Distance-Based Outlier Detection: Consolidation and Renewed Bearing

Gustavo H. Orair   Carlos H. C. Teixeira
Wagner Meira Jr.
Department of Computer Science
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil

Ye Wang   Srinivasan Parthasarathy
Department of Computer Science and
Engineering
The Ohio State University
Columbus, USA
Contact Author
Email:srini@cse.ohio-state.edu

ABSTRACT

Detecting outliers in data is an important problem with interesting applications in a myriad of domains ranging from data cleaning to financial fraud detection and from network intrusion detection to clinical diagnosis of diseases. Over the last decade of research, distance-based outlier detection algorithms have emerged as a viable, scalable, parameter-free alternative to the more traditional statistical approaches.

In this paper we assess several distance-based outlier detection approaches and evaluate them. We begin by surveying and examining the design landscape of extant approaches, while identifying key design decisions of such approaches. We then implement an outlier detection framework and conduct a factorial design experiment to understand the pros and cons of various optimizations proposed by us as well as those proposed in the literature, both independently and in conjunction with one another, on a diverse set of real-life datasets. To the best of our knowledge this is the first such study in the literature. The outcome of this study is a family of state of the art distance-based outlier detection algorithms.

Our detailed empirical study supports the following observations. The combination of optimization strategies enables significant efficiency gains. Our factorial design study highlights the important fact that no single optimization or combination of optimizations (factors) always dominates on all types of data. Our study also allows us to characterize when a certain combination of optimizations is likely to prevail and helps provide interesting and useful insights for moving forward in this domain.

1. INTRODUCTION

Outlier detection is a fundamental step in a large number of data quality, data management, and data analysis tasks. Examples abound ranging from its use as a first class operation for tasks such as medical diagnostics [15], image

analysis [24], and network intrusion detection [16, 17] to its use as a preprocessing step for assessing the quality of data and as a precursor to various data mining algorithms that are heavily influenced by the presence of outliers.

In layman's terms, the noted physicist Stephen Hawkins defines an outlier as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". The problem of outlier detection has been around for over a century and and has been the focus of much research in the statistics literature. Here, data is assumed to follow a parametric distribution and the objects that do not fit properly the model are considered outliers. However, such approaches are limited in the sense that the data distribution and underlying parametric formulation may be unknown or hard to determine [19]. Another limitation with these approaches is that they typically do not scale well to large or even moderately large datasets.

Several non-parametric approaches that do not rely on data distributions have been proposed. The first class of methods proposed were typically an artifact of clustering algorithms [18, 7, 28, 11]. Here the idea is to label as outliers those points that were not a core part of any clusters. A limitation of such approaches is the lack of a strong definition as to what constitutes an outlier since the definition is somewhat implicit from algorithm implementation and significantly depends on the algorithm's parameters. It is also the case that even the outlier definition is often domain-dependent.

Computational geometry inspired approaches for outlier detection, based on depth and convex hull computations, have been around for the last four decades [25]. These approaches rely on the principle that outliers lie at the border of the data space. The main idea here is, given a cloud of points, to identify convex hulls at multiple depths (layers). At the end of this process points in the few outermost convex hull layers are reported as outliers. The dependency on convex hull calculations significantly hampers the scalability of such methods on high dimensional data and even in low dimensions these methods can be quite expensive with large datasets [14].

To address this limitation, several authors defined the notion of an outlier based on density (of neighborhood) or distance (of neighbors). Density-based approaches determine outliers by analyzing the object's neighborhood density [6, 20] and thus have the opportunity to identify interesting out-

liers missed by other methods. Note that in such methods distance plays an implicit role in determining the neighborhood and local density. However, while these methods offer improved performance when compared to methods inspired on statistical or computational geometry principles, they are not scalable [14].

Explicit distance-based approaches, based on the well-known nearest-neighbor principle, were first proposed by Ng and Knorr [13] and employ a well-defined distance metric to detect outliers, that is, the greater is the distance of the object to its neighbors, the more likely it is an outlier. Distance-based approaches have been the subject of much research withiin the database and data analytics communities [3, 5, 10, 2]. It is a relatively non-parametric approach that has been shown to scale quite nicely to large datasets of moderate to high dimensionality and is the focus of this paper. The basic algorithm for such distance-based algorithms, the nested loop (NL) algorithm, calculates the distance between each pair of objects and then set as outliers those that are far from most objects. The NL algorithm has quadratic complexity, with respect to the number of objects, making it unsuitable for mining very large databases such as those found in government audit data, clinical trials data, and network data. As a result, a good portion of the research in the literature to date on this problem has focused on identifying practical sub-quadratic algorithms [22]. In the quest for designing of such algorithms, researchers have identified several important optimizations and algorithms such as the use of compact data structures [22, 10], the benefits of pruning and randomization [5], among others. In fact, each new proposal, including some by ourselves is inevitably backed by a strong set of empirical results showcasing the benefits of the new method over competitive strawman or the previous state of the art algorithms. However, as we demonstrate in this study, research in this important domain has lost sight of some important optimizations along the way. Our hypothesis when we started this study is that several important optimizations that are tied down to specific implementations are often ignored by more recent advances in this field. It is thus, in our opinion, extremely important to consolidate the efforts of many and understand these interactions carefully before proposing new ideas.

In this paper we take a consolidated view of this problem and include the important optimizations suggested in this domain in the past and also include some new ones within the context of this tudy. A fundamental optimization underpinning almost all approaches proposed to date is the ability to perform an approximate nearest neighbor search as this information is critical to estimate how likely a point is an outlier, given the data seen thus far. Beyond this optimization (which is an essential component of all competitive strategies in the field), we believe that optimizations proposed by various researchers may be categorized as a form of pruning of the search space or a ranking based optimization. Each of these may be further categorized further into two sub categories. In each category and sub-category we will briefly discuss how extant research proposals fit, and, as noted above, we discuss a new proposal in one sub category. We then describe a framework wherein each of these optimization strategies can be independently applied in isolation or collectively, allowing us to better understand the interactions among them. This framework facilitates a factorial design study on several real and synthetic datasets enabling us to understand the utility of different optimization strategies and to help understand the interactions among them.

