COMPLETE AND SUFFICIENT SPATIAL DOMINATION OF MULTIDIMENSIONAL RECTANGLES

Tobias Emrich
Harman International
Tobias.Emrich@harman.com

Hans-Peter Kriegel
Ludwig Maximilian University of Munich
kriegel@dbs.ifi.lmu.de

Andreas Züfle
George Mason University
azufle@gmu.edu

Peer Kröger
Ludwig Maximilian University of Munich
kroeger@dbs.ifi.lmu.de

Matthias Renz
Christian-Albrechts-Universität zu Kiel
mr@informatik.uni-kiel.de

ABSTRACT

Rectangles are used to approximate objects, or sets of objects, in a plethora of applications, systems and index structures. Many tasks, such as nearest neighbor search and similarity ranking, require to decide if objects in one rectangle A may/must/must not be closer to objects in a second rectangle B, than objects in a third rectangle R. To decide this relation of "Spatial Domination" it can be shown that using minimum and maximum distances it is often impossible to detect spatial domination. This spatial gem provides a necessary and sufficient decision criterion for spatial domination that can be computed efficiently even in higher dimensional space. In addition, this spatial gem provides an example, pseudocode and an implementation in Python.

1 Introduction

Minimal bounding rectangles (MBRs) are used as object approximations in a plethora of different applications. For example, MBRs are used to bound spatially extended objects and MBRs are used as spatial key for spatial access methods such as the R-Tree [1, 2]. MBR approximations have also become very popular for uncertain databases [3, 4, 5, 6, 7] to approximate all possible locations of an uncertain object.

Rectangular approximations are commonly integrated into spatial query processing as a filter-step, to identify true hits and true drops efficiently based on rectangular approximations only. Most types of spatial/similarity queries, including k-nearest neighbor (kNN). queries, reverse k-nearest neighbor (RkNN) queries, and ranking queries, commonly require the following information. Given three rectangles A, B, and R in a multi-dimensional space $\mathbb{R}^d$, the task is to determine whether object A is definitely closer to R than B w.r.t. a distance function defined on the objects in $\mathbb{R}^d$. If this is the case, we say A *dominates* B w.r.t. R. An example of such a situation is depicted in Figure 1. This concept of domination is a central problem to identify true hits and true drops (pruning). For example, in case of a 1NN query around R, we can prune B if it is dominated by A w.r.t. R and for an R1NN query around R, we can prune B if A dominates R w.r.t. B.

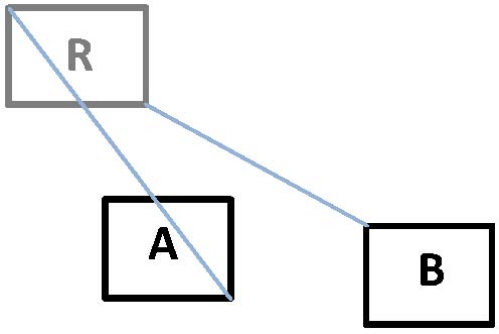

Figure 1: Spatial pruning on MBRs.

The domination problem is trivial for point objects. However, applied to rectangles, the domination problem is much more difficult to solve. Traditionally, the minimal distance and maximal distance between rectangles (min/max-dist) are used to decide which object is closer to another object: If the maximum distance between R and A is lower than

Figure 2: MBR pruning example: Incompleteness of Min/Max

the minimum distance between R and B, then, for any points $r \in R$, $a \in A$ and $b \in B$, it must hold that a is closer to r than b. While this implication is correct, the backward direction does not hold. Thus, min/max-dist provides a sufficient but not a complete decision criterion. To illustrate this problem, consider the example shown in Figure 2. In this example, we can guarantee that point a spatially dominates point b with respect to R. That is, any point in R must be closer to point a than to point b, as any point in R is located on the same side of the equi-distance line H_{ab} (or Voronoi-line) between a and b. Yet, in this example, we still have $MaxDist(a, R) > MinDist(b, R)$, thus the min/max-dist approach does not allow to identify this spatial domination. The error of min/max-dist is incurred by using two different locations of R for the computation of mindist and maxdist.

