UNCERTAIN DATA CLUSTERING THROUGH VORONOI DIAGRAMS & R-TREE INDEX

K.Radhika[1], G. Bhargavi[2], S.Srinivasulu[3]

[1]*Student, Computer Science and Engineering, Prakasam Engineering College,A.P.,India,*
radhika.konduru0125@gmail.com
[2]*Associate Professor, Computer Science and Engineerin, Prakasam Engineering College, A.P.,India,*
gbhargavi2007@rediffmail.com
[3]*Head of the Department, Computer Science and Engineering, Prakasam Engineering College,A.P.,India,*
sreenivasulusadineni@gmail.com

Abstract

Increasing quantity of data with uncertainty has been arising from various applications such as sensor network measurements, record linkage, and as output of mining algorithms. There is a problem in clustering uncertain objects whose locations are described by probability density functions (pdfs). Handling of uncertain objects is very inefficient using UK-means algorithm because, the UK-means computes expected distances (EDs) between objects and cluster representatives. Numerical integrations are used to compute arbitrary pdfs, expected distances. These are costly operations. In this paper, the pruning techniques that are based on Voronoi diagrams are proposed to reduce the number of expected distance calculations. We analytically proven that These are more effective than the basic bounding-box-based techniques previously defined. Then R-tree index is introduced to organize the uncertain objects so as to reduce pruning overheads. Experiments are conducted to evaluate the effectiveness of these techniques.

Index Terms: *network management, mining algorithms, uncertainty, probability density functions, expected distances clustering, R-tree index.*

---------- *** ----------

1. INTRODUCTION

Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters. Clustering is a main task of explorative data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

The two key features of clustering through *k*-means which take it efficient are often regarded as its biggest drawbacks:

- Euclidean distance is used as a metric and variance is used as a measure of cluster scatter.
- The number of clusters *k* is an input parameter: an inappropriate choice of *k* may yield poor results. That is why, when performing k-means, it is important to run diagnostic checks for determining the number of clusters in the data set.
- Convergence to a local minimum may produce counterintuitive ("wrong") results.

Many efficient algorithms have been devised to solve the clustering problem efficiently. Traditionally, clustering algorithms deal with a set of objects whose positions are accurately known. The goal is to find a way to divide objects into clusters so that the total distance of the objects to their assigned cluster centers is minimized.

Although it is simple, the problem model does not address the situations where object locations are uncertain. i.e., data uncertainty arises naturally and often inherently in many applications. For example, physical measurements can never be 100 percent precise in theory. Thus, in practice, there are limitations in measuring devices induce uncertainty to the measured values.

As an example, in the application of clustering a set of mobile devices, a leader can be elected for each cluster, which can then coordinate the work within its cluster after grouping mobile devices into clusters. In this, the data may be collected by a cluster leader from its cluster's members, this data may

be processed, and the data can be sent to a central server via an access point in batch. Only short-ranged signals are required in local communication within a cluster for which a higher bandwidth is available. Batch communication can be used in long-ranged communication between the leaders and the mobile network. This results in better bandwidth utilization and energy conservation.

In fact, the device locations are uncertain. A mobile device may deduce and report its location by comparing the strengths of radio signals from mobile access points. Unfortunately, such deductions are susceptible to noise. Furthermore, locations are reported periodically. A location value is unknown and can only be estimated by considering a last reported value between two sampling times instances. Typically, such an uncertainty model considers factors such as the speed of the moving devices and other geometrical constraints (such as road network, etc.). In other applications (such as tracking animals using wireless sensors), the whereabouts of objects are sampled. The samples of a mobile object are collected to construct a probability distribution that indicates the occurrence probabilities of the object at various points in space.

In this paper, we consider the problem of clustering uncertain objects whose locations are specified by uncertainty regions over which arbitrary probability density functions (pdfs) are defined. And also concentrated on the problem of clustering objects with location uncertainty. an object is represented by a pdf over the space R^m rather than a single point in space. It is assumed that each object is confined in a finite region so that the probability density outside the region is zero. Thus each object can be bounded by a finite bounding box. This assumption is realistic because, the probability density of an object is high only within a very small region of concentration. The probability density is negligible outside the region. (For example, the uncertainty region of a mobile device can be limited by the maximum speed of the device.)

The problem of clustering uncertain objects was first described in [5] in which the UK-means algorithm was proposed. UK-means is a generalization of the traditional k-means algorithm to handle objects with uncertain locations. It is an iterative procedure. Each iteration consists of two steps.

In step 1, each object o_i is assigned to a cluster whose representative (a point) is the one closest to o_i among all representatives. This step can be called as cluster assignment.

In step 2, the representative of each cluster is updated by the means of all the objects that are assigned to the cluster. In cluster assignment, the closeness between an object and a cluster is measured by some simple distance such as euclidean distance.

The difference between UK-means and k-means is that under UK-means, objects are represented by pdfs instead of points. Also, UK-means computes expected distances (EDs) between objects and cluster representatives instead of simple euclidean distances during the cluster assignment step. Since many EDs are computed, the major computational cost of UK-means is the evaluation of EDs, which involves numerical integration using a large number of sample points for each pdf. To improve efficiency, Ngai et al. [6] introduced some pruning techniques to avoid many ED computations. The pruning techniques make use of bounding boxes over objects as well as the triangle inequality to establish lower and upper bounds of the EDs. Using these bounds, some candidate clusters are eliminated from consideration when UK-means determines the cluster assignment of an object. The corresponding computation of expected distances from the object to the pruned clusters is thus not necessary and is avoided.

