = \sum_{x_i} \begin{cases} 1, & max(1, \lceil m \cdot x_{i,d} \rceil) = i \\ 0, & else \end{cases} \tag{8}$$

5.2 Relevant intervals

The determination of relevant intervals is a computationally cheap process. For a histogram with k bins and d dimensions, a simple χ^2 statistic is calculated at most $d \cdot k$ times. Even for relatively large k and d values this step is cheap in comparison to other steps, therefore its parallelization will not result in a considerable speedup.

5.3 Cluster core generation

The cluster core generation, as depicted in Algorithm 1, involves two computationally costly steps, namely *candidate generation* (line 6) and *candidate proving* (lines 3,7). In the candidate generation step, a set of p-signatures out of given intervals and already *proved* candidates is determined. A candidate is proved if its support significantly exceeds the expected support. The counting of the support and testing for significance is called *candidate proving.*

Algorithm 1 Cluster Core Generation

```
1: Input: Intervals $I_1, .., I_n$
2: $Cand_1 = \{\{I_i\} | i \in \{1..n\}\}$
3: $Proven_1$ = Prove Candidates$(Cand_1)$
4: $k = 2$
5: while $Proven_{k-1}$ is not empty do
6:     $Cand_k$ = A Priori Candidate Generation$(Proven_{k-1})$
7:     $Proven_k$ = Prove Candidates$(Cand_k)$
8:     $k$++
9: end while
10: $Proven = \bigcup_{i=1}^{k} Proven_i$
11: $Proven$ = Filter maximal Cluster Cores$(Proven)$
12: Output:Cluster Cores $\bigcup_{i=1}^{k} Proven_i$
```

Generating a candidate set $Cand_{p+1}$ of $(p+1)$-signatures out of a set of p-signatures is achieved by joining each two p-signatures, that have $p-1$ intervals in common. Having a set of k p-signatures, there are $c = \frac{k \cdot (k-1)}{2}$ candidate pairs for merging. Due to the quadratic complexity, the signature candidate generation consumes an unfeasible amount of time for large k, if run on a single processor. Therefore MapReduce is employed for the parallelization of large candidate set generation using $m = \lfloor \frac{c}{T_{gen}} \rfloor$ mappers and zero reducers, for $c > 2T_{gen}$. Since each MR job adds some overhead, small candidate sets are generated in a serial manner. The threshold T_{gen} should be chosen, such that the parallel version is equally fast to the serial version if $c = T_{gen}$ and depends on the available cluster. On our cluster $T_{gen} = 4 \cdot 10^7$ performed best.
The set of p-signatures is send to each mapper via the distributed cache. A set of $\frac{c}{m}$ indices is provided to each mapper, which indicates the candidate pairs to be processed. The indices range from

1 to c, each representing a pair of p-signatures. If a pair can be merged to a $(p+1)$-signature, the mapper writes the resulting signature to the result file. The main program collects the generated $(p+1)$-signature candidates while ignoring duplicates (two pairs of p-signatures might result in the same $(p+1)$-signature).

In order to reduce I/O overhead of MR jobs, candidates are not proven at each level. If the number of generated candidates on a level j is small, the candidates on level $j+1$ are generated based on $Cand_j$ instead of $Proven_j$ (cf. line 6). When the set of collected candidates c_{sum} of several levels exceeds a threshold T_c, all candidates are proven with a single MR job. We use the following heuristic to decide where to stop the collection of candidates on level j:

$$|Cand_j| = 0 \vee (c_{sum} > T_c \wedge |Cand_j| > |Cand_{j-1}|) \Rightarrow Stop.$$

Again, the optimal setting of T_c depends on the available cluster. On our cluster $T_c = 3 \cdot 10^4$ performed best. Continuing the candidate collection as the candidate set shrinks, has proven to be most efficient throughout all of our experiments.
The multi-level candidate generation increases the total number of candidates that have to be proven as a priori pruning becomes less effective. On the other hand, it saves I/O overhead of otherwise necessary additional MR jobs. Our heuristic aims for a good trade-off of these effects.

Proving a candidate p-signature involves the determination of its actual support. Therefore the support of each generated candidate must be calculated. This corresponds to a single MR job, where each mapper is provided all candidates $\hat{\mathbf{S}}_{all}$ and determines the set of signatures $\hat{\mathbf{S}}_{in}(x_i)$ for each data point x_i of its split, with $\hat{\mathbf{S}}_{in}(x_i) = \{\mathbf{S} \in \hat{\mathbf{S}}_{all} | x_i \in SuppSet(\mathbf{S})\}$. The reducers sums up the support counts of each cluster reported by the mappers.

To determine $\hat{\mathbf{S}}_{in}(x_i)$, each signature in $\hat{\mathbf{S}}_{all}$ is queried for containment of x_i. Since a total of 10^5 and more candidates is common, counting the support needs to be more efficient. To this end, we introduce the *Rapid Signature Support Counter* (RSSC).