This paper, in our opinion, makes three significant and non-trivial contributions. First, we present a full fledged factorial design study to understand the impact of the main optimization strategies presented in the literature, including one that is novel to this paper. To the best of our knowledge we are not aware of another such study that attempts to consolidate these efforts and is attempting to understand the contributions of different optimizations and the interactions among them. Second, the results of our factorial design study enables us to identify interesting and important insights. For example, it is worth noting here that an optimization strategy proposed as part of a carefully constructed strawman implementation from Ramaswamy et al [22], and not part of their main proposal, is by far one of the more important factors for this problem. Additionally, we find that while certain combinations of optimizations work very well in general on both real and synthetic datasets, there is no single combination of optimizations that is always the best. Third, an artifact of our effort is a general purpose framework that is available to the broader research community, enabling researchers to replicate our results and to also build upon them. Another artifact of our study is that it helps us get a renewed bearing on interesting directions for future work on this problem which is discussed at the very end of this paper.

2. BACKGROUND

In this section we begin by defining the problem and then review some of the previous efforts for detecting outliers. Along the way we identify a taxonomy of optimizations and ideas proposed for this problem. We classify several state of the art algorithms based on their fundamental optimization strategy. For expository simplicity we have summarized the common notations used in the rest of this paper within Table 1.

2.1 Problem Definition

As noted earlier, distance-based techniques for outlier detection have become quite popular, due to their relatively non-parametric nature and due to their simplicity and ability to scale to large datasets. However, it is important to formally define the notion of an outlier in order to proceed. To date there are three main definitions of outliers:

1. Outliers are objects with fewer than k neighbors in the database, where a neighbor is an object that is within a distance R [13].

2. Outliers are the n objects presenting the highest distance values to their respective k^{th} nearest neighbor (*the kNN definition*) [22].

3. Outliers are the n objects presenting the highest average distance to their respective k nearest neighbors [3].

All three definitions exemplify Hawking's definition, that is, the greater is the distance of the object to its neighbors, the more likely it is an outlier. The first definition originally proposed by Knorr and Ng [13] relies on both the definition of a neighborhood (R) as well as the number of neighbors k in order to determine whether a point is an outlier or not. The choice of R is an additional parameter that the user

Symbol	Description
n	Number of outliers in result set to be identified in the database
k	Number of nearest neighbours to be considered
$D^k(p)$	Distance between point p and its kth closest neighbour
D^k_{min}	Smallest distance between a point and its kth nearest neighbour from result set
$|P|$	Number of objects (size) of a partition P
$R(P)$	The MBR diagonal value of a partition P
$MINDIST$	Minimum distance between an object and MBR structure
$MAXDIST$	Maximum distance between an object and MBR structure

will need to set which may sometimes be hard to determine apriori. The definition also does not lend itself to certain optimization approaches. In fact, for a suitable choice of R, it can be shown that *Knorr and Ng's definition* leads to a quadratic algorithm where all distances between pairs of objects must be calculated in the worst case.

Another measure that quantifies such information for each object p is $D^k(p)$, which is the distance between the object p and its kth-nearest neighbor. This measure was proposed by Ramaswamy et al [22], and is the basis of the second definition of an outlier. An added benefit of this definition is that it allows the outliers to be transparently ranked and places less of a demand user's apriori knowledge about the objects in the database in that it relies on one fewer parameter to set. A variant of this definition is considered by Angiulli et al [3].

Note that several contemporary algorithms, for example ORCA [5] and RBRP [9, 10] do admit all three definitions within their respective frameworks and thus some optimizations will apply to all definitions, whereas others are a bit more restrictive. Among the three, however, the kNN definition by Ramaswamy et al [22] is certainly more frequently used and is the focus of this work. We should note that many of our results also have been evaluated on the third definition (which is closely related to the second) and the results we find for them are along similar lines to what is reported in this paper.

2.2 Canonical Algorithm

According to the *kNN definition*, the simplest algorithm for distance-based outlier detection is the Simple Nested Loop (SNL). The SNL (not shown) just computes all $D^k(p)$ values, which are the distances from each point p to the kth nearest neighbor. Then, the top-n outliers are selected as the n points associated with the largest $D^k(p)$ values. This algorithm's complexity is quadratic – $O(N^2)$. The canonical form of an optimized variant of the nested loop algorithm is shown in Algorithm 1. This algorithm is also quite straightforward and while it still retains a quadratic worst case complexity it does offer significant potential for optimization as we shall discuss shortly.

3. OPTIMIZATIONS

Given the canonical algorithm presented in the previous section, we are now in a position to discuss some of the key optimizations proposed in the literature. These are described next.

3.1 Approximate Nearest Neighbor Search

Algorithm 1 Algorithm 1: Canonical Distance-Based Outlier Detection Algorithm

Procedure: Search for Outliers
Inputs: k, number of neighbours considered; n, number of outliers to be identified; D, the set of points
Outputs: O, the outlier result set
Let: Nearest(o, S, k) returns k elements from S that are the nearest to o
Let: Maxdist(o, S) returns the maximum distance between o and an element in set S
Let: TopOutlier(S, n) returns the top n outliers in S based on the distance to their k^{th} nearest neighbor.
Begin
1: $O \leftarrow \emptyset$ {Make the outlier result set empty}
2: $D^k_{min} \leftarrow 0$ {Reset the pruning threshold}
3: **for each** object o in D **do**
4:    $Neighbours(o) \leftarrow \emptyset$ {Make the neighbors's set from o empty}
5:    $D^k(o) \leftarrow 0$ {Reset the k nearest neighbor distance}
6:    {Searching for neighbours of object o}
7:    **for each** object v in D, where $v \neq o$ **do**
8:       Neighbours(o) = Nearest(o, Neighbours(o)$\cup$ v, k)
9:       $D^k(o)$ = Maxdist(o, Neighbours(o))
10:      **if** |Neighbours(o)| = k and $D^k_{min} > D^k(o)$ **then**
11:         *break* {Discard this object as outlier, ANNS rule}
12:      **end if**
13:   **end for**
14:   O = TopOutliers($O \cup o$, n)
15:   **if** $| O | = n$ **then**
16:      D^k_{min} = min($D^k(o)$ for all o in O)
17:   **end if**
18: **end for**
End

One of the most important pruning rules in the context of outlier mining was defined by Ramaswamy et al [22] and is often refereed to as *Approximate Nearest Neighbour Search* – ANNS(Def. 1). ANNS focuses on just the top-n outliers, maintaining such a set during the execution of the algorithm. Let D^k_{min} be the the shortest distance between any object in the result set and its kth nearest neighbor (see Algorithm 1). Assume that for a given point p we are processing its distance to its nearest neighbors ($D^k(p)$). Since $D^k(p)$ monotonically decreases as we process other neighbors at any point in the calculation, the current value is an upper-bound on its eventual value. If the current value (upper bound) becomes smaller than D^k_{min}, the point p cannot be an outlier and can thus be pruned. Approaches that can quickly identify a set of approximate neighbors for a data point can thus be used to obtain an upper bound on ($D^k(p)$).

