Open Access

Open Journal of Databases (OJDB)
Volume 7, Issue 1, 2020

www.ronpub.com/ojdb
ISSN 2199-3459

Branch-and-Bound Ranked Search by Minimizing Parabolic Polynomials

Peter Poensgen, Ralf Möller

Institute for Information Systems, University of Lübeck, Ratzeburger Allee 160, 23562 Lübeck, Germany,
{poensgen, moeller}@ifis.uni-luebeck.de

Abstract

The Branch-and-Bound Ranked Search algorithm (BRS) is an efficient method for answering top-k queries based on R-trees using multivariate scoring functions. To make BRS effective with ascending rankings, the algorithm must be able to identify lower bounds of the scoring functions for exploring search partitions. This paper presents BRS supporting parabolic polynomials. These functions are common to minimize combined scores over different attributes and cover a variety of applications. To the best of our knowledge the problem to develop an algorithm for computing lower bounds for the BRS method has not been well addressed yet.

Type of Paper and Keywords

Short Communication: *top-k queries, query answering, top-k ranking, parabolic polynomials, scoring functions, Branch-and-Bound Ranked Search, minimal bounding rectangle*

1 Introduction

Supporting efficient top-k processing in database systems is a relatively recent and active line of research. The answer to a top-k query in a relational database is a ranked set of the k tuples that fit "best" to the selection condition. Efficient processing is usually the most important requirement for applications, in particular if huge amounts of data are involved. A common way for computing the relevant top-k objects is the application of a scoring (ranking) function. In relational database systems ranking functions will be used in the Order-By statement of a top-k query. In general, they are multivariate functions that assign an aggregation over partial attributes (scores). The answer to a top-k query is a ranked set of the k tuples in the database that match "best" the selection condition.

An example of such databases is a patient database used in clinical studies, and the databases provides the blood pressure values systolic (*sys*) and diastolic (*dia*) as well as the pulse rate (*pulse*) for each patient. A typical application scenario, which deals with parabolic polynomials when computing top-k queries over the patient database, is described in Example 1. We use the application scenario as the running example in the paper to explain our work.

Example 1: Searching for suitable probands

In order to test a drug for the treatment of slowed heart rate (bradycardia) and very fast heart rate (tachycardia), a sample of patients needs to be determined. Since side effects on (normal) blood pressure should be investigated, the systolic and diastolic values must be included. The patient data is stored in a patient table *R* in a relational database. To identify a sample of suitable candidates, the doctor looks in the patient database for people whose systolic value (*sys*) is around 120, the diabolic value (*dia*) is around 80 and whose pulse sequence (*pulse*) deviates significantly from the mark 50 either upwards or downwards. The top-k query for patients in the 40- to 50-year-old age group could be written as follows:

```
SELECT Top k *
FROM R
WHERE age BEETWEEN 40 AND 50
ORDER BY $\omega_1(sys-120)^2+\omega_2(dia-80)^2+$
             $\omega_3(puls-50)^2$ ASC
```

The point $q=(120,80,50)$ is the so-called query-point and $\omega=(\omega_1,\omega_2,\omega_3)$ with $\omega_i\neq 0$ is a vector whose components weight the three attributes in order to show their relevance and the corresponding sign determine which attribute is to be particularly close (positive weighting) to, or as far away as possible (negative weighting) from the corresponding coordinate of the query point. For our example $\omega_1,\omega_2>0$ and $\omega_3<0$ must be chosen so that the problem can be solved by minimizing the scoring function of the ORDER-BY statement. If the values of one or more attributes are very small in relation to the others, the corresponding exponent can be increased instead of a scalar weighting. However, for the method presented in this work, the exponents may only be natural numbers.

In this paper the semantic of a top-k selection query is given by

```
SELECT TOP k attribute-list
FROM R
[WHERE…]
ORDER BY $f(t)$ ASC
```

where f is a function of several variables that assigns a numeric score to any tuple t (see [2]). Note that a scoring function does not need to consider all attributes of the table.

1.1 Motivation and Challenges

A naive way for answering top-k queries is to consider a complete dataset of N tuples of a given relation, compute the value of a scoring function for each tuple, while maintaining and finally returning the k lowest-ranked tuples. This algorithm is in $O(N\,log(k))$ (see [5], and since k is small and fixed, the procedure is called sequential search.

Our problem is not monotone: The underlying ranking function in Example 1 described above measures the distance by squaring the difference of two expressions. Such requests can also be modeled differently to come to the same solution. An alternative way is to use the absolute value function. However, note the fact that this requirement cannot be modeled with the help of a monotone scoring function.

In general, minimum problems cannot be converted into maximum ones: The task of finding top-k tuples from a given relation can either be defined by a maximization or a minimization Order-By criterion. Basically, a maximum query can always be turned into a minimum one by switching the sign of the objective function; i.e., both queries will answer the same question. If the scoring function f is monotone increasing the negative $-f$ is obviously monotone decreasing. Therefore, in the context of monotonicity most proposed techniques assume just one case, mostly the maximization problem, e.g. [14] and [18]. In other words, in the monotonic case the calculation of top-k data points can always be reduced to the maximization of the underlying ranking function. However, this method is generally not applicable to non-monotonic functions. Parabolic polynomials of Example 1 are convex functions and the complement of convexity is concavity and thus solutions which assume convexity e.g. [13] are not applicable.