The RSSC uses a bitmap representation of the problem, which is defined as follows. Each signature $\mathbf{S}_j$ is given a temporary $id_j \in \{0, .., |\hat{\mathbf{S}}_{all}| - 1\}$. A bit vector b of length $|\hat{\mathbf{S}}_{all}|$ represents a subset $\hat{\mathbf{S}}_{sub} \subseteq \hat{\mathbf{S}}_{all}$ with:

$$\mathbf{S}_j \in \hat{\mathbf{S}}_{sub} \iff b_{id_j} = 1$$

For each attribute $a \in A_{rel}$ (cf. Equation 3), we define a binning B_a. We select the bins, such that for each bin $b_{a,i}$ a bit vector $v^{a,b}$ can be given, such that for a given data point x_i the following holds

$$\forall_{\mathbf{S}_j = \{..,I_{j,a},..\} \in \hat{\mathbf{S}}_{all}} v^{a,b_i}_{id_j} = 0 \iff x_i \notin SuppSet(I_{j,a}),$$

where b_i denotes the bin in which x_i belongs on attribute a. Such a binning can be derived for attribute a by taking all upper and lower bounds of provided intervals on a as separators. Figure 3 depicts an example for such a binning with $|\hat{\mathbf{S}}_{all}| = 4$.
The desired set $\hat{\mathbf{S}}_{in}(x_i)$ is then given by the bit vector b^{in_i} with:

$$b^{in_i} = \bigwedge_{a \in A_{rel}} v^{a,b_i}$$

Having the RSSC bit masks calculated, mappers only have to bin each point in each $a \in A_{rel}$ according to B_a and aggregate the selected bit masks with the logical AND operator, both of which are very fast calculations. This way, $\hat{\mathbf{S}}_{in}(x_i)$ is calculated far more

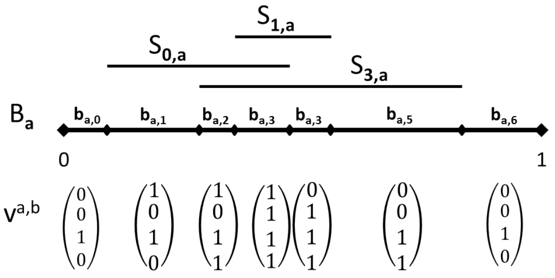


Figure 3: A binning B_a with bit vectors $v^{a,b}$ for each bin on attribute a with four signatures. Since a is not relevant for S_2, the corresponding bit vector entries are set to 1

efficiently compared to the simple approach.
The bit masks are calculated by the main program beforehand and then passed to the mappers via distributed cache. Since the RSSC bit masks can be calculated with only two scans of $\hat{\mathbf{S}}_{all}$, the overhead is negligible.

Filtering cluster cores out of $Proven$ (cf. line 11 in Algorithm 1) is fast even for large $Proven$ sets, such that it's not necessary to parallize this task.

5.4 EM

In order to initialize the EM algorithm sample means μ_C and covariances Σ_C of cluster cores have to be calculated twice. In the first iteration, μ_C and Σ_C are calculated using only the support sets of the cluster cores in order to assign outliers using the Mahalanobis distance. In the second iteration, μ_C and Σ_C are calculated using the support sets of cluster cores plus the assigned outliers.
Sample means and covariances can be calculated by two MR jobs. The first MR job calculates the linear sum l_C of the members and linear and squared sum w_C and w_{C^2} of the member weights for each cluster C:

$$l_C = \sum_{x_i} w_{C,i} x_i, \quad w_C = \sum_{x_i} w_{C,i}, \quad w_{C^2} = \sum_{x_i} w_{C,i}^2,$$

with $w_{C,i}$ being the weight of object i in cluster C. The results of the first MR job are used to calculate the sample mean $\mu_c = \frac{l_C}{w_C}$ for each cluster C and passed to the second MR job, which calculates the sample covariances σ_C of cluster C as:

$$\Sigma_C = \frac{w_C}{w_C^2 - w_{C^2}} \sum_{x_i} w_{C,i}(x_i - \mu_C)(x_i - \mu_C)^T$$

The statistics Σ_C and μ_C which include the outliers serve as the initial Gaussian mixture model for EM. Each EM step needs another two MR jobs [16], where the first job calculates new estimates $\tilde{\mu}_C$ and $\tilde{w}_C$, while the second job provides new covariance estimates $\tilde{\Sigma}_C$.

5.5 Outlier detection

For a cluster C and a member x of C, the Mahalanobis distance between x and C is compared to the critical value of a χ^2_n distribution (cf. Section 4.2.2). The *naive* version uses the means and covariances of the clusters as provided by the EM algorithm. Thus, a single MR job with map phase only is necessary, which is called the *OD* job. Each mapper of the OD job computes for each point x the cluster C it belongs to according to the Gaussian mixture distribution provided by the EM algorithm and the Mahalanobis distance between x and C. Finally, x is written back to the result file augmented with an additional membership attribute, which is set to the id of C or -1, if it is considered an outlier.

