

Clustering Uncertain Data using Voronoi Diagrams and R-Tree Index

Ben Kao

Sau Dan Lee

Foris K. F. Lee

David W. Cheung

Wai-Shing Ho

Department of Computer Science, The University of Hong Kong

{kao, sdlee, kflee, dcheung, wsho}@cs.hku.hk

Abstract—We study the problem of clustering uncertain objects whose locations are described by probability density functions (pdf). We show that the UK-means algorithm, which generalises the k-means algorithm to handle uncertain objects, is very inefficient. The inefficiency comes from the fact that UK-means computes expected distances (ED) between objects and cluster representatives. For arbitrary pdf's, expected distances are computed by numerical integrations, which are costly operations. We propose pruning techniques that are based on Voronoi diagrams to reduce the number of expected distance calculation. These techniques are analytically proven to be more effective than the basic bounding-box-based technique previously known in the literature. We then introduce an R-tree index to organise the uncertain objects so as to reduce pruning overheads. We conduct experiments to evaluate the effectiveness of our novel techniques. We show that our techniques are additive and, when used in combination, significantly outperform previously known methods.

Index Terms—Uncertainty, Clustering, Object hierarchies, Indexing methods

I. INTRODUCTION

Clustering is a technique that has been widely studied and applied to many real-life applications. Many efficient algorithms, including the well known and widely applied k-means algorithm, 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 centres is minimised.

Although simple, the problem model does not address situations where object locations are uncertain. Data uncertainty, however, arises naturally and often inherently in many applications. For example, physical measurements can never be 100% precise in theory (due to Heisenberg's *Uncertainty Principle*). Limitations of measuring devices thus induce uncertainty to the measured values in practice.

As another example, consider the application of clustering a set of mobile devices [1]. By grouping mobile devices into clusters, a leader can be elected for each cluster, which can then coordinate the work within its cluster. For example, a cluster leader may collect data from its cluster's members, process the data, and send the data to a central server via an access point in batch [2], [3]. In this way, local communication within a cluster only requires short-ranged signals, for which

a higher bandwidth is available. Long-ranged communication between the leaders and the mobile network only takes place in the form of batch communication. This results in better bandwidth utilisation and energy conservation.

We remark that device locations are uncertain in practice. 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. Between two sampling time instances, a location value is unknown and can only be estimated by considering a last reported value and an uncertainty model [4]. 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 (pdf's) are defined.

Traditional clustering methods were designed to handle point-valued data and thus cannot cope with data uncertainty. One possible way to handle data uncertainty is to first transform uncertain data into point-valued data by selecting a representative point for each object before applying a traditional clustering algorithm. For example, the centroid of an object's pdf can be used as such a representative point. However, in [5], it is shown that considering object pdf's gives better clustering results than the centroid method.

In this paper we concentrate on the problem of clustering objects with location uncertainty. Rather than a single point in space, an object is represented by a probability density function (pdf) over the space R^m being studied. We assume that each object is confined in a finite region, so that the probability density outside the region is zero. Each object can thus be bounded by a finite bounding box. This assumption is realistic because in practice 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 generalisation of the traditional k-

means algorithm to handle objects with uncertain locations. The traditional k-means clustering algorithm (for point-valued data) 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. We call this step *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 pdf's 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, [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 are thus not necessary and are 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% 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 unorganised 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.

The rest of the paper is organised as follows. We mention a few related works in Section II. In Section III, we formally define the problem. In Section IV, we first briefly describe the UK-means algorithm and the bounding-box-based pruning techniques. After that, we discuss our Voronoi-diagram-based pruning techniques, followed by our R-tree-based boosting technique. We present experimental results in Section V, comparing our new techniques with existing ones. Finally, Section VI concludes the paper.

II. RELATED WORKS

Data uncertainty has been broadly classified into existential uncertainty and value uncertainty. Existential uncertainty appears when it is uncertain whether an object or a data tuple exists. For example, a data tuple in a relational database could be associated with a probability that represents the confidence of its presence [9], [10]. Value uncertainty, on the other hand, appears when a tuple is known to exist, but its values are not known precisely. A data item with value uncertainty is usually represented by a pdf over a finite and bounded region of possible values [11], [12], [13], [5]. In this paper, we study the problem of clustering objects with value (e.g., location) uncertainty.

One well-studied topic on value uncertainty is “imprecise query processing.” An answer to such a query is associated with a probabilistic guarantee on its correctness. Some example studies on imprecise data include indexing structures for range query processing [11], nearest neighbour query processing [12], and imprecise location-dependent query processing [13].

Depending on the application, the result of cluster analysis can be used to identify the (locally) most probable values of model parameters [14] (e.g., means of Gaussian mixtures), to identify high-density connected regions [15] (e.g., areas with high population density), or to minimise an objective function (e.g., the total within-cluster squared distance to centroids [16]). For model parameters learning, by viewing uncertain data as samples from distributions with hidden parameters, the standard Expectation-Maximisation (EM) framework [14] can be used to handle data uncertainty [17].

There has been growing interest in uncertain data mining. In [5], the well-known k-means clustering algorithm is extended to the UK-means algorithm for clustering uncertain data. In that study, it is empirically shown that clustering results are improved if data uncertainty is taken into account during the clustering process. As we have explained, data uncertainty is usually captured by pdf's, which are generally represented by sets of sample values. Mining uncertain data is therefore computationally costly due to information explosion (sets of

samples vs. singular values). To improve the performance of UK-means, CK-means [18] introduced a novel method for computing the EDs efficiently. However, that method only works for a specific form of distance function. For general distance functions, [6] takes the approach of pruning, and proposed pruning techniques such as min-max-dist pruning. In this paper, we follow the pruning approach and propose new pruning techniques that are significantly more powerful than those proposed in [6]. In [19], guaranteed approximation algorithms have been proposed for clustering uncertain data using k-means, k-median as well as k-centre.

Apart from studies in partition-based uncertain data clustering, other directions in uncertain data mining include density-based clustering (e.g., FDBSCAN [15]), frequent itemset mining [20] and density-based classification [21]. For density-based clustering, two well-known algorithms, namely, DBSCAN and OPTICS have been extended to handle uncertain data. The corresponding algorithms are called FDBSCAN [15] and FOPTICS [22], respectively. In DBSCAN, the concepts of core objects and reachability are defined. Clusters are then formed based on these concepts. In FDBSCAN, the concepts are re-defined to handle uncertain data. For example, under FDBSCAN, an object o is a core object if the probability that there is a “good number” of other objects that are close to o exceeds a certain probability threshold. Also, whether an object y is “reachable” from another object x depends on both the probability of y being close to x and the probability that x is a core object. FOPTICS takes a similar approach of using probabilities to modify the OPTICS algorithm to cluster uncertain data.

Clustering of uncertain data is also related to fuzzy clustering, which has long been studied in fuzzy logic [23]. In fuzzy clustering, a cluster is represented by a fuzzy subset of objects. Each object has a “degree of belongingness” with respect to each cluster. The fuzzy c-means algorithm is one of the most widely used fuzzy clustering methods [24], [25]. Different fuzzy clustering methods have been applied on normal or fuzzy data to produce fuzzy clusters [26], [27]. A major difference between the clustering problem studied in this paper and fuzzy clustering is that we focus on hard clustering, for which each object belongs to exactly one cluster. Our formulation targets for applications such as mobile device clustering, in which each device should report its location to exactly one cluster leader.

