Mining Top-n Local Outliers in Large Databases*

Wen Jin Anthony K. H. Tung Jiawei Han
Intelligent Database Systems Research Lab
School of Computing Science
Simon Fraser University
Burnaby, B.C.,Canada V5A 1S6
Email: {wjin, khtung, han}@cs.sfu.ca

ABSTRACT

Outlier detection is an important task in data mining with numerous applications, including credit card fraud detection, video surveillance, etc. A recent work on outlier detection has introduced a novel notion of **local outlier** in which the *degree* to which an object is outlying is dependant on the density of its local neighborhood, and each object can be assigned a **Local Outlier Factor** (LOF) which represents the likelihood of that object being an outlier. Although the concept of local outliers is a useful one, the computation of LOFvalues for every data objects requires a large number of k-nearest neighbors searches and can be computationally expensive. Since most objects are usually not outliers, it is useful to provide users with the option of finding only n most outstanding local outliers, i.e., the top-n data objects which are most likely to be local outliers according to their LOFs. However, if the pruning is not done carefully, finding top-n outliers could result in the same amount of computation as finding LOFfor all objects. In this paper, we propose a novel method to efficiently find the top-n local outliers in large databases. The concept of "micro-cluster" is introduced to compress the data. An efficient micro-cluster-based local outlier mining algorithm is designed based on this concept. As our algorithm can be adversely affected by the overlapping in the micro-clusters, we proposed a meaningful cut-plane solution for overlapping data. The formal analysis and experiments show that this method can achieve good performance in finding the most outstanding local outliers .

1. INTRODUCTION

Outlier detection is an important data mining activity with numerous applications, including credit card fraud detection, discovery of criminal activities in electronic commerce, video surveillance, pharmaceutical research, and weather prediction.

An outlier is an observation that deviates so much from other observations so that it arouses suspicions that it is generated by a different mechanism [6]. Studies on outlier detection are numerous and can be grouped into five general categories. The first is **distribution-based**, where outliers are observations which deviate from a standard distribution (e.g., Normal, Poisson, etc.) [1]. Outlier detection can also be **depth-based** which relies on the computation of different layers of k-d convex hulls. In depth-based methods, outliers are objects in the outer layer of these hulls. The third category of outlier detection method is **distance-based**. Introduced by Knorr and Ng in [7], a distance-based outlier in a dataset D is an object with $pct\%$ of the objects in D having a distance of more than d_{min} away from it. This notion of distance-based outlier generalizes many notions from distribution-based approaches and is further extended in [9] so that outliers can be more efficiently discovered and ranked. In many clustering algorithms like [8], DBSCAN [4], BIRCH [10] and CURE [5], outliers are by-products of clustering and we refer to such outlier-detection method as clustering-based.

While the above four categories of outlier detection are interesting and useful in their own rights, our paper will only focus on the fifth category which detect **local outliers**-based on the local density of an object's neighborhood. We refer to this category as **density-based**. The concept of a local outlier is an important one since in many applications, different portions of a dataset can exhibit very different characteristics, and it is more meaningful to decide on the outlying possibility of an object based on other objects in its neighborhood. In view of this, [3] defines the concept of a **local outlier factor**(LOF), which is intuitively a measure of difference in density between an object and its neighborhood objects. Unfortunately, the work done in [3] requires the computation of LOFfor all objects which is rather expensive because it requires a large number of k-nearest-neighbors queries. As it is observed that many objects have LOFvalues which are very close and are unlikely to be interesting outliers, we believe that a system should provide users with the option of constraining a search to only the top-n local outliers instead of computing the LOFof every object in a database.

Given our motivation, it is obvious that finding LOFfor all objects and then selecting the top-n among them is not a solution in our consideration. However, if careful pruning is

*The work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC-A3723) and the Networks of Centres of Excellence of Canada (NCE/IRIS-3 and NCE/GEOID)

KDD 2001 San Francisco, California USA

not done, finding the top-n LOFcan be as expensive as the above naive method. This is because unlike global outlier detection in which a data point can be immediately pruned off if it cannot be an outlier, in local outlier detection the points deletion may affect the LOFvalues of those objects in its neighborhood. In short, we have a catch-22 situation here: Without knowing where the possible local outliers are, we cannot prune off our computation of density information; however, without computing density information, we do not know where the possible top-n outliers are.