The *MVB* version requires mean $\hat{\mu}_C$ and covariance $\hat{\Sigma}_C$ of the minimum volume ball MVB_C for each cluster C as explained in Section 4.2.2. These statistics can be extracted within three MR jobs. The first of which calculates mean and radius of each MVB_C. The following two jobs compute $\hat{\mu}_C$ and $\hat{\Sigma}_C$ for each cluster C as in the EM initialization step, but take only the points in MVB_C into account.
Mean m_C and radius r_C for each MVB_C are approximated with a single MR job, where each mapper j calculates m_C^j and r_C^j for its split and the reducer aggregates the statistics by taking the dimension-wise median of the means and radii provided by the mappers.
We define the dimension-wise median $Mdd(X)$ of a set of d-dimensional vectors X as

$$Mdd_j(X = \{x_1, ...x_n\}) = Md(x_{1,j}, .., x_{n,j}), \forall j \in \{1, .., d\},$$

where Md is the sample median.

In order to compute m_C^j and r_C^j, mapper j caches the set of all data points X_{split} of the current split. In the clean up phase of the mapper, X_{split} is sorted once w.r.t. to each dimension to calculate $m_C^j = Mdd(X_{split})$. Next, the Euclidean distance between each point in X_{split} and m_C^j is calculated and stored in an array G. The radius of MVB_C^j is then given by $r_C^j = Md(G)$. In the end, the mapper writes m_C^j and r_C^j for each cluster C to the output.
After calculating MVB' means and covariances, the OD job is run using $\hat{\mu}_C$ and $\hat{\Sigma}_C$ instead of μ_C and Σ_C.

5.6 Attribute Inspection

In this step, one MR job is needed to calculate a histogram for each cluster core. To this end, Equation 8 is slightly adjusted for this task to

$$bin_{C,d,i} = \sum_{x_i \in C} \begin{cases} 1, & max(1, \lceil mx_{i,d} \rceil) = i \\ 0, & else \end{cases}$$

with $bin_{C,d,i}$ being the ith bin on dimension d on the histogram of cluster C.
When AI proving is used (cf. Section 4.2.3), another job has to be run, in order to compute the support for the augmented signatures. This is done exactly as in the cluster core generation step.

5.7 Interval Tightening

At this point, for each cluster, the support set and the set of relevant dimensions are determined. The interval bounds for the output signatures are calculated in a single MR job, where each mapper calculates the minimum (maximum) value within its split for the relevant dimensions of each cluster. The reducer aggregates the provided values by repeated extracting the minimum (maximum) for each dimension and cluster.

6. P3C$^+$-MR-Light ALGORITHM

During our experimental evaluation, we made an interesting observation that resulted in the P3C$^+$-MR-Light algorithm. Due to the proposed changes in our P3C$^+$ the generated cluster cores already provided an extremely good approximation of the hidden clusters. Moreover, the quality of the results even dropped if the following EM and outlier detection steps were executed. The main reason for this phenomenon is the fact that the outlier detection step is often not able to identify all outlier objects correctly such that the bounds of the resulting found clusters are 'blurred' in the interval tightening step.

Given the subsets of all relevant attributes A_{rel} (cf. Equation 3) and a cluster core K that approximates a hidden cluster C, two outlier data points x^- and x^+ that are likely to cause the blurring effect, have the following properties. Let A_C be the set of relevant attributes of C and $a_{blur} \in A_C$. The data points x^- and x^+ have to be assigned to C in EM phase, and should be close to the center μ_C of C in all attributes $a \in A_{rel}^b = A_{rel} \setminus \{a_{blur}\}$, e.g. $x_a = \mu_C, \forall a \in A_{rel}^b$. In order to blur the bounds of the interval $I_{a_{blur}}$ of K extremely, we assume $a_{a_{blur}}^- = 0$ and $a_{a_{blur}}^+ = 1$. Since x^- and x^+ perfectly match cluster C in all relevant dimensions but one, EM will output C as the most probable cluster for both points. The outlier detection algorithm will not be able to identify the outliers for the same reason. Finally, the interval tightening step will calculate the interval $I_{a_{blur}} = [0, 1]$ for K, resulting in a very poor approximation of C in attribute $I_{a_{blur}}$.
Obviously, the risk of having blurring data points, rises with growing data set size. We observed that the decreasing quality of clustering quality of P3C$^+$ with increasing data set size (cf. Section 7) is directly correlated to the blurring effect.
For that reason, we also propose the P3C$^+$-MR-Light algorithm that circumvents the blurring effect by avoiding a partitioning of the data set, as it is performed during the EM phase. P3C$^+$-MR-Light consists of all but the EM- and outlier detection steps of the P3C$^+$ algorithm.
The histogram building in the attribute inspection step (cf. Section 4.2.3) requires a mapping $m(x) : D \to \hat{K} \cup \{O\}$, that maps each data point to either a single cluster or to the set of outliers O. Since P3C$^+$-MR-Light doesn't have such a partition, a mapping $m'(x) : D \to 2^{\hat{K}}$ is defined as:

$$K \in m'(x) \iff x \in SuppSet(K).$$

The histogram for a cluster core K is calculated as:

$$bin_{K,d,i} = \sum_{x_i \in SuppSet(K) \wedge |m'(x_i)|=1} \begin{cases} 1, & max(1, \lceil mx_{i,d} \rceil) = i \\ 0, & else \end{cases}$$

In order to avoid finding attributes mistakenly relevant due to the redundancy problem discussed in Section 4.2.1, we exclude those data points from the histograms, that contribute to the support sets of more than a single cluster core.