Formally we may define this as:

DEFINITION 1. *We may disregard an object p as a candi-*

date outlier if, while computing its $D^k(p)$, $D^k(p) < D^k_{min}$.

This optimization is fundamental to almost all recent algorithms proposed in the literature as we note later when discussing the taxonomy of existing work.

3.2 Pruning

A fundamental strategy used in a myriad of data processing and data analytic workloads is to prune the search space by leveraging appropriate domain insights. In fact, the ANNS optimization discussed above is indeed also a form of pruning – the reason we separated it out is because of its universal appeal in the current domain of interest. We have observed that several algorithms proposed in the literature rely on a preprocessing step to cluster or partition the data space to facilitate such pruning. In these proposals, the goal is organize the objects into clusters or partitions so that it is not necessary to perform all distance calculations. Some approaches applied this strategy based on the inherent indexing data-structures that is omnipresent in modern database systems, achieving good scalability in the presence of a large number of objects, but not in terms of the number of dimensions, being affected by the curse of dimensionality known to afflict such structures [13, 6, 22]. An alternative is to simply pre-cluster the data (as noted above) and work in a similar manner with the resulting clustering arrangement. In this context we distinguish two clustering-based prune strategies:

Pruning Partitions during Search for Neighbors:

The PPSN strategy prunes partitions that are far from an object while searching for neighbors. In [23], the authors introduce some fundamental pruning metrics for an R*-Tree while searching for the nearest neighbors. They leverage the concept of MBR (*"Minimum Bounding Rectangle"*), which is a spatial structure that represents the smallest hyper-rectangle that embed all objects that belong to a given node. They also propose two pruning thresholds inherent to each node and based on the MBR. These thresholds are bounds on the distance between an object and any other object that belongs to the cluster. More specifically, let $MINDIST$ be the shortest distance from all objects in a partition to an object p, if $MINDIST$ becomes greater than the current $D^k(p)$, none of the objects in that partition can be among the k-nearest neighbors of p. The early identification of partitions that cannot contain neighbors reduce significantly the number of neighbors to be evaluated.

This strategy was applied to solve the outlier detection problem by Ramaswamy et al [22]. In this work they proposed an algorithm (*Index-Based Join*) based on indexes. The idea is to traverse the R*-Tree in a top-down fashion while searching for neighbors, exploiting the tree structure for improving the search. The index-based algorithm they propose explicitly performs the aforementioned ANNS pruning rule. Also, an additional pruning rule for entire partitions while searching for neighbors is applied. It allows pruning entire subtrees that contain irrelevant objects to the kth-NN search for an object p. It should be noted that as the algorithm traverses the R*-tree looking for nearest

neighbors, it is affected by the curse of dimensionality, not being a good choice for datasets containing more than 10 dimensions.

For datasets with larger dimensionality, a pre-clustering step that clusters the data and then pre-computes a similar set of minimum bounding rectangles can be suitably leveraged. The advantage of this approach is that it avoids the limitations of the index-structure based approaches while still retaining the advantages for distance-based outlier detection.

Pruning Partitions during Search for Outliers:

PPSO prunes outlier candidates based on summary statistics from the underlying clustering or partitioning arrangement, while PPSN prunes objects candidates during the search for nearest neighbors. This strategy prunes all objects from a partition, based on the knowledge that it cannot contain outliers. Ramaswamy et al [22], in his third algorithm, propose a two-phase partition-based algorithm, which eliminates whole partitions that do not contain outliers. This elimination is based on summary statistics generated in the clustering phase. They argue that, since this preprocessing step is performed at the granularity of partitions rather than objects, this strategy eliminates a significant number of objects as outlier candidates. The rationale they adopt is to use bounds associated with the partitions to first estimate D^k_{min}, which is a lower bound of the $D^k(p)$) value for an outlier. The D^k_{min} values are then used to prune partitions that could not contain outliers. Later, another partition bound is used to prune partitions that may not contain k^{th}-NN for points from a partition P. The resulting partitions are computed using the index-based algorithms.

It should be noted that the effectiveness of this strategy has an inherent dependency on the quality of the partitioning or clustering algorithm. Additionally, this is a strategy that is likely to be more effective when the number of partitions is quite high (requiring a significant penalty in terms of preprocessing costs), since for large partitions it is unlikely to be terribly effective. Moreover, as in the previous approach, one can either apply this strategy based on an index-structure based decomposition of the data space – limiting its applicability to high dimensional data or a clustering based approach which may be more broadly applicable to higher dimensional problems.

3.3 Ranking

The next set of strategies evaluated in the scope of this work are ranking strategies, which aim to improve the efficiency of the ANNS pruning rule ($D^k_{min} > D^k(o)$). Given both sides of the inequality of the pruning rule, it is intuitive that the performance of ANNS depends on the order of evaluation of both objects and their neighbors. Based on this observation, we may distinguish two sub-category of optimization strategies, which implicitly define when they are applicable:

Ranking Objects Candidates for Neigbours:

The ROCN strategy ranks the order in which neighbors of a point are processed, and aims to reduce the

current value of $D^k(p)$ faster, enabling one to get a better upper bound on the true value and thus ensuring that the ANNS pruning rule is triggered earlier in the computation. Bay and Schwabacher proposed ORCA [5], an algorithm based on nested loops, randomized ranking and ANNS, defined in their work as Simple Pruning Rule. That work shows that ORCA has near linear time performance on many large real data set. Notice that, in this case, randomization is just a non-deterministic ranking strategy applied by the authors. They show that when the objects are in random order, the algorithm gives near linear time performance, even though it is quadratic in the worst case. Moreover, notice that the authors randomized the neighbor ranking but indirectly also randomized the outlier candidates (which we shall discuss next). From their work it is hard to understand the implications of which of these is more effective. RBRP [10] also leverages the idea of ranking neighbors based on a simple projection based approach within partitions. RBRP also exploits this idea across partitions as we shall discuss shortly.