Voronoi diagram is a well-known geometric structure in computational geometry. It has also been applied to clustering. For example, Voronoi trees [7] have been proposed to answer Reverse Nearest Neighbour (RNN) queries [28]. Given a set of data points and a query point q , the RNN problem [29] is to find all the data points whose nearest neighbour is q . The TPL algorithms proposed in [30] uses more advanced pruning techniques to solve this problem efficiently.

An R-tree [31] is a self-balancing tree structure that resembles a B+-tree, except that it is devised for indexing multi-dimensional data points to facilitate proximity-based searching, such as k-nearest neighbour (kNN) queries. R-trees are well studied and widely used in real applications. They are also available in many RDBMS products such as

SQLite, MySQL and Oracle. An R-tree conceptually groups the underlying points hierarchically and records the minimum bounding rectangle (MBR) of each group to facilitate answering of spatial queries. While most existing works on R-tree concentrate on optimising the tree for answering spatial queries, we use R-trees in this paper in a quite innovative way: We exploit the hierarchical grouping of the objects organised by an R-tree to help us check pruning criteria in batch, thereby avoiding redundant checking.

III. DEFINITIONS

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

$$\begin{aligned} f_i(x) &\geq 0 \quad \forall x \in R^m \\ \int_{x \in R^m} f_i(x) dx &= 1 \end{aligned}$$

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 $x \in R^m \setminus A_i$.

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

$$\text{ED}(o_i, y) = \int_{x \in A_i} d(x, y) f_i(x) dx. \quad (1)$$

Now, given an integer constant k , the problem of clustering uncertain data is to find a set of cluster representative points $C = \{c_1, \dots, c_k\}$ and a mapping $h : \{1, \dots, n\} \rightarrow \{1, \dots, k\}$ so that the sum of squared expected distance

$$\sum_{i=1}^n [\text{ED}(o_i, c_{h(i)})]^2$$

is minimised.

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 Equation (1) still holds if we replace “ $x \in A_i$ ” with “ $x \in \text{MBR}_i$ ”. This fact can be exploited for optimisation when computing ED.

IV. ALGORITHMS

We first give a short description of the UK-means algorithm [5] and existing pruning techniques [6] that improve UK-means. Then, we present our new pruning techniques that are based on Voronoi diagrams and finally, we introduce our performance booster based on R-trees.

Algorithm 1 UK-means

```

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

```

A. UK-means

UK-means (see Algorithm 1) [5] is an adaptation of the well known k-means algorithm to handle data objects with uncertain locations.

Initially, k arbitrary points c_1, \dots, 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 , $\text{ED}(o_i, c_j)$ is computed for all $c_j \in C$. Object o_i is then assigned to cluster c_{j^*} that minimises ED, i.e., $h(i) \leftarrow j^*$. Next, each cluster representative c_j is recomputed as the centroid of all o_i 's that are assigned to cluster j . The two steps are repeated until the solution $C = \{c_1, \dots, c_k\}$ and $h(\cdot)$ 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 4–6 in Algorithm 1 with the following:

```

1:  $Q_i \leftarrow C$  /*candidate clusters*/
2: Apply a pruning algorithm
3: if  $|Q_i| = 1$  then /*only one candidate remains*/
4:    $h(i) \leftarrow j$  where  $c_j \in Q_i$ 
5: else
6:   for all  $c_j \in Q_i$  do /*remaining candidates*/
7:     Compute  $\text{ED}(o_i, c_j)$ 
8:    $h(i) \leftarrow \arg \min_{j: c_j \in Q_i} \{\text{ED}(o_i, c_j)\}$ 

```

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.

B. 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

$$\begin{aligned} \text{MinD}(o_i, c_j) &= \min_{x \in \text{MBR}_i} d(x, c_j) \\ \text{MaxD}(o_i, c_j) &= \max_{x \in \text{MBR}_i} d(x, c_j) \\ \text{MinMaxD}(o_i) &= \min_{c_j \in C} \{\text{MaxD}(o_i, c_j)\} \end{aligned}$$

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 optimisation, 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  $c_j \in C$  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:

$$\text{ED}(o_i, c_j) \leq \text{ED}(o_i, y) + d(y, c_j) \quad (2)$$

$$\text{ED}(o_i, c_j) \geq |\text{ED}(o_i, y) - d(y, c_j)| \quad (3)$$

for any point $y \in R^m$. (Equation (2) 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 pre-compute $\text{ED}(o_i, y)$. Then, evaluating the bounds using the inequalities involves only an addition, a subtraction, and an evaluation of distance $d(y, c_j)$, which are relatively cheap. Note that y is fixed for each object while c_j , a cluster representative, changes across different iterations in UK-means. 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 Equations (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, $d(c'_j, c_j)$ 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 pre-determined fixed points y and hence no pre-computation of $\text{ED}(o_i, y)$ is needed. In the following discussions, we consider the cluster-shift 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.

C. 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 $C = \{c_1, \dots, c_k\}$, the Voronoi diagram divides the space R^m into k cells $V(c_j)$ with the following property:

$$d(x, c_p) < d(x, c_q) \quad \forall x \in V(c_p), c_q \neq c_p \quad (4)$$

The boundary of a cell $V(c_p)$ and its adjacent cell $V(c_q)$ consists of points on the *perpendicular bisector*, denoted $c_p|c_q$ 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 mid-point 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

$$c_p, c_q \in C,$$

$$\begin{aligned} d(x, c_p) &< d(x, c_q) \quad \forall x \in H_{p/q}, \\ d(x, c_p) &= d(x, c_q) \quad \forall x \in c_p|c_q. \end{aligned} \quad (5)$$

Here is how we use Voronoi diagram for pruning in UK-means: In each iteration, we first construct the Voronoi diagram from the k cluster representative points, $C = \{c_1, \dots, c_k\}$. 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 Equations (1) and (4) that:

$$\text{ED}(o_i, c_j) < \text{ED}(o_i, c_q) \quad \forall c_q \in C \setminus \{c_j\}.$$

Note that in this case, no ED is computed. All clusters except c_j are pruned. An example is illustrated in Figure 1(a), 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 $\text{ED}(o_i, c_j)$ is strictly smaller than $\text{ED}(o_i, c_q)$ for all $c_q \neq c_j$. The Voronoi-cell pruning method, denoted as VD, can be summarised by the pseudo code in Algorithm 3.