An important contribution of this paper is the introduction of a new set of pruning techniques for the UK-means algorithm that are based on Voronoi diagrams [7]. These new pruning techniques take into consideration the spatial relationship among the cluster representatives. We prove that Voronoi-diagram-based technique is theoretically more effective than the basic bounding-box-based technique.

Another technique we propose in this paper is the partial ED evaluation method, which can be shown to further save the computation costs of UK-means. Indeed, our pruning techniques are so effective that over 95 percent of the ED evaluations can be eliminated [8]. With a highly effective pruning method, only few expected distances are computed, and thus, the once dominating ED computation cost no longer occupies the largest fraction of the execution time. The overheads incurred in realizing the pruning strategy (e.g., the testing of certain pruning conditions) now become relatively significant. To further reduce execution time, we have developed a performance boosting technique based on R-trees index. Instead of treating the uncertain objects as an unorganized set of objects, as in the basic k-means algorithm and derivatives, we index the uncertain objects using a bulk loaded R-tree. Each node in the R-tree represents a rectangular region in space that encloses a group of uncertain objects. The idea is to apply Voronoi-diagram-based pruning techniques on an R-tree node instead of on each individual object that belongs to the node. Effectively, we prune in batch. Our experiment shows that using an R-tree significantly reduces pruning overheads.

It is important to note that these two types of techniques have different goals. The Voronoi-diagram-based techniques aim at reducing the amount of ED calculations at the expense of some "pruning cost." The R-tree-based booster aims at reducing this pruning cost, which becomes dominant once the number of ED calculations have been tremendously reduced.

Since our pruning techniques are orthogonal to the ones proposed in [6], it is possible to integrate the various pruning techniques to create hybrid algorithms. Our empirical study shows that the hybrid algorithms achieve significant performance improvement.

2 DEFINITIONS

Consider a set of objects O = {o_1, o_2, . . . o_n} in an m-dimensional space R^m with a distance function d : R^m X R^m → R giving the distance d(x,y) ≥ 0 between any points x, y ∈ R^m. Associated with each object is a pdf f_i : R^m → R, which gives the probability density of o_i at each point x ∈ R^m. By the definition of pdf, we have (for all i = 1,2, . . . , n)

$f_i(x) \geq 0 \quad \forall x \in R^m$

$$\int_{x \in R^m} f_i(x)dx = 1$$

Further, we assume that the probability density of o_i is confined in a finite region A_i so that $f_i(x) = 0$ for all x ∈ R^m \ A_i.

We define the expected distance between an object o_i and any point y ∈R^m:

$$ED(o_i,y) = \int_{x \in A_i} d(x,y) f_i(x) dx$$

Now, given an integer constant k, the problem of clustering uncertain data is to find a set of cluster representative points C={c_1, c_2,. . . ,c_k } and a mapping h : {1, . . . , n} → {1, . . ., k} so that the sum of squared expected distance $\sum[ED(o_i, c_{h(i)})]^2$ is minimized.

To facilitate our discussion on bounding-box-based algorithms, we use MBR_i to denote the minimum bounding rectangle of object o_i. MBR_i is the smallest box, with faces perpendicular to the principal axes of R^m, that encloses A_i. Note that it still holds if we replace "x ∈ Ai" with "x ∈ MBR_i." This fact can be exploited for optimization when computing ED.

3 ALGORITHMS

We first give a short description of the UK-means algorithm and existing pruning techniques that improve Kmeans. Then, we present our new pruning techniques that are based on Voronoi diagrams, and finally, we introduce our performance booster based on R-trees.

3.1 UK-Means

UK-means (see Algorithm 1) is an adaptation of the wellknown k-means algorithm to handle data objects with uncertain locations.

Algorithm 1. UK-means
1: Choose k arbitrary points as c_j (j = 1, . . . , k)
2: repeat
3: for all o_i ∈ O do /*assign objects to clusters*/
4: for all c_j ∈C do
5: Compute ED(o_i, c_j)
6: h(i) ← arg $\min_{j:c_j \in C} \{ED(o_i, c_j)\}$
7: for all j = 1, . . . , k do/*readjust cluster representatives*/
8: c_j ← centroid of {o_i ∈ O | h(i) = j}
9: until C and h become stable

Initially, k arbitrary points c_1, . . . , c_k are chosen as the cluster representatives. Then, UK-means repeats the following steps until the result converges. First, for each object o_i, ED(o_i,c_j) is computed for all c_j ∈ C. Object o_i is then assigned to cluster c_{j^*} that minimizes ED, i.e., h(i) ← j*. Next, each cluster representative c_j is recomputed as the centroid of all o_is that are assigned to cluster j. The two steps are repeated until the solution C = (c_1, . . . , c_k) and h(.) converge.

The UK-means algorithm is inefficient. This is because UK-means computes ED for every object-cluster pair in every iteration. So, given n objects and k clusters, UK-means computes nk EDs in each iteration. The computation of an ED involves numerically integrating a function that involves an object's pdf. In practice, a pdf is represented by a probability distribution matrix, with each element of the matrix representing a sample point in an MBR. To accurately represent a pdf, a large number of sample points are needed. The computation cost of an integration is thus high.

