Hypersphere Dominance: An Optimal Approach

Cheng Long, Raymond Chi-Wing Wong, Bin Zhang, Min Xie
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, HongKong
{clong, raywong, zhangbin}@cse.ust.hk, mxieaa@ust.hk

ABSTRACT

Hyperspheres are commonly used for representing uncertain objects (in uncertain databases) and for indexing spatial objects (in spatial databases). An interesting operator on hyperspheres called *dominance* is to decide for two given hyperspheres whether one *dominates* (or *is closer than*) the other wrt a given query hypersphere. In this paper, we propose an approach called *Hyperbola* which is *optimal* in the sense that it gives neither false positives nor false negatives and runs in linear time wrt the dimensionality. To the best of our knowledge, *Hyperbola* is the first optimal approach for the dominance problem on hyperespheres with any dimensionality. We also study an application of the dominance problem which relies on the dominance operator as the core component. We conducted extensive experiments on both real and synthetic datasets which verified our approaches.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

Keywords

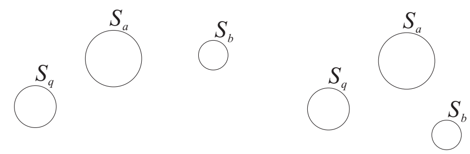
Hypersphere dominance; Hyperbola; Pruning

1. INTRODUCTION

Hypershperes are commonly used in uncertain databases and spatial databases. In uncertain databases, using a hypersphere to represent an uncertain object is widely adopted [6, 26, 2, 8]. For example, in GIS applications, the location of an object is measured by some GPS devices which may yield some *imprecise* measurements. Usually, a hypersphere for the location is given to describe the uncertain region that the object is located at. In spatial databases, many existing index structures such as M-tree [9], VP-tree [10], SS-tree [31], SS$^+$-tree [20] and SR-tree [18], rely on hyperspheres for efficient spatial queries. It is found in [31, 20, 18] that manipulating with hyperspheres in their indexing structures is very effective for answering similarity search queries in high-dimensional space compared with conventional well-known indexing structures based on hyperrectangles such as R-tree and R*-tree.

SIGMOD'14, June 22–27, 2014, Salt Lake City, UT, USA.

http://dx.doi.org/10.1145/2588555.2593669.



(a) (b)

Figure 1: An example illustrating the dominance problem

Given two hyperspheres S_a and S_b, and a query hypersphere S_q, we say that S_a *dominates* S_b wrt S_q if and only if for any query point q in S_q, the distance between q and any point in S_a is smaller than the distance between q and any point in S_b. That is, $\forall q \in S_q, \forall a \in S_a, \forall b \in S_b$, we have $Dist(a, q) < Dist(b, q)$ where $Dist(p, p')$ denotes the distance between two given points p and p'. For example, Figure 1(a) shows that S_a dominates S_b wrt S_q while Figure 1(b) shows that S_a does not dominate S_b wrt S_q.

In this paper, we study a *spatial dominance problem* (or the dominance problem in short) which is to determine whether a hypersphere S_a dominates another hypersphere S_b wrt a query hypersphere S_q.

Spatial dominance is a fundamental operator for pruning in lots of spatial queries. One example is the k nearest neighbor (kNN) query where we want to find k nearest neighbors from a query S_q. If k is set to 1, then we can discard S_b if we know that there exists an object S_a which dominates S_b wrt S_q. Another example is the reverse k nearest neighbor (RkNN) query where we want to find which objects consider a query object S_q as one of their k nearest neighbors. If k is set to 1, we can discard S_b if S_a dominates S_q wrt S_b. Some other examples include inverse ranking queries and dominating queries.

We consider three requirements for evaluating a method M for the spatial dominance problem, namely *correctness*, *soundness* and *efficiency* which are borrowed from [14] [1].

- *Correctness*: If method M returns $true$, then S_a dominates S_b wrt S_q.

- *Soundness*: If method M returns $false$, then S_a does not dominate S_b wrt S_q.

- *Efficiency*: The time complexity of method M is low. Specifically, our desired time complexity is linear to the dimensionality (i.e., $O(d)$).

All the above three requirements are essential to the spatial dominance problem since (1) a method which is not correct might prune some hypershperes that *should not be pruned* (this corresponds to an instance of "false positive"), which further implies that some

[1] The original term for "soundness" in [14] is "completeness".

Methods	Correct?	Sound?	Efficient?
MinMax decision criterion [26, 15]	Yes	No	Yes
MBR decision criterion [14]	Yes	No	Yes
GP decision criterion [22]	Yes	No	Yes
Trigonometric decision criterion [12]	No	Yes	Yes
Hyperbola (Our Method)	Yes	Yes	Yes

Table 1: Summary of existing methods for the spatial dominance problem on hyperspheres

solutions might be missed, (2) a method which is not sound might leave some hyperspheres that *should be pruned* not pruned (this corresponds to an instance of "false negative"), which introduces more burden on post-processing the remaining hypershperes, and (3) a method which is not efficient is undesirable since the spatial dominance operator is usually executed frequently. We say that a method is *optimal* if it satisfies all these three requirements.

Unfortunately, existing methods cannot address our dominance problem well. In fact, none of them are optimal. The *MinMax decision criterion* [26, 15], the *MBR decision criterion* [14] and the *GP decision criterion* [22], three of the existing methods, satisfy the correctness requirement and the efficiency requirement, but they do not satisfy the soundness requirement. The *Trigonometric decision criterion* [12], another existing method, satisfies the soundness requirement and the efficiency requirement, but it does not satisfy the correctness requirement. Table 1 shows the summary of four existing methods for the spatial dominance problem on hyperspheres.

Motivated by this, in this paper, we propose a new method called *Hyperbola* which can meet all the above three requirements, i.e., *Hyperbola* is optimal. This method utilizes an interesting geometry property based on a hyperbola and solves the dominance problem efficiently. Intuitively, we can construct a hyperbola based on the information about two given hyperspheres S_a and S_b. According to this hyperbola, we can partition the space into two parts. We then determine whether S_a dominates S_b wrt a query S_q by checking whether S_q is in one part of the partitioned space.

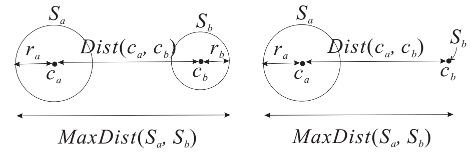
The following shows our contributions.

- Firstly, we develop a new method called *Hyperbola* for the dominance problem which is based on some geometry properties. To the best of our knowledge, *Hyperbola* corresponds to the first optimal approach (i.e., *Hyperbola* is correct, sound and efficient) for the dominance problem in any dimensional space.
- Secondly, we study an application, namely kNN query of the dominance problem, which relies on the dominance operator as its core component.
- Thirdly, we conducted extensive experiments with both synthetic and real datasets which verified our approaches.

The rest of the paper is organized as follows. Section 2 provides the formal definition of the dominance problem and discusses the adaptions of some existing decision criteria. Section 3 introduces a new decision criterion and based on this decision criterion, Section 4 introduces our *Hyperbola* method. Section 5 gives the related work and Section 6 studies an application of the dominance operator. Section 7 gives the empirical study and Section 8 concludes the paper.

2. PROBLEM DEFINITION & ADAPTIONS OF EXISTING SOLUTIONS

We give the formal definition of our spatial dominance problem in Section 2.1 and discuss the adaptions of some existing dominance decision criteria in Section 2.2.



(a) (b)

Figure 2: An example illustrating the definition of $MaxDist(\cdot, \cdot)$

2.1 The Dominance Problem

Consider three hyperspheres in the d-dimensional space, S_a, S_b and S_q. Each hypersphere has a *center* c which is a d-dimensional point and a *radius* r which is a non-negative real value. We denote the centers (radii) of S_a, S_b and S_q by c_a, c_b and c_q (r_a, r_b and r_q), respectively. Note that a d-dimensional point could be regarded as a d-dimensional hypersphere with its radius equal to 0.

Given a d-dimensional point p, we denote its i-th coordinate by $p[i]$. In this paper, we use the Euclidean distance as the distance metric, i.e., the distance between two d-dimensional points p and p', denoted by $Dist(p, p')$, is defined as follows.

$$Dist(p, p') = \sqrt{\sum_{i=1}^{d}(p[i] - p'[i])^2} \quad (1)$$

DEFINITION 1 (DOMINANCE). *Given three hyperspheres S_a, S_b and S_q (S_q is used as a query hypersphere), S_a is said to* dominate *S_b wrt S_q iff for any $q \in S_q$, any $a \in S_a$ is closer to q than any $b \in S_b$. That is,*

$$\forall q \in S_q, \forall a \in S_a, \forall b \in S_b : Dist(a, q) < Dist(b, q) \quad (2)$$

□

We define $Dom(S_a, S_b, S_q)$ as an indicator of whether S_a dominates S_b wrt S_q. Specifically, $Dom(S_a, S_b)$ is true if S_a dominates S_b wrt S_q and is false otherwise. To illustrate, consider Figure 1. We know that $Dom(S_a, S_b, S_q)$ is true for Figure 1(a) while $Dom(S_a, S_b, S_q)$ is false for Figure 1(b).

The dominance problem studied in this paper is defined as follows.

PROBLEM 1 (DOMINANCE PROBLEM). *Given three hyperspheres S_a, S_b and S_q, the* dominance problem *is to determine whether $Dom(S_a, S_b, S_q)$ is true or not.* □

Two d-dimensional hyperspheres S_a and S_b are said to *overlap* iff $Dist(c_a, c_b) \leq r_a + r_b$. We have the following lemma.