Algorithm 3 Voronoi-cell Pruning (VD)

- 1: Compute the Voronoi diagram for $C = \{c_1, \dots, c_k\}$.
 - 2: **for all** $c_j \in C$ **do**
 - 3: **if** $\text{MBR}_i \subseteq V(c_j)$ **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 Equation (5), we can deduce that $\text{ED}(o_i, c_p) < \text{ED}(o_i, c_q)$, and c_q is pruned from Q_i . The expected distance $\text{ED}(o_i, c_q)$ is not computed. The bisector-pruning method (abbreviated as Bi) is summarised 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** $\text{MBR}_i \subseteq H_{p/q}$ **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 ($j = 1, \dots, k$), 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., $\text{MaxD}(o_i, c_r) = \text{MinMaxD}(o_i)$. We consider two cases:

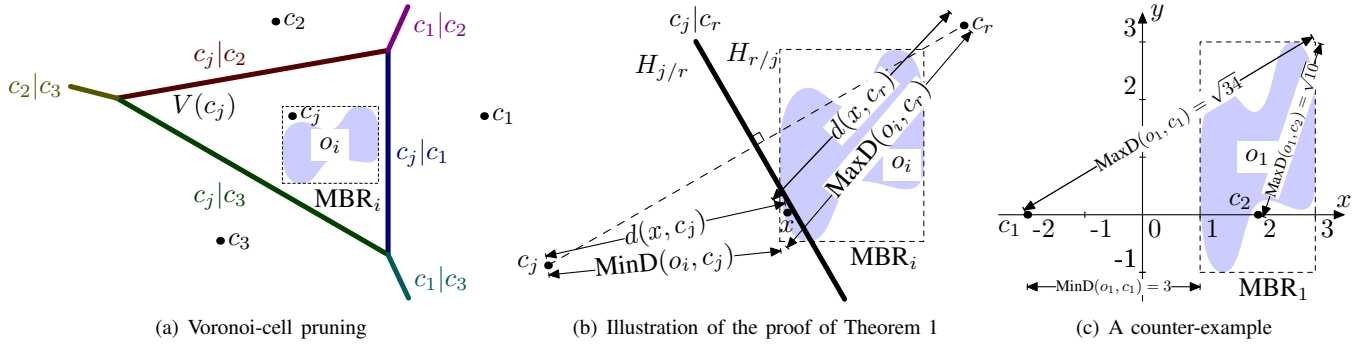


Fig. 1. Voronoi-cell pruning and bisector pruning

Case 1: $r = j$. Then,

$$\begin{aligned} \text{MinD}(o_i, c_j) &\leq \text{MaxD}(o_i, c_j) && \text{by definition} \\ &= \text{MaxD}(o_i, c_r) && \text{since } r = j \\ &= \text{MinMaxD}(o_i) && \text{by definition of } c_r \end{aligned}$$

Since MinMax-BB prunes cluster j only when $\text{MinD}(o_i, c_j) > \text{MinMaxD}(o_i)$, 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 $H_{j/r} \cup (c_j|c_r)$. Now, consider a point x in $\text{MBR}_i \cap (H_{j/r} \cup (c_j|c_r))$, as illustrated in Figure 1(b). We have:

$$\begin{aligned} \text{MinD}(o_i, c_j) &\leq d(x, c_j) && \text{since } x \in \text{MBR}_i \\ &\leq d(x, c_r) && \text{since } x \in H_{j/r} \cup (c_j|c_r) \\ &\leq \text{MaxD}(o_i, c_r) && \text{by definition of MaxD} \\ &= \text{MinMaxD}(o_i) && \text{by definition of } c_r \end{aligned}$$

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. Figure 1(c) shows such an example in R^2 with 2 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 $\text{MinD}(o_1, c_1) = 3$; $\text{MaxD}(o_1, c_1) = \sqrt{34}$ and $\text{MaxD}(o_1, c_2) = \sqrt{10}$. So, we have $\text{MinMaxD}(o_1) = \sqrt{10} > \text{MinD}(o_1, c_1)$ 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

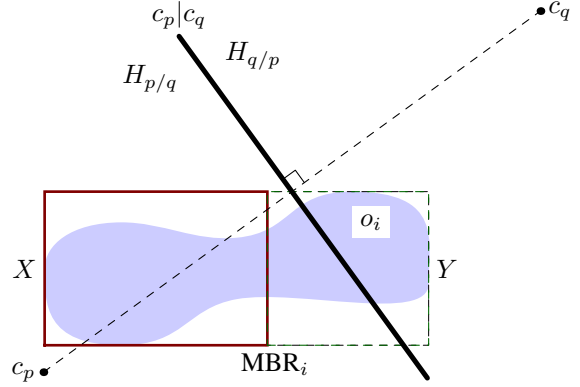


Fig. 2. Partial ED Computation

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 ($j = 1, \dots, k$), we can move the computation of the Voronoi diagram to the outermost loop in the UK-means algorithm as a further optimisation. We call the resulting algorithm VDBi (employing both VD and Bi techniques).

D. Partial ED Computation

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

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 ($X \cup Y = \text{MBR}_i$ and $X \cap Y = \emptyset$) such that $X \subseteq V(c_p)$, as shown in Figure 2. The expected distance $\text{ED}(o_i, c_p)$ can then be decomposed as a sum of two “smaller” integrals:

$$\begin{aligned} \text{ED}(o_i, c_p) &= \int_{x \in \text{MBR}_i} d(x, c_p) f_i(x) dx \\ &= \int_{x \in X} d(x, c_p) f_i(x) dx + \int_{x \in Y} d(x, c_p) f_i(x) dx \end{aligned}$$

$$\stackrel{\text{def}}{=} ED_X(o_i, c_p) + ED_Y(o_i, c_p).$$

Similarly, we have $ED(o_i, c_q) = ED_X(o_i, c_q) + ED_Y(o_i, c_q)$.

Now, since $X \subseteq V(c_p)$, by Equation (4), 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 $ED_Y(o_i, c_q)$ 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 $ED(o_i, c_q)$ 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.

E. 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. As we will see later (Section V), our pruning techniques are so effective that in many cases more than 95% 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 organise 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 super-groups, 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 minimises the volumes of the MBRs. A natural choice is to group objects using an R-tree structure.

1) *R-Trees*: The R-tree [31] 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 multi-dimensional spatial data to support faster proximity-based queries. The tree has the property that each internal

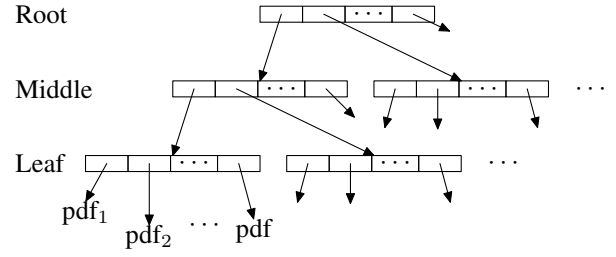


Fig. 3. Structure of an R-Tree

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 centred 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 Figure 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 fanout factors of the internal and leaf nodes.

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 utilisation.

Each internal node of the tree corresponds to a super-group, 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 [31]. It builds an R-tree from bottom up (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 Figure 4. The figure shows the MBRs of 36 uncertain objects. Suppose the fanout factor of leaf nodes is 4. Then, each leaf node will contain 4 uncertain objects and 9 leaf nodes are needed. This

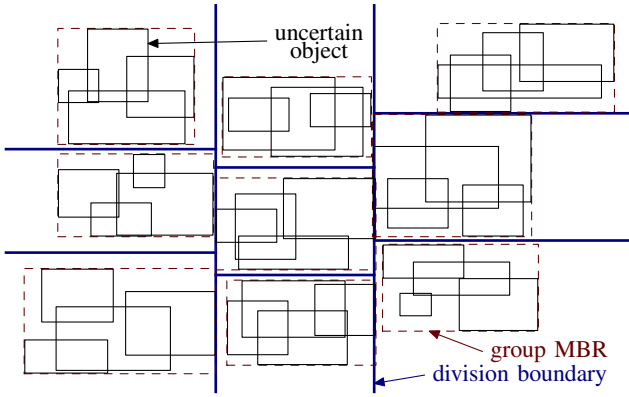


Fig. 4. Bulk-loading algorithm for R-Tree

means we need to divide the 36 objects into 9 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' centres according to their x -coordinates, and then dividing them into 3 groups of 12 members each. Next, for each stripe, we independently divide the 12 members into 3 groups of 4 members each, after sorting by the y -coordinates. In this way, we have divided the 36 objects into 9 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 hyper-group is formed. That hyper-group is the root of the R-tree.