Another existing method for detecting spatial domination on rectangle has been proposed in [8] which exploits convexness of rectangles to check only the combination of corners of all three rectangles A, B, and R. The drawback of this approach is the high run-time, which scale exponentially in the dimensionality of the data space.

This spatial gem revisits a necessary and sufficient decision criterion for spatial domination of rectangles [9] that allows to efficiently scale to high dimensionality. We summarize the theory, describe application, illustrate examples and provides an implementation in Python.

2 Necessary and Sufficient Domination of Rectangles

The problem of spatial domination is formally defined as follows.

Definition 1 (Domination)**.** *Let $A, B, R \subseteq \mathbb{R}^d$ be rectangles and dist be an L_P norm to measure distance between points in $\mathbb{R}^d$ such as Euclidean distance ($P = 2$) and Manhattan distance ($P = 1$). The rectangle A dominates B w.r.t. R iff for all points $r \in R$ it holds that every point $a \in A$ is closer to r than any point $b \in B$, i.e.*

$$Dom_R(A, B) \Leftrightarrow \forall r \in R, \forall a \in A, \forall b \in B : dist(a, r) < dist(b, r) \tag{1}$$

Equation 1 enumerates an (uncountably) infinite number of triples of possible point locations and thus, cannot be computed directly in this form. An decision criterion that is complete, sufficient, and can be computed in $O(d)$ time has been proposed in [9] as follows:

Theorem 1 (Complete and Sufficient Domination)**.** *Let $A, B, R \subseteq \mathbb{R}^d$ be rectangles and dist be an L_P norm to measure distance between points in $\mathbb{R}^d$. Further, let MinDist(X, x) and MaxDist(X, x) denote the minimum and maximum distance between a (one-dimenstional) interval $X = [X^{min}, X^{max}]$ and a scalar x, respectively:*

$$MinDist(X, x) = \left\{ \begin{array}{ll} 0, & \text{for } x \in X \\ X^{min} - x, & \text{for } x < X^{min} \\ x - X^{max}, & \text{for } x > X^{max} \end{array} \right\} \qquad MaxDist(X, x) = \max(|x - X^{min}|, |x - X^{max}|)$$

Then, the following equivalence holds:

$$Dom_R(A, B) \Leftrightarrow \sum_{i=1}^{d} \max_{r \in \{R_i^{min}, R_i^{max}\}} (\textit{MaxDist}(A_i, r)^P - \textit{MinDist}(B_i, r)^P) < 0 \tag{2}$$

Proof. A detailed inference of Theorem 1 can be found in [9]. □

Example 1. *Let us revisit the example in Figure 2 using Euclidean distance ($P = 2$), and assume the following coordinates of points and rectangles in this example: $a = (0, 2)$, $b = (0, 0)$, $R_{min} = (2, 2)$, and $R_{max} = (10, 4)$. Using Equation 2 we obtain, for the first dimensions ($i = 1$):*

$$\max_{r_1 \in \{R_1^{min}, R_1^{max}\}} (\textit{MaxDist}(A_1, r_1)^2 - \textit{MinDist}(B_1, r_1)^2) =$$

$$\max_{r_i \in \{2,10\}} (\textit{MaxDist}(A_1, r_1)^2 - \textit{MinDist}(B_1, r_1)^2) =$$

$$\max(\textit{MaxDist}([0,0], 2)^2 - \textit{MinDist}([0,0], 2)^2)), \textit{MaxDist}([0,0], 10)^2 - \textit{MinDist}([0,0], 10)^2) =$$

$$\max(4 - 4, 100 - 100) = max(0, 0) = 0$$

and for the second dimension we get:

$$\max_{r_2 \in \{R_2^{min}, R_2^{max}\}} (\textit{MaxDist}(A_2, r_2)^2 - \textit{MinDist}(B_2, r_2)^2) =$$

$$\max_{r_i \in \{2,4\}} (\textit{MaxDist}(A_2, r_2)^2 - \textit{MinDist}(B_2, r_2)^2) =$$

$$\max(\textit{MaxDist}([2,2], 2)^2 - \textit{MinDist}([0,0], 2)^2)), \textit{MaxDist}([2,2], 10)^2 - \textit{MinDist}([0,0], 10)^2) =$$

$$\max(0 - 4, 64 - 100) = max(-4, -36) = -4$$

The sum over the two dimensions yields $0 - 4 = -4$, and since $-4 < 0$, the inequation of Equation 2 is satisfied, and thus, $Dom_R(a, b)$ holds.