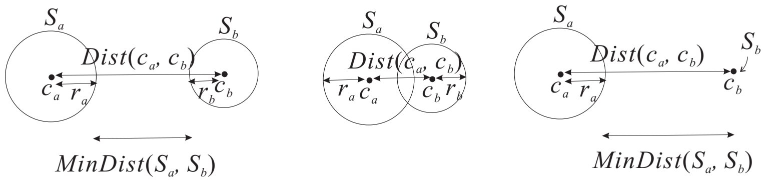
LEMMA 1 (OVERLAPPING CASE). *If S_a and S_b overlap, then $Dom(S_a, S_b, S_q)$ is false.* □

PROOF. Let p be a point in both S_a and S_b and p' be any point in S_q. Consider $a = p$, $b = p$ and $q = p'$. We have $Dist(a, q) = Dist(b, q)$ which implies that $Dom(S_a, S_b, S_q)$ is false. □

The above lemma suggests that if S_a and S_b overlaps, then we immediately know that $Dom(S_a, S_b, S_q)$ is false.

2.2 Adaptions of Existing Decision Criteria

There are some existing decision criteria originally proposed for hyperrectangles and some others originally designed for hyperspheres. In this section, we focus on the existing decision criteria originally proposed for hyperrectangles, which are closely related to our dominance problem. They are the *MinMax decision criterion*



(a) (b) (c)

Figure 3: An example illustrating the definition of $MinDist(\cdot, \cdot)$

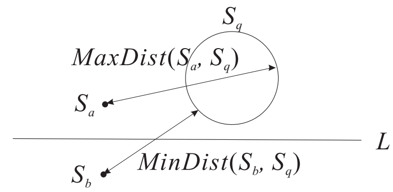


Figure 4: An example illustrating the proof of Lemma 3

[26, 15], the *MBR decision criterion* [14] [2] and the *corner-based decision criterion* [13]. In the next section, we will describe the existing decision criteria originally proposed for hyperspheres.

MinMax decision criterion: The *MinMax decision criterion* is denoted by $DC_{MinMax}(S_a, S_b, S_q)$. Before we describe it, we first give some notations which will be used in this criterion.

The *maximum distance* between S_a and S_b, denoted by $MaxDist(S_a, S_b)$, is defined to be the maximum distance between a point in S_a and a point in S_b. It is easy to verify that

$$MaxDist(S_a, S_b) = Dist(c_a, c_b) + r_a + r_b \quad (3)$$

For example, Figure 2(a) shows the maximum distance between two hyperspheres S_a and S_b. Figure 2(b) shows the maximum distance between a hypersphere S_a with non-zero radius and a hypersphere S_b with zero radius (or a point).

The *minimum distance* between S_a and S_b, denoted by $MinDist(S_a, S_b)$, is defined to be the minimum distance between a point in S_a and a point in S_b. It is easy to verify that $MinDist(S_a, S_b) =$

$$\begin{cases} Dist(c_a, c_b) - r_a - r_b & \text{if } Dist(c_a, c_b) > r_a + r_b \\ 0 & \text{otherwise (overlapping case)} \end{cases} \quad (4)$$

For example, Figure 3(a) shows the minimum distance between two non-overlapping hyperspheres S_a and S_b. Figure 3(b) shows the minimum distance between two overlapping hyperspheres S_a and S_b. Figure 3(c) shows the minimum distance between a hypersphere S_a with non-zero radius and another hypersphere S_b with zero radius (or a point).

With the above notations, we are ready to describe $DC_{MinMax}(S_a, S_b, S_q)$ which checks whether the maximum distance between S_a and S_q is strictly smaller than the minimum distance between S_b and S_q. Specifically, $DC_{MinMax}(S_a, S_b, S_q)$ is true if

$$MaxDist(S_a, S_q) < MinDist(S_b, S_q)$$

and false otherwise.

The MinMax decision criterion is simple, but it does not satisfy all three desired requirements for an optimal dominance operator introduced in Section 1 (i.e., correctness, soundness and efficiency). Specifically, the MinMax decision criterion is correct but not sound, which is shown in the following two lemmas.

LEMMA 2 (CORRECTNESS OF $DC_{MinMax}(S_a, S_b, S_q)$). *If $DC_{MinMax}(S_a, S_b, S_q)$ is true, then $Dom(S_a, S_b, S_q)$ is true.* □

PROOF. Since $DC_{MinMax}(S_a, S_b, S_q)$ is true, by definition, we have $MaxDist(S_a, S_q) < MinDist(S_b, S_q)$. Thus, we deduce that $\forall q \in S_q, \forall a \in S_a, \forall b \in S_b : Dist(a, q) \leq MaxDist(S_a, S_q) < MinDist(S_b, S_q) \leq Dist(b, q)$. □

[2] "MBR decision criterion" corresponds to the "$DDC_{optimal}$ decision criterion" in [14].

LEMMA 3 (NON-SOUNDNESS OF $DC_{MinMax}(S_a, S_b, S_q)$). *$DC_{MinMax}(S_a, S_b, S_q)$ is false does not imply that $Dom(S_a, S_b, S_q)$ is false.* □

PROOF. We prove this lemma by constructing an example such that $DC_{MinMax}(S_a, S_b, S_q)$ is false and $Dom(S_a, S_b, S_q)$ is true.

Consider a two-dimensional space containing one hypersphere S_a and another hypersphere S_b where their radii are equal to 0. The x-coordinates of the centers of both hyperspheres are the same but the the y-coordinate of the center of S_a is larger than that of the center of S_b. Figure 4 shows these two hyperspheres. Let L be the perpendicular bisector of the line connecting S_a and S_b. We can construct a hypersphere S_q with non-zero radius above L such that $MaxDist(S_a, S_q) > MinDist(S_b, S_q)$ as shown in the figure. Thus, $DC_{MinMax}(S_a, S_b, S_q)$ is false.

Note that for each $q \in S_q$, we know that $Dist(c_a, q) < Dist(c_b, q)$ which essentially implies that $\forall q \in S_q, \forall a \in S_a, \forall b \in S_b : Dist(a, q) < Dist(b, q)$ (i.e., $Dom(S_a, S_b, S_q)$ is true). □

Note that the MinMax decision criterion is sound only when S_q is a point.

In addition, the MinMax decision criterion can be determined in $O(d)$ time since both $MaxDist(S_a, S_q)$ and $MinDist(S_b, S_q)$ can be computed in $O(d)$ time.

MBR decision criterion: The *MBR decision criterion* denoted by $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ was proposed by [14], where $\mathcal{R}_a, \mathcal{R}_b$ and $\mathcal{R}_q$ are hyperrectangles (instead of hyperspheres studied in this paper). Similar to our dominance operator in the context of hyperspheres, $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ determines whether $\mathcal{R}_a$ dominates $\mathcal{R}_b$ wrt $\mathcal{R}_q$ in the context of hyperrectangles. According to [14], $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ is correct, sound and efficient *in the context of hyperrectangles*.

We propose to adapt $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ for our problem (in the context of hyperspheres), and denote the adapted decision criterion by $DC_{MBR}(S_a, S_b, S_q)$ which is described as follows. Let $\mathcal{R}_a$, $\mathcal{R}_b$ and $\mathcal{R}_q$ be the *minimum bounding hyperrectangles* of S_a, S_b and S_q, respectively. We define $DC_{MBR}(S_a, S_b, S_q)$ to be true if $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ is ture and false otherwise.

Unfortunately, similar to the MinMax decision criterion, this adapted MBR decision criterion does not satisfy all three desired requirements for the dominance operator. Specifically, the adapted MBR decision criterion is correct but is not sound, which will be shown in the following two lemmas.

LEMMA 4 (CORRECTNESS OF $DC_{MBR}(S_a, S_b, S_q)$). *If $DC_{MBR}(S_a, S_b, S_q)$ is true, then $Dom(S_a, S_b, S_q)$ is true.* □

PROOF. Let $\mathcal{R}_a, \mathcal{R}_b$ and $\mathcal{R}_q$ be the minimum bounding hyperrectangles of S_a, S_b and S_q, respectively. Since $DC_{MBR}(S_a, S_b, S_q)$ is true, by definition, we know that $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ is true. Since $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ is correct [14], we deduce that $\forall q \in \mathcal{R}_q, \forall a \in \mathcal{R}_a, \forall b \in \mathcal{R}_b : Dist(a, q) < Dist(b, q)$ which further implies that $\forall q \in S_q, \forall a \in$

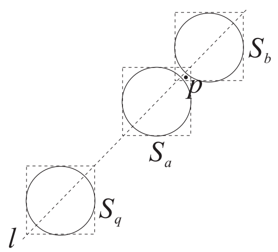


Figure 5: An example illustrating the proof of Lemma 5

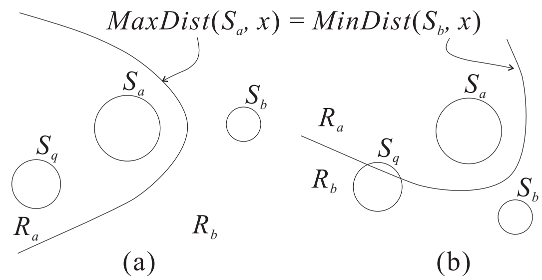


Figure 6: An example illustrating the Voronoi-based approach

$S_a, \forall b \in S_b : Dist(a, q) < Dist(b, q)$ (i.e., $Dom(S_a, S_b, S_q)$ is true). □

LEMMA 5 (NON-SOUNDNESS OF $DC_{MBR}(S_a, S_b, S_q)$). *$DC_{MBR}(S_a, S_b, S_q)$ is false does not imply that $Dom(S_a, S_b, S_q)$ is false.* □

PROOF. We prove this lemma by constructing an example such that $DC_{MBR}(S_a, S_b, S_q)$ is false and $Dom(S_a, S_b, S_q)$ is true.