2) *Group-based pruning*: With an R-tree in place, we already have a multi-level 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 super-group) 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 c_p 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 sub-groups 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 sub-groups 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 thanks to this inheritance of candidate lists, all descendants of e must be assigned to c_r . In this case, we can further optimise 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 $20 \times 15 = 300$ 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 centres 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 o_i under subtrees rooted at n **do**
- 5: $h(i) \leftarrow j$ where $c_j \in Q$
- 6: **else**
- 7: $m \leftarrow e$'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.

F. Hybrid Algorithms

We have discussed a number of pruning methods: MinMax-BB, cluster-shift (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 cluster representatives 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

TABLE I
HYBRID ALGORITHMS USED IN EXPERIMENTS

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

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 re-applied. The pruning achieved by MinMax-BB at different levels along a path of the R-tree is thus accumulative.

We have selected a few combinations of the techniques for performance evaluation. Table I 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 Voronoi-diagram-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 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 been reduced from C to a smaller set Q' as a result of applying pruning to N 's ancestor nodes. We have two options:
 - 1) Apply VD using the (previously constructed) global Voronoi diagram (then followed by applying Bi); or
 - 2) 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 (1), 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 (2), 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. Recall 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 cluster-shift (CS) in some of our performance graphs. In our experiments, we have executed two versions of each of the five algorithms listed in Table I:

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 semi-synthetic data sets under all 10 algorithms in Sections V-B and V-C. 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 5 algorithms instead of 10, our presentation, in particular the performance graphs, is more readable.

V. EXPERIMENTS

We have performed a series of experiments to compare the performance of our new techniques with existing ones [6]. We compare the algorithms VDBi, VDBiP, RBi, RMinMax-BB to the old MinMax-BB algorithm. We have also studied alternative versions of the 5 algorithms in which cluster-shift (CS) pruning is added. All the algorithms are implemented in OpenJDK 1.6 on AMD64 platform running Linux kernel 2.6.24. Experiments are carried out on a PC with Intel Core2 Duo 2.83GHz CPU and 4GB of main memory. For computation of Voronoi diagrams, we employ the popular `qhull` program¹[32].

A. Data Sets

We have used two types of data sets in the experiments. The first type is *synthetic* data generated in the same way as [6]. For each data set, a set of n MBRs are generated in the m -dimensional space $[0, 100]^m$. Each MBR's side length is generated randomly, but bounded above by d . The MBR is then divided into s grid cells, each corresponding to a pdf sample point. Each sample point is associated with a randomly generated probability value, normalised so that the sum of probabilities of the MBR is equal to 1. These probability values give a discretised representation of the pdf f_i of the corresponding object.

The second type, *semi-synthetic* data, is based on real data. The real data is a geographical data set containing 53145 points in 2 dimensions representing geographic locations in the Long Beach County of California in the United States. This data set contains point-valued data. We transform this real data set into many uncertain data sets by replacing each data point with an MBR and pdf generated in the same way as described above. By varying the parameters d and s , we get many semi-synthetic data sets.

In each set of experiments, we generate a data set as described above, as well as k random points to serve as the initial cluster centres. The data set and initial cluster centres are then fed to the algorithms. The clustering results from all algorithms are compared to ensure that they are the same. For each set of parameters, 5 sets of experiments are run and the average values are taken and reported.

The parameters used for the experiments are summarised in Table II. The rightmost column of the table shows the baseline values of the various parameters.

¹<http://www.qhull.org/>

TABLE II
PARAMETERS FOR THE EXPERIMENTS

Parameter	Description	Baseline Value
n	no. of uncertain objects	20000
k	number of clusters	50
d	max. side length of MBR	5
s	no. of samples per object	196
m	no. of dimensions	2
b	block size of R-tree nodes	675

TABLE III
BASELINE RESULTS ON SYNTHETIC DATA SET

Algorithm	without cluster-shift		with cluster-shift	
	t_I (ms)	N_{ED}	t_I (ms)	N_{ED}
MinMax-BB	959	0.88	954	0.66
VDBi	764	0.75	750	0.55
VDBiP	772	0.59	767	0.45
RMinMax-BB	420	0.88	367	0.66
RBi	410	0.75	371	0.55

B. Results of Baseline Experiment

We carried out the first set of experiments using the parameters shown in Table II on synthetic data sets. The results are shown in Table III. Depending on the data set, it takes 85–186 iterations before the solution converges. (Note that for each data set, every algorithm takes the same number of iterations.) To mask out the effects of such variations, we report the average execution time *per iteration* (t_I) in the second and fourth columns. The value t_I is defined as the total execution time taken divided by the number of iterations executed. The column N_{ED} shows the average number of ED calculations per object per iteration. The value N_{ED} is defined as the total number of ED calculations divided by the number of objects ($n = 20000$ in our baseline setting) and by the number of iterations. Columns 2–3 show the results for the plain pruning algorithms, whereas columns 4–5 show the results when these algorithms are combined with the cluster-shift (CS) technique [6].

It is obvious from the first three columns of Table III that our new algorithms all output-perform MinMax-BB significantly. The Voronoi-diagram-based pruning methods can save 20% of the execution time. Employing an R-tree index on MinMax-BB brings about a time saving of $> 55\%$. The fastest algorithm is RBi, which gives a time reduction of 57% w.r.t. MinMax-BB. All these five algorithms blend well with the cluster-shift technique. Cluster-shift delivers up to 13% reduction in execution time when combined with the other algorithms. This reduction is relatively minor when compared to Voronoi-diagram-based pruning and R-tree boosting. Moreover, the reduction is quite consistent throughout all our experiments. So, we will omit the cluster-shift technique in the rest of this section for brevity.

To compare the pruning effectiveness of the algorithms, we need to examine the N_{ED} -column. Since N_{ED} is the number of ED calculations per object per iteration, a smaller value indicates more effective pruning. Note that if we had performed the same experiment with UK-means (Algorithm 1, which does not perform any pruning), the number N_{ED} for UK-means would be k [6]. This is because in each iteration,

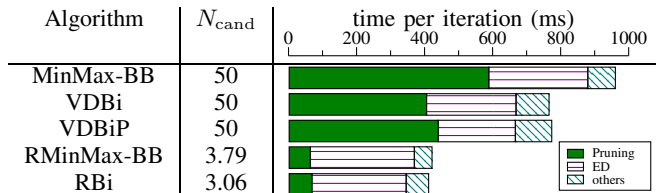


Fig. 5. Breakdown of Execution Time (per Iteration)

UK-means computes for each object all k expected distances from the object to the k cluster representatives. In our baseline setting, $k = 50$ and therefore N_{ED} for UK-means would be 50. From Table III, we see that the pruning algorithms are very effective. All of the pruning algorithms reduce N_{ED} from $k = 50$ (UK-means) to below 0.88. That is a reduction of more than 98%. We can see that VDBi is more effective than MinMax-BB, whereas RBi is more effective than RMinMax-BB, thereby confirming Theorem 1: Bisector pruning is stronger than MinMax-BB pruning. Note also that RMinMax-BB (resp. RBi) always gives the same N_{ED} value as MinMax-BB (resp. VDBi). This is because the function of R-tree boosting is to enable batch processing in the application of a pruning method. It thus lowers the pruning overheads and improves pruning *efficiency*. However, an R-tree index does not affect the pruning *effectiveness*. Therefore, it lowers execution time but does not affect N_{ED} . To better understand the benefits of applying an R-tree index, let us examine a breakdown of t_I (for the case without cluster-shift) in Figure 5.