In the following section we proceed with evaluation of the P3C$^+$-MR, P3C$^+$-MR-Light and the BoW algorithms.

7. EXPERIMENTS

In this section, we evaluate our novel P3C$^+$ approach and compare it with the original P3C algorithm in Section 7.4. Then, we proceed with evaluation of the MapReduce based solutions P3C$^+$-MR, P3C$^+$-MR-Light and BoW on large data sets in Section 7.5.

7.1 Data set description

For evaluation of P3C$^+$-MR and BoW algorithm we use several synthetic and real world data sets that are described in this section.

Synthetic data. To generate the synthetic data we vary three parameters: (1) number of clusters with different dimensionalities,

(2) data set size and (3) percentage of noise objects. The number of clusters is equal to 3, 5 or 7. The databases consists of 10^4, 10^5, 10^6, $5 \cdot 10^6$, 10^7, $5 \cdot 10^7$ and as extreme setting 10^9 (one billion!) objects. The percentage of noise is set to 0%, 5%, 10% and 20% of the database size. All data sets have 50 dimensions. The one billion data set is ≈ 0.2 TByte large.

Clusters are generated in a hyperrectangular shape. The interval width of a hidden cluster on a relevant attribute varies between 0.1 and 0.3. A cluster is distributed on each relevant interval following a Gaussian distribution with $\sigma = 1$. Data points are uniformly distributed on irrelevant attributes. The clusters have 2 to 10 dimensions and may overlap with others clusters on relevant attributes. In fact, each generated data set contains at least two clusters that overlap.

Real world data. For evaluation of P3C$^+$ algorithm we make use of the available data set 'Colon cancer' from the standard machine learning repository UCI[1]. It consists of 62 objects with 2000 dimensions and is annotated by information whether a person has cancer or not. This data set is used for comparison of our P3C$^+$ and the original P3C algorithms only.

7.2 Used measures

To evaluate our solution we employed E4SC, F1, RNIA and CE [11] evaluation measures. However, in this work we only report the results of the E4SC since it is able to detect the most important differences of a subspace/projected clustering results from the ground truth including cluster merges, wrong subspaces, wrong object assignment and many more.
A further reason for this restriction is that the remaining measures showed different drawbacks. F1, as a full space clustering measure, is not able to punish clusters in wrong subspaces and often reported too good quality for in reality bad clustering results. The CE measure showed itself as too sensitive in the case of cluster splits that often punished our competitors too much. Therefore we provide only the E4SC quality measure in this work and refer the interested reader to our web page[2] with remaining results.

7.3 Parameter settings

In all experiments the parameters were set to: $\alpha_{\chi^2} = 0.001$, $\alpha_{poi} = 0.01$ and the number of samples per reducer in the BoW variant was set to 100.000.

To determine an optimal value for the θ_{cc} parameter, we executed the P3C$^+$-MR algorithm on all data sets and varied θ_{cc} in the range $[0.05, 0.5]$. In the end, the θ_{cc} was calculated as the median of all optimal values over all databases and is equal to 0.35.

7.4 Evaluation of the P3C$^+$ model

In this section we evaluate our P3C$^+$ model and show how single steps improve the overall quality of the algorithm.

7.4.1 Outlier detection

In order to evaluate the quality of outlier detection techniques we evaluated the E4SC of clustering results on databases with 10k, 100k and 1Mio objects for all noise levels (0%, 5%, 10%, 20%) and 3, 5 and 7 number of clusters. The results of the evaluation are depicted in Figure 4. To keep the plots large enough we omit the plot for noise level 0%. But it shows the same behavior as the remaining plots. The dotted lines represent the 'naive' outlier detection technique based on mean and covariance computed from all points of a cluster and the solid lines depict the results of our MVB based outlier detector. We observe that except for a single case, (noise level 10%, 5 clusters) the MVB outlier detector leads to a considerable better clustering quality, i.e., the resulting clusters much better approximate the hidden clusters. This confirms our expectation that the MVB provides a more stable estimate for the real mean and covariance of a cluster and that way leads to a better outlier detection.

However, for the largest data set (1 million objects) we observe that the quality of all approaches decreases. The probable explanation is that a larger number of objects in a cluster lead to a worse estimate by the MVB and that way less outliers are identified. The exact MVE estimator will probably results in a better clustering quality but as was stated earlier the calculation of MVE is a computationally expensive step. Due to our focus on large data sets we therefore leave this point not evaluated. However, as last remark we observe that a sophisticated outlier detection step is crucial for good quality of the P3C and also P3C$^+$ algorithms.