Consider a two-dimensional space containing three hyperspheres with the same radii equal to r, S_a, S_b and S_q. The centers of these hyperspheres are along a virtual line l with slope equal to 1 such that the distance between the center of S_a and the center of S_q is $4 \cdot r$ and the distance between the center of S_b and the center of S_q is $6r + \delta$ where δ is a small number. Figure 5 shows these hyperspheres. In this example, it is easy to verify that $Dom(S_a, S_b, S_q)$ is true. Besides, we have that $\mathcal{R}_a$ (i.e., the MBR of S_a) intersects with $\mathcal{R}_b$ (i.e., the MBR for S_b). This, however, implies that $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ is false which further implies that $DC_{MBR}(S_a, S_b, S_q)$ is false. □

It is shown in [14] that the time complexity of determining $DC_{MBR}(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_q)$ is $O(d)$. Since the adapted decision criterion $DC_{MBR}(S_a, S_b, S_q)$ requires $O(d)$ time for constructing three hyperrectangles from three hyperspheres, the time complexity of determining $DC_{MBR}(S_a, S_b, S_q)$ is $O(d)$.

Corner-based decision criterion: The *corner-based decision criterion* denoted by $DC_{Corner}(S_a, S_b, S_q)$ was proposed by [13]. Similar to the MBR decision criterion, the corner-based decision criterion is designed for the dominance problem in the context of hyperrectangles. Thus, this decision criterion cannot be used for our dominance problem in the context of hyperspheres directly. Besides, the time complexity of executing this decision criterion is $O(2^d)$ which is prohibitively expensive for high-dimensional data and thus we do not adapt it in this paper as what we did for the MBR decision criterion.

3. DOMINANCE CONDITION

We discuss Voronoi-based approaches for capturing our hypersphere dominance condition in Section 3.1 and then propose our own approach in Section 3.2.

3.1 Voronoi-based Approach

One may propose to use a Voronoi-based approach for our dominance problem. Specifically, we partition the whole space into two regions R_a and R_b where R_a contains S_a and R_b contains S_b such that if S_q falls in R_a, then S_a dominates S_b wrt S_q. The two partitioned regions can be determined by a boundary represented by a curve in the form of "$MaxDist(S_a, x) = MinDist(S_b, x)$". Figure 6(a) shows the two regions with this boundary for the example in Figure 1(a) where $Dom(S_a, S_b, S_q)$ is true. Figure 6(b) shows the case in Figure 1(b) where $Dom(S_a, S_b, S_q)$ is false.

LEMMA 6. *The whole hypersphere S_q is in region R_a iff $Dom(S_a, S_b, S_q)$ is true.* □

PROOF. The whole hypersphere S_q is in region R_a

$$\Leftrightarrow \quad \forall q \in S_q, MaxDist(S_a, q) < MinDist(S_b, q)$$
$$\Leftrightarrow \quad \forall q \in S_q, \forall a \in S_a, \forall b \in S_b, Dist(a, q) < Dist(b, q)$$
$$\Leftrightarrow \quad Dom(S_a, S_b, S_q) \text{ is true}$$

□

A Voronoi-based approach looks promising for solving our dominance problem, but how to implement this approach efficiently on a d-dimensional space is not an easy task. Up to now, this dominance problem on hyperspheres has not been solved optimally although it is fundamental. This is mainly due to the difficulty in finding the shape of the region R_a in a high-dimensional space, one of the well-known challenges of computing a high-dimensional Voronoi diagram. To the best of our knowledge, there are only three existing studies [22, 12, 32] using a Voronoi-based approach for our dominance problem on hyperspheres as a component of their proposed algorithms, but they have some deficiencies.

- Firstly, the Voronoi-based approach studied in [22] called the *GP decision criterion*, originally used for a RkNN query, is not optimal for our dominance problem when the dimensionality is greater than 2. Specifically, it is optimal for 2-dimensional datasets only. As claimed in [22] that finding the optimal solution in a d-dimensional dataset is very complex where $d > 2$, [22] proposed an approximate solution for the dominance problem. Specifically, when $d > 2$, it transforms a d-dimensional dataset to a 2-dimensional dataset and adopts the method, originally designed for the 2-dimensional data, on this transformed dataset. Since a high-dimensional dataset is transformed to a low-dimensional dataset, some information is lost and thus optimality cannot be achieved.

- Secondly, [12] also adopted the Voronoi-based approach called the *Trigonometric decision criterion*, originally used for an all-nearest-neighbor query, is unfortunately not correct thought it is sound and efficient for our dominance problem. Detailed description of this adaption can be found in the appendix.

- Thirdly, the Voronoi-based approach studied in [32] called the *UV-Diagram decision criterion*, originally used for a 1NN query, is restricted to 2-dimensional datasets only. It is not clear how the approach can be extended to high-dimensional datasets such that the time complexity of the approach can be $O(d)$. Besides, the query object studied in [32] is a point only, but not a hypersphere.

In this paper, we propose a Voronoi-based method, which is correct and sound, for our dominance problem on data of any dimensionality. With some properties, this method can be done in $O(d)$ time, the first optimal approach in the literature.

Since there are some details of the above existing decision criteria, for clarity, we show the correctness, soundness and efficiency of these existing decision criteria in the appendix. The results of these decision criteria can be found in Table 1. Since the *UV-Diagram decision criterion* is restricted to 2-dimensional datasets only, we do not include it in the table nor in the appendix.

3.2 Our Approach

In this section, we derive a condition called the *minimum distance difference (MDD) condition* which is used to determine whether $Dom(S_a, S_b, S_q)$ is true or not. This condition has an interesting geometry property which can determine the boundary described above and can help to solve the dominance problem efficiently. We will discuss this interesting property in Section 4.

According to Definition 1, $Dom(S_a, S_b, S_q)$ is equivalent to determining whether Expression (2) is true or not.

With the notations $MaxDist(\cdot)$ and $MinDist(\cdot)$, it is easy to verify that Expression (2) is equivalent to the following.

$$\forall q \in S_q : MaxDist(S_a, q) < MinDist(S_b, q)$$

By (3), the above expression could be re-written as follows.

$$\forall q \in S_q : Dist(c_a, q) + r_a < MinDist(S_b, q) \quad (5)$$

Consider two cases. *Case 1:* $\forall q \in S_q : Dist(c_b, q) > r_b$. In this case, by (4), we know that

$$\forall q \in S_q : MinDist(S_b, q) = Dist(c_b, q) - r_b$$

Expression (5) can be re-written as follows.

$$\forall q \in S_q : Dist(c_b, q) - Dist(c_a, q) > r_a + r_b \quad (6)$$

Case 2: $\exists q \in S_q : Dist(c_b, q) \leq r_b$. This case is not possible. We show by contradiction. Suppose that there exists a point $q \in S_q$ such that $Dist(c_b, q) \leq r_b$. By (4), we know that $MinDist(S_b, q) = 0$. We derive that $Dist(c_a, q) + r_a < 0$, which leads to a contradiction that $Dist(c_a, q) + r_a$ must be non-negative.

By combining the above two cases, we conclude that Expression (5) is equivalent to the following expression.

$$\forall q \in S_q : Dist(c_b, q) - Dist(c_a, q) > r_a + r_b$$

The above expression can be further re-written as follows.

$$Min_{q \in S_q}(Dist(c_b, q) - Dist(c_a, q)) > r_a + r_b \quad (7)$$

The above expression is called the **minimum distance difference (MDD)** condition. Therefore, the dominance problem reduces to the problem of determining whether the MDD condition is true or not which we solve in Section 4.

4. ALGORITHM HYPERBOLA

Section 4.1 presents the high-level idea of our proposed algorithm called *Hyperbola*. Section 4.2 elaborates on some steps of *Hyperbola* in detail. Section 4.3 gives the theoretical analysis and the time complexity of *Hyperbola*.

4.1 Algorithm Hyperbola

Consider two cases. The first case is called the *overlapping case* in which S_a and S_b overlap. The second case is called the *non-overlapping case* in which S_a and S_b do not overlap.

Algorithm 1 Algorithm *Hyperbola*

Input: S_a, S_b and S_q
Output: a boolean value denoting whether $Dom(S_a, S_b, S_q)$ is true

```
1: if S_a and S_b overlap then
2:    return false
3: else
4:    P ← the hyperbola represented in the form of
      "Dist(c_b, x) − Dist(c_a, x) = r_a + r_b"
5:    R_a ← the region containing c_a which is one of the regions
      partitioned by P
6:    if S_q is in R_a then
7:       return true
8:    else
9:       return false
```

Consider the overlapping case. According to Lemma 1, we know that $Dom(S_a, S_b, S_q)$ is false. There is no need to check with the MDD condition.

Consider the non-overlapping case. We have to check with the MDD condition in order to determine whether $Dom(S_a, S_b, S_q)$ is true or not. In the following, we observe an interesting geometry property for the MDD condition which helps to check the MDD condition efficiently.

Let P be the hyperbola represented in the following form.

$$Dist(c_b, x) - Dist(c_a, x) = r_a + r_b \quad (8)$$

where x is a d-dimensional point along the hyperbola which has two focal points, namely c_a and c_b. We use P to partition the space into two regions and denote by R_a the one that contains c_a. Then, we have the following property.

LEMMA 7 (NON-OVERLAPPING CASE). *The whole hypersphere S_q is in R_a iff the MDD condition is satisfied.* □

PROOF. The whole hypersphere S_q is in R_a

$$\begin{aligned} &\Leftrightarrow \forall q \in S_q, q \text{ is in } R_a \\ &\Leftrightarrow \forall q \in S_q, Dist(c_b, q) - Dist(c_a, q) > r_a + r_b \\ &\Leftrightarrow Min_{q \in S_q}(Dist(c_b, q) - Dist(c_a, q)) > r_a + r_b \\ &\Leftrightarrow \text{the MDD condition is satisfied} \end{aligned}$$

□

The above lemma suggests that in order to determine whether the MDD condition is satisfied or not, we can check whether S_q is in R_a. If yes, we know that the MDD condition is satisfied and thus $Dom(S_a, S_b, S_q)$ is true; otherwise, we know that the MDD condition is not satisfied and thus $Dom(S_a, S_b, S_q)$ is false. This corresponds to the idea of our *Hyperbola* algorithm which we present in Algorithm 1.