The execution time can be broken down into 3 components: (1) the time spent on ED calculations, (2) the time spent on pruning and (3) other costs. The time spent on pruning involves a huge number of Euclidean distance computations (for MinMax-BB) or checking against Voronoi-cell boundaries (for VD and Bi). The number of such calculations can be indicated by N_{cand} , the average number of candidate object-cluster pairs per iteration per object, shown in the second column of Figure 5. Without an R-tree, this number equals $k = 50$, i.e. the number of clusters. With the help of an R-tree, many cluster candidates are eliminated before we perform the pruning tests. As a result, N_{cand} drops to below 4, which is a 92% reduction. As the number of candidate object-cluster pairs to be considered decreases significantly, the associated pruning cost is greatly reduced. We can see from the third column of Figure 5 that while R-tree boosting has virtually no effect on the time spent on ED calculations, it does significantly cut down the pruning costs. For example, it reduces the pruning cost *per iteration* of MinMax-BB by 89% from 587ms to 62.4ms. The *total* time spent on the R-tree construction is only 800ms, which translates to less than 10ms *per iteration*. This cost is included in the “others” cost component. The tree construction time is thus negligible compared with t_I . We emphasise again that the R-tree needs only be constructed once and is reused in each iteration. So, this cost is very low, especially when the number of iterations is high.

For algorithms that employ Voronoi diagrams, there is a time cost incurred in the computation of the diagrams. In our experiments, the average cost of Voronoi diagram construction is around 10ms per iteration, which is only 1.1%

TABLE IV
BASELINE RESULTS ON SEMI-SYNTHETIC DATA SET

Algorithm	without cluster-shift		with cluster-shift	
	t_I (ms)	N_{ED}	t_I (ms)	N_{ED}
MinMax-BB	2629	0.91	2617	0.68
VDBi	2079	0.76	2018	0.56
VDBiP	2128	0.58	2096	0.46
RMinMax-BB	1165	0.91	1080	0.68
RBi	1047	0.76	924	0.56

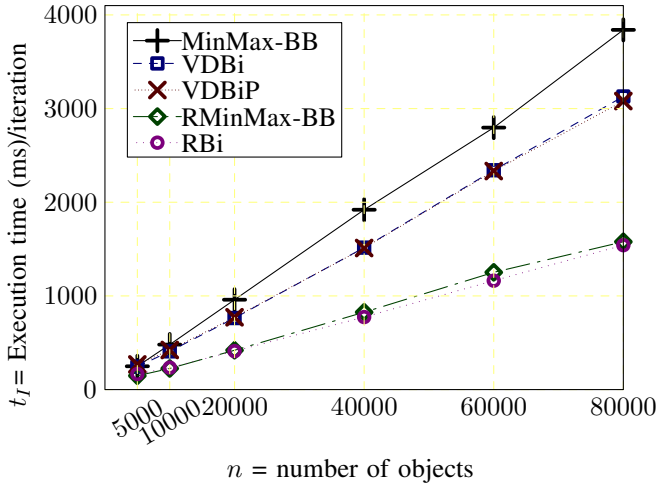


Fig. 6. Effects of n on Execution Time per Iteration

of t_I of MinMax-BB. So, the cost is negligible. The diagram construction time is certainly paid off by the huge number of ED calculations saved. Recall that bisector pruning is strictly stronger than MinMax-BB pruning as proved in Theorem 1. So, we can conclude that Voronoi-diagram-based pruning is a more practical and effective pruning technique than MinMax-BB. Combined with R-tree boosting and the cluster-shift technique, much better pruning efficiency and effectiveness can be achieved.

C. Results on Semi-synthetic Data Set

We have done a similar experiment on semi-synthetic data sets. The parameters $n = 53145$ and $m = 2$ are constrained by the nature of the data. The other parameters are maintained at their baseline values given in Table II. The results are given in Table IV. The relative performances are very similar to those obtained from the synthetic data set. Therefore, in the rest of this section, we present only the experimental results for synthetic data for which parameter values can be controlled to conduct various sensitivity studies.

D. Effects of the Number of Objects

In our next set of experiments, we vary the number of uncertain objects, n , from 5000 to 80000. Other parameters are given their baseline values (Table II). The resulting execution times per iteration (t_I) are plotted against n in Figure 6. It can be seen that the execution time per iteration grows linearly with the number of uncertain objects. This is because as long as the pruning effectiveness and the effects of R-tree boosting remain stable, the total number of ED computations

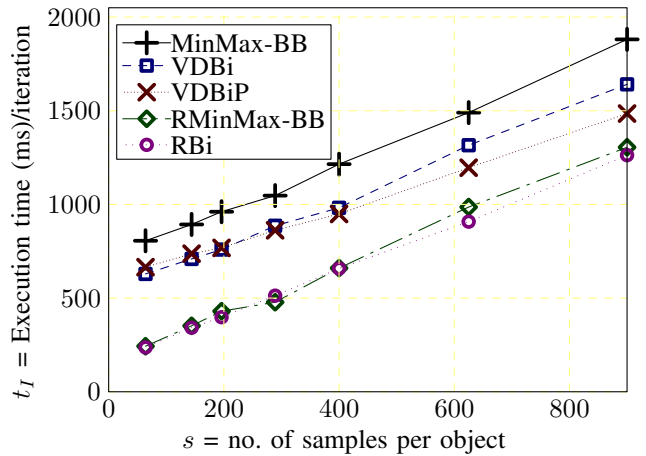


Fig. 7. Effects of s on Execution Time per Iteration

and the amount of pruning overheads will be proportional to the number of uncertain objects being handled. The algorithms are thus scalable w.r.t. n .

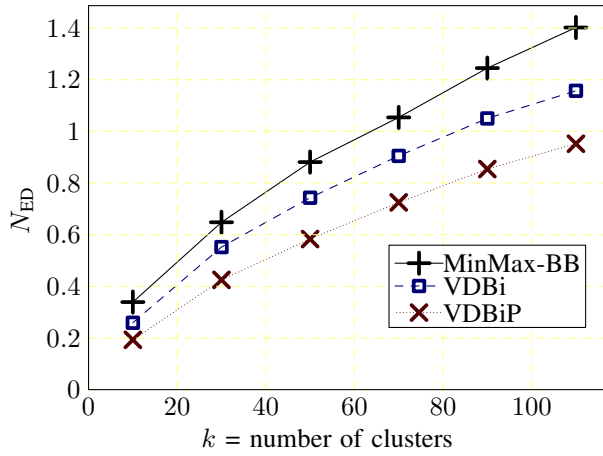
We have also studied the variation of N_{ED} against n . The finding is that N_{ED} is insensitive to n and hence have roughly the same values as given in Table III. In other words, the pruning effectiveness of the algorithms is insensitive to the number of uncertain objects.

E. Effects of the Number of Samples

Next, we vary s , the number of samples used to represent an object's pdf. In this experiment, s is varied from 64 to 900. The execution times *per iteration* taken by the algorithms are plotted in Figure 7.