To improve the performance of UK-means, we need to reduce the time spent on ED calculations, because it dominates the execution time of the algorithm. One way to achieve this is to avoid ED computations whenever possible. To incorporate pruning into UK-means, we replace lines 3-6 in Algorithm 1 with the following:

1: Q_i ←C /*candidate clusters*/
2: Apply a pruning algorithm
3: if $|Q_i| = 1$ then /*only one candidate remans*/

4: h(i) ← j where $c_j \in Q_i$
5: else
6: for all $c_j \in Q_i$ /*remaning candidates*/
7: Compute ED(o_i, c_j)
8: h(i) ← arg $\min_{j:c_j \in Q_i} \{ED(a_i, c_j)\}$

For a given object o_i, the set Q_i stores the set of candidate

cluster representatives that are potentially the closest to o_i. Initially, Q_i = C, the set of all cluster representatives. In line 2, a pruning algorithm is applied to prune candidate representatives from Q_i that are guaranteed to be not the closest to object o_i. If all but one candidate cluster remains in Q_i, object o_i is assigned to that cluster. Otherwise, we compute the expected distances between o_i and each remaining cluster in Q_i. Object o_i is then assigned to the cluster that gives the smallest expected distance. We describe a few pruning algorithms in the following sections. A good pruning algorithm should desirably reduce the set Q_i to a very small cardinality so that the number of ED calculations that need to be performed in line 7 is as few as possible.

3.2 MinMax Pruning

Several pruning techniques that are based on bounds on ED have been proposed in [6]. In the MinMax approach, for an object o_i and a cluster representative c_j, certain points in MBR_i are geometrically determined. The distances from those points to c_j are computed to establish bounds on ED. Formally, we define

$$\text{MinD}(o_i, c_j) = \min_{x \in MBR_i} d(x, c_j)$$

$$\text{MaxD}(o_i, c_j) = \max_{x \in MBR_i} d(x, c_j)$$

$$\text{MinMaxD}(o_i) = \min_{c_j \in C} \{MaxD(o_i, c_j)\}$$

It should be obvious that $\text{MinD}(o_i,c_j) \leq \text{ED}(o_i, c_j) \leq \text{MaxD}(o_i,c_j)$. Then, if $\text{MinD}(o_i, c_p) > \text{MaxD}(o_i, c_q)$ for some cluster representatives c_p and c_q, we can deduce that $\text{ED}(o_i,c_p) > \text{ED}(o_i,c_q)$ without computing the exact values of the EDs. So, object o_i will not be assigned to cluster p (since there is another cluster q that gives a smaller expected distance from object o_i). We can thus prune away cluster p without bothering to compute $\text{ED}(o_i,c_p)$. As an optimization, we can prune away cluster p if $\text{MinD}(o_i,c_p) > \text{MinMaxD}(o_i)$. This gives rise to the MinMax-BB (bounding box) pruning algorithm (Algorithm 2).

Algorithm 2. MinMax-BB Pruning

1: for all $c_j \in C$ do /*for a fixed object o_i*/
2: Compute $\text{MinD}(o_i,c_j)$ and $\text{MaxD}(o_i,c_j)$.
3: Compute $\text{MinMaxD}(o_i)$.
4: for all cj $\in$ C do
5: if $\text{MinD}(o_i,c_j) > \text{MinMaxD}(o_i)$ then
6: Remove c_j from Q_i

Depending on data distribution, the pruning condition $\text{MinD}(o_i,c_j) > \text{MinMaxD}(o_i)$ potentially removes many clusters from consideration in line 6. This avoids many ED computations at the expense of computing MinD and MaxD. We remark that computing MinD and MaxD requires us to consider only a few points on the perimeter of an object's MBR, instead of all points in its pdf. Thus, computing MinD and MaxD is much simpler than computing ED and it does not involve evaluating an integral. They can be computed much faster than ED.

Another pruning technique proposed in [6] makes use of the inequalities:

$$\text{ED}(o_i,c_j) \leq \text{ED}(o_i,y) + \text{d}(y,c_j)$$

$$\text{ED}(o_i,c_j) \geq \left|\text{ED}(o_i, y) - \text{d}(y, c_j)\right|$$

for any point $y \in R^m$. (Second Equation is indeed the triangle inequality.) These inequalities give bounds on $\text{ED}(o_i,c_j)$ based on $\text{ED}(o_i,y)$. If we can compute the latter efficiently, then we can find the bounds efficiently. One possibility is to choose (for each object) certain fixed points as y and precompute $\text{ED}(o_i,y)$. Then, evaluating the bounds using the inequalities involves only an addition, a subtraction, and an evaluation of distance $\text{d}(y,c_j)$, which are relatively cheap. Note that y is fixed for each object, while c_j, a cluster representative, changes across different iterations in UKmeans.

So, for an object o_i, by computing one expected distance $\text{ED}(o_i,y)$, we are able to obtain bounds for many EDs that involve o_i and any cluster representative c_j. Another pruning method proposed in [6] is called "cluster-shift" (CS). Consider a cluster j whose representatives in two consecutive iterations are c_j' and c_j in that order.

If $\text{ED}(o_i,c_j')$ has been calculated, then we can use c_j' as y in (2) and (3) to bound $\text{ED}(o_i,c_j)$. An appealing aspect of CS is that in the later iterations, as the solution converges, $\text{d}(c_j', c_j)$ is generally very small, making the bounds very tight. It is shown in [6] that the cluster-shift method is very effective in pruning. Also, it does not require any predetermined fixed points y, and hence, no precomputation of $\text{ED}(o_i,y)$ is needed.

In the following discussions, we consider the cluster-hift method instead of the fixed-point method.

Now, the MinMax-BB algorithm can be augmented with the cluster-shift technique to tighten the bounds on EDs. This leads to more effective pruning at little additional cost.