In this paper, we first propose a novel method for top-n local outliers mining that avoid computation of LOFfor most objects if n<<N, N being the size of the database. The main idea in our solution is to compress the data into "micro-clusters" similar to those in [10] and efficiently estimate an upper bound and lower bound on LOFof each object in the database. By comparing the upper and lower bound for the LOFof each object(micro-cluster), it will be immediately clear that many of the objects cannot be in the list of top-n local outliers. As such, only a small amount of computation will be required for those candidate objects that are possibly among the top-n.

As a central idea of our algorithm, we will need to estimate the distance between a point o to a group of points in a micro-cluster. Since each micro-cluster is represented by a circle centered at the mean of its points, a very imprecise estimation will be made if the point o lies within the circle. To solve this overlapping problem, we introduce a novel "cut-plane" method to identify the boundary between a data point and a micro-cluster.

The paper is organized as follows. In section 2, we present the motivation and definition. In section 3, properties of micro-cluster are introduced. In Section 4, we propose a micro-cluster based top-n LOFalgorithm. In section 5, a performance evaluation is made and the results are analyzed. Section 6 concludes the paper.

2. DEFINITION OF LOCAL OUTLIERS

In this section, we briefly review the definition of local outliers. Interested reader are referred to [3] for more details.

Let D be a database. Let p, q, o be some objects in D. Let k be a positive integer. We use $d(p, q)$ to denote the **Euclidean distance** between objects p and q.

DEFINITION 1. **(k-distance of p)**
The k-distance of p, denoted as k-distance(p) is defined as the distance $d(p, o)$ between p and o such that:

1. *for at least k objects $o' \in D\backslash\{p\}$ it holds that $d(p, o') \leq d(p, o)$, and*
2. *for at most $(k-1)$ objects $o' \in D\backslash\{p\}$ it holds that $d(p, o') < d(p, o)$*

□

Intuitively, k-distance(p) provides a measure on the sparsity or density around the object p. When the k-distance of p is small, it means that the area around p is dense and vice versa.

DEFINITION 2. **(k-distance neighborhood of p)**
The k-distance neighborhood of p contains every object whose distance from p is not greater than the k-distance, is denoted as

$$N_k(p) = \{q \in D\backslash\{p\} |\ d(p,q) \leq k\text{-}distance(p)\}.$$

□

Note that since there may be more than k objects within k-distance(p), the number of objects in $N_k(p)$ may be more than k. Later on, the definition of LOF is introduced, and its value is strongly influenced by the k-distance of the objects in its k-distance neighborhood.

DEFINITION 3. **(reachability distance of p w.r.t object o)**
The reachability distance of object p with respect to object o is defined as

$$reach\text{-}dist\_k(p,o) = \max\ \{k\text{-}distance(o), d(p,o)\}.$$

□

DEFINITION 4. **(local reachability density of p)**
The local reachability density of an object p is the inverse of the average reachability distance from the k-nearest-neighbors of p.

$$lrd_k(p) = 1/[\frac{\sum_{o \in N_k(p)} reach - dist_k(p,o)}{|N_{k(p)}(p)|}]$$

□

Essentially, the local reachability density of an object p is an estimation of the density at point p by analyzing the k-distance of the objects in $N_k(p)$. The local reachability density of p is just the reciprocal of the average distance between p and the objects in its k-neighborhood. Based on local reachability density, the local outlier factor can be defined as follows.

DEFINITION 5. **(local outlier factor of p)**

$$LOF_k(p) = \frac{\sum_{o \in N_{k(p)}} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}$$

□

LOF is the average of the ratios of the local reachability density of p and those of p's k-nearest-neighbors. Inituitively, p's local outlier factor will be very high is it's local reachability density is much lower than those of its neighbors.