From the figure, we see that the execution times of the algorithms generally increase linearly as s increases. This is because the time to compute an ED grows linearly with s . We observe from the figure that when s is large (e.g., $s > 400$), the relative performance of the algorithms is mostly consistent with that observed in our baseline experiment (Table III). When s is small, however, VDBiP starts to take more time than VDBi. This is because computing an ED is less costly when s is smaller. As a result, the extra pruning cost incurred under Partial-ED pruning cannot be paid back by the time saved on the reduced number of (light-weight) ED calculations. All in all, for the wide range of values we have tested, Voronoi-diagram-based pruning and R-tree boosting can both deliver high efficiency. The combination RBi has the best performance.

Note that the value of s only affects the amount of time taken to compute an ED, not the *number* of ED computations. In the following experiments, we will concentrate on measuring the pruning effectiveness of the algorithms. Therefore, in the following sections, we will omit execution times and report N_{ED} only. We have already established that for large values of s , the relative efficiency of the algorithms is similar to that reported in Table III.

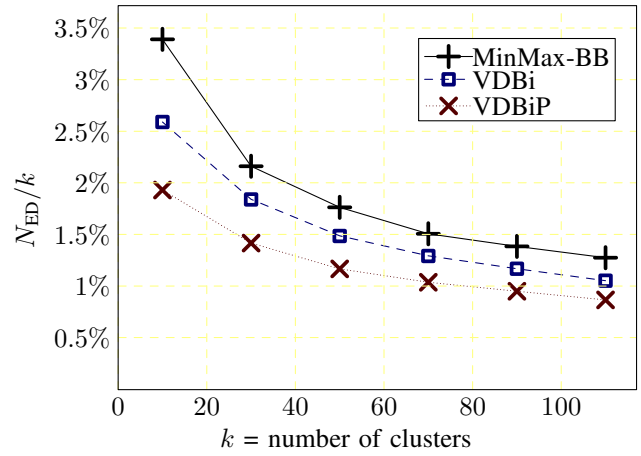
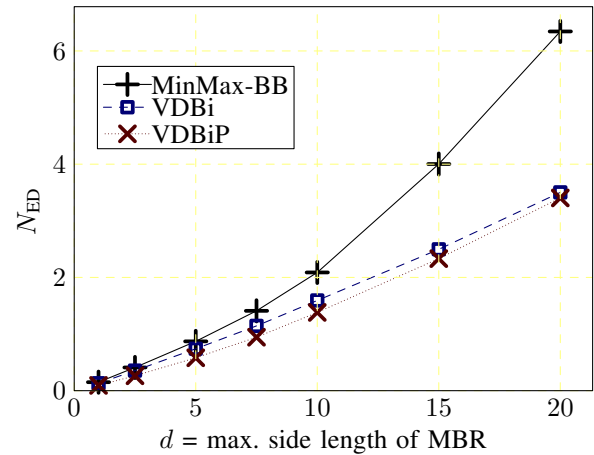
Fig. 8. Effects of k on N_{ED}

F. Effects of the Number of Clusters

In another experiment, we vary the number of clusters, k , from 10 to 110. The other parameters are kept at their baseline values. Note that since R-tree boosting does not affect N_{ED} , we have omitted the curves for RMinMax-BB and RBi for clarity. We do this for all subsequent plots of N_{ED} . Figure 8 shows the results. We see from the graph that N_{ED} increases with k . This is because with a larger number of clusters, cluster representatives are generally less spread out. It is therefore less likely that the pruning algorithms will be able to prune all but one cluster for a given object. Hence, more ED will have to be computed to determine the cluster assignment. For example, under VD pruning, a larger number of clusters implies smaller Voronoi cells. It is thus less likely that an object is found to be enclosed entirely within a particular Voronoi cell so that all but one cluster representative are pruned.

From Figure 8, we see that the N_{ED} curves are always significantly lower than k . Recall that UK-means performs k ED computations per object per iteration, Figure 8 thus shows that all three pruning algorithms are very effective for a wide range of values of k . To better illustrate the algorithms' pruning effectiveness with respect to the basic UK-means algorithm, we plot N_{ED}/k against k in Figure 9. The figure thus shows the fraction of expected distances computed by the various algorithms compared with UK-means.

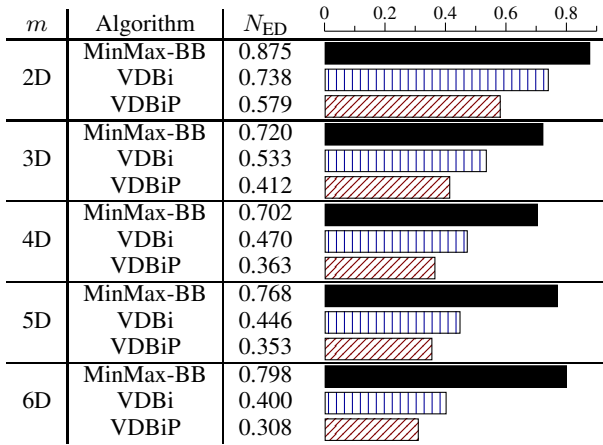
From the figure, we see that the values of N_{ED}/k are very small. The pruning algorithms are thus very effective. For example, when $k = 10$, MinMax-BB and VDBiP computed 3.39% and 1.93% of the EDs computed by UK-means, respectively. These translate into a pruning effectiveness of 96.6% and 98.1%, respectively. Also, we see that N_{ED}/k decreases as k increases for all three pruning algorithms. In other words, the fraction of ED pruned by the algorithms increases when there are more clusters. The pruning effectiveness of VDBi is seen to be consistently better than MinMax-BB over the whole range of k value. By achieving additional pruning using partial ED computation, VDBiP performs even better than VDBi.

Fig. 9. N_{ED}/k vs. k Fig. 10. Effects of d on N_{ED}

G. Effects of the Size of MBR

To study the effect of the extent of uncertainty on the algorithms' performance, we vary d , the maximum side length of an object's MBR, from 1.0 to 20. Other parameters are kept at their baseline values. Essentially, a larger MBR implies a larger uncertainty region and so an object's location is *more uncertain*. The results are shown in Figure 10.

We can see from the graph that N_{ED} increases as the size of the MBRs increases. For MinMax-BB, a bigger MBR causes the gap between MinD and MaxD to be bigger, making the pruning condition $\text{MinD}(o_i, c_j) > \text{MinMaxD}(o_i)$ less likely to be satisfied. Hence, the pruning effectiveness of MinMax-BB drops significantly. In addition, as the size of the MBRs increases, it is more likely that the MBRs overlap with multiple Voronoi cells. This causes VD pruning and Bi pruning to fail more often. Even though the pruning effectiveness decreases when d becomes larger, Figure 10 shows that the overall pruning effectiveness is still very impressive (recall that N_{ED} for the basic UK-means algorithm is 50). Comparing MinMax-BB and VDBiP, the latter prunes 33%–46% of the EDs computed by the former. Our Voronoi-diagram-based pruning algorithms are thus seen to outperform the corresponding MinMax-based algorithms by a significant amount.

Fig. 11. Effects of Number of Dimensions on N_{ED}

Number of dimensions (m)	2	3	4	5	6
Average inter-cluster distance	51.9	66.3	77.7	87.9	96.7
MinMax-bound tightness	3.21	3.79	4.29	4.75	5.18

TABLE V
FACTORS AFFECTING N_{ED} OF MINMAX-BB

H. Effects of the Number of Dimensions

We now vary parameter m from 2 to 6 to see the effects of the number of dimensions on N_{ED} . The results are shown in Figure 11.