7.4.2 Redundancy filter and effect size enhancement

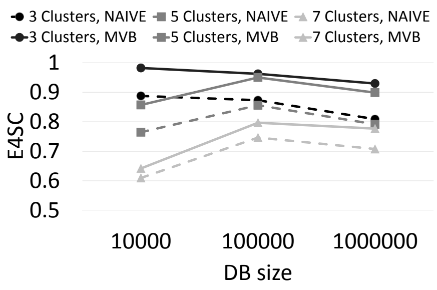
Figure 5 depicts the exemplary results obtained by redundancy filter described in the Section 5 for the synthetic data sets with 10k and 100k objects, 5 clusters and noise level of 20% percent. Each plot consists of three curves: 'Optimal' that provides the number of hidden clusters (5), 'Poisson' that shows the behavior of the original Poisson test of the original P3C algorithm and 'Combined', that is a combination of 'Poisson' and effect size test described in Section 4.1.2. The Figures 5(a) and 5(c) depict the number of generated cluster cores without redundancy filtering, while Figures 5(b) and 5(d) show the effect of redundancy filter.

The first observation from Figures Figures 5(a) and 5(c) is that the original 'Poisson' filter always significantly overestimate the number of cluster cores in the data set for large threshold values. This effect is expectable since a larger threshold means that a smaller deviation of the cluster core support from the expected support is sufficient to consider the deviation as significant. Moreover, we observe that for the larger data sets (Figure 5(c)) the overestimation begins at a smaller threshold value ($\approx 10^{-60}$) while in the 10k data set this effect occurs at $\approx 10^{-10}$. As explained in Section 4.1.2 this effect stems from the growing power of the hypothesis test. Our proposed solution to use effect size (the 'Combined' curves) considerably improves the overall result. For investigated threshold values the effect size test leads to a stagnation of the clusters core number at 40 (42) cluster for 10k (100k) data sets that is much closer to the exact number of hidden clusters (5).

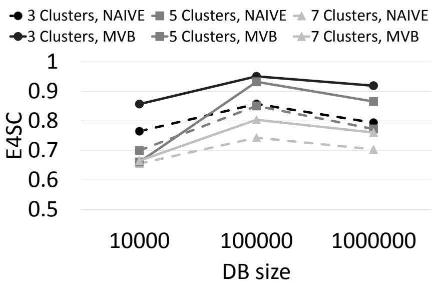
The largest improvement of the results shows the redundancy filtering technique from Section 5 depicted in Figures 5(b) and 5(d). Already for thresholds $\geq 10^{-40}$ even the 'Poisson' test almost stabilizes at the exact number of hidden clusters and shows a deviation in two cases 10^{-4} for 10k data set, and 10^{-2} for a 100k data set only. The 'Combined' test shows its superiority over 'Poisson' test also in this case, as it delivers constantly correct number of cluster for all thresholds $\geq 10^{-40}$ in the 10k data set and all investigated threshold values in the 100k data set.

As side remark, we want to point out that computation of cumu-

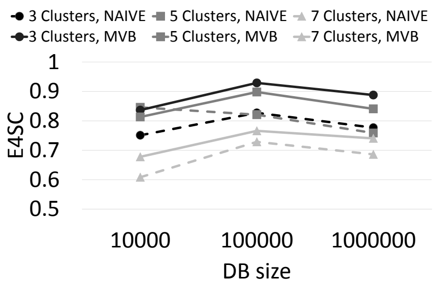
[1]`http://archive.ics.uci.edu/ml/`
[2]`http://dme.rwth-aachen.de/en/P3CMR`



(a) Noise level 5%.

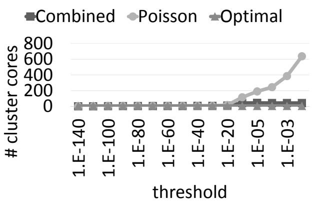


(b) Noise level 10%.

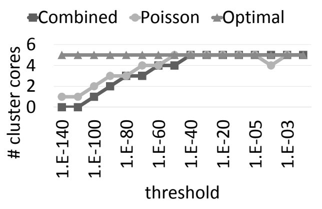


(c) Noise level 20%.

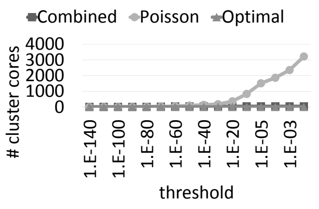
Figure 4: Comparison of naive and MVB outlier detection steps.



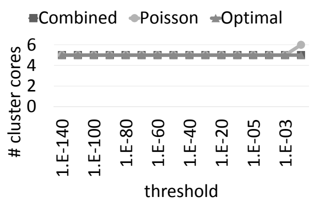
(a) 10k objects, 20% noise, no redundancy filtering.



(b) 10k objects, 20% noise, with redundancy filtering.



(c) 100k objects, 20% noise, no redundancy filtering.



(d) 100k objects, 20% noise, with redundancy filtering.

Figure 5: Effect of redundancy filtering and effect size statistics.

lative probabilities with values $\geq 10^{-10}$ become infeasible due to inaccuracy of floating point arithmetic. To overcome this problem, we transform the Poisson distribution in a Gaussian distribution with appropriate parameters and determine the amount of standard deviations of the threshold. Then, we can easily determine if an observed value should be considered as significant by comparison of the amount of standard deviations of the observed and expected supports.