3.3 Pruning with Voronoi Diagram

MinMax-based pruning techniques improve the performance of UK-means significantly by making use of efficiently evaluable bounds on ED to avoid many ED computations. However, these techniques do not consider the geometric structure of R^m or the spatial relationships among the cluster representatives. One important innovation of this paper is the introduction of Voronoi diagrams [7] as a method to exploit the spatial relationships among the cluster representatives to achieve a very effective pruning. We will show in this section that the Voronoi diagram-based pruning techniques are theoretically strictly stronger than MinMax-BB.

We start with a definition of Voronoi diagram and a brief discussion of its properties. Given a set of points $C = \{c_1, \ldots, c_k\}$, the Voronoi diagram divides the space R^m into k cells $V(c_j)$ with the following property:

$D(x,c_p) < d(x,c_q) \quad \forall x \in (c_p), \; c_q \neq c_p.$

The boundary of a cell $V(c_p)$ and its adjacent cell $V(c_q)$ consists of points on the perpendicular bisector, denoted by $c_p|c_q$ between the points c_p and c_q. The bisector is the hyperplane that is perpendicular to the line segment joining c_p and c_q that passes through the midpoint of the line segment. This hyperplane divides the space R^m into two halves. We denote the half containing c_p (but excluding the hyperplane itself) as $H_{p/q}$. Thus, $H_{p/q}$, $H_{q/p}$, and $c_p|c_q$ form a partition of the space R^m. Further, we have the following properties:

$\forall$ distinct $c_p, c_q \in C$,

$D(x,c_p) < d(x,c_q) \quad \forall x \in H_{p/q}$,
$D(x,c_p) = d(x,c_q) \quad \forall x \in c_p|c_q$

Here is how we use Voronoi diagram for pruning in UKmeans: In each iteration, we first construct the Voronoi diagram from the k cluster representative points, $C = \{c_1, \ldots, c_k\}$. The Voronoi diagram leads to two pruning methods: The first one is Voronoi cell pruning. For each object o_i, we check if MBR_i lies completely inside any Voronoi cell $V(c_j)$. If so, then object o_i is assigned to cluster c_j. This is because it follows from the above equations that
$ED(o_i,c_j) < ED(o_i,c_q) \quad \forall c_q \in C \setminus \{c_j\}$.

Note that in this case, no ED is computed. All clusters except c_j are pruned. An example is illustrated in Fig. 1a in which $V(c_j)$ is adjacent to $V(c_1)$, $V(c_2)$, and $V(c_3)$. Since MBR_i lies completely in $V(c_j)$, all points belonging to o_i lie closer to c_j than any other c_q. It follows that $ED(o_i,c_j)$ is strictly smaller than $ED(o_i,c_q)$ for all $c_q \neq c_j$. The Voronoi cell pruning method, denoted as VD, can be summarized by the pseudocode in Algorithm 3.

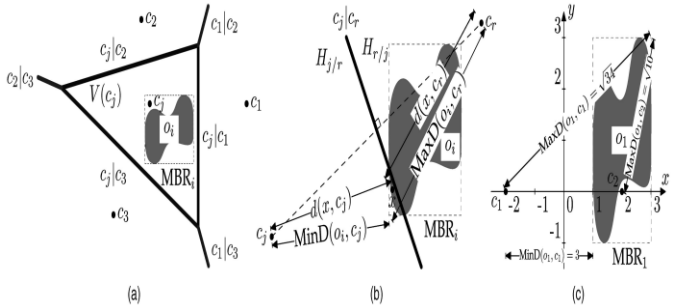

Fig. 1. Voronoi cell pruning and bisector pruning. (a) Voronoi cell pruning. (b) Illustration of the proof of Theorem 1. (c) A counterexample.

Algorithm 3. Voronoi cell Pruning (VD)

1: Compute the Voronoi diagram for $C = \{c_1, \ldots, c_k\}$.
2: for all $c_j \in C$ do
3: if $MBR_i \subseteq V(c_j)$ then
4: $Q_k \leftarrow \{c_j\}$ /*The one and only one candidate*/

The other pruning method is bisector pruning. Bisectors are the side products of Voronoi diagram construction, and thus, they are available at little extra cost. Given an object o_i, we consider every pair of distinct cluster representatives c_p, c_q from C. We check if MBR_i lies completely in $H_{p/q}$. If it does, then by (5), we can deduce that $ED(o_i,c_p) < ED(o_i, c_q)$ and c_q is pruned from Q_i. The expected distance $ED(o_i,c_q)$ is not computed. The bisector-pruning method (abbreviated as Bi) is summarized in Algorithm 4.

Algorithm 4. Bisector Pruning (Bi)
1: Extract all $H_{p/q}$ from Voronoi diagram for C
2: for all distinct $c_p,c_q \in C$ do
3: if $MBR_i \subseteq H_{p/q}$ then
4: remove c_q from Q_i

In the following theorem, we show that bisector pruning is strictly stronger than MinMax-BB in terms of pruning effectiveness.

Theorem 1. For any object $o_i \in O$ and cluster j (j = 1, . . . , k), if bisector pruning does not prune away candidate cluster j, then neither does MinMax-BB.

Proof :

Let c_r be the cluster representative that gives the smallest MaxD with object o_i, i.e., $MaxD(o_i,c_r) = MinMaxD(o_i)$. We consider two cases:

Case 1: r = j. Then,

$MinD(o_i,c_j) \leq MaxD(o_i, c_j)$ by definition
$= MaxD(o_i,c_r)$ since r = j
$= MinMaxD(o_i)$ by definition of cr

Since MinMax-BB prunes cluster j only when $MinD(o_i,c_j) > MinMaxD(o_i)$, we conclude that MinMax-BB does not prune away cluster j in this case. The theorem thus holds in this case.