THEOREM 1. *Algorithm 1 is correct and sound.* □

PROOF. This could be easily verified by Lemma 7 and the equivalence between the MDD condition and the dominance condition. □

According to the above theorem, algorithm *Hyperbola* satisfies the first two requirements for our dominance problem. In the following, we introduce the detailed steps of *Hyperbola* in Section 4.2 and show that *Hyperbola* runs in $O(d)$ time (i.e., *Hyperbola* is efficient) in Section 4.3.

4.2 Detailed Steps

Algorithm 1 looks straightforward but how to perform the step of determining whether S_q is in R_a *efficiently* needs more careful design. Note that this step has to be performed only in the case that S_a and S_b do not overlap.

In this paper, we propose the following two-step method to determine whether S_q is in R_a.

- **Step 1 (Finding Minimum Distance):** We find the minimum distance d_{min} between hyperbola P and point c_q.
- **Step 2 (Checking Minimum Distance):** We conclude that S_q is in R_a if $d_{min} > r_q$ and c_q is inside R_a; otherwise, we conclude that S_q is not in R_a.

The correctness of the above two-step method is obvious and its time complexity is $O(d)$ since Step 1 could be done in $O(d)$ which will be shown in the following Section 4.3 and Step 2 also runs $O(d)$ time which could be verified easily.

4.3 Theoretical Analysis & Time Complexity

Now, we present an efficient method to find the minimum distance d_{min} between the hyperbola P and the point c_q in $O(d)$ time.

4.3.1 An Explicit Expression of P

In this subsection, we give an explicit expression of P.

Let $r_{ab} = r_a + r_b$. From (8), P is represented in the following form.

$$Dist(c_b, x) - Dist(c_a, x) = r_a + r_b$$

Since $r_{ab} = r_a + r_b$, by (1), it can be re-written as follows.

$$\left(\sqrt{\textstyle\sum_{i=1}^{d}(c_b[i] - x[i])^2}\right)^2 = \left(\sqrt{\textstyle\sum_{i=1}^{d}(c_a[i] - x[i])^2} + r_{ab}\right)^2$$

The above expression can be simplified as follows.

$$\textstyle\sum_{i=1}^{d}(c_b[i] - x[i])^2 - \sum_{i=1}^{d}(c_a[i] - x[i])^2 - r_{ab}^2 = 2r_{ab}\sqrt{\textstyle\sum_{i=1}^{d}(c_a[i] - x[i])^2}$$

Squaring both sides of the above expression results in the following expression.

$$\left(\textstyle\sum_{i=1}^{d} c_b^2[i] - \sum_{i=1}^{d} c_a^2[i] + 2\sum_{i=1}^{d} x[i](c_a[i] - c_b[i]) - r_{ab}^2\right)^2 = 4r_{ab}^2\left(\textstyle\sum_{i=1}^{d} x^2[i] + \sum_{i=1}^{d} c_a^2[i] - 2\sum_{i=1}^{d} x[i]c_a[i]\right) \quad (9)$$

The above expression looks a little bit complicated to proceed. Fortunately, we can make use of the hyperbola property and can simplify the above expression by transforming points to a new coordinate system.

In a typical hyperbola, there are two fixed points (or focal points) and the point along the hyperbola should satisfy that the difference between its distance from one of the fixed points and its distance from the other fixed point is equal to a fixed value r. In a typical representation, one of the fixed points has the coordinates equal to $(-\alpha, 0, 0, ..., 0)$ and the other fixed point has the coordinates equal to $(\alpha, 0, 0, ..., 0)$ where α is a non-negative real number. Note that there are $d-1$ zeros in the coordinates of both fixed points which can be used to simplify some derivations. The sufficient condition that this hyperbola exists is that $r < 2\alpha$.

Motivated by the above observation, since the two fixed points are c_a and c_b in our hyperbola, we perform a coordinate transformation for our hyperbola such that in the new coordinate system, c_a has its coordinates equal to $(-\alpha, 0, 0, ..., 0)$ and c_b has its coordinates equal to $(\alpha, 0, 0, ..., 0)$. Here, in our hyperbola, α is set to $Dist(c_a, c_b)/2$ and r is set to r_{ab}. Note that $r_{ab} < 2\alpha$, which satisfies the condition that a hyperbola exists. This is because we perform to check whether S_q is in R_a only when know that S_a and S_b does not overlap, which means that $r_a + r_b < Dist(c_a, c_b)(= 2\alpha)$.

With this coordinate transformation, c_a, c_b and c_q in the original coordinate system is transformed to c'_a, c'_b and c'_q in the new coordinate system, respectively. In the following, for simplicity, we do not change the notations of c_a, c_b and c_q. Instead, we just substitute the coordinate of these points in the new coordinate system. It is easy to verify that the coordinate transformation takes $O(d)$ time.

After the coordinate transformation, Expression (9) denoting the hyperbola can be written as follows.

$$(4\alpha x[1] + r_{ab}^2)^2 = 4r_{ab}^2\left(\textstyle\sum_{i=1}^{d} x^2[i] + \alpha^2 + 2\alpha x[1]\right)$$

With some simple derivations, we have the following.

$$4r_{ab}^2 \textstyle\sum_{i=1}^{d} x^2[i] + 4r_{ab}^2\alpha^2 - 16\alpha^2 x^2[1] - r_{ab}^4 = 0 \quad (10)$$

Let $F(x) = 4r_{ab}^2 \sum_{i=1}^{d} x^2[i] + 4r_{ab}^2\alpha^2 - 16\alpha^2 x^2[1] - r_{ab}^4$. Thus, the hyperbola P can be written in the following form.

$$F(x) = 0$$

4.3.2 Finding d_{min} Between P and c_q

We want to find the minimum distance d_{min} between P and c_q. Note that P can be written in the form of "$F(x) = 0$". What we want to solve is the following constrained optimization problem.

Minimize $Dist(c_q, x)$
subject to $F(x) = 0$

The solution of above optimization corresponds to d_{min}. Besides, the above optimization is equivalent to the following optimization where the objective function is changed from $Dist(c_q, x)$ to $Dist(c_q, x)^2$ (since $Dist(c_q, x)$ is non-negative).

Minimize $Dist(c_q, x)^2$
subject to $F(x) = 0$

By the Lagrange multipliers [4], we just need to consider the corresponding Lagrange function $G(x)$ as follows.

$$G(x) = Dist(c_q, x)^2 + \lambda \cdot F(x)$$

where λ is a Lagrange multiplier which is a real number.

It is easy to verify the following gradient of $G(x)$.

$$\begin{cases} \frac{\partial G(x)}{\partial x[1]} & = & -2(c_q[1] - x[1]) + \lambda(8r_{ab}^2 x[1] - 32\alpha^2 x[1]) \\ \frac{\partial G(x)}{\partial x[2]} & = & -2(c_q[2] - x[2]) + \lambda 8r_{ab}^2 x[2] \\ \frac{\partial G(x)}{\partial x[3]} & = & -2(c_q[3] - x[3]) + \lambda 8r_{ab}^2 x[3] \\ & \vdots & \vdots \\ \frac{\partial G(x)}{\partial x[d]} & = & -2(c_q[d] - x[d]) + \lambda 8r_{ab}^2 x[d] \end{cases}$$

Setting the gradient of $G(x)$ to 0 results in the following equations.

$$\begin{cases} c_q[1] - x[1] & = & \lambda(4r_{ab}^2 x[1] - 16\alpha^2 x[1]) \\ c_q[2] - x[2] & = & \lambda 4r_{ab}^2 x[2] \\ c_q[3] - x[3] & = & \lambda 4r_{ab}^2 x[3] \\ & \vdots & \vdots \\ c_q[d] - x[d] & = & \lambda 4r_{ab}^2 x[d] \end{cases} \quad (11)$$

From (11), we derive the following equations.

$$x[1] = \frac{c_q[1]}{1 + 4r_{ab}^2\lambda - 16\alpha^2\lambda} \quad (12)$$

$$x[i] = \frac{c_q[i]}{4r_{ab}^2\lambda+1} \text{ where } 2 \leq i \leq d \quad (13)$$

We insert the above d equations for $x[1], x[2], ..., x[d]$ into $F(x) = 0$. We have

$$\frac{(16\alpha^2-4r_{ab}^2)c_q^2[1]}{(1+4r_{ab}^2\lambda-16\alpha^2\lambda)^2} + r_{ab}^4 - 4r_{ab}^2\alpha^2 = \frac{4r_{ab}^2(\sum_{i=2}^d c_q^2[i])}{(4r_{ab}^2\lambda+1)^2}$$

Let $a_1 = (16\alpha^2 - 4r_{ab}^2)c_q^2[1]$, $a_2 = r_{ab}^4 - 4r_{ab}^2\alpha^2$, $a_3 = 4r_{ab}^2(\sum_{i=2}^d c_q^2[i])$, $a_4 = 4r_{ab}^2$ and $a_5 = 4r_{ab}^2 - 16\alpha^2$. The above equation can be simplified as follows.

$$\frac{a_1}{(1+a_5\lambda)^2} + a_2 = \frac{a_3}{(1+a_4\lambda)^2}$$

It can be further expressed in the following quartic form.

$$A\lambda^4 + B\lambda^3 + C\lambda^2 + D\lambda + E = 0 \quad (14)$$

where

$$\begin{aligned} A &= a_2a_4^2a_5^2 \\ B &= 2a_2a_4^2a_5 + 2a_2a_4a_5^2 \\ C &= a_1a_4^2 + a_2a_4^2 + 4a_2a_4a_5 + a_2a_5^2 - a_3a_5^2 \\ D &= 2a_1a_4 + 2a_2a_4 + 2a_2a_5 - 2a_3a_5 \\ E &= a_1 + a_2 - a_3 \end{aligned}$$

We know that the solutions for a quartic equation can be found in $O(1)$ time [17]. Thus, we can find the solutions for Equation (14) in $O(1)$ time. Besides, at most four solutions for λ can be obtained. For each solution for λ, we can compute the value for x according to Equation (12) and Equation (13) in $O(1)$ time, and then can compute the distance $Dist(c_q, x)$ in $O(d)$ time. We pick the smallest distance value among all computed distance values as the final d_{min} value. Thus, d_{min} can be computed in $O(d)$ time.