It can be seen from the figure that the pruning effectiveness of VDBi and VDBiP is consistently better than that of MinMax-BB for all the number of dimensions experimented. Comparing each algorithm across different values of m , we find that MinMax-BB attains a minimum N_{ED} at 4 dimensions. There are two competing factors here affecting N_{ED} . On the one hand, since the number of clusters k is kept constant but the volume of the space $[0, 100]^m$ (expressed as a multiple of the unit hypercube) increases with m , the average distance between any two cluster centres increases rapidly with m . As an example, Table V shows the average inter-cluster distance measured in the experiment. Consequently, the cluster centres become more spread apart, and there is a higher chance for the pruning criterion of MinMax-BB to be met. This has an effect of driving down N_{ED} as m rises. On the other hand, the maximum side length of the MBR of the uncertain objects is fixed at d . As m increases, the difference $\text{MaxD}(o_i, c_j) - \text{MinD}(o_i, c_j)$ (for any uncertain object o_i and cluster representative c_j) increases as well. The empirical evidence is shown in Table V. Consequently, the bounds MinD and MaxD become looser and looser as m increases. With looser bounds, the pruning effectiveness of MinMax-BB decreases. This causes N_{ED} to increase as m increases. These two effects compete with each other and affect N_{ED} at the same time. Apparently, from Figure 11, the first effect dominates from 2D up to 4D. So, N_{ED} for MinMax-BB goes down from 2D to 4D. Beyond that, the second effect starts to dominate. This explains why N_{ED} goes up again for MinMax-BB beyond 4D.

For algorithms based on VD and Bi pruning, the general trend is that the higher the number of dimensions, the lower the value of N_{ED} , and hence the more effective the pruning

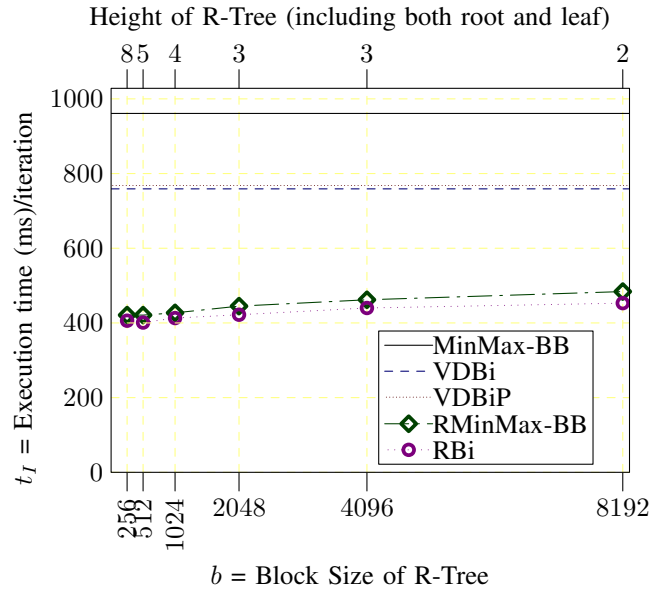


Fig. 12. Effects of Block Size on Execution Time per Iteration

is. Moreover, this decrease in N_{ED} is more remarkable than MinMax-based pruning. To see why this is so, we note that the average volume of each Voronoi-cell is $100^m/k$. On the other hand, the average volume of the MBR of an uncertain object is d^m . So, on average the volume ratio of MBR to Voronoi-cell is $k(d/100)^m = k(5/100)^m$ (because we use the baseline value $d = 5$). As this ratio decreases rapidly with m , the likelihood that an MBR lies completely within a Voronoi-cell increases, thus increasing the probability of successful Voronoi-cell pruning and also bisector pruning. Therefore, as the number of dimension increases, the pruning effectiveness of these algorithms increases. Judging from these results, we conclude that our Voronoi-diagram-based pruning techniques are not only more effective, but also scale better than MinMax-BB with the number of dimensions.

I. Effects of R-Tree Block Size

Finally, we test the effects of the block size of R-tree nodes on R-tree boosting. The block size affects the height of the R-tree built, its compactness, the granularity of the groups and also the size of the MBR of each group. Its effect is thus complex. As we have explained previously, R-tree boosting improves pruning efficiency but has no effect on the pruning effectiveness. So, we examine the execution time of the algorithms. The results are shown in Figure 12. The numbers along the top side of the figure show the heights of the R-trees constructed. For example, at a block size of 2048 bytes, the R-tree is 3-level tall. The algorithms MinMax-BB, VDBi and VDBiP do not employ R-tree, and hence they are not affected by variations in b . Nevertheless, we have included horizontal lines, with values taken from Table III, to show t_I for these algorithms in Figure 12 for reference.

We observe that, in general, execution time increases slightly with block size. With smaller blocks, the number of nodes in the R-tree increases, and so does the height of the R-tree. This has a positive effect on pruning-cost reduction

because a deeper R-tree allows more opportunities for batching the pruning computation, which can be applied to a larger number of nodes at more diverse granularities. Moreover, a smaller block holds fewer child entries, giving smaller MBRs for the parent group. These smaller MBRs allow pruning to be done early (i.e., at the higher-level nodes), which favours R-tree boosting.

On the other hand, smaller blocks imply a larger number of R-tree nodes. This has the adverse effect of more processing and thus a slower execution time. So, when the block size is extremely small, such costs would increase up to a level that the time savings brought about by R-tree boosting cannot compensate. This is observed in Figure 12 at $b = 256$. This block size is too small, generating an R-tree with as many as 8 levels. The overhead incurred in traversing and processing so many nodes is too high. So, it eventually takes more time to complete than the case when $b = 512$.

Nevertheless, the general observation is that as long as the block size is not smaller than a critical value, using smaller block sizes with RMinMax-BB and RBi results in shorter execution time.

J. Summary

Our experiments have shown that our new techniques, namely Voronoi-diagram-based pruning and R-tree boosting, are highly effective. They can bring down the execution time of UK-means as well as MinMax-BB significantly. Further pruning effectiveness and performance gain can be achieved by combining them with cluster-shift. We have found that RMinMax-BB and RBi, when combined with cluster-shift, give the best performance, with RBi out-performing RMinMax-BB when the number of objects is moderately large.

VI. 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 UK-means 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' pdf's is large. We have mentioned the existing pruning techniques MinMax-BB and cluster-shift (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% 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 organise the uncertain objects. This R-tree boosting technique turns out to cut down the pruning costs significantly.

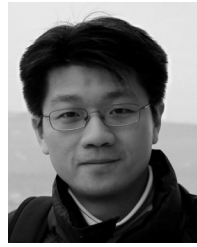
We have also noticed that some of the pruning techniques and R-tree boosting can be effectively combined. Employing different pruning criteria, the combination of these different techniques yield very impressive pruning effectiveness.

We have conducted extensive experiments to evaluate the relative performance of the various pruning algorithms and combinations. The results show that our new pruning techniques outperform MinMax-BB consistently over a wide range of experimental parameters. The overhead of computing Voronoi diagrams for our Voronoi-diagram-based technique is paid off by the large number of ED calculations saved. The overhead of building an R-tree index also gets compensated by the large reduction of pruning costs. The experiments also consistently demonstrated that the hybrid algorithms can prune more effectively than the other algorithms. Therefore, we conclude that our innovative techniques based on Voronoi diagrams and R-tree index are effective and practical.