Case 2: $r \neq j$. The bisector $c_j|c_r$ is well defined and the space R^m can be partitioned into $\{H_{r/j}, c_j|c_r; H_{j/r}\}$. We consider 2 subcases:

Case 2a: MBR_i lies completely in $H_{r/j}$. In this case, cluster j will be pruned by line 4 in Algorithm 4. So, the theorem holds for this case because the antecedent is not satisfied.

Case 2b: MBR_i overlaps with $H_{j/r} \bigcup (c_j|c_r)$. Now, consider a point x in $MBR_i \cap (Hj/r \bigcup (c_j|c_r))$, as illustrated in Fig. 1b. We have

$MinD(o_i,c_j) \leq d(x,c_j)$ since $x \in MBR_i$
$\leq d(x, c_r)$ since $x \in H_{j/r} \bigcup (c_j|c_r)$
$\leq MaxD(o_i,c_r)$ by definition of MaxD
$= MinMaxD(o_i)$ by definition of c_r.

Again, the pruning criterion of MinMax-BB is not satisfied and MinMax-BB cannot prune away cluster j. The theorem thus holds.

Hence, we conclude that if bisector pruning does not prune away cluster j, neither does MinMax-BB.

The converse of the theorem, however, does not hold. That is, there are cases in which MinMax-BB fails to prune a cluster while bisector pruning can. Fig. 1c shows such an example in R^2 with two clusters. Suppose $c_1 = (-2,0)$ and $c_2 = (2,0)$. Then, $c_1|c_2$ is the line x = 0, i.e., the y-axis. Now, consider an object o_1 with MBR_1 bounded by the lines x = 1, x = 3, y = -1, y = 3. Since MBR_1 lies completely in H_2=1, bisector pruning can prune away cluster 1. How about MinMax-BB? Note that $MinD(o_1, c_1) = 3$, $MaxD(o_1,c_1) = \sqrt{34}$ p ; and $MaxD(o_1,c_2) = \sqrt{10}$. So, we have $MinMaxD(o_1) = \sqrt{10} > MinD(o_1,c_1)$, and hence, the pruning condition of MinMax-BB is not satisfied. So, MinMax-BB cannot prune away cluster 1.

We have thus shown that bisector pruning is strictly stronger than MinMax-BB in terms of pruning effectiveness. Note that in implementation, bisectors are a side product of Voronoi diagram computation. It is, therefore, advantageous to perform both Voronoi cell pruning and bisector pruning together. As the Voronoi diagram and bisectors depend only on the cluster representatives c_j (j = 1, . . . , k), we can move the computation of the Voronoi diagram to the outermost loop in the UK-means algorithm as a further optimization. We call the resulting algorithm VDB_i (employing both VD and B_i techniques).

3.4 Partial ED Computation

Given two cluster representatives cp and cq and an object o_i, bisector pruning prunes cluster q if $MBR_i \subseteq H_{p/q}$. If no bisector that involves cluster q can be found to prune cluster q, the expected distance $ED(o_i,q)$ would have to be computed. Interestingly, it is not necessary that we compute the complete integral of $ED(o_iq)$. Our next pruning technique attempts to prune a cluster by computing ED partially.

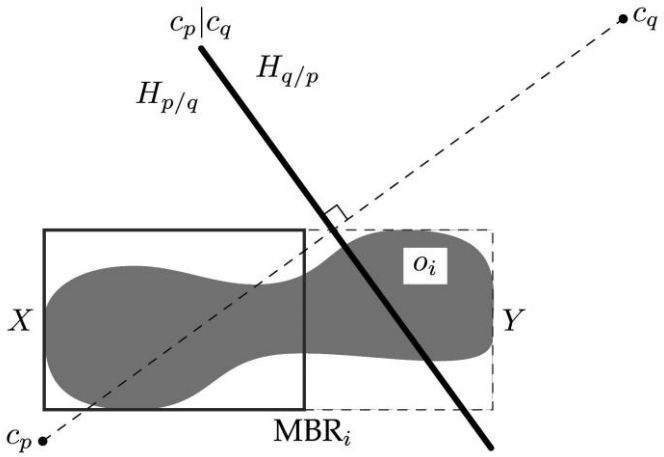

Fig. 2. Partial ED computation.

Again, consider two clusters p and q and an object o_i. If MBR_i intersects the bisector $c_p|c_q$, neither Voronoi cell nor bisector pruning is applicable. In this case, we partition MBR_i into two parts X and Y ($X \bigcup Y = MBR_i$ and $X \cap Y = \Phi$) such that X

$\subseteq V(c_p)$, as shown in Fig. 2. The expected distance $ED(o_i,c_p)$ can then be decomposed as a sum of two "smaller" integrals:

$$ED(o_i,c_p) = \int_{x \in MBR} d(x,c_p) f_i(x) dx$$
$$= \int_{x \in X} d(x,c_p) f_i(x) dx + \int_{x \in Y} d(x,c_p) f_i(x) dx$$
$$= ED_X(o_i,c_p) + ED_Y(o_i,c_p)$$

Similarly, we have $ED(o_i,c_q) = ED_X(o_i,c_q) + ED_Y(o_i,c_q)$. Now, since $X \subseteq V(c_p)$, we know that $ED_X(o_i,c_p) < ED_X(o_i,c_q)$. We compute the integrals $ED_Y(o_i,c_p)$ and $ED_Y(o_i,c_q)$ and if $ED_Y(o_i,c_p) < ED_Y(o_i,c_q)$, we can conclude that $ED(o_i,c_p) < ED(o_i,c_q)$. Cluster q can thus be pruned without computing $ED_X(o_i,c_q)$. Otherwise, if q cannot be pruned, we have to compute $ED(o_i,c_q)$ later, but we need not do so from scratch. We only need to compute $ED_X(o_i,c_q)$, and then, add it to the already computed value of $ED_X(o_i,c_q)$ to get $ED(o_i,c_q)$. Therefore, the effort spent on computing $ED_Y(o_i,c_q)$ can be reused for computing complete ED later if necessary. Thus, the partial computation of EDðoi; cqÞ involves little overhead. We incorporate the above idea of partial ED computation into VDBi to improve the pruning power of the algorithm. We call the resulting algorithm VDBiP.