LEMMA 8. *d_{min} can be computed in $O(d)$ time.* □

THEOREM 2. *The time complexity of* Hyperbola *as shown in Algorihtm 1 is $O(d)$.* □

According to the above theorem, *Hyperbola* satisfies the third requirement for our dominance problem.

5. RELATED WORK

We study the related work of hyperspheres, spatial dominance and existing queries using dominance in Section 5.1, Section 5.2 and Section 5.3, respectively.

5.1 Hyperspheres

There are a lot of existing studies about hyperspheres which we study with two branches. The first branch includes the queries in uncertain databases where the uncertain objects are usually represented with hyperspheres due to the imprecise measurements of these objects. Some example include [6, 26, 2, 8].

The second branch includes the indexes whose index entries are represented in the form of hyperspheres. Some representative examples are SS-tree [31], SS$^+$-tree [20], SR-tree [18], M-tree [9] and VP-tree [10]. [31] and [20] reported that SS-tree and its variation, SS$^+$-tree, outperform the conventional well-known indexing structure, R*-tree, in similarity search queries in a high-dimensional space, which is commonly used in the literature of image and video retrieval. SR-tree is a hybrid tree structure which integrates the advantage of R*-tree and the advantage of SS$^+$-tree in order to speed up nearest neighbor queries in a high-dimensional space. Both M-tree and VP-tree are tree structures which are designed to index objects in a metric space.

5.2 Spatial Dominance

The spatial dominance operator has been studied in the context of hyperrectangles [14]. Four decision criteria were studied in [14]. The first decision criterion is the MinMax decision criterion. Same as the case for hyperspheres, this decision criterion is correct but not sound for hyperrectangles and runs in $O(d)$ time. The second decision criterion is the Voronoi-based decision criterion which is correct and sound for hyperrectangles, but runs in $O(2^d)$ time. The third criterion is the corner-based decision criterion which is correct and sound for hyperrectangles, but runs in $O(2^d)$ time. The fourth decision criterion is the MBR decision criterion which is correct and sound for hyperrectangles, and runs in $O(d)$ time [14]. However, the MBR decision criterion cannot be applied to hyperspheres directly since it makes use of the property of hyperrectangles which do not hold for hyperspheres. In particular, this decision criterion requires that the maximum distance between a hyperrectangle r and another hyperrectangle r' is split into d components where each component is the maximum distance between r and r' on one dimension. Since each dimension of a hyperrectangle is represented in the form of a closed interval (with a minimum value and a maximum value on this dimension), the splitting process can be done conveniently for hyperrectangles which is not the case for hyperspheres. These criteria are all based on hyperrectangles instead of hyperspheres studied in this paper. Note that different applications use different shapes to represent uncertain objects. In some applications, an uncertain object is represented by a hypersphere as described in Section 5.1 and in some other applications, it is represented by a hyperrectangle. In this paper, we focus on the former case.

To the best of our knowledge, we are the first to propose a dominance operator in the context of hyperspheres, which is correct, sound and efficient, for any dimensionality.

5.3 Existing Queries Using Dominance

There are an abundance of existing queries which depend on the dominance definition studied in this paper. In the context of points (which could be regarded as special cases of hyperspheres), those queries are kNN queries [26, 15], RkNN queries [29], inverse ranking queries [21], dominating queries [33], spatial skyline queries [27] and reverse skyline queries [11].

In the context of hyperspheres, similar queries also adopt this dominance definition. Consider kNN queries. Among many existing studies on kNN queries in uncertain databases [32, 34, 25, 3, 7, 30, 28, 16, 19] adopting the dominance definition (or similar definitions), only [32] is closely related to us. Specifically, some [16, 30, 28] only focus on these queries on the data containing points (not hyperspheres) with an uncertain query object represented in some forms (e.g., hyperspheres). Some [25, 7] only focus on these queries on the data containing uncertain data with a query point (not a hypersphere). Some [34] focus on uncertain data in the form of the shapes other than hyperspheres (e.g., hyperrectangles). Some techniques like [19] with sampling may introduce errors for the queries when the number of sampling points is insufficient. Some techniques like [3] choose k objects "independently" without considering the correlation of the objects in the answer set of the queries. [32] studied the 1NN query on the database containing hyperspheres. However, it is different from ours. Firstly, we study kNN query where k could be any positive integer while [32] focuses on 1NN query. Secondly, we study the dominance problem for any dimensionality while [32] focuses on the dominance problem embedded in the queries on 2-dimensional dataset only. Thirdly, the query object in our kNN query is a hypersphere while the query object in the 1NN query studied in [32] is a point.

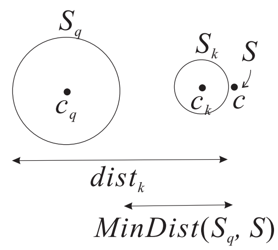


Figure 7: An example illustrating the proof of Lemma 10

Consider RkNN queries. [1] focuses on objects represented in the form of hyperrectangles and [5] works on a discrete space only instead of a continuous space inside a hypersphere as studied in this paper. [22] studies a RkNN query which involves a spatial dominance problem similar to us. Nevertheless, when the dimensionality d of the dataset is greater than 2, [22] transforms the dataset to a 2-dimensional dataset with some information loss and the resulting approach is non-optimal for the spatial dominance problem.

Consider all-nearest neighbor queries. [12] adopts the dominance operator for all-nearest neighbor queries considering S_q as a hypersphere, S_a as a point only and S_b as either a point or a hypersphere. However, as we described before, though the adapted method considering all S_q, S_q and S_b as hyperspheres is sound and efficient, it is not correct.

Consider inverse ranking queries. [23] studies these queries when the data objects are represented in the form of hyperrectangles, but not hyperspheres.

Consider dominating queries. [24] focuses on a discrete space instead of a continuous space as studied in this paper.

6. APPLICATION

In the previous section, we describe how to determine whether $Dom(S_a, S_b, S_q)$ is true or not efficiently. In this section, we discuss how to use $Dom(S_a, S_b, S_q)$ in a popular query, namely the k nearest neighbor (kNN) query. Although there are other applications using $Dom(S_a, S_b, S_q)$ (e.g., reverse kNN queries [22], inverse ranking queries [21] and dominating queries [33, 24]), for the sake of space, we study the kNN query only.

As described in Section 5, although there are a vast number of studies for the kNN query on uncertain databases [32, 34, 25, 3, 7, 30, 28, 16, 19], surprisingly, none of them study the kNN query on hyperspheres with any dimensionality which we study in this section.

Let D be a set of N hyperspheres, $S_1, S_2, ..., S_N$. We define a *kNN query* in the context of *hyperspheres* as follows.

DEFINITION 2 (kNN QUERY). *Given a query hypersphere S_q, a k* nearest neighbor (kNN) query *of S_q is to find a set of hyperspheres in D which are not dominated by S_k wrt S_q where S_k is a hypersphere in D which has the k-th smallest maximum distance to S_q.*

Note that if there are multiple hyperspheres in D which has the k-th smallest maximum distance to S_q, all these hyperspheres are kept in the answer set of this query. Notation S_k in the above definition is applied to each of these hyperspheres. In the following discussion, we assume that there is only one S_k for ease of discussion. All techniques can be extended easily to the case where there exist multiple hyperspheres for S_k.

The existing algorithms [26, 15] can be adapted for this kNN query in the context of hyperspheres. Specifically, these algorithms have to maintain a *best-known list L* storing hyperspheres/points found so far when the algorithms are being executed. This list is updated when a better hypersphere is found during the execution process.

Before describing how we maintain the list L in our adapted algorithm, we give the following lemmas first.

LEMMA 9. *Let S_q be a query hypersphere and D' be a subset of D. Let L be a list of hyperspheres of the kNN query on D' with the query hypersphere as S_q and $dist_k$ be the k-th smallest maximum distance of a hypersphere in L to S_q. Given a hypersphere $S \in D \setminus D'$, if $dist_k$ is smaller than the minimum distance of S to S_q, then S is not in the answer set of the kNN query on $D' \cup \{S\}$ with the query hypersphere as S_q.* □

PROOF. Let S_k be the hypersphere in L which has the k-th smallest maximum distance to S_q equal to S_k. Since $dist_k$ is smaller than the minimum distance of S to S_q, we have $MaxDist(S_k, S_q) < MinDist(S, S_q)$. Thus, $DC_{MinMax}(S_k, S, S_q)$ is true. By Lemma 2, $Dom(S_k, S, S_q)$ is true. S is dominated by S_k wrt S_q. Thus, S is not in the answer set of the kNN query on $D' \cup \{S\}$ with the query hypersphere as S_q. □

The above lemma suggests that given a best-known list L corresponding to the answer set of the kNN query on D' (containing the hyperspheres accessed so far) with the query hypersphere as S_q, whenever we access a hypersphere S, if $dist_k$ is smaller than the minimum distance of S to S_q where $dist_k$ is the k-th smallest maximum distance of a hypersphere in L to S_q, we can prune S since S is not in the answer set of the kNN query on $D' \cup \{S\}$ (and even on D).