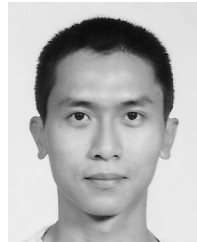
REFERENCES

- [1] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*, 2nd ed. Ganga-Jamuna Press, 2006, ISBN 0-9709544-1-7.
- [2] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *33rd Annual Hawaii International Conference on System Sciences (HICSS)*, IEEE, Maui, Hawaii, U.S.A., 4th-7th Jan. 2000.
- [3] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, San Francisco, U.S.A., 30th Mar.-3rd Apr. 2003.
- [4] O. Wolfson and H. Yin, "Accuracy and resource consumption in tracking and location prediction," in *SSTD*, ser. Lecture Notes in Computer Science, vol. 2750. Santorini Island, Greece: Springer, 24-27 Jul. 2003, pp. 325-343.
- [5] M. Chau, R. Cheng, B. Kao, and J. Ng, "Uncertain data mining: An example in clustering location data," in *PAKDD*, ser. Lecture Notes in Computer Science, vol. 3918. Singapore: Springer, 9-12 Apr. 2006, pp. 199-204.
- [6] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip, "Efficient clustering of uncertain data," in *ICDM*. Hong Kong, China: IEEE Computer Society, 18-22 Dec. 2006, pp. 436-445.
- [7] F. K. H. A. Dehne and H. Noltemeier, "Voronoi trees and clustering problems," *Inf. Syst.*, 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," in *ICDM*. Pisa, Italy: IEEE Computer Society, 15th-19th Dec. 2008, pp. 333-342.
- [9] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," *The VLDB Journal*, vol. 16, no. 4, pp. 523-544, 2007.
- [10] D. Barará, H. Garcia-Molina, and D. Porter, "The management of probabilistic data," *IEEE Trans. Knowl. Data Eng.*, vol. 4, no. 5, pp. 487-502, 1992.
- [11] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Querying imprecise data in moving object environments," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1112-1127, 2004.
- [12] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter, "Efficient indexing methods for probabilistic threshold queries over uncertain data," in *VLDB*. Toronto, Canada: Morgan Kaufmann, 31 Aug.-3 Sept. 2004, pp. 876-887.
- [13] J. Chen and R. Cheng, "Efficient evaluation of imprecise location-dependent queries," in *ICDE*. Istanbul, Turkey: IEEE, 15-20 Apr. 2007, pp. 586-595.

- [14] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. B39, pp. 1–38, 1977.
- [15] H.-P. Kriegel and M. Pfeifle, "Density-based clustering of uncertain data," in *KDD*. Chicago, Illinois, USA: ACM, 21–24 Aug. 2005, pp. 672–677.
- [16] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings Fifth Berkeley Symposium on Math. Stat. and Prob.* University of California Press, 1967, pp. 281–297.
- [17] H. Hamdan and G. Govaert, "Mixture model clustering of uncertain data," in *Proceedings of the 14th IEEE International Conference on Fuzzy Systems*, 25 May 2005, pp. 879–884.
- [18] S. D. Lee, B. Kao, and R. Cheng, "Reducing UK-means to K-means," in *The 1st Workshop on Data Mining of Uncertain Data (DUNE)*, in conjunction with the 7th IEEE International Conference on Data Mining (ICDM), Omaha, NE, USA, 28 Oct. 2007.
- [19] G. Cormode and A. McGregor, "Approximation algorithms for clustering uncertain data," in *PODS*, M. Lenzerini and D. Lembo, Eds. Vancouver, BC, Canada: ACM, 9th–11th Jun. 2008, pp. 191–200.
- [20] C. K. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data," in *PAKDD*, ser. Lecture Notes in Computer Science, vol. 4426. Nanjing, China: Springer, 22–25 May 2007, pp. 47–58.
- [21] C. C. Aggarwal, "On density based transforms for uncertain data mining," in *ICDE*. Istanbul, Turkey: IEEE, 15–20 Apr. 2007, pp. 866–875.
- [22] H.-P. Kriegel and M. Pfeifle, "Hierarchical density-based clustering of uncertain data," in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*. Houston, Texas, USA: IEEE Computer Society, 27–30 Nov. 2005, pp. 689–692.
- [23] E. H. Ruspini, "A new approach to clustering," *Information and Control*, vol. 15, no. 1, pp. 22–32, 1969.
- [24] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 32V57, 1973.
- [25] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York, USA: Plenum Press, 1981.
- [26] M. Sato, Y. Sato, and L. C. Jain, *Fuzzy Clustering Models and Applications*. Physica-Verlag, 1997.
- [27] M. Tabakov, "A fuzzy clustering technique for medical image segmentation," in *2006 International Symposium on Evolving Fuzzy Systems*, Sep. 2006, pp. 118–122.
- [28] I. Stanoi, M. Riedewald, D. Agrawal, and A. E. Abbadi, "Discovery of influence sets in frequently updated databases," in *Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001)*. Roma, Italy: Morgan Kaufmann, 11–14 Sep. 2001, pp. 99–108.
- [29] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Dallas, Texas, USA: ACM, 16–18 May 2000, pp. 201–212.
- [30] Y. Tao, D. Papadias, and X. Lian, "Reverse kNN search in arbitrary dimensionality," in *VLDB*. Toronto, Canada: Morgan Kaufmann, 31 Aug.–3 Sept. 2004, pp. 744–755.
- [31] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis, *R-Trees: Theory and Applications*. Springer, 2005, ISBN 1-85233-977-2.
- [32] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.
- [33] *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB 2004)*. Toronto, Canada: Morgan Kaufmann, 31 Aug.–3 Sept. 2004.
- [34] *Proceedings of the 23rd International Conference on Data Engineering*. Istanbul, Turkey: IEEE, 15–20 Apr. 2007.



Ben Kao received the B.Sc. degree in computer science from the University of Hong Kong in 1989, the Ph.D. degree in computer science from Princeton University in 1995. From 1989–1991, he was a teaching and research assistant at Princeton University. From 1992–1995, he was a research fellow at Stanford University. He is currently associate professor of the Department of Computer Science at the University of Hong Kong. His research interests include database management systems, data mining, real-time systems, and information retrieval systems.



Sau Dan Lee is a Post-doctoral Fellow at the University of Hong Kong. He received his Ph.D. degree from the University of Freiburg, Germany in 2006 and his M.Phil. and B.Sc. degrees from the University of Hong Kong in 1998 and 1995. He is interested in the research areas of data mining, machine learning, uncertain data management and information management on the WWW. He has also designed and developed backend software systems for e-Business and investment banking.



Foris K. F. Lee is a M.Phil. student in the Department of Computer Science, The University of Hong Kong, under the supervision of Dr. Benjamin Kao. His research interest is uncertainty and data clustering



David Wai-lok Cheung received the M.Sc. and Ph.D. degrees in computer science from Simon Fraser University, Canada, in 1985 and 1989, respectively. Since 1994, he has been a faculty member of the Department of Computer Science in The University of Hong Kong. His research interests include database, data mining, database security and privacy. Dr. Cheung was the Program Committee Chairman of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2001), Program Co-Chair of PAKDD 2005, Conference Chair of PAKDD 2007, and the Conference Co-Chair of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009).



Wai-Shing Ho teaches in the Department of Computer Science, The University of Hong Kong. He received his Bachelor of Engineering and Doctor of Philosophy degrees from The University of Hong Kong in 1999 and 2005 respectively. His research interests include uncertain data classification and clustering, and online analytical processing of sequence data.