Example 1 is generally not a NNS-problem: As mentioned in [15] a nearest neighbor search (NNS) is a form of proximity search. Given a data point q and an integer *k*, the output of a so-called k-NN query contains the k objects closest to q, where proximity is computed by means of a distance function. Nearest neighbour queries can be considered as a special case of top-k queries, where the ranking function corresponds to the distance among the objects. Example 1 with positive weightings is a k-NN query, whereas its generalization given by negative weightings is not. The last one is a top-k selection query with a special family of non-monotonic ranking functions.

1.2 Contributions

Our approach developed in this paper addresses the problem of answering non-monotonic top-k queries by minimizing parabolic polynomials. We introduced the minimum principle for this class of functions that guarantees lower bounds for the BRS algorithm. Our contributions towards this goal are summarized here:

- We introduce the class of parabolic polynomials and show why these polynomials are suitable candidates for the Branch-and-Bound ranked search algorithm.

- We develop the MinScore algorithm for calculating the minimum (lower bounds) for parabolic polynomials on minimal bounding rectangles (MBRs), which makes the BRS available to a wide range of practically relevant (non-monotonic) applications.

- We analyze the Branch-and-Bound Ranked Search method (BRS) and show that the algorithm takes $O(log(N)log(k))$ if the number of rectangles at the lowest inner level of the R-tree is chosen as a function of the number of data points N.

The remainder of this paper is organized as follows: Section 2 discusses related work and Section 3 introduces related background of BRS. Section 4 presents our research work: the Minimum Principle of Parabolic Polynomials and the Min-Score Algorithm in bounded search for top-k solutions in BRS. We discuss and analyze the research results in Section 5, and conclude our study in Section 6.

2 Related Work

The Branch-and-bound principle is a fundamental and widely-used methodology for calculating exact solutions to optimization problems. This approach, first proposed by Lang and Doig [8], depends on efficient estimation of lower or upper bounds of regions on the search space and uses an R-tree to develop efficient search algorithms. Clausen [1] gives an overview about the search strategy, the branching strategy and some pruning rules of Branch-and-Bound and illustrates the method and design issues through three examples. In [10] Morrison and colleagues present recent research advances in the design of Branch-and-Bound algorithms. In addition, search strategies are described that affect the computing time required for the Branch-and-Bound process.

Branch-and-Bound is also suitable for next neighbor search [7] and [15], for skyline retrieval [11] and for processing of ranking queries introduced by Tao and colleagues (Branch-and-Bound Ranked Search, BRS [18]). This paper also shows how the BRS works for monotone scoring functions, which always assume their maximum at the upper right or lower left edge of hyper rectangles. Monotonic functions are the classical example to which a maximum principle applies. In convex optimization, the maximum principle says that the maximum of a function in its domain is attained on the boundary of that domain (see [17]). This solves the problem of finding suitable bounds for monotonous functions.

The BRS is also suitable for non-monotonous functions, as long as the bounds for these functions can be determined efficiently. Quasi-convex functions, as shown in [13], generalize the monotonic approach. The maximum of quasi-convex functions is to be found on any vertex of a hyper rectangle. For both functional classes, the algorithm for determining the upper bounds needed for the BRS is quite simple. The upper bound can be specified as function value of the edge where the function assumes its maximum.

The proposed family of ranking functions in [14] is closest to our approach. It combines the idea of so-called attractive and repulsive dimensions. Based on a given data point $q = (q_1, ... q_n)$ (query point) the introduced linear *SD-Score* function measures the distance between each relevant data point $p = (p_1, ... p_n)$ and q in each coordinate. The goal is to find the k highest scoring points of such a ranking function. The authors note, that they take a more direct approach and develop precomputation based index structures specifically for the proposed class of linear scoring functions. Instead of maximizing (non-monotone) scoring functions, the method developed in our paper is dealing with the challenge of minimizing them.

Ad-hoc ranking functions are addressed by [20], with the restriction that the function is lower-bounded. A ranking function f is lower-bounded in a region of its variables domains, if the lower bound of f in this region can be derived. The authors present an index-merge framework that performs progressive search over a space of states composed by joining index nodes. The main idea is to exploit existing B-Tree and R-Tree indexes of ranking predicates to create a search space of possible query answers.

To solve the problem, the function must be minimized. However, this and all other methods known to us that minimize the scoring function are not suitable to solve the minimum problem of BRS. Parabolic polynomials are minimal on the edge of a hyper rectangle and therefore a minimum principle holds for these functions. This makes them suitable candidates for the BRS algorithm. How to determine the local minima on rectangles for these functions is the focus of this paper.

3 Preliminaries

In this part, the fundamentals of the Branch-and-Bound framework are presented briefly, which provide necessary information in order to better understand the approach developed in this paper.

The BRS method is essentially based on an R-tree with minimal bounding rectangles (MBRs) for partitioning, and it requires an MBR scoring function for deciding which node is to be examined next. The Branch-and-Bound framework has been applied extensively to develop efficient search algorithms based on R-Trees [18]. An R-tree [6] is a common access method for multi-dimensional objects. Its key idea is to group nearby objects and represent them as a minimum bounding rectangle in the next higher level. MBRs at the same level are recursively clustered into