Ranking Objects Candidates for Outlier:

Ranking outlier candidates (ROCO) tries to determine which objects are more likely to be outliers. The objective here is to focus on the value of D^k_{min} (increasing it) so as to trigger the ANNS pruning rule earlier on in the process. Recently, Wu and Jermaine [27] proposed a Bayesian method for guessing extreme object values from a database, and demonstrated its applicability to distance-based outlier detection by adopting the above principle. They estimate the kth-NN distance for each object p. These estimates are then used as a ranking wherein objects with greater k^{th}-NN distances will be considered first as candidate outliers. Clearly, when processed in this ranked order, one would expect to see a fast increase in the value of D^k_{min} thereby enabling the efficacy of the ANNS pruning rule. The Bayesian approach is also used together with probabilistic pruning that is performed periodically. It consists of a trained model that is used to estimate the kth-NN distance while computing additional neighbor points, whenever this estimate holds with high probability, it is used to prune the object. The probabilistic pruning rule they apply is simply an extension of the ANNS pruning rule to their model. It should be noted here that this approach does not produce an exactly identical list of outliers as the other methods given that it is a probabilistic approach but it is a good example of the use of this ranking strategy. A problem with this approach is that the estimation procedure itself may be quite expensive (for the purposes of outlier detection) – although to be fair we should note that application context in this paper was not limited to distance-based outlier detection.

We next discuss how the pre-processing clustering or partitioning step may be used for improving the search for neighbors, supporting a ranking of the objects according to the distances between partitions. The rationale here is to exploit the ranking idea in conjunction with partitioning as follows:

ROCN with Partitions:

RBRP [10] explores how the pruning rule from ANNS may be enhanced in terms of efficiency as a consequence of faster determination of nearest neighbors at multiple levels. RBRP is a two-phase algorithm where the first phase is a partitioning phase that bins the data space into grid-like partitions. RBRP exploits ranking of neighbors within a partition by employing a fastmap [8] style projection of points within a partition and employs a ranking across partition via the natural binning process employed in the first phase. This two level partitioning is used for sorting the search space for nearest neighbors as follows. The objects are first compared to other objects in the same partition (using the projection), and, if it is necessary to evaluate objects from other partitions, it also ranks the partitions so that nearer partitions are evaluated first, resulting in a pruning strategy that is shown to have log-linear performance on average. We should note that Index and Partition-based algorithms from Ramaswamy et al [22] also exploit a simpler strategy (limited to the partition level – not within a partition).

ROCO with Partitions:

Novel to this paper we present an approach to rank partitions based on their likelihood of containing outliers in the context of distance-based outlier detection. The synopsis of this approach is as follows. During the clustering or partitioning step we compute useful statistical synopses of each cluster and use them to rank clusters according to their likelihood to contain outliers. Our strategy exploits the ROCO strategy to increase D^k_{min} values faster, and it is expected to improve the overall efficacy of the ANNS pruning rule. To perform the ranking, we applied a first clustering phase that generates partitions that is similar to the recursive partitioning phase proposed by RBRP [10]. While doing so we maintain relevant statistical summaries for each bin (cluster) such as the number of objects within a bin, the partition volume and the density of the partition. For example, the second heuristic ranks the partitions by their spatial size, so that larger partitions are considered first, because they tend to have low-density regions that may contain outliers. We anticipate that ranking partitions by density will increase D^k_{min} values faster and is likely to achieve better performance overall.

4. TAXONOMY OF EXISTING WORK

We are now in a position to present a taxonomy of the key approaches presented thus far in the literature as they relate to the kNN definition of outliers. An overview is presented in Table 2.

As noted already, there are several enhancement techniques that aim to reduce the number of distance calculations. Many of them have the same fundamental optimization idea, but explore different ways to achieve it. Thus, we divide the algorithms and their strategies into groups, as shown in Table 2. This division was performed according to the following criteria:

- Does the algorithm have a clustering preprocessing phase?

- What is the pruning strategy used?

- Does it rank objects candidates for neighbors?

- Does it rank objects candidates for outliers?

The taxonomy is fairly self explanatory but it is interesting to note that almost all recent algorithms rely on the ANNS pruning rule. We next discuss the key elements of our framework.

5. THE DIODE FRAMEWORK

Before we discuss the experimental results, we describe the framework for DIstance-based Outlier DEtection (DIODE) [1] used to perform the experiments, and the strategies employed to improve the effectiveness of the pruning process. The techniques implemented in DIODE focus on partitions for sake of pruning and ranking objects, and also applying the ANNS rule. DIODE covers all dimensions from our taxonomy and supports the evaluation of each dimension in isolation and also in combination.

The baseline method in DIODE employs a clustering pre-processing step. Currently, two kinds of clustering algorithms are supported. The first is an RBRP-style recursive partitioning algorithm and the second is a divisive bisection k-means algorithm that relies on a fast-map style projection to identify the initial centroids at each step. Both algorithms can be thresholded to return partitions of a user-defined size. For our experiments we ensured that the size of any partition cannot be more than 16000 points (that is, at any point in the algorithm, if a partition is larger than this threshold, we sub-partition it further). Both clustering algorithms scale quite well to large datasets (linear performance) and produce a reasonable set of partitions. Our framework also supports the ability to compute important summary statistics about the clusters or partitions (e.g., radius, diameter, density etc.) similarly to BIRCH [28].

Once the preprocessing has taken place, the meta-level clusters can be used to effectively support the basic ANNS pruning rule by first examining nearest neighbors within a cluster partition and then subsequently looking at neighbors from other partitions. This step is part of our *baseline method*. In order to facilitate the factorial design study and also to enable researchers to leverage this platform, the current implementation of the framework also includes various pruning and ranking strategies that are key for the scalability of the distance-based algorithms, as discussed earlier. We briefly discuss specifics of how they are implemented within the DIODE framework.

PPSO : DIODE implementation prunes outlier candidate partitions in a manner similar to the approach presented by Ramaswamy et al [22]. We employ the current $D^k min$, and, whenever we look for an outlier, we check whether $D^k min$ is large enough so that it is not necessary to look for additional outliers in the partition. Note this is a best effort adaptation of Ramaswamy's original proposal targeted at high dimensional data, since estimators based on distances among partitions often lose significance here. We note that this pruning rule is applied on demand only when we

need to analyze the partition objects P. Since D^k_{min} is monotonically increasing with the execution of the algorithm, we may expect that once it is large enough, a significant number of partitions will be pruned.

PPSN : DIODE also prunes partitions while searching for neighbors. This pruning aims to identify objects that do not need to be taken into account as nearest neighbors from an object p. Roussopoulos et al [23] proposed such pruning using MINDIST, an estimation of the smallest distance between an object p and a partition P. DIODE extends this definition to support categorical data by adapting the k-prototypes definition of distances in mixed attribute datasets [12]. It also uses the $D^k(p)$ to improve the efficiency of the pruning rule, as follows. Assume that $D^k(p)$ has been determined for object p by analyzing a subset of the objects, and, as discussed, its current value is an upper bound of the actual $D^k(p)$. Hence a partition P may be discarded while searching for p's neighbors if the shortest distance estimated between the object p and P is greater than the current $D^k(p)$. Note that $D^k(p)$ decreases monotonically as we apply PPSN.