3. ALGORITHM FOR FINDING TOP-N LOCAL OUTLIERS

Given the earlier definition of local outliers, our problem is to find the top-n outliers in terms of LOF values when n and k are provided by the users. In this section, we will first analyze this problem and then introduce various concepts involved in our algorithm for finding top-n local outliers efficiently.

3.1 Problem Analysis

From the definitions given in the previous section, our analysis shows that an upper bound and lower bound on the LOF of an object p can be obtained if an upper and lower bound on the local reachability density of p and objects in $N_k(p)$ are available. More specifically, we have the following theorem:

THEOREM 3.1. *Let $lrd_k(o)$.lower and $lrd_k(o)$.upper denote the lower and upper bound on the local reachability density of an object o, and, $o \in N_k(p)$, then*

$$\frac{Min\{lrd_k(o).lower\}}{lrd_k(p).upper} < LOF_k(p) < \frac{Max\{lrd_k(o).upper\}}{lrd_k(p).lower}$$

Proof: Since LOF is the average of the ratios of $lrd_k(p)$ and its k-neighborhood objects, this average must be higher than $Min\{lrd_k(o)|o \in N_k(p)\}/lrd_k(p)$. By taking the lower bound for $lrd_k(o)$ and the upper bound for $lrd_k(p)$, we reduce this estimate even further and thus we obtained a lower bound for $LOF_k(p)$. By similar reasoning, we will obtain an upper bound for $LOF_k(p)$ by taking the maximum of $lrd_k(o).upper$ and dividing it by $lrd_k(p).lower$. □

As a result of Theorem 3.1, we have the following corollary which we will use to find an upper bound and lower bound for the local reachability density of every object.

COROLLARY 3.1. *Given an object p, $o \in N_k(p)$, its reachability distance with respect to its k-neighborhood is bounded:*

$$\frac{1}{Max\{reach\text{-}dist_k(p,o)\}} \leq lrd_k(p) \leq \frac{1}{Min\{reach\text{-}dist_k(p,o)\}}$$

Explanation: *Since the local reachability density of p is the inverse of the average of reachability distance, it is not difficult to see that the average will be lower than the maximum of reach-dist$_k(p,o)$ and higher than the minimum of reach-dist$_k(p,o)$, $o \in N_k(p)$. Thus taking their inverse will give the respective upper and lower bound.* □

Given the above corollary, we will now have to use the following theorem to bound the maximum and minimum value of $reach\text{-}dist_k(p,o)$. We derive the following theorem:

THEOREM 3.2. *Let the upper and lower bound on the k-distance of object o be denoted by k-distance(o).upper and k-distance(o).lower respectively. We use d(o,p).upper and d(o,p).lower to denote the upper, lower bound on the distance between object o and p, then we have the following two inequalities*

$$Max(d(o,p).lower, k\text{-}distance(o).lower) \leq reach\text{-}dist(p,o)$$

and

$$reach\text{-}dist(p,o) \leq Max(d(o,p).upper, k\text{-}distance(o).upper)$$

□

The above theorem is derived naturally based on the definition of reachability distance. With the derivation of the above theorem, we can see that the computation of an upper and lower bound on the LOF of all points in a dataset is dependent on an efficient way to find an upper and lower bound on the distance between points and the k-distance of each point.

3.2 Concepts

We will now look at the various concepts used in our algorithm for finding top-n local outliers. The concept we use is called "micro-clustering" which is similar to those in [10]. We compress the data into small clusters and represent each small cluster using some statistical information. We define a **micro-cluster** as follows:

DEFINITION 6. **(Micro-Cluster)**
A micro-cluster $MC(n,c,r)$ is a summarized representation of a group of data $p_1, \ldots, p_n$, which are so close together that they are likely to belong to the same cluster. Here, $c = \frac{\sum_{i=1}^{n} p_i}{n}$, is the mean center while $r = max\{d(p_i,c)\}$, i=1, ..., n, is the radius. □

To ensure that not too much accuracy is sacrificed by using micro-cluster, we limit radius of each micro-cluster to be below a user-specified threshold, maxradius. When a micro-cluster with a radius higher than maxradius is found, it is split by choosing two data that are farthest apart in the micro-cluster as seeds and reassigning other data left to the nearest seed. Since we use a micro-cluster to represent a dataset in outlier detection, the distance measurement to a micro-cluster must also be defined.