Based on the traditional rule of pruning used in the literature, one may think that if $dist_k$ is larger than or equal to the minimum distance of S to S_q, S must be in the answer set of the kNN query on $D' \cup \{S\}$, which, however, is not always true in our case.

LEMMA 10. *Let S_q be a query hypersphere and D' be a subset of D. Let L be a list of hyperspheres of the kNN query on D' with the query hypersphere as S_q and $dist_k$ be the k-th smallest maximum distance of a hypersphere in L to S_q. Given a hypersphere $S \in D \setminus D'$, if $dist_k$ is larger than or equal to the minimum distance of S to S_q, it is possible that S is not in the answer set of the kNN query on $D' \cup \{S\}$ with the query hypersphere as S_q.* □

PROOF. We show this lemma by constructing an example that S is not in the answer set of the kNN query on $D' \cup \{S\}$ with the query hypersphere as S_q.

Let $k = 1$. Consider a two-dimensional space containing three hyperspheres, S_k, S_q and S. Let r_k, r_q and r be the radii of S_k, S_q and S, respectively. Let c_k, c_q and c be the centers of S_k, S_q and S, respectively. Besides, $r_q > r_k$ and r is near to zero. Suppose that the centers of these hyperspheres are along a horizontal line where (1) the distance between c_q and c_k is larger than $r_q + r_k$, (2) c_k along the line segment between c_q and c, (3) the distance between c_k and c is equal to $r_k + \delta$ where δ is a very small constant. Figure 7 shows these hyperspheres. Let $D' = \{S_k\}$.

In this example, we can verify that $dist_k$ (which is equal to $Dist(c_q, c_k) + r_q + r_k$) is larger than or equal to the minimum distance of S to S_q (which is equal to $Dist(c_q, c_k) - r_q + r_k + \delta$). Besides, $\forall q \in S_q, \forall a \in S_k, \forall s \in S : Dist(a, q) < Dist(s, q)$. This implies that $Dom(S_k, S, S_q)$ is true. S is dominated by S_k wrt S_q. Thus, S is not in the answer set of the kNN query on $D' \cup \{S\}$ with the query hypersphere as S_q. □

According to the above lemma, it is possible that S is not in the answer set of the kNN query on $D' \cup \{S\}$ with the query hypersphere as S_q if $dist_k$ is larger than or equal to the minimum distance of S to S_q. In this case, in order to determine whether S is in the answer set, we have to check whether S_k dominates S wrt S_q where S_k is a hypersphere in L which has the k-th smallest maximum distance to S_q.

Now, based on the above lemmas, we are ready to describe how we adapt these existing algorithms to our kNN query as follows. Let $\mathcal{A}$ be one of the existing algorithms. Before $\mathcal{A}$ is executed, L is initialized to $\emptyset$. Whenever $\mathcal{A}$ determines a hypersphere S which is a potential candidate to be inserted into L, it performs the following steps.

- If the number of hyperspheres maintained in L is smaller than k, it inserts S into L.
- Otherwise, it performs the following steps. There are at least k hyperspheres in L. Let $dist_k$ be the k-th smallest maximum distance between S_q and a hypersphere in L. Let $dist_{max}$ and $dist_{min}$ be the maximum distance between S and S_q, and the minimum distance between S and S_q, respectively. It performs different steps under different cases based on $dist_k$.
  - *Case 1:* $dist_{max} \leq dist_k$.
    In this case, it inserts S into L. It removes each hypersphere S' in L such that S_k dominates S' wrt S_q where S_k is the new hypersphere in L which has the k-th smallest maximum distance to S_q.
  - *Case 2:* $dist_{min} \leq dist_k < dist_{max}$.
    We check whether S_k dominates S wrt S_q. If yes, it prunes S. Otherwise, it inserts S into L.
  - *Case 3:* $dist_{min} > dist_k$.
    It prunes S.

According to the above two lemmas, it is easy to verify the correctness of the above adapted algorithm of $\mathcal{A}$.

THEOREM 3. *The adapted algorithm of $\mathcal{A}$ returns the optimal answer for the kNN query.* □

7. EMPIRICAL STUDIES

Four real datasets were used in our experiments, namely NBA, Color, Texture and Forest. *NBA* is a data set downloaded from the NBA official website containing Great NBA Players' technical statistics from 1960 to 2001. NBA has 17,265 points with 17 dimensions. *Color* and *Texture* are two datasets each of which contains 68,040 image features (or points in our context) extracted from a Corel image collection. *Color* is 9-dimensional and *Texture* is 16-dimensional. *Forest* is a dataset containing 82,012 10-dimensional points that was derived from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. For each point in the real dataset, we generated a corresponding hypersphere by using the point as the center of the hypersphere to be generated and sampling a real number with Gaussian distribution $\mathcal{N}(\mu, \sigma)$ as the radius of the hypersphere. μ is user parameter which will be studied in our experiments and σ is set to $\mu/4$ by default.

Synthetic datasets were also used in our experiments. We generated a dataset containing N hyperspheres in the d-dimensional space as follows. Firstly, we generated N d-dimensional points as the centers of the hyperspheres to be generated by sampling d coordinates for each point. The sampled coordinates follow the Gaussian distribution with its mean equal to 100 and its standard deviation equal to 25. Secondly, for each data point generated, we constructed a hypersphere with its center as this point and its radius as a real number sampled similarly as for the real datasets. In our experiments, we study the effects of data size N, dimensionality d and parameter μ. The settings of these factors are shown in Table 2 where the default values are shown in bold.

Parameter	Values
Average radius value (μ)	5, 10, **50**, 100
No. of hyperspheres (N)	20k, 60k, **100k**, 140k, 180k
Dimensionality (d)	2, 4, **6**, 8, 10
Parameter k for kNN	1, **10**, 20, 30

Table 2: Parameter settings for synthetic datasets

All algorithms were implemented in C/C++ and ran on a CentOS linux server with a 2.66GHz processor and 4GB memory.

In the following, we conducted experiments on the hypersphere problem in Section 7.1 and on the kNN problem in Section 7.2.

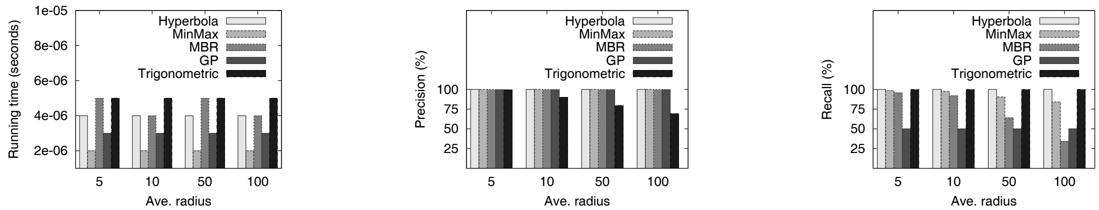
7.1 Hypersphere Dominance

In this part, we study five methods, namely *Hyperbola*, *MinMax*, *MBR*, *GP* and *Trigonometric*, for the hypersphere dominance problem defined in Section 2. Specifically, for each reported experiment, we created a workload containing 10,000 random queries each involving three hyperspheres S_a, S_b and S_q selected from the dataset randomly, and ran the algorithm 10 times. We took the average results as the final reported results.

We used three measures, namely *execution time*, *precision* and *recall*. *Execution time* means the running time of the algorithm. *Precision* and *recall* are defined as follows. We used the results (i.e., "true" or "false") returned by *Hyperbola* as ground truth (note that *Hyperbola* is the only algorithm which is both correct and sound). The *precision* of an algorithm is defined to be $TP/(TP + FP)$ where TP and FP correspond to the number of "true positives" and the number of "false positives" of this algorithm, respectively, when a workload of 10,000 queries is performed. Note that an algorithm which is correct has its *precision* always equal to 100%. The *recall* of an algorithm is defined to be $TP/(TP + FN)$ where TP is as defined above and FN correspond to the number of "false negatives" of this algorithm when a workload of 10,000 queries is performed. Note that an algorithm which is sound has its *recall* always equal to 100%.

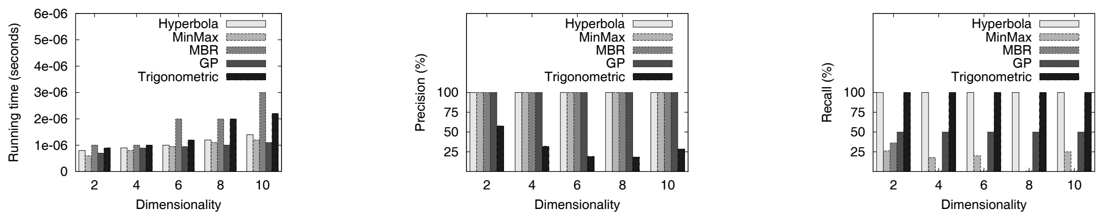
Effects of Ave. Radius μ. The results on the real dataset *NBA* are shown in Figure 8. Consider the results of execution time in Figure 8(a). We notice that *MinMax* runs the fastest, followed by *GP*, *Hyperbola*, *MBR*, and *Trigonometric*. Consider the precision results in Figure 8(b). They show that all algorithms except *Trigonometric* have their precision always equal to 100%, which verified our theoretical analysis that all algorithms except *Trigonometric* are correct. Besides, *Trigonometric* has its precision worse and worse when μ increases. Consider the recall results in Figure 8(c). Only *Hyperbola* and *Trigonometric* have their recall always equal to 100%, which verified our theoretical analysis that only *Hyperbola* and *Trigonometric* are sound.