7.5 Evaluation on large synthetic data sets

7.5.1 Quality results

Figure 6 depicts the E4SC quality results of both BoW variants (Light and MVB), P3C$^+$-MR and P3C$^+$-MR-Light on synthetic datasets with up to $5 \cdot 10^7$ data points of dimensionality 50 and all examined noise levels.

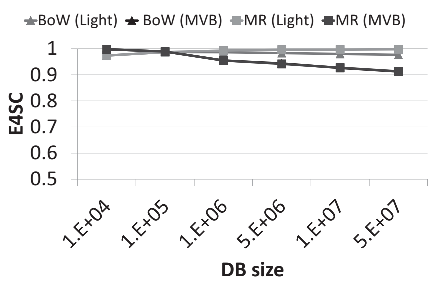
In general, we observe that the P3C$^+$-MR-Light and BoW (Light) perform better than their full equivalents with the MVB based outlier detectors. This superiority shows itself as well for data sets with growing noise levels as also for large data sets. Furthermore, the clustering quality of the P3C$^+$-MR-Light approach even increases in most cases with growing database size, while the quality of all other approaches decreases.
These results have several reasons. As we already mentioned earlier, the employed outlier detection techniques plays a significant role for the overall clustering result quality. Although MVB based outlier detector showed a substantial improvement in comparison to the naive outlier detector, it still not able to sufficiently good remove outlier objects. This on its hand enlarges the regions of the clusters and leads to worse quality. Due to sampling approach, both BoW variants additionally suffer from errors based on not sufficiently precisely approximated data distribution. Since the overall clustering results is computed from all subresults, even a small shift of a cluster in a single subresult will decrease the quality of the solution. With a growing data set size, the probability that one of the samples follows another distribution than the whole data set increases such that the above described case occurs.

Although due to the redundancy filtering all approaches are always able to identify the right number of clusters, they show a decreasing quality with a growing number of hidden clusters in the data. This behavior arises from a combination of effects. First, as the number of clusters grows, the number of points decreases such that the identification of correct borders of the clusters become harder. Second, for a larger number of clusters the number of overlaps between clusters grows, again resulting in more 'blurred' cluster borders. Third, with growing number of clusters, the probability that a point will be associated to a wrong cluster mean during the EM clustering step or the outlier detector is not able to detect a point as outlier also grow, such that the overall clustering quality decreases.

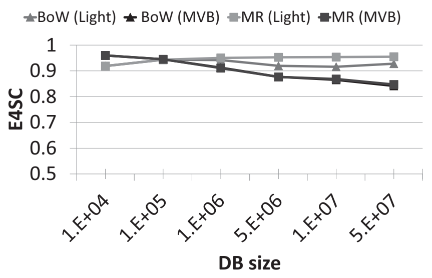
7.5.2 Runtime results

In this section we evaluate the runtimes of BoW (Light and MVB variants), P3C$^+$-MR (MR (MVB) and MR (naive)) and P3C$^+$-MR-Light (MR (Light)) algorithms on the data sets with 10^4 up to $5 \cdot 10^7$ objects. In all experiments, the number of reducers is set to 112.
First, we compare the runtimes of P3C$^+$-MR with the naive and MVB based outlier detectors. A more complex computation of MVB estimator results in a $10\% - 20\%$ runtime overhead for all data set sizes. Considering higher clustering quality achieved by the MVB-based solution (cf. Section 7.4.1) this overhead seems to be acceptable for real applications.

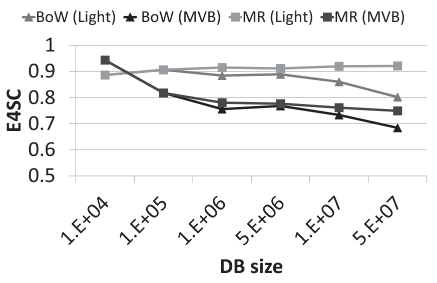
Next, we examine the BoW algorithm. The obtained results are consistent with our expectations. The BoW algorithms scale linearly with the database size and with the number of used reducers. Since each reducer processes the same amount of data and the runtime of every single job is almost constant, the overall runtime equals the time needed by the map step and a time needed by reducers to process all sampled data sets. The main computa-



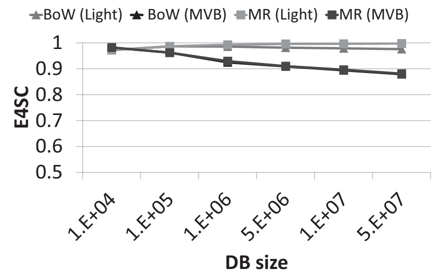
(a) 3 clusters, 0% noise.



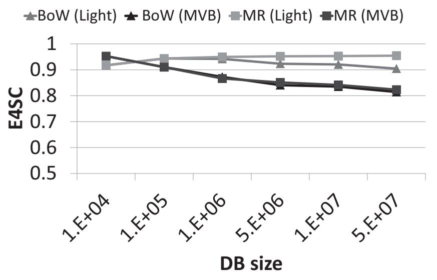
(b) 5 clusters, 0% noise.