In the rest of this paper, let D be a database and M be a set of micro-clusters. A point in D is denoted as $p(p_1, ..., p_m)$ and a micro-cluster in M is denoted as $MC(n,c,r)$.

THEOREM 3.3. *Let p be an object and $MC(n,c,r)$ a micro-cluster. If p is greater than a distance of r from c, then the minimum distance between an object p and a micro-cluster MC will be:*

$$Dist_{Min}(p, MC) = d(p,c) - r$$

and the maximum distance between p and MC will be:

$$Dist_{Max}(p, MC) = d(p,c) + r$$

□

Figure 1 illustrates the maximum and minimum distance between a point p and a micro-cluster, $MC(n,c,r)$ when p is not within a distance of r from c. Intuitively, any point within the micro-cluster $MC(n,c,r)$ will be at a distance of at least $Dist_{Min}(p, MC)$ and at most $Dist_{Max}(p, MC)$ from p. The situation is different if p is within a distance of r from the center of the micro-cluster. In such case, while the maximum distance remains unchanged, the minimum distance must be set to 0. Such a situation, however, is undesirable since we will be estimating the k-distance of a point based on its distance to its neighboring micro-clusters, and a k-distance of 0 will mean extremely high density. To overcome this problem, our solution is to ensure that each object in a micro-cluster $MC(n,c,r)$ is in fact nearest to c than to the center of any other micro-clusters. This can be achieved by first forming micro-clusters using an algorithm like BIRCH [10], fixing the centers of each micro-cluster and then do a simple redistribution of the objects to the nearest center. Note that such a process can be efficiently done since the BIRCH structure resides in the main memory. With such an assumption, we will now define the concept of a **cut plane**.

DEFINITION 7. **(Cut-Plane)**
Let MC_i, MC_j be two micro-clusters, a cut-plane *for MC_i and MC_j, denoted as $cp(MC_i, MC_j)$, is a hyperplane that is perpendicular to the line between c_i and c_j and divide the line into exactly two halves.*

□

We illustrate a cut plane between two micro-clusters MC_i and MC_j in Figure 2. With the definition of a cut-plane, we will now define the minimum distance between p and a micro-cluster $MC_j(n_j, c_j, r_j)$ when p is within a distance of r_j from c_j.

DEFINITION 8. **(Min distance between an object and a micro-cluster with overlapping)**
Let p be an object within a micro-cluster $MC_i(n_i, c_i, r_i)$. If p is within a radius of r_j to a micro-cluster $MC_j(n_j, c_j, r_j)$,

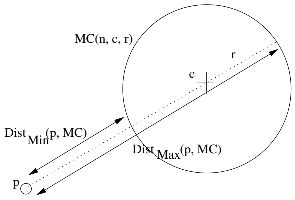

Figure 1: Min/Max distance between an object and a micro-cluster without overlapping.

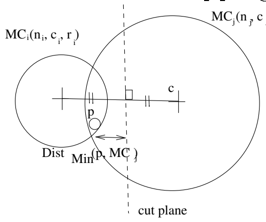

Figure 2: Minimum distance between an object and a micro-cluster with overlapping.

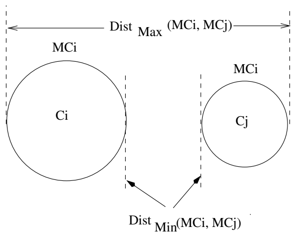

Figure 3: Min/Max distance of micro-clusters.

let $d(p, cp(MC_i, MC_j)$ denotes the perpendicular distance of p from cut-plane $cp(MC_i, MC_j)$. Then the minimum distance between an object p and a micro-cluster MC_j is $d(p, cp(MC_i, MC_j))$. □

We illustrate the computation of the minimum distance between p and an overlapping micro-cluster MC_j in Figure 2. Note that such a formulation is based on the assumption that p is always nearer to the center of MC_i than MC_j. We now define the minimum and maximum distance between two micro-clusters.