There are two different ranking approaches for improving the outlier detection algorithms as we have discussed. Both ranking strategies may be both employed within a partition and across partitions. We adapt this strategy for our analysis. Note that both strategies may be combined for a better performance since they target different aspects of the ANNS pruning rule. It is important to realize that the ranking strategies are performed in the beginning of the processing, i.e., before the pruning rules are applied.

ROCN : We employed the same approach of the RBRP algorithm, i.e., ranking the search for neighbors in a partition level using estimated distances among them. These estimates could be determined by calculating the distances between the centers (centroids) of the partitions or even between MBR structures (in the case of categorical data).

ROCO : For ranking objects that are candidates for outliers we adopted a density-based heuristic, which is based on the intuition that low-density regions (partitions) tend to contain higher-score objects. We define density as $\frac{|P|}{R(P)}$, where $|P|$ is the number of objects in partition P, and $R(P)$ is the MBR diagonal length of P.

There are other aspects of our framework that also may be of interest to the general user such as an effort related to parallelizing the framework on a cloud system but these are not germane to the context of this work. The strategies described in this section form the basis of our full fledged factorial design study presented next.

6. EXPERIMENTAL RESULTS

In this section, we present a summary of our factorial design study evaluating the different pruning and ranking strategies discussed earlier. We begin by briefly describing the datasets used in our evaluation and the experimental setup.

Table 2: Summary of distance-based outlier detection strategies

Method	Clustering	ANNS	PPSN	PPSO	ROCN	ROCO	Algorithms
1	No	No	No	No	No	No	Nested Loop [22]
2	No	Yes	No	No	No	No	Enhanced Nested Loop with ANNS
3	No	Yes	No	No	Yes	No[a]	ORCA [5]
4	No	Yes	No	No	No	Yes	Wu and Jermaine [27]
5	Yes	Yes	No	No	Yes	No	RBRP [10]
6	Yes	Yes	Yes	No	Yes	No	Index-based join algorithm [22]
7	Yes	Yes	Yes	Yes	Yes	No	Partition-based algorithm [22]
8	Yes	Yes	Yes	Yes	No	No	MIRO [26]
9	Yes	Yes	No	No	No[b]	Yes	Our new approach presented in this paper
10	Yes	Yes	No	No	Yes	Yes	Our new approach + ROCN partition-based

[a]The ORCA algorithm performs ranking outlier candidates collaterally
[b]Our new algorithm starts the search for neighbors of p in p's partition

Table 3: Database descriptions

Database	# Objects	# Attributes	
		# Cont	# Cat
Government Auctions	268,170	6	7
KddCup1999	4,898,430	34	8
Forest Covertype	581,012	10	45
Uniform30D	1,000,000	30	0
ClusteredData	500,000	30	0
ClusteredData with noise	500,500	30	0

6.1 Experimental Setup

Datasets: We used various real and synthetic databases. These databases were chosen for being representative and diverse in terms of their characteristics or the associated problems, which arise from different areas. A standard t-test was used to assess the statistical significance of results across multiple optimizations unless otherwise noted. A brief summary of the databases we used is presented in Table 3 and described below.

- *Government Auctions*: The database contains records associated with purchases made by various government institutions from Brazil. Details on this dataset can be found elsewhere [21].

- *KddCup1999*: This data set contains a set of records that represent connections to a military computer network where there have been multiple intrusions and attacks by unauthorized users. The raw binary TCP data from the network has been processed into features such as connection duration, protocol type, number of failed logins, and so forth. This data set was obtained from the UCI KDD archive [4].

- *Forest CoverType*: Database with the forest cover type for 30 x 30 meter cells obtained from US Forest Service (USFS) on Rocky Mountain Region [4].

- *Uniform30D*: This is a synthetic database with 30 dimensions where the attribute values were generated randomly between (0.5, -0.5), resulting in a uniform distribution.

- *ClusteredData*: This synthetic database was generated based on various well-defined uniform and Gaussian distributions in a multi-dimensional space where all attributes are in the range (2, -2).

- *ClusteredData with noise*: It consists of the Clustered-Data database augmented with a few noisy objects (almost 0.1% of objects) that follow a uniform distribution between (2, -2). The ClusteredData and ClusteredData with noise databases contain well-defined clusters and will be used to evaluate the impact of noise on the algorithms' performance.

It is important to note that on all real datasets the numeric attributes were normalized according to a normal distribution and categorical values were converted to an integer representation.

Experimental Environment: The experiments were performed on an AMD Athlon 64 3200+ with 2 GB RAM. In all experiments we looked for the top 30 outliers ($n = 30$), using $k = 5$ as the number of nearest neighbors unless otherwise noted[2]. We would like to reiterate that in all results reported here the outliers found by all approaches are identical. All algorithms were implemented in C.

Factorial Design: As noted earlier, the majority of the existing approaches rely on the ANNS pruning rule. As a result this is a part of our baseline approach and also a part of all factor combinations we evaluate. Additionally, an increasing number of algorithms proposed recently in the literature rely on a fast clustering (pre-processing) step. This is also a part of our baseline and all factor combinations we evaluate. We evaluated the factors present in Table 4 in our experiments. We performed experiments that either enable "+1" or disable "-1" each of the four factors (strategies) resulting in a full 2^4 factorial design experiment. All experimental configurations for all datasets were averaged over 10 runs unless otherwise noted.

The factorial design model tries to explain the data assessing the impact for each factor and possible interactions of factors as defined by Equation 1.

$$y = Q_0 + \sum_{i \in F} Q_i \times x_i \qquad (1)$$

where $F =$
$$\{A, B, C, D, AB, AC, AD, BC, BD,$$
$$CD, ABC, ABD, ACD, BCD, ABCD\}$$

[2]The largest partition size was 16000 objects [10].