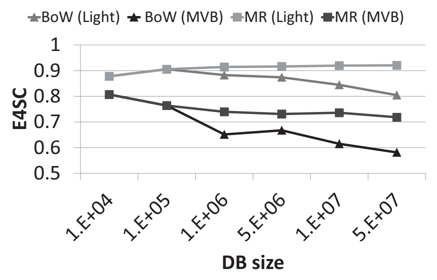
(c) 7 clusters, 0% noise.



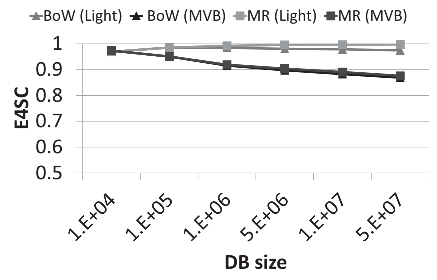
(d) 3 clusters, 5% noise.



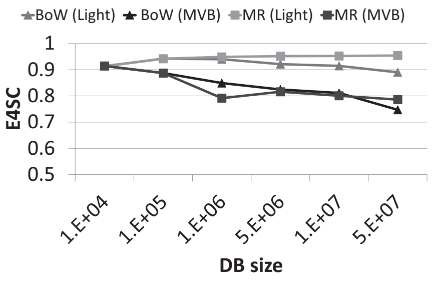
(e) 5 clusters, 5% noise.



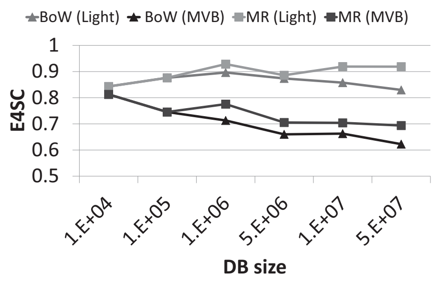
(f) 7 clusters, 5% noise.



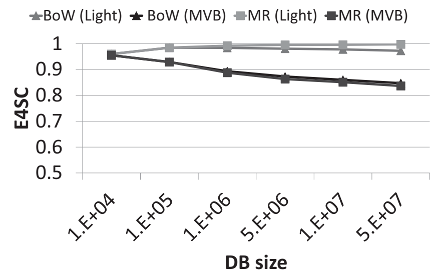
(g) 3 clusters, 10% noise.



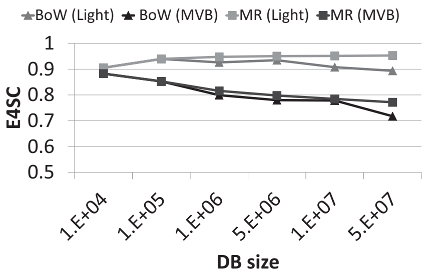
(h) 5 clusters, 10% noise.



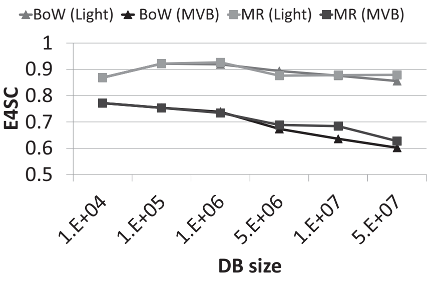
(i) 7 clusters, 10% noise.



(j) 3 clusters, 20% noise.



(k) 5 clusters, 20% noise.



(l) 7 clusters, 20% noise.

Figure 6: Quality results of BoW (Light and MVB variants), P3C$^+$-MR (MR (MVB)) and P3C$^+$-MR-Light (MR (Light)) for different cluster sizes, noise ratios and database sizes.

tional bottleneck of the BoW algorithm is therefore the number of sampled data sets. If this number is smaller than the number of available reducers, BoW will not be able to make use of the complete performance of a cluster. On the other hand, for large data sets, every reducer will have to process several data blocks. However, in general, for clusters with sufficiently many reducers, BoW provides an ideal workload distribution.

At the same time, P3C$^+$-MR and P3C$^+$-MR-Light algorithms also possess very good workload properties and in the case of P3C$^+$-MR-Light do not have reducer-dependent bottlenecks. As described in previous sections, all MapReduce jobs mainly consists of a map step and a final reduce step, performed by a single reducer. Therefore, the runtime equals the runtime of a single map step multiplied by the number of MapReduce jobs needed for clustering determination. The time of a single map step grows linearly with the database size and is not limited by the number of reducers. The main limitation is therefore the performance of the storage and network systems. In our case, the P3C$^+$-MR-Light algorithm outperforms BoW (MVB) even for small data sets and is comparably fast with BoW (Light) variant.

In comparison to other approaches P3C$^+$-MR shows the worst run-

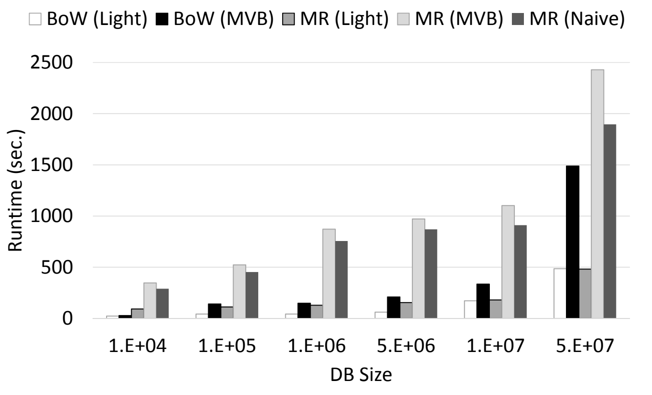


Figure 7: Runtime results.

times. Those arise from a larger number of MapReduce jobs that have to be performed and especially from the multiple iterations of the EM algorithm.