3 Implementation

This section provides an implementation for the complete and sufficient spatial domination decision criterion sketched in Section 2 and described in detail in [9]. Pseudocode can be found in Algorithm 1, which takes three d-dimensional rectangle and an L_P norm as input, and decides if A spatially dominates B with respect to R. The algorithm iterates of all dimensions in Line 2. For each dimension, the algorithm uses the minimum and maximum point of rectangle R, and compares minDist and maxDist of these points to rectangles A and B in lines 4-5. Algorithms to compute maxDist and minDist are shown in Algorithm 2 and Algorithm 3, respectively. The larger of the two is aggregated into the final result in lines 6-10.

For an implementation of Algorithm 1 in Python, see `https://github.com/azufle/Spatial_Gems_Spatial_Domination`.

Algorithm 1: Complete and Sufficient Domination of MBRs

input : P // Employed L_P norm to measure distance
input : d // Number of dimensions
input : A, B, R // Three Rectangles in R^d.
output : A boolean predicate to decide if A spatially dominates B w.r.t. R

1 $sum \leftarrow 0$ // Initialize the sum
2 **for** $i \in 1 : d$ // Iterate over each dimension
3 **do**
4 $\quad max1 \leftarrow MaxDist(A_i, R_i^{min})^P - MinDist(B_i, R_i^{min})^P$ // First argument of the maximum
5 $\quad max2 \leftarrow MaxDist(A_i, R_i^{max})^P - MinDist(B_i, R_i^{max})^P$ // Second argument of the maximum
6 $\quad$ **if** $max1 \geq max2$ // Add the larger argument to the sum
7 $\quad$ **then**
8 $\quad\quad sum \leftarrow sum + max1$
9 $\quad$ **else**
10 $\quad\quad sum \leftarrow sum + max2$
11 **end**
12 **return** $sum < 0$ // Return True if $sum < 0$ and False otherwise

Algorithm 2: Maximum Distance between an Interval and a Scalar

input : $X = [X^{min}, X^{max}]$ // An interval from X^{min} to X^{max}
input : x // A scalar in $\mathbb{R}$
output : The maximum distance between X and x (for any point in X)

```
if |x − X^min| ≥ |x − X^max| then
    return |x − X^min|
else
    return |x − X^max|
```

Algorithm 3: Minimum Distance between an Interval and a Scalar

input : $X = [X^{min}, X^{max}]$ // An interval from X^{min} to X^{max}
input : x // A scalar in $\mathbb{R}$
output : The minimum distance between X and x (for any point in X)

```
if x < X^min then
    return X^min − x
else if x ≤ X^max then
    return 0                                  // Case where x ∈ X
else
    return x − X^max                          // Case where x > X^max
```

4 Scenarios and Applications

In this section, we will show how the concepts of spatial domination can be used to accelerate candidate pruning used in similarity search and recommendation systems.

4.1 Reverse Nearest Neighbor Search

The reverse k-nearest neighbor query problem is given a point q, retrieve all the data points that have q as one of their k nearest neighbors. The geometrical pruning based solution of this problem introduced in [10] overcomes the problem of other RkNN approaches including 1) support arbitrary values of k, 2) can efficiently deal with database updates, and 3) are applicable to arbitrary-dimensional feature spaces. The basic operation of determining if an object or a page region of a spatial index, e.g. an R-tree, can be pruned based on a bisecting hyperplane defined by two multi-dimensional points can be efficiently solved with our spatial domination solution.