1475

Table 4: Factors employed on Experimental Design

Factor	Description
A	ROCO
B	ROCN
C	PPSO
D	PPSN

Table 5: Algorithms evaluated

A	B	C	D	Comment
-1	-1	-1	-1	Baseline
-1	-1	-1	+1	PPSN
-1	-1	+1	-1	PPSO
-1	-1	+1	+1	PPSN+PPSO
-1	+1	-1	-1	ROCN
-1	+1	-1	+1	ROCN+PPSN
-1	+1	+1	-1	ROCN+PPSO
-1	+1	+1	+1	ROCN+PPSO+PPSN
+1	-1	-1	-1	ROCO
+1	-1	-1	+1	ROCO+PPSN
+1	-1	+1	-1	ROCO+PPSO
+1	-1	+1	+1	ROCO+PPSN+PPSO
+1	+1	-1	-1	ROCO+ROCN
+1	+1	-1	+1	ROCO+ROCN+PPSN
+1	+1	+1	-1	ROCO+ROCN+PPSO
+1	+1	+1	+1	All Enabled

$$\text{and } x_i = \begin{cases} -1 & \text{if } i \text{ is disabled;} \\ +1 & \text{if } i \text{ is enabled.} \end{cases}$$

Q_0 is the average value from all executions regardless of the configuration adopted. The column Q_i shows the value of the factors and interactions on execution time. For example, if we have the values for Q_0, Q_A, Q_B and Q_{AB} equal to 60, -10, -8 and 3 seconds, respectively, we can see that employing ROCO (factor A) may reduce the execution time by 20 seconds (10s added to the average when disabled and 10s reduced when enabled) while employing ROCN may reduce by 16 seconds. Further, if employing both strategies combined we may reduce execution time by 30 seconds (20+16-6 from example).

Another comparison metric is the Sum of Squares Total (SS_T), which shows the total variation of y across all runs. The column SS_i presents the sum of squares due to the factor i, and it represents the portion of Total Sum of Squares (SS_T) that is explained by the factor i. Then, in column $SS_i(\%)$, we show the percentage of variation explained by the factor.

For each experiment, we report the raw execution times of the outlier detection process for all factor combinations in Table 6. Statistically significant factors and factor combinations are also reported for each dataset we consider in Table 7.

The results presented next are generated following a 90% confidence t-test. All results that are not significant in a 90% confidence t-test interval are discarded.

6.2 Results

ClusteredData: In Tables 6 and 7(a) we can see the config-

urations that present significant variation while mining the clustered dataset. A reasonable null hypothesis here could be that due to the well-defined clusters that are inherent to this dataset we expect all optimizations to be very effective here. Clearly, when viewed from the running time, this is true (the lowest running time is reported when all factors are enabled). However it is interesting to observe that, in particular, PPSO did not provide significant improvements on this dataset. We can explain this result as follows. First, there are few opportunities to apply PPSO to this dataset because it is well clustered and it is difficult to prune entire partitions. Second, when drilling down into the details of the experiment, we found that the baseline ANNS pruning rule is already fairly effective downplaying the effects of this optimization. On the other hand, the other strategy that prunes partitions during neighborhood search, PPSN, has shown to be the most effective strategy. First, PPSN has ample opportunity to be employed since it is a critical step during the evaluation of every point in the data. Second, the fact that this dataset is well behaved (clustered) actually significantly aids the performance of this pruning strategy as one would expect. One can also observe this from the fact that this pruning strategy is able to explain more than 60% of the data variation. Notice that there is a significative interaction between factors A and D, which reduces the effectiveness of ROCO to less than 50% (Q_A is equal to -20.95 and Q_{AD} is equal to 11.75), meaning that, since PPSN is being employed, the effectiveness of ROCO reduces significantly, but it is still relevant for the algorithm. Such behavior is explained by the fact that ROCO increases D_{min}^k fast so that a smaller number of objects search for neighbors out of the outlier candidate partition. When PPSN is applied, the high cost associated with searching for neighbors in other partitions is minimized and reduces the effectiveness of ROCO. As we observe for ROCO, ROCN also is a somewhat useful optimization in isolation but since it also interacts significantly with PPSN the overall benefits when compared to PPSN is low. The best performing strategy when viewed from raw execution time is to turn on all the optimizations, resulting in a speedup of about seven fold over the baseline.

ClusteredData with Noise: The database ClusteredData with noise synthesizes the ideal scenario where there are several well-defined clusters and few outliers. The results in this database (Tables 6 and 7(b)) were the only ones where PPSO provided significant execution time reductions. None of the strategy interactions are able to eliminate the need for any of the characteristics and, thus, the best configuration applies all strategies. However once again due to the repeated applicability of the optimization when searching for neighbors the PPSN strategy was found to be the most effective strategy (explaining 35% of the data variance in isolation) here followed by the ROCO, ROCN and PPSO strategies. Notice that the average execution time applying all strategies reduced the execution time from 141 s to 8 s.

CoverType: The evaluations using the database Forest Covertype (Tables 6 and 7(c)) again show that the most effective strategy for reducing the execution time was PPSN (in isolation, the execution time is 232.7s). This dataset also illustrates an interesting effect of the interactions among various optimization strategies that is unique to this kind of a study. For example, ROCN applied in isolation results in an execution time of 262.31s. ROCO in isolation results in an

Factors				Average Time in Datasets (s)					
A	B	C	D	CD[a]	CD w/noise[b]	Covertype[c]	KDDCup1999	Auctions[d]	Uniform30D
-1	-1	-1	-1	182.05	141.65	327.03	1317.15	57.88	2967.11
-1	-1	-1	+1	44.85	33.68	232.7	242.42	51.23	2944.05
-1	-1	+1	-1	171.76	100.41	329.47	1249.95	55.91	2900.83
-1	-1	+1	+1	49.66	26.79	214.71	207.79	51.31	2994.69
-1	+1	-1	-1	132.82	77.71	262.31	1249.49	44.01	1439.62
-1	+1	-1	+1	43.4	21.53	212.33	228.11	38.08	1394.74
-1	+1	+1	-1	134.12	70.71	263.04	1160.42	43.98	1421.29
-1	+1	+1	+1	42.84	14.66	206.13	203.07	42.31	1416.17
+1	-1	-1	-1	91.89	40.6	280.83	748.24	11.89	2933.52
+1	-1	-1	+1	26.15	17.15	183.5	215.32	11.1	2919.02
+1	-1	+1	-1	102.31	37.38	289.01	727.44	10.88	3010.55
+1	-1	+1	+1	29.49	13.11	196.38	179.5	10.4	2947.04
+1	+1	-1	-1	90.21	31.43	233.87	713.77	12.23	1356.74
+1	+1	-1	+1	26.85	11.58	**169.76**	223.59	**9.6**	**1350.59**
+1	+1	+1	-1	74.7	26.3	230.33	676.32	11.37	1395.32
+1	+1	+1	+1	**24.65**	**8.17**	183.94	**178.4**	10.27	1395.79

[a]Clustering Database

[b]Clustering Database with noise

[c]Forest Covertype

[d]Auctions Government dataset

execution time of 280.83s. However, given that both PPSN and ROCN target neighborhood search, it is expected that the two have a higher degree of interaction. This is also what we observe from the experimental results, and it is confirmed by the fact that Q_B is equal to -18.24 and Q_D is equal to -38.3, but Q_{BD} is equal to 11.35, highlighting the interaction among these two optimization strategies. On the other hand, while ROCO offers less benefits in isolation, it does not have a significant interaction with PPSN, thus suggesting that coupling those two optimizations yields a greater benefit than coupling PPSN and ROCN. It should be noted that, despite the interactions between ROCN, ROCO, and PPSN, their use is still relevant for reducing the execution time of the algorithms. Thus, the best configuration was the combination of ROCN, ROCO and PPSN, which achieved a speedup of about 2 over the baseline configuration. PPSO was found to be ineffective in the context of this dataset.