The sub-linear runtime of P3C$^+$-MR and P3C$^+$-MR-Light stems from the fact that larger inputs are processed by a larger number of mappers. And since we do not artificially split the input files in smaller chunks, the higher number of mappers resulted in an overall better runtime. For sufficiently large databases the runtime will become lineary proportional to the size of the database.

We also executed the BoW (Light) and P3C$^+$-MR-Light on the *1 billion* data set with 100 dimensions. On this huge data set the P3C$^+$-MR-Light algorithm clearly showed its superiority. While BoW (Light) needed over 9500 sec., the P3C$^+$-MR-Light produced the result in $\approx$ 4300 seconds.

7.6 Real world data

In the last experiment we compared the quality of our P3C$^+$ algorithm to the original P3C algorithm by comparing the accuracies of the clustering results on the 'colon cancer' data set used in [2]. With 71% accuracy the P3C$^+$ outperformed the original P3C algorithm that achieved 67%. However, we report this result only for the sake of completeness since the presented results on synthetic data sets better show the properties and the quality of our P3C$^+$ algorithm.

8. CONCLUSION

In this work we presented an extended version of the P3C algorithm that is capable to process huge data sets. This novel P3C$^+$ algorithm was implemented in the MapReduce framework and two versions P3C$^+$-MR and P3C$^+$-MR-Light algorithms were proposed. We compared our solution with existing approach BoW, and showed the superior accuracy of P3C$^+$-MR at costs of higher runtimes and I/O costs. In the end, we showed that an adjusted P3C$^+$-MR-Light algorithm performs best for large data sets at lower runtime and I/O costs as P3C$^+$-MR and BoW while producing highest accuracy results.

9. REFERENCES

[1] Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data **3**(1) (March 2009) 1:1–1:58

[2] Moise, G., Sander, J., Ester, M.: P3C: A Robust Projected Clustering Algorithm. In: ICDM, IEEE Computer Society (2006) 414–425

[3] Cordeiro, R.L.F., Jr., C.T., Traina, A.J.M., López, J., Kang, U., Faloutsos, C.: Clustering very large multi-dimensional datasets with MapReduce. In Apté, C., Ghosh, J., Smyth, P., eds.: KDD, ACM (2011) 690–698

[4] Aggarwal, C.C., Procopiuc, C.M., Wolf, J.L., Yu, P.S., Park, J.S.: Fast algorithms for projected clustering. In Delis, A., Faloutsos, C., Ghandeharizadeh, S., eds.: SIGMOD Conference, ACM Press (1999) 61–72

[5] Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.M.: A Monte Carlo algorithm for fast projective clustering. In Franklin, M.J., Moon, B., Ailamaki, A., eds.: SIGMOD Conference, ACM (2002) 418–427

[6] Silva, Y.N., Reed, J.M., Tsosie, L.M.: Mapreduce-based similarity join for metric spaces. In: Proceedings of the 1st International Workshop on Cloud Intelligence. Cloud-I '12, New York, NY, USA, ACM (2012) 3:1–3:8

[7] Seidl, T., Fries, S., Boden, B.: MR-DSJ: Distance-Based Self-Join for Large-Scale Vector Data Analysis with MapReduce. In Markl, V., Saake, G., Sattler, K.U., Hackenbroich, G., Mitschang, B., Härder, T., Köppen, V., eds.: BTW. Volume 214 of LNI., GI (2013) 37–56

[8] Moise, G., Sander, J.: Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In Li, Y., Liu, B., Sarawagi, S., eds.: KDD, ACM (2008) 533–541

[9] Nagesh, H.S., Goil, S., Choudhary, A.N.: A Scalable Parallel Subspace Clustering Algorithm for Massive Data sets. In: ICPP. (2000) 477–484

[10] Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: OSDI. (2004) 137–150

[11] Günnemann, S., Färber, I., Müller, E., Assent, I., Seidl, T.: External evaluation measures for subspace clustering. In Macdonald, C., Ounis, I., Ruthven, I., eds.: CIKM, ACM (2011) 1363–1372

[12] Rousseeuw, P.J., Zomeren, B.C.v.: Unmasking multivariate outliers and leverage points. Journal of the American Statistical Association **85**(411) (1990) pp. 633–639

[13] Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics). 1 edn. Wiley (September 1992)

[14] Hyndman, R.J.: The problem with sturges rule for constructing histograms (1995)

[15] Richardson, J.: Measures of effect size. Behavior Research Methods, Instruments, Computers **28**(1) (1996) 12–22

[16] Chu, C.T., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G.R., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. In Schölkopf, B., Platt, J.C., Hoffman, T., eds.: NIPS, MIT Press (2006) 281–288