3.5 Indexing the Uncertain Objects

The above pruning techniques all aim at reducing the number of ED calculations, which dominates the execution time of UK-means. Our pruning techniques are so effective that in many cases, more than 95 percent of the ED calculations are pruned. The cost of other computations, such as the pruning overhead, now becomes relatively significant. In order to further reduce the execution time, we have devised further techniques to reduce the pruning costs.

Observing that Voronoi-diagram-based pruning techniques (namely, VD and Bi) takes advantage of the spatial distribution of the cluster representatives, it is natural to ask whether we can also make use of the spatial distribution of the uncertain objects. Can we organize the objects so that nearby objects are grouped together and processed in batch to avoid repeating similar computations, such as similar pruning condition testing? If we first divide the uncertain objects into groups, we can obtain an MBR for each group (the minimum rectangle enclosing all objects in the group). With these MBRs, we can apply MinMax-BB, VD, and Bi pruning onto the groups. This allows cluster candidate pruning at the group level. In the ideal case where a single cluster is assigned to a whole group, all the member objects of that group get assigned the cluster at once, saving the computations needed to assign clusters to each member individually. This saving is potentially significant. Furthermore, we can group the groups into supergroups, forming a hierarchy. Our pruning techniques can then be applied to different levels in the hierarchy in a top-down manner. To get good pruning effectiveness, the grouping should be done in a way that minimizes the volumes of the MBRs. A natural choice is to group objects using an R-tree structure.

3.5.1 R-Trees

The R-tree is a tree structure that can be considered a variant of B+-Tree. As such, it is a self-balancing tree with all leaf nodes at the same depth from the root node. The main difference is that R-tree is designed for indexing multidimensional spatial data to support faster proximity based queries. The tree has the property that each internal node stores also the MBR for all the objects stored under that subtree. Data items are not stored in the internal nodes. R-tree facilitates spatial query processing. As an example, to locate all objects that are within a distance d from a certain query point x, one starts from the root node and only needs to descend (recursively) into the child nodes whose MBRs intersect the sphere with radius d centered at x. This brings the search to only those few leaf nodes containing the objects being searched for.

In our implementation, we use an R-tree like the one depicted in Fig. 3. Each tree node, containing multiple entries is stored in a disk block. Based on the size of a disk block, the number of entries in a node is computed. The height of the tree is a function of the total number of objects being stored, as well as the fan-out factors of the internal and leaf nodes.

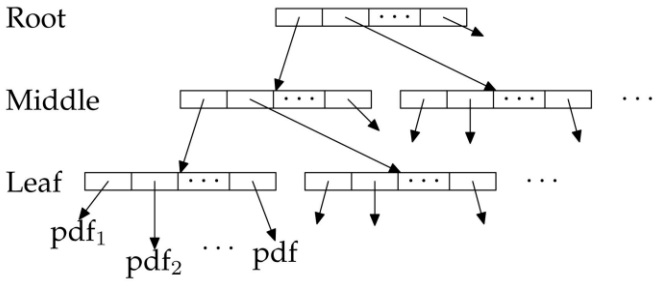

Fig. 3. Structure of an R-tree.

Each leaf node corresponds to a group of uncertain objects. Each entry in the node maps to an uncertain object. The following information are stored in each entry:

- The MBR of the uncertain object.
- The centroid of the uncertain object.
- A pointer to the pdf data of the object.

Note that the pdf data are stored outside the tree to facilitate memory utilization.
Each internal node of the tree corresponds to a supergroup, which is a group of groups. Each entry in an internal node points to a child group. Each entry contains the following information:

- The MBR of the child group.
- The number of objects under the subtree at this child.
- The centroid of the objects under the subtree at this child.
- A pointer to the node corresponding to the child.
-

Note that storing the number of objects under the subtree at a child node and the corresponding centroid location allows efficient readjustment of cluster representatives at the end of each iteration of UK-means (steps 7-8, Algorithm 1).

To build an R-tree from a database of uncertain objects, we use a bulk-load algorithm based on the Sort-Tile-Recursive algorithm . It builds an R-tree from bottomup(as opposed to repeated insertion from the top) and has the advantages of building a more fully filled tree, with smaller MBRs for the internal nodes, and a shorter construction time. We illustrate this algorithm with a 2D example shown in Fig. 4. The figure shows the MBRs of 36 uncertain objects. Suppose the fan-out factor of leaf nodes is 4. Then, each leaf node will contain four uncertain objects and nine leaf nodes are needed. This means that we need to divide the 36 objects into nine groups. We first work on the x-dimension and try to divide the 36 objects into $\sqrt{9} = 3$ vertical stripes. This is done by sorting the objects' ;-