KddCUP: The KddCup database is quite large, but it contains large clusters of identical objects. As shown in Tables 6 and 7(d) we definitely see that PPSO does not provide significant gains. Again the rationale is the same as for the clustered datasets – there are few opportunities for applying it and, moreover, the interactions with the ANNS pruning rule make this strategy relatively ineffective. Surprisingly, in this dataset we also find that ROCN is not very useful either (the reduction in running time is about 60s). However, PPSN provides significant gains in isolation and also when it is combined with ROCO, although there is a somewhat significant interaction among them when combined. In order to further analyze these results in more detail, we verified the number of comparisons performed by the algorithm, where we observe that PPSN alone is very efficient, since it reduces the number of comparisons by a factor of 20 (from 2 billion to 100 million), explaining more than 70% of the data variation. An interesting observation here is that even though ROCN and PPSO do not help much in isolation,

when combined on top of the benefits realized from ROCO and PPSN they still yield significant benefits when viewed from the perspective of overall speedup (going from a 5 fold speedup to well over 6 fold speedup when all optimizations are enabled).

Government Auctions: When analyzing the results from the Auctions Government Database in Tables 6 and 7(e), we can see that PPSN was not effective. We believe that it happens because this database has several categorical attributes and our MINDIST definition is not able to estimate well the minimum distance between a point p and any object from a partition. In this database, the ROCO is very effective and, combined to ANNS, it achieves significative gains, decreasing running time from 57 to 12 seconds. The application of the other strategies is not able to improve the execution time significantly. Notice that, although ROCN provides gains for this database when applied in isolation, the interactions between ROCN and ROCO are quite significant and affect the algorithm's efficiency, since Q_A is -18.56, Q_B is -3.05, and Q_{AB} is 2.95, that is, even in the cases where ROCN is significant and are able to improve the execution time, when applied together with ROCO, it is not as effective and the execution time gains are very small. The best average time (9.6 seconds) is associated with the ABD configuration resulting in a 6 fold speedup over the baseline.

Uniform 30D: For the Uniform30D database (Tables 6 and 7(f)), we found ROCN to be the most effective, providing a speedup larger than 2, and the average execution time reduces from 2900 to 1400 seconds. All other strategies did not work very well on this dataset. Note that this is a worst-case situation for the pruning methods – since the opportunity to apply pruning will be quite limited in such datasets. The two ranking methods in conjunction provide the best overall benefits, realizing a speedup of 2.2 over the baseline.

Table 7: Factor Values on several Datasets

(a) Clustering Database

Factor	SS_i	SS_i (%)	Q_i
0	1004512.17		79.24
A	70244.51	14.43%	-20.95
B	10330.25	2.12%	-8.04
D	299262.32	61.48%	-43.25
AB	2381.00	0.49%	3.86
AD	22103.82	4.54%	11.75
BD	6727.95	1.38%	6.48
ABD	1785.43	0.37%	-3.34

(b) Clustering Database with noise

Factor	SS_i	SS_i (%)	Q_i
0	282955.88		42.05
A	56786.96	21.88%	-18.84
B	13818.15	5.32%	-9.29
C	3781.77	1.46%	-4.86
D	90027.69	34.69%	-23.72
AB	4749.95	1.83%	5.45
AC	1334.71	0.51%	2.89
AD	27070.40	10.43%	13.01
BD	3910.69	1.51%	4.94
ABD	2220.27	0.86%	-3.73

(c) Forest Covertype

Factor	SS_i	SS_i (%)	Q_i
0	9098036.87		238.46
A	49027.79	10.75%	-17.50
B	53256.79	11.68%	-18.24
D	237497.69	52.07%	-38.53
BD	20621.63	4.52%	11.35

(d) KDDCup1999

Factor	SS_i	SS_i (%)	Q_i
0	56655569.77		595.06
A	3013547.76	9.31%	-137.24
D	23751322.26	73.36%	-385.29
AD	2567085.61	7.93%	126.67

(e) Auctions Government

Factor	SS_i	SS_i (%)	Q_i
0	139497.65		29.53
A	55123.98	68.40%	-18.56
B	1484.46	1.84%	-3.05
AB	1389.72	1.72%	2.95

(f) Uniform30D

Factor	SS_i	SS_i (%)	Q_i
0	756337514.69		2174.19
B	96822880.87	98.56%	-777.91
AC	25349.34	0.03%	12.59

7. CONSOLIDATION AND RENEWED BEARING

We were able to show which strategies are more effective for each type of database and measure their impact in terms of execution time. It is fairly clear from our factorial design study results that PPSN is a very important optimization strategy among the ones we evaluated (note that ANNS was applied in the baseline). Such result is not that surprising as it has the most opportunity to be applied (during every neighborhood search process for every point evaluated). From Amdahl's principle and basic intuition it is what one would expect. It is worth emphasizing that PPSN originally proposed by Ramaswamy et al [22] was a part of their competitive strawman and not a part of the best performing algorithm on the datasets they evaluated on. Perhaps, as a result of such design decision, it is not always exploited by contemporary algorithms targeting this problem.

As our experimental results show, in most of the databases, PPSO has not been effective w.r.t. reducing the execution time. This result disagrees with Ramaswamy et al's findings [22], where the partition-based algorithm, whose main strategy is PPSO, outperformed both the Nested Loop and the index-based algorithms. However, we need to recall that, while Ramaswamy employed just ANNS to the Nested-Loop, we employed ANNS in all experimental configurations. In general, we were also able to apply ANNS reducing the effectiveness of the former whenever PPSO is employed. We believe that this may help partially explain the differences between our and Ramaswamy's results. We should also note that most of our results are on datasets with much larger dimensionality than those considered by Ramaswamy et al.

Another conclusion we reached is that the strategies employed in isolation are often not a good predictor of how they will behave in conjunction due to interaction effects. In most of the datasets, for example, we observed that the interaction between the strategies ROCN and PPSN was often positive thus limiting the overall effectiveness of combining them.