Effects of Dimensionality (d). The results are shown in Figure 9(a) (execution time), Figure 9(b) (precision) and Figure 9(c) (recall). We have the following observations. First, each algorithm has its running time slightly increase when the dimensionality increases. This could be easily explained by the fact that each algorithm involves some distance computations whose cost grows with the dimensionality. Second, consistent with the results shown in Figure 8, *Hyperbola* runs slightly slower than *MinMax* and *GP* (the reason is probably that MinMax simply computes only two distances, namely the maximum one and the minimum one, which is



(a) Execution time (b) Precision (c) Recall

Figure 8: Effects of the Ave. Radius μ for the Dominance problem (NBA)



(a) Execution time (b) Precision (c) Recall

Figure 9: Effects of the Dimensionality d for the Dominance problem (Synthetic)

cheap and GP always does the computations in the 2D space only (since in case of a higher dimensional space, it transforms the space to a 2D one)), but it runs faster than both *MBR* and *Trigonometric* (the reason is probably that *MBR* involves an additional adaption step and *Trigonometric* involves some trigonometric computations which are costly). Third, again, it shows that *Hyperbola* is the only algorithm that is both correct and sound.

Experiments on Real Datasets. The results are shown in Figure 10. Consider the results of execution time in Figure 10(a). As could be noticed, the same pattern found on the synthetic datasets, i.e., *MinMax* is the fastest, followed by *GP*, *Hyperbola*, *MBR* and *Trigonometric*, could be found in the real datasets as well. The results of precision (in Figure 10(b)) and those of recall (in Figure 10(c)) verify that *Hyperbola* is the only algorithm that is both accurate and sound.

Additional Experiments. This part includes some additional experiments. First, we conducted experiments on synthetic datasets in a high-dimensional space. Specifically, we vary the dimensionality with 25, 50, 75, 100, and Figure 11 shows the results of the execution time. Second, we conducted experiments on synthetic datasets where we generated the coordinates and also the radii by using different distributions. We consider two distributions, namely Gaussian distribution and Uniform distribution. Gaussian distribution is used by default in our experiments and our method of generating the coordinates (and also the radii) using Gaussian distribution have been explained in the beginning part of Section 7. Our method of generating the coordinates (and also the radii) using Uniform distribution is to first specify a range and then sample a value from the range randomly. The ranges we used for generating coordinates and radii are both [0, 200]. We denote by "G-U" if the generated coordinates follow Gaussian distribution and the generated radii follow Uniform distribution. The meanings for "G-G", "U-G" and "U-U" are defined in an analogous way. Figure 12 shows the results of execution time. We notice that *Hyperbola* and *Trigonometric* favor the datasets in Gaussian distribution slightly and while other algorithms are not affected by the distributions significantly.

7.2 kNN Query

We index our dataset with an SS-Tree [31] which is a popular index for hyperspheres. We adopted two well-known strategies for searching over the SS-Tree we built, namely *DF* [26] which is a depth-first search strategy and *HS* [15] which is a best-first

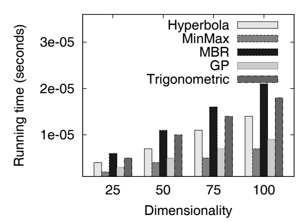


Figure 11: Execution time for the Dominance problem (Datasets in high dimensional space)

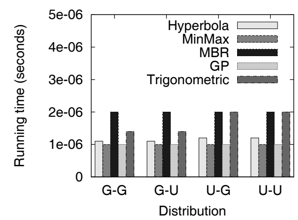


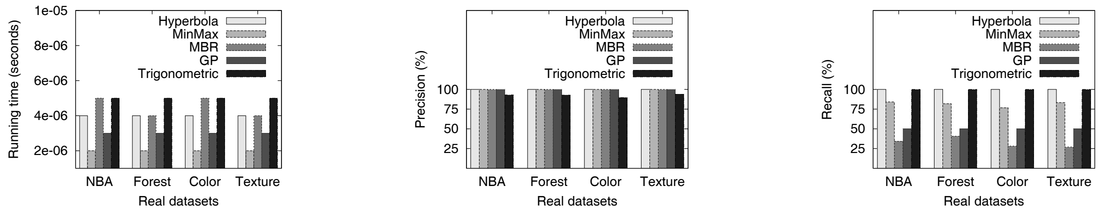
Figure 12: Execution time for the Dominance problem (Datasets in different distributions)

search strategy. We adopted the *Hypherboloa*, *MinMax*, *MBR* and *GP* for performing the hypersphere dominance operator involved in the algorithm and denote the corresponding algorithm by *DF(Hyper)*, *DF(MinMax)*, *DF(MBR)* and *DF(GP)* for *DF*, and *HS(Hyper)*, *HS(MinMax)*, *HS(MBR)* and *HS(GP)* for *HS*. Note that we did not adopt *Trigonometric* for our experiments since it is not correct which implies some hyperspheres that are among the kNN could be missed by the algorithms based on *Trigonometric*.

We used two measures, namely *query time* and *precision*. *Query time* means the time of answering a kNN query and *precision* is defined to be the total number of hyperspheres correctly returned by the algorithm divided by the total number of hyperspheres returned by the algorithm. Note that we did not use the *recall* measure here since for the kNN problem, we find *all* hypersheres that correspond to the kNN of a query hypershphere which means that no "false negatives" are allowed (i.e., all algorithms have the *recall* equal to 100%).

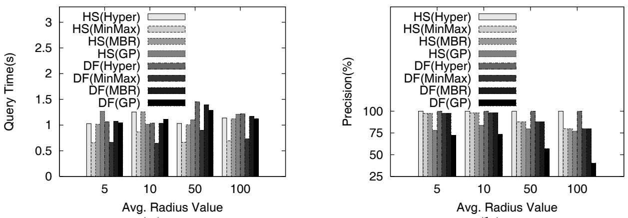
Effect of Ave. Radius μ. The results on synthetic datasets are shown in Figure 13(a) (query time) and Figure 13(b) (precision). We have the following observations. First, the algorithms based on *MinMax* have the smallest query time and those based on other methods run comparably fast. Second, the algorithms based on *Hyperbola* have the precision consistently equal to 100% and the algorithms based on other pruning methods have the precision always smaller than 100% (e.g., as low as 40%).

Effect of Parameter k. The results on the synthetic datasets are shown in Figure 14(a) (query time) and Figure 14(b) (precision).



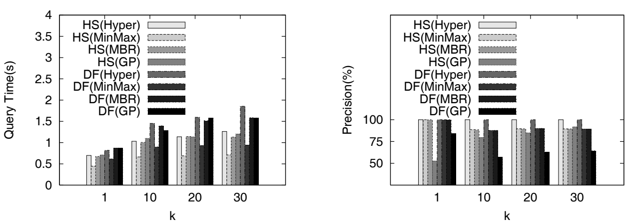
(a) Execution time (b) Precision (c) Recall

Figure 10: Experimental results on real datasets for the Dominance problem



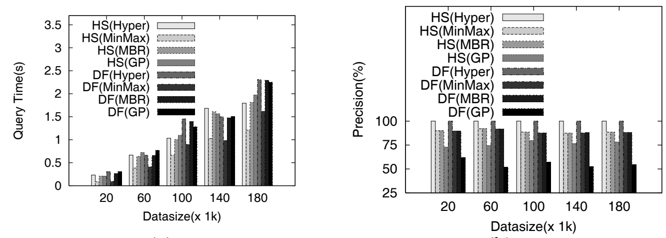
(a) (b)

Figure 13: Effect of Ave. Radius μ for kNN Queries (Synthetic)



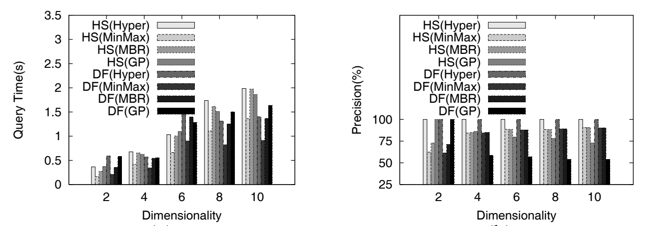
(a) (b)

Figure 14: Effect of k for kNN queries (Synthetic)



(a) (b)

Figure 15: Effect of Data Size N for kNN Queries (Synthetic)



(a) (b)

Figure 16: Effect of Dimensionality d for kNN Queries (Synthetic)

We have the following observations. First, the query times of all algorithms increase when k increases. This is reasonable since a larger k means that a longer best-known list L has to be maintained by the algorithm which costs more time. Second, the setting of k has no clear effect on the precision of the algorithms.

Effect of Data Size (N). The results are shown in Figure 15(a) (query time) and Figure 15(b) (precision). We notice that the query times of all algorithms increase when N increases and the precision of all algorithms is not affected by N significantly.

Effect of Dimensionality (d). The results are shown in Figure 16(a) (query time) and Figure 16(b) (precision). We notice that the query times of all algorithms increase when d increases and the precision of all algorithms is not affected by d significantly.

8. CONCLUSION

In this paper, we studied an important problem called the dominance problem, for which we proposed a new method called *Hyperbola* which is optimal for the dominance problem. We also studied a kNN query which relies on the dominance operator as one of its components. Finally, we conducted experiments which verified our methods. One interesting future research direction is to study how to solve the dominance problem efficiently when the radii of the hyperspheres change over time and/or when some distance metrics other than Euclidean distance are adopted.

Acknowledgements: We are grateful to the anonymous reviewers for their constructive comments on this paper. The research is supported by grant FSGRF14EG34.

9. REFERENCES

[1] T. Bernecker, T. Emrich, H.-P. Kriegel, M. Renz, S. Zanki, and A. Zufle. Efficient probabilistic reverse nearest neighbor query processing on uncertain data. In *VLDB*, 2011.