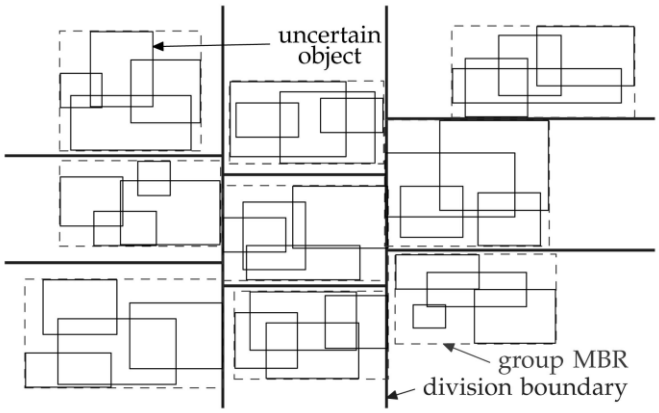

Fig. 4. Bulk-loading algorithm for R-tree.

centers according to their x-coordinates, and then, dividing them into three groups of 12 members each. Next, for each stripe, we independently divide the 12 members into three groups of four members each, after sorting by the y-coordinates. In this way, we have divided the 36 objects into nine groups, for each of which a leaf node is constructed. Higher levels up the tree are built by grouping in a similar fashion, and this is repeated until a single hypergroup is formed. That hypergroup is the root of the R-tree.

3.5.2 Group-Based Pruning

With an R-tree in place, we already have a multilevel grouping of the uncertain objects. In addition, the information kept at each node helps us do pruning in batch, thereby further boosting the performance of the pruning algorithms.

Instead of repeating the cluster assignment to each uncertain object one after another as in UK-means, we traverse recursively down the tree, starting from the root node. We examine each entry of the root node. Each entry e represents a group (or supergroup) of uncertain objects. The MBR of e is readily available from the R-tree. Using this MBR, we can apply our pruning techniques MinMax-BB, VD, or Bi to prune away candidate clusters in the same way as explained before. These techniques work here because they are guaranteed to prune away a cluster representatives cp if there is for sure another cluster representatives $c_q \neq c_p$ such that all points in the MBR are closer to c_q than to c_p. Since this property holds for all points within the group's MBR, it also holds for all subgroups and uncertain objects under the subtree.

In other words, the list of remaining candidates can be passed from ancestors to descendants along any R-tree path. This is a significant performance boost: if a cluster representative c_p is pruned at an R-tree entry e, then c_p needs not be considered for all subgroups and uncertain objects within the whole subtree of e. This saves a lot of repeated computations.

In case only a single cluster representative c_r is left after the pruning, then due to this inheritance of candidate lists, all descendants of e must be assigned to cr. In this case, we can further optimize by bulk-assigning c_r to the whole subtree. There is no need to process each uncertain object under the subtree individually. If this kind of subtree pruning happens at higher levels of the R-tree, a lot of processing can be saved. For instance, suppose an entry e has only one candidate cluster left. Suppose further that e has 20 child nodes, each having 15 leaf-level children. Then, e has 20 X 15 = 300 uncertain objects as grandchildren. In this example, by processing one entry e, we will be able to save the time for processing 300 objects. The higher up the tree this happens, the more tremendous the saving can be. The cluster centroids stored in internal nodes are useful in this scenario for readjusting the cluster centers at the end of the iteration, eliminating the need

to access and process the centroid of every individual uncertain object under the subtree.

To sum up, using an R-tree, we can replace steps 3-6 of Algorithm 1 with a call ProcessInternalNode(r, C), where r is the R-tree's root node and C is the set of all clusters. The recursive procedure ProcessInternalNode is shown in Algorithm 5.

Algorithm 5. ProcessInternalNode

Input: n an R-tree internal node
Q a set of candidate clusters
1: for all child entry e of n do
2: Apply a pruning technique to Q using e's MBR
3: if $|Q| = 1$ then /*only one candidate remains*/
4: for all uncertain object oi under subtrees rooted at n do
5: h(i) ← j where $c_j \in Q$
6: else
7: m ← e's R-tree node
8: if m is leaf node then
9: Call ProcessLeafNode(m, Q)
10: else
11: Call ProcessInternalNode(m, Q) /*recursively*/

The handling of leaf nodes is quite similar, and hence, not repeated. Procedure ProcessLeafNode differs from ProcessInternalNode in which the recursive part (steps 7- 11) is replaced by ED-calculations (for the remaining candidates in Q) and assigning the closest cluster to the uncertain object.
It should be pointed out that construction of this R-tree is very efficient. In addition, it only needs to be done once because every iteration shares the same R-tree. So, the cost of the R-tree construction averaged over all iterations is very negligible. When handling databases where an R-tree on objects is already in place, this cost is effectively zero.

3.6 Hybrid Algorithms

We have discussed a number of pruning methods: MinMax-BB, CS, Voronoi cell (VD), bisector (Bi), and Partial-ED (P). We remark that some of these pruning methods can be employed in combination. For example, candidate clusterrepresentatives can first be pruned by Voronoi-diagram method, then by Bisector pruning, and finally, by Partial-ED pruning. Moreover, some of the pruning methods work well with an R-tree index to achieve group processing. For example, if MinMax-BB is applied to an internal node N such that it reduces the set of candidate cluster representatives Q to a smaller set Q', the reduced set Q' can be passed along to the child nodes of N where MinMax-BB is reapplied. The pruning achieved by MinMax-BB at different levels along a path of the R-tree is thus accumulative.

Algorithm	ED Pruning Techniques				R-tree index
	MinMax-BB	Voronoi-cell (VD)	Bisector (Bi)	Partial ED (P)	(R)
MinMax-BB	✓				
VDBi		✓	✓		
VDBiP		✓	✓	✓	
RMinMax-BB	✓				✓
RBi			✓		✓

TABLE 1
Hybrid Algorithms Used in Experiments

We have selected a few combinations of the techniques for performance evaluation. Table 1 shows five such combinations. In the table, the first column gives the algorithm names and each row indicates the techniques used under the selected algorithm. Here are some justifications of our choices:

- We do not combine MinMax-BB with Voronoidiagram- based methods. This is because we have shown that Bisector pruning is strictly stronger than MinMax-BB pruning (see Theorem 1).