DEFINITION 9. **(Min/max distance between MC_i and MC_j)**
Let $MC_i(n_i, c_i, r_i)$ and $MC_j(n_j, c_j, r_j)$ be two micro-clusters, then the minimum distance between MC_i and MC_j is:

$$Dist_{Min}(MC_i, MC_j) = d(c_i, c_j) - r_i - r_j$$

and the maximum distance between MC_i and MC_j is,

$$Dist_{Max}(MC_i, MC_j) = d(c_i, c_j) + r_i + r_j$$ □

Note that the definition assumes that the two micro-clusters have no overlap. When overlap occurs, then the minimum distance between the two micro-clusters will be 0. Having defined the minimum and maximum distance between two micro-clusters, we will now provide a corollary which will be used to find the upper and lower bound for the k-distance of an object with respect to the micro-clusters around it.

COROLLARY 3.2. *Let p be an object and $MC(n, c, r)$ be the micro-cluster that contains p. Let $MC_1(n_1, c_1, r_1)$, ... , $MC_l(n_l, c_l, r_l)$ be a set of micro-clusters that could* **potentially** *contain the k -nearest neighbors of p. For ease of discussion, we will treat the other $(n-1)$ objects as micro-clusters, i.e. each object o_i is a micro-cluster $MC_i(1, o_i, 0)$. Thus we will now have $l+n-1$ micro-clusters.*

1. *Let $\{Dist_{Min}(p, MC_1), \ldots, Dist_{Min}(p, MC_{l+n-1})\}$ be sorted in increasing order, then a lower bound on the k-distance of p, denoted as k_{min}-distance(p) will be $Dist_{Min}(p, MC_i)$ such that $n_1 + \ldots + n_i \geq k$, and $n_1 + \ldots + n_{i-1} < k$*

2. *Let $\{Dist_{Max}(p, MC_1), \ldots, Dist_{Max}(p, MC_{1+n-1})\}$ be sorted in increasing order, then an upper bound on the k-distance of p, denoted as k_{max}-distance(p) will be $Dist_{Max}(p, MC_i)$ such that $n_1 + \ldots + n_i \geq k$ and $n_1 + \ldots + n_{i-1} < k$.*

□

Given a micro-cluster MC containing $p_1, \ldots, p_n$, we will use k_{max}-*distance*(MC) to denote Max(k_{max}-distance(p_1), ... , k_{max}-distance(p_i)) and k_{min}-*distance*(MC) to denote Min(k_{min}-distance(p_1), ... , k_{min}-distance(p_i)). We now derive a bound for the internal reachability of a micro-cluster.

DEFINITION 10. **(Internal reachability bound of a micro-cluster)**
We define the internal reachability bounds of a micro-cluster $MC(n, c, r)$ as follows:

1. *$r_{max}(MC) = Max(2r, Max(k_{max}$-distance$(MC)))$*

2. *$r_{min}(MC) = k_{min}$-distance(MC)*

□

Intuitively, given two objects p and o within micro-cluster MC, $r_{max}(MC)$ and $r_{min}(MC)$ respresents the maximum and minimum bound for *reach-distance*(p, o). This definition is used for estimating the reachability distance bound of a pair of data within one micro-cluster.

DEFINITION 11. **(External reachability bound of two micro-clusters)**
We define the external reachability bounds of a micro-cluster MC_i with respect to another micro-cluster MC_j as follow:

1. *$r_{max}(MC_i, MC_j) =$
$Max(Dist_{max}(MC_i, MC_j), k_{max}$-distance$(MC_j))$*

2. *$r_{min}(MC_i, MC_j) =$
$Max(Dist_{min}(MC_i, MC_j), k_{min}$-distance$(MC_j))$*

□

Intuitively, given two objects p and o within micro-cluster MC_i and MC_j respectively, r_{max} and r_{min} represents the maximum, minimum bound for *reach-distance*(p, o). This definition is used for estimating the reachability distance bound of a pair of data in different micro-clusters. One thing to note is that the external reachability bound of two micro-clusters will **NOT** be affected by overlapping between the micro-clusters as long as we have good estimation of the k-distance bound for the objects inside the micro-clusters. Having introduced the various concepts, we will have a look at our algorithm in the next section.