Finally, it is remarkable the fact that no strategy has shown to be effective for all databases. In the Auctions Government database we showed that the combination of ROCO and ANNS is able to achieve one of the best execution times. Another interesting case is that almost no strategy is able to reduce the execution time for Uniform30D database, what was expected since most of the strategies assume that there are well defined outliers. In this database, just ROCN + ANNS was able to be effective, being able to reduce the execution time by more than 50% (from 2900 to 1400 seconds). For the other datasets PPSN was the dominant optimization often in conjunction with one of the others. That said, we did observe significant improvements when we employed ROCN, PPSN and ROCO in conjunction with one another as can be seen from the summary table reporting all execution times. Factors ABD or ABCD are always the top performing approach in terms of overall speedup.

Moving forward, we do believe that there is scope for further analysis and further optimizations. In this work we did not consider the preprocessing time or the interactions between the partitioning and clustering algorithm and the optimizations and factors we considered. We believe that this is an important limitation of the current study but found that it is hard to handle all the parameters involved (which clustering algorithm, how many partitions etc.) as each of them can play a role in understanding the pros and cons of various factors.

In terms of optimizations, we believe that many steps discussed in this process can be further enhanced to yield greater performance benefits. One promising direction, in our opinion, is the use of Locality Sensitive Hashing [1] as a unified method to aid in both pruning and ranking strategies we discussed. It has also been shown to be useful for approximate nearest neighbor search algorithms which is a fundamental part of the canonical distance-based algorithms. We believe that this is a promising direction of further study and well worth pursuing. More generally faster methods to preprocess the data, faster methods to approximate the appropriate bounds of the two sides of the ANNS pruning rule are also worth pursuing.

8. CONCLUSIONS

In this work, we considered the domain of distance-based outlier detection, have exploited the design space of such algorithms, and identified a region in the design space that has been relatively unexplored.

We presented new strategies for ranking outliers and processing them in order of their approximate ranks and performed experimental results supporting that these ranking strategies could improve existing algorithms. Furthermore, as part of our quest to understand how various optimizations interact with one another, we conducted a full factorial design experiment implemented on the DIODE framework. From this experimental study we were able to draw several interesting conclusions. First, our proposed optimization is extremely useful for a range of datasets. Second, no single combination of various proposals (including our own) is always optimal. The specific combination that is optimal varies greatly by dataset characteristics but we did find that three of the optimization strategies proposed in the literature, when used in conjunction, typically provides good speedup.

We consolidate our results and provided some general directions for future study. We are currently performing a case study and identifying outliers in an Government Auction database, where these outliers may be associated with illegal transactions, fraud or even an incorrect information. We planned to investigate and propose much better ROC strategies since there is still room for enhancements and they may be achieved with simple strategies. Also, we are in the process of developing an adaptive parallel algorithm that executes local search for neighbors for a selected pool by ranking strategies before examining the data for outliers. We believe that this strategy may not need a specified threshold by the user and may prune out current overhead, thereby achieving linear speedup.

9. ACKNOWLEDGEMENTS

10. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[2] F. Angiulli and F. Fassetti. Very efficient mining of distance-based outliers. In M. J. Silva, A. H. F. Laender, R. A. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, and A. O. Falcão, editors, *CIKM*, pages 791–800. ACM, 2007.

[3] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *PKDD '02: Proc. of the 6th European Conf. on Principles of Data Mining and Knowledge Discovery*, pages 15–26, London, UK, 2002. Springer-Verlag.

[4] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *SIGKDD Explor. Newsl.*, 2(2):81–85, 2000.

[5] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *9th ACM SIGKDD Int. Conf. on Knowledge Discovery on Data Mining*, 2003.

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proc. of the 2000 ACM SIGMOD Int. Conf. on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 93–104. ACM, 2000.

[7] M. Ester, J. Kriegel, H. P.and Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial fatabases with noise. In *In Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[8] C. Faloutsos and K. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 163–174. ACM New York, NY, USA, 1995.

[9] A. Ghoting, S. Parthasarathy, and M. E. Otey. Fast mining of distance-based outliers in high-dimensional datasets. *6th SIAM Int. Conf. on Data Mining*, April 2005.

[10] A. Ghoting, S. Parthasarathy, and M. E. Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Min. Knowl. Discov.*, 16(3):349–364, 2008.

[11] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD '98: ACM SIGMOD Int. Conf. on Management of data*, pages 73–84, New York, NY, USA, 1998. ACM.

[12] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min. Knowl. Discov.*, 2(3):283–304, 1998.

[13] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB '99: 25th Int. Conf. on Very Large Data Bases*, pages 211–222, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[14] H. Kriegel, P. Kroger, and A. Zimek. Outlier Detection Techniques. In *Tutorial at the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2009.

[15] J. Laurikkala, M. Juhola, and E. Kentala. Informal identification of outliers in medical data. In *The Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*. Citeseer, 2000.

[16] M. Mahoney and P. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 376–385. ACM New York, NY, USA, 2002.

[17] M. Mahoney and P. Chan. Learning rules for anomaly detection of hostile network traffic. In *Proceedings of the Third IEEE International Conference on Data Mining*, page 601. Citeseer, 2003.

[18] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *20th Int. Conf. on Very Large Data Bases, 1994, Santiago, Chile*, pages 144–155. Morgan Kaufmann Publishers, 1994.

[19] K. Ord. Outliers in statistical data : V. barnett and t. lewis, 1994, 3rd edition, (john wiley & sons, chichester), isbn 0-471-93094. *Int. Journal of Forecasting*, 12(1):175–176, March 1996.

[20] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *19th International Conference on Data Engineering, 2003. Proceedings*, pages 315–326, 2003.

[21] Projeto Tamandua, 2006. http://tamandua.speed.dcc.ufmg.br/.

[22] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD '00: Proc. ACM SIGMOD Int. Conf. on Management of data*, pages 427–438, New York, NY, USA, 2000. ACM Press.

[23] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *SIGMOD '95: ACM SIGMOD Int. Conf. on Management of data*, pages 71–79, New York, NY, USA, 1995. ACM.

[24] P. Torr and D. Murray. Outlier detection and motion segmentation. *Sensor Fusion VI*, 2059:432–443, 1993.

[25] J. Tukey. *Exploratory data analysis*. Addison-Wesley, 1977.

[26] N. Vu and V. Gopalkrishnan. Efficient Pruning Schemes for Distance-Based Outlier Detection. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, page 175. Springer, 2009.

[27] M. Wu and C. Jermaine. A bayesian method for guessing the extreme values in a data set? In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 471–482. VLDB Endowment, 2007.

[28] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, 1996.