- We do not consider VD pruning when an R-tree index is used. Under VD pruning, a Voronoi diagram is constructed on the set of all cluster representatives C. Let us call this diagram the"global Voronoi diagram." Assume that we use an R-tree index. Now, consider applying a pruning combination (e.g., VDBi) at an internal node (say N) where the set of candidate cluster representatives has beenreduced from C to a smaller set Q' as a result of applying pruning to N's ancestor nodes.

We have two options:

- Apply VD using the (previously constructed) global Voronoi diagram (then followed by applying Bi); or
- construct a new Voronoi diagram based on the reduced set of representatives Q' and use that for VD pruning.

Our experiments show that none of the two options results in good performance. For first option, since the same global Voronoi diagram is used for VD pruning, it does not take advantage of the reduction in the candidate set (from C to Q') achieved by pruning performed at the ancestor nodes of N. For second option, the construction of a different Voronoi diagram for each R-tree node is too costly and is thus counter-productive.

- We do not combine Partial-ED pruning with R-tree boosting. This is simply because the Partial-ED method cannot be applied to an R-tree node. Recal that Partial-ED is applied when the MBR of an uncertain object intersects a bisector and in such case, a partial integration of the object is computed. Since an R-tree node represents not a single object but a group of objects, no such partial integrations can be computed. Thus, Partial-ED is not applicable.

- We omit CS in some of our performance graphs. In our experiments, we have executed two versions of each of the five algorithms listed in Table 1: one with CS pruning and another without. There are thus altogether 10 algorithms. We will present the results of our baseline experiments on synthetic and semisynthetic data sets under all 10 algorithms. For the rest of the experiments, however, we omit the CS versions of the algorithms for two reasons. First, the CS method has been previously studied and we did not find much new insight to it in our study. Second, with five algorithms instead of 10, our presentation, in particular the performance graphs, is more readable.

4 CONCLUSIONS

In this paper, we have studied the problem of clustering uncertain objects whose locations are represented by probability density functions. We have discussed the Ukmeans algorithm [5], which was the first algorithm to solve the problem. We have explained that the computation of expected distances dominates the clustering process, especially when the number of samples used in representing objects' pdfs is large. We have mentioned the existing pruning techniques MinMax-BB and CS [6]. Although these techniques can improve the efficiency of UK-means, they do not consider the spatial relationship among cluster representatives, nor make use of the proximity between groups of uncertain objects to perform pruning in batch.

To further improve the performance of UK-means, we have first devised new pruning techniques that are based on Voronoi diagrams. The VDBi algorithm achieves effective pruning by two pruning methods: Voronoi cell pruning and bisector pruning. We have proved theoretically that bisector pruning is strictly stronger than MinMax-BB. Furthermore, we have proposed the idea of pruning by partial ED calculations and have incorporated the method in VDBiP.

Having pruned away more than 95 percent of the ED calculations, the execution time has been significantly reduced. It has been reduced to such an extent that the originally relatively cheap pruning overhead has become a dominating term in the total execution time. To further improve efficiency, we exploit the spatial grouping derived from an R-tree index built to organize the uncertain objects. This R-tree boosting technique turns out to cut down the pruning costs significantly.

We have also noticed that some of the pruning techniques and R-tree boosting can be effectively combined. Employing different pruning criteria, the combination of these different techniques yields very impressive pruning effectiveness.

ACKNOWLEDGMENTS

We would like to express our sincere thanks to Sri. Dr. Kancharla Ramaiah Secretary and Correspondent, Prakasam Engineering College, Kandukur, A.P. India for his support with providing research environment . We are extremely thankful to our collogues , friends and family members who are cooperated in this work.

REFERENCES

[1] P. Misra and P. Enge, Global Positioning System: Signals, Measurements, and Performance, second ed. Ganga-Jamuna Press, 2006.
[2] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," Proc. IEEE 33rd Ann. Hawaii Int'l Conf. System Sciences (HICSS), Jan. 2000.
[3] S. Bandyopadhyay and E.J. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," Proc. IEEE INFOCOM, Apr. 2003.
[4] O. Wolfson and H. Yin, "Accuracy and Resource Consumption in Tracking and Location Prediction," Proc. Symp. Spatial and Temporal Databases (SSTD), pp. 325-343, July 2003.
[5] M. Chau, R. Cheng, B. Kao, and J. Ng, "Uncertain Data Mining: An Example in Clustering Location Data," Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD), pp. 199-204, Apr. 2006.
[6] W.K. Ngai, B. Kao, C.K. Chui, R. Cheng, M. Chau, and K.Y. Yip, "Efficient Clustering of Uncertain Data," Proc. IEEE Int'l Conf. Data Mining (ICDM), pp. 436-445, Dec. 2006.
[7] F.K.H.A. Dehne and H. Noltemeier, "Voronoi Trees and Clustering Problems," Information Systems, vol. 12, no. 2, pp. 171-175, 1987.
[8] B. Kao, S.D. Lee, D.W. Cheung, W.-S. Ho, and K.F. Chan,

"Clustering Uncertain Data Using Voronoi Diagrams," Proc. IEEE Int'l Conf. Data Mining (ICDM), pp. 333-342, Dec. 2008.
[9] N.N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," The VLDB J., vol. 16, no. 4, pp. 523-544, 2007.