[2] G. Beskales, M. A. Soliman, and I. F. IIyas. Efficient search for the top-k probable nearest neighbors in uncertain database. *PVLDB*, 2008.

[3] G. Beskales, M. A. Soliman, and I. F. Ilyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. In *VLDB*, 2008.

[4] G. Bliss. Lectures on the calculus of variations. *Chicago Univ. Press*, 1947.

[5] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei. Probabilistic reverse nearest neighbor queries on uncertain data. In *TKDE*, 2010.

[6] J. Chen and R. Cheng. Efficient evaluation of imprecise location-dependent queries. In *ICDE*, 2007.

[7] R. Cheng, L. Chen, J. Chen, and X. Xie. Evaluating probability threshold k-nearest neighbor queries over uncertain data. In *EDBT*, 2009.

[8] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *TKDE*, 2004.

[9] P. Ciaccia, M. Patella, and P. Zezula. M-tree an efficient access method for similarity search in metric spaces. In *VLDB*, 1997.

[10] T. cker Chiueh. Content-based image indexing. In *VLDB*, 1994.

[11] E. Dellis and B. Seeger. Efficient computation of reverse skyline queries. In *VLDB*, 2007.

[12] T. Emrich, F. Graf, H.-P. Kriegel, M. Schubert, and M. Thoma. Optimizing all-nearest-neighbor queries with trigonometric pruning. In *Scientific and Statistical Database Management*, 2010.

[13] T. Emrich, H.-P. Kriegel, P. Kröger, M. Renz, and A. Züfle. Constrained reverse nearest neighbor search on mobile objects. In *SIGSPATIAL*, 2009.
[14] T. Emrich, H.-P. Kriegel, P. Kröger, M. Renz, and A. Züfle. Boosting spatial pruning: on optimal pruning of mbrs. In *SIGMOD*, 2010.
[15] G. Hjaltason and H. Samet. Distance browsing in spatial databases. *TODS*, 1999.
[16] H. Hu and D. L. Lee. Range nearest neighbor query. In *TKDE*, 2006.
[17] Irving and R. S. *Integers, polynomials, and rings*. Springer-Verlag, 2004.
[18] N. Katayama and S. Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *SIGMOD*, 1997.
[19] H.-P. Kriegel, P. Kunath, and M. Renz. Probabilistic nearest-neighbor query on uncertain objects. In *DASFAA*, 2007.
[20] R. Kurniawati, J. Jin, and J. Shepherd. The SS+-tree: An improved index structure for similarity searches in a high-dimensional feature space. In *Proc. 5th Storage and Retrieval for Image and Video Databases SPIE*, 1997.
[21] C. Li. Enabling data retrieval: By ranking and beyond. In *Ph.D. Dissertation, University of Illinois at Urbana-Champaign*, 2007.
[22] X. Lian and L. Chen. Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data. In *VLDBJ*, 2009.
[23] X. Lian and L. Chen. Probabilistic inverse ranking queries over uncertain data. In *DASFAA*, 2009.
[24] X. Lian and L. Chen. Top-k dominating queries in uncertain databases. In *EDBT*, 2009.
[25] V. Ljosa and M. K. Singh. Apla: Indexing arbitrary probability distributions. In *ICDE*, 2007.
[26] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Acm Sigmod Record*, volume 24, pages 71–79, 1995.
[27] M. Sharifzadeh and C. Shahabi. The spatial skyline queries. In *VLDB*, 2006.
[28] Z. Song and N. Roussopoulos. K-nearest neighbor search for moving query point. In *SSTD*, 2001.
[29] Y. Tao, D. Papadias, and X. Lian. Reverse knn search in arbitrary dimensionality. In *VLDB*, 2004.
[30] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *VLDB*, 2002.
[31] D. White and R. Jain. Similarity indexing with the SS-tree. In *icde*, page 516. Published by the IEEE Computer Society, 1996.
[32] X. Xie, R. Cheng, M. L. Yiu, L. Sun, and J. Chen. Uv-diagram: A voronoi diagram for uncertain spatial databases. In *VLDBJ*, 2012.
[33] M. L. Yiu and N. Mamoulis. Efficient processing of top-k dominating queries on multi-dimensional data. In *VLDB*, 2007.
[34] P. Zhang, R. Cheng, N. Mamoulis, M. Renz, A. Zufile, Y. Tang, and T. Emrich. Voronoi-based nearest neighbor search for multi-dimensional uncertain databases. In *ICDE*, 2013.

APPENDIX

A. NON-OPTIMALITY OF EXISTING DECISION CRITERIA

In this part, we show that neither the *GP decision criterion*, denoted by $DC_{GP}(S_a, S_b, S_q)$, nor the *Trigonometric decision criterion*, denoted by $DC_{Tri}(S_a, S_b, S_q)$, are optimal.

GP decision criterion: The GP decision criterion [22] is correct and efficient but not sound. We briefly describe the major idea of this method/decision criterion. Firstly, it transforms three d-dimensional points c_a, c_b and c_q into three 2-dimensional points c'_a, c'_b and c'_q with the following transformation. Given a d-dimensional point x, its *transformed* 2-dimensional point u is defined to be a point where $(u[1])^2 = \sum_{i=1}^{d-1}(x[i])^2$ and $(u[2])^2 = (x[d])^2$. Note that it is shown in [22] that for any two d-dimensional points x and y with its two transformed 2-dimensional points x' and y', $dist(x', y') \leq dist(x, y)$, which means that $dist(x', y')$ can be regarded as a lower bound on $dist(x, y)$. Based on this property, this method guarantees that if $DC_{GP}(S_a, S_b, S_q)$ is true, then $Dom(S_a, S_b, S_q)$ is true. Thus, it is correct. However, since the pairwise distance $dist(x', y')$ between two transformed 2-dimensional points is not exactly equal to the pairwise distance $dist(x, y)$ between the two corresponding d-dimensional points, it is possible that $DC_{GP}(S_a, S_b, S_q)$ is false but $Dom(S_a, S_b, S_q)$ is true. In other words, it is not sound. Detailed description can be found in [22]. Besides, the time complexity of determining $DC_{GP}(S_a, S_b, S_q)$ is $O(d)$ [22], which means that it is efficient.

Trigonometric decision criterion: The Trigonometric decision criterion [12] is sound and efficient but not correct, which is shown in the lemmas in the following.

Before we show the lemmas, we describe this method. There are two phases. The first phase is the preprocessing phase. In this phase, this method finds the optimal value of the *variant* of the function for the MMD condition (i.e., the function at the left hand side of Inequality (7)). Specifically, Inequality (7) can be re-written as "$Min_{q \in S_q}(Dist(c_b, q) - Dist(c_a, q) - (r_a + r_b)) > 0$". It defines a function $f_{S_a,S_b}(q) = Dist(c_b, q) - Dist(c_a, q) - (r_a + r_b)$ where $S_a(S_b)$ contains the information about c_a and r_a (c_b and r_b), and q is a variable denoting a d-dimensional point. Since it is difficult to find a nice formula of the derivative of function $f_{S_a,S_b}(q)$, the method defines another function $g_{S_a,S_b}(q) = [Dist(c_b, q)]^2 - [Dist(c_a, q)]^2 - (r_a + r_b)$ whose derivative can be obtained easily. This function g can be regarded as a variant of the function for the MMD condition. Note that g is a quadratic function and thus has two possible optimal values. Based on this function g, it finds the derivative of this function. It derives a nice formula F_{S_a,S_b} which computes the two possible solutions of the optimal value of function g based on its derivative in $O(d)$ time. The second phase is the query phase. In this phase, the method takes S_a and S_b as inputs and plugs them into the formula F_{S_a,S_b}, finally computing two possible solutions q_1 and q_2. Then, it determines whether one of the following conditions is satisfied: (1) $f_{S_a,S_b}(q_1)$ and $f_{S_a,S_b}(q_2)$ have different signs, and (2) $f_{S_a,S_b}(q_1) = 0$ or $f_{S_a,S_b}(q_2) = 0$. If one of the conditions is satisfied, then this method returns false. Otherwise, it returns true.

LEMMA 11 (NON-CORRECTNESS OF $DC_{Tri}(S_a, S_b, S_q)$). *$DC_{Tri}(S_a, S_b, S_q)$ is true does not imply that $Dom(S_a, S_b, S_q)$ is true.* □

Proof Sketch: The major idea of the claim in the lemma is that optimizing g is not equivalent to optimizing f. Thus, it is easy to give a counter example showing that $DC_{Tri}(S_a, S_b, S_q)$ is true but $Dom(S_a, S_b, S_q)$ is false. A counter example in a 2-dimensional space can be $c_a = (20, 8)$, $c_b = (8, 10)$, $c_q = (16, 16)$, $r_a = 0.4$, $r_b = 0.3$ and $r_q = 0.3$. □

LEMMA 12 (SOUNDNESS OF $DC_{Tri}(S_a, S_b, S_q)$). *If $DC_{Tri}(S_a, S_b, S_q)$ is false, then $Dom(S_a, S_b, S_q)$ is false.* □

PROOF. Let q_1 and q_2 are the two solutions of the optimal value of function g used in the Trigonometric decision criterion. Since $DC_{GP}(S_a, S_b, S_q)$ is false, we know that one of the following conditions is satisfied: (1) $f_{S_a,S_b}(q_1)$ and $f_{S_a,S_b}(q_2)$ have different signs, and (2) $f_{S_a,S_b}(q_1) = 0$ or $f_{S_a,S_b}(q_2) = 0$. Since we know that function f is continuous, there exists a point q_o such that $f_{S_a,S_b}(q_o) = 0$. Thus, Inequality (7) is not satisfied. $Dom(S_a, S_b, S_q)$ is false. □

As we described above, the time complexity of determining $DC_{Tri}(S_a, S_b, S_q)$ (in the query phase) is $O(d)$, which means that this method is efficient.