4. MICRO-CLUSTER-BASED ALGORITHM

In this section, we look at our micro-cluster-based algorithm for finding top-n local outliers. The general steps of our algorithm is as follows:

1. Preprocessing.
2. Computing LOF bound for micro-clusters.
3. Rank *top-n* local outliers.

4.1 Preprocessing

In this step, the input data are pre-processed so that efficient determination of k-distance bound can be done. Preprocessing can further be divided into three steps as follows.

1. Load data into CF Tree. A sequential scan is performed on the database and the objects are inserted into a CF-tree [10]. At the end of the process, the centers of all the CFs are recorded and inserted into a memory-based X-tree [2].

2. Fix CF node and generate micro-clusters. A second scan is then performed on the database and the objects are assigned to their nearest centers computed in the previous steps. In the process, the number of objects and the radius of each micro-cluster are recorded.

3. Insert micro-clusters into X-tree. Each micro-cluster is bounded by its MBR and is inserted into a different memory-based X-tree from step (1).

4.2 Computing LOF Bound for Micro-Clusters

To compute an upper and lower bound for micro-clusters, it requires one scan through the database and one scan through the micro-clusters. The scan through the database will first estimate a bound for the k-distance for each micro-cluster MC, i.e., k_{max}-$distance(MC)$ and k_{min}-$distance(MC)$. The second scan will determine a bound for the reachability distance and thus the LOF for each micro-cluster. We describe the details as follows.

ALGORITHM 1. *Algorithm k-distance-bound.* **Input:** *A set of micro-clusters* $MC_1, \ldots, MC_l$.
Output: k_{max}-$distance(MC_i)$, k_{min}-$distance(MC_i)$ $1 \le i \le l$.
Method:

1. **for each** *micro-cluster* MC_i **do**
2. *find a micro-clusters set* P *of* MC_i *by Colloary 3.2;*
3. k_{min}-$distance(MC_i) = \infty$; k_{max}-$distance(MC_i) = 0$;
4. **for each** *object* p *in* MC_i **do**
5. *get* $k_{min}(p)$, $k_{max}(p)$ *w.r.t* P *& objects in* MC_i.
6. **if** $k_{min}(p) < k_{min}$-$distance(MC_i)$
7. k_{min}-$distance(MC_i) = k_{min}(p)$;
8. **if** $k_{max}(p) > k_{max}$-$distance(MC_i)$
9. k_{max}-$distance(MC_i) = k_{max}(p)$;

This algorithm is used to compute the k-distance bound for each micro-cluster. Since the micro-clusters are in an X-tree, step 2 can be efficiently performed [2] by examining the MBR of the micro-clusters and computing the cut-plane between two micro-clusters if needed.

After computing the k-distance bound for each micro-cluster, we scan through the micro-clusters to compute the LOF bound using Algorithm k-distance-bound.

ALGORITHM 2. *Algorithm LOF-bound.*
Input: $MC_1, \ldots, MC_l$ *and their k-distance-bound.*
Output: *LOF bound for each micro-cluster.*
Method:

1. **for each** *micro-cluster* MC_i **do**
2. *find a micro-clusters set* P *of* MC_i *by Colloary 3.2;*
3. *get internal* $r_{max}(MC_i)/r_{min}(MC_i)$*bound for* MC_i.
4. $LOF(MC_i).upper = r_{max}(MC_i)/r_{min}(MC_i)$;
   $LOF(MC_i).lower = r_{min}(MC_i)/r_{max}(MC_i)$;
5. **for each** *micro-cluster* $MC_j \in P$ **do**
6. *get external* $r_{max}(MC_i, MC_j)$,$r_{min}(MC_i, MC_j)$
7. **if** $LOF(MC_i).upper < r_{max}(MC_i, MC_j)/r_{min}(MC_i, MC_j)$
8. $LOF(MC_i).upper = r_{max}(MC_i, MC_j)/r_{min}(MC_i, MC_j)$;
9. **if** $LOF(MC_i).lower > r_{min}(MC_i, MC_j)/r_{max}(MC_i, MC_j)$
10. $LOF(MC_i).lower = r_{min}(MC_i, MC_j)/r_{max}(MC_i, MC_j)$;

This algorithm is used to compute the LOF bound of a micro-cluster. In steps 3 and 4, it first handles data within one micro-cluster. Then in steps 6-10, it considers those potential neighbor micro-clusters to obtain their lower and upper bounds.

4.3 Rank Top-n Local Outliers

Given an upper and lower bound for the LOF of micro-cluster, we can easily rank Top-n local outliers.

ALGORITHM 3. *Algorithm Rank-TOPn-LOF.*
Input: *A set of* $MC_1, \ldots, MC_l$ *and their LOF_bound.*
Output: *TOP-n local outliers.*
Method:

1. *Sort the first n micro-clusters in ascending LOF.lower order label the minimal one Min-TOPn-LOF;*
2. **for** *any other micro-clusters* MC_i **do**
3. **if** $LOF(MC_i).upper <$ *Min-TOPn-LOF*
4. **then** *delete* MC_i;
5. **else if** $LOF(MC_i).lower >$ *Min-TOPn-LOF*
6. **then** *delete the current sorted n-th micro-cluster;*
7. *add* MC_i *into current top n sorted micro-clusters;*
8. *re-sort current n micro-clusters,*
   label the n-th as Min-TOPn-LOF;
9. **for** *any data in the remaining micro-clusters* MC'_i **do**
10. *calculate detailed LOF value;*
11. *prune those that are impossible to become TOP-n LOFs;*
12. *Obtain TOP-n local outliers;*

5. EXPERIMENTS

We compare our micro-cluster TOP-n LOF algorithm with X-tree-based LOF method [3]. We did experiments on both real and synthetic data sets but will only show the result for the synthetic data due to lack of space. The experiments are conducted on an PentiumIII-450 PC with 256MB main memory under Windows NT4.0 and implemented in Visual C++6.0. The cost of time in the experiments includes building an index tree.

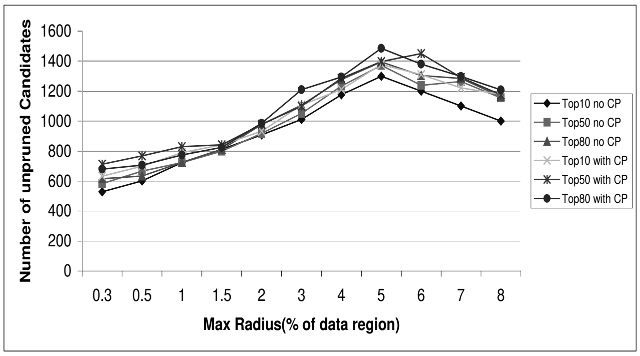

Figure 4: No. of unpruned candidates vs max-radius

The synthetic datasets we used for our experiments are generated using Gaussian random distribution. We evaluate the performance based on three aspects.