4.2 Multi-Preference Recommendation

Multi-preference recommendation and multi-criteria decision making based on top-k query processing has become a hot topic and has been studied extensively in the last two decades. In the context of multi criteria top-k query processing, computational geometry driven top-k query processing models have gained a lot of interest in recent years [11, 12, 13, 14, 15] and could be successfully adapted to several variants of top-k queries in multi-criteria settings [12]. The principal idea of the above mentioned line of work is to translate object domination according to multi-criteria preferences into hyperplane bisection of the multidimensional preference space. A key operation shared by all these approaches is, given a halfspace S defined by hyper-plane bi-section of the multidimensional preference space, and an axis parallel rectangle R, the determination of which of the following cases is true: case 1) R is completely covered by S, case 2) R intersects S, or case 3) R does not intersect S at all, as illustrated in Figure 3.

References

[1] A. Guttman. R-Trees: A dynamic index structure for spatial searching. In *ACM SIGMOD*, pages 47–57, 1984.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An efficient and robust access method for points and rectangles. In *ACM SIGMOD*, 1990.

[3] George Beskales, Mohamed A. Soliman, and Ihab F. IIyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. *Proc. VLDB Endow.*, 1(1):326–339, 2008.

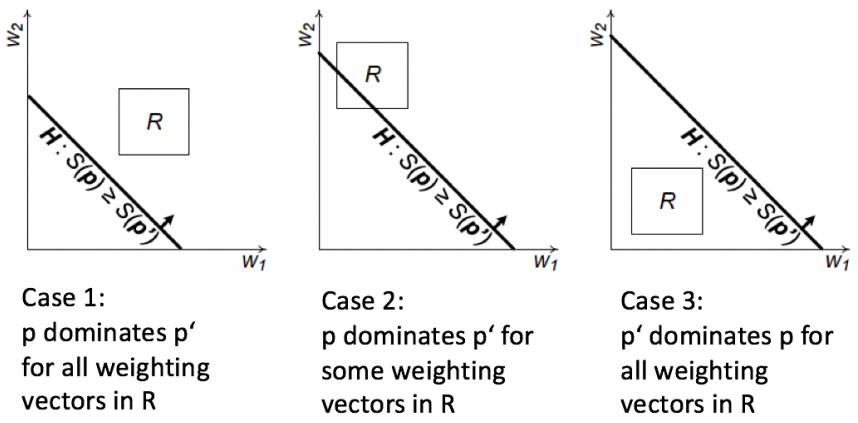

Figure 3: R-domination identification based on hyperplane bisection [12]

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

[5] R. Cheng, J. Chen, M. Mokbel, and C. Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *ICDE*, 2008.

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

[7] X. Lian and L. Chen. Probabilistic inverse ranking queries over uncertain data. In *DASFAA*, 2009.

[8] Tobias Emrich, Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Andreas Züfle. Incremental reverse nearest neighbor ranking in vector spaces. In *International Symposium on Spatial and Temporal Databases*, pages 265–282. Springer, 2009.

[9] Tobias Emrich, Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Andreas Züfle. Boosting spatial pruning: on optimal pruning of mbrs. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 39–50. ACM, 2010.

[10] Yufei Tao, Dimitris Papadias, and Xiang Lian. Reverse knn search in arbitrary dimensionality. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 744–755. VLDB Endowment, 2004.

[11] Kyriakos Mouratidis. Geometric top-k processing: Updates since mdm'16 [advanced seminar]. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pages 1–3. IEEE, 2019.

[12] Kyriakos Mouratidis and Bo Tang. Exact processing of uncertain top-k queries in multi-criteria settings. *Proceedings of the VLDB Endowment*, 11(8):866–879, 2018.

[13] Bo Tang, Kyriakos Mouratidis, and Man Lung Yiu. Determining the impact regions of competing options in preference space. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 805–820. ACM, 2017.

[14] Guolei Yang and Ying Cai. Querying a collection of continuous functions. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1783–1795, 2018.

[15] Li Qian, Jinyang Gao, and HV Jagadish. Learning user preferences by adaptive pairwise comparison. *Proceedings of the VLDB Endowment*, 8(11):1322–1333, 2015.