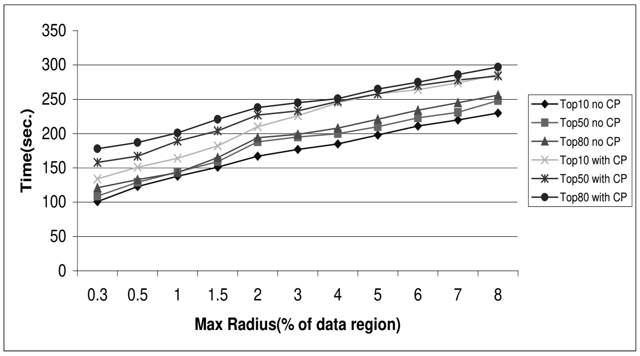

Figure 5: Running time vs max-radius

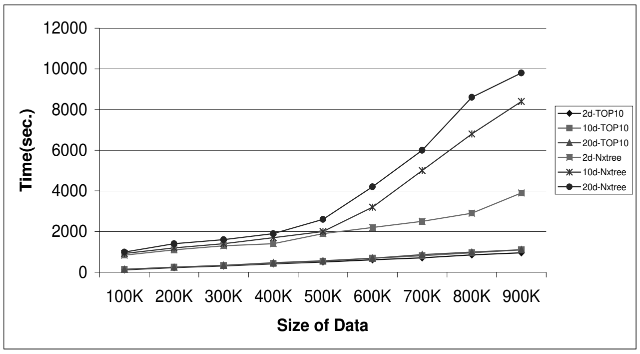

Figure 6: Running time vs.dataset size

First, we investigate the pruning effect of TOP-n LOF under the different max-radius of micro-clusters. Here the max-radius is viewed as the percentage of the dataset's radius. The number of candidate micro-clusters are plotted against max-radius for $n = 10$, $n = 50$ and $n = 80$ in Figure 4 for two versions of our algorithm, one with cut plane (abbreviated as "with CP") and one without (abbreviated as "without CP"). When a cut-plane is considered, it shows that more micro-clusters will be pruned since much of the overlapping data, which originally could not be separated, now has a discernible distance from other data. In addition, when the max-radius of a micro-cluster increases, less micro-clusters will be pruned since more accuracy is lost and the lower and upper bounds for the LOF of these micro-clusters are more interleaved . Subsequently, as max-radius continue to increase, the number of unpruned candidates decrease, this is due to the overall decrease in the total number of micro-clusters rather than an effect of our pruning.

Second, we plot the runtime of our algorithms against the size of micro-clusters in Figure 5. It is clear that when cut-plane is used, the runtime is usually lower than when it is not used since more micro-clusters have been pruned. We can also see that although there are very few candidate micro-clusters when the max-radius is large, the running time at these values still increase. This is because search and computation time for each of these micro-cluster is large as each of them contains more data points,

Third, to investigate the scalability of our algorithm with regard to dimensionality and dataset size, we plot the running time of our algorithm for a top-10 local outlier query against the size of the dataset in Figure 6. The dimensionality of the dataset are set at 2, 10 and 20. The graph shows that the micro-cluster TOP-n LOF mining method outperforms naïve X-tree-based method by a big margin especially for high number of dimensions.

6. CONCLUSIONS

In this paper, we have proposed a novel and efficient method for mining top-n local outliers. The strength of the method is at that it avoids computation of LOFfor most objects if $n << N$, where N is the size of the database. The formal analysis and performance evaluation show that our method is not only efficient in computation but also effective at ranking most understandable local outliers.

Finding top-n local outliers is a new and promising research topic in data mining. The future work may include finding strong local outlier groups and finding (nested) local outlier at multiple levels of granularity.

Acknowledgment: Thanks to Helen Pinto and Joyce Lam for proof reading the initial version of this paper.

7. REFERENCES

[1] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley & Sons, 1994.

[2] S. Berchtold, D. Keim, and H.-P. Kriegel. The X-tree: An efficient and robust access method for points and rectangles. In *Proc. 1996 Int. Conf. Very Large Data Bases (VLDB'96)*, pages 28–39, Bombay, India, Sept. 1996.

[3] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, Dallas, Texas, 2000.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, Portland, Oregon, Aug. 1996.

[5] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 73–84, Seattle, WA, June 1998.

[6] D. Hawkins. *Identification of Outliers.* Chapman and Hall, London, 1980.

[7] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. 1998 Int. Conf. Very Large Data Bases (VLDB'98)*, pages 392–403, New York, NY, Aug. 1998.

[8] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 144–155, Santiago, Chile, Sept. 1994.

[9] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, Dallas, Texas, 2000.

[10] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pages 103–114, Montreal, Canada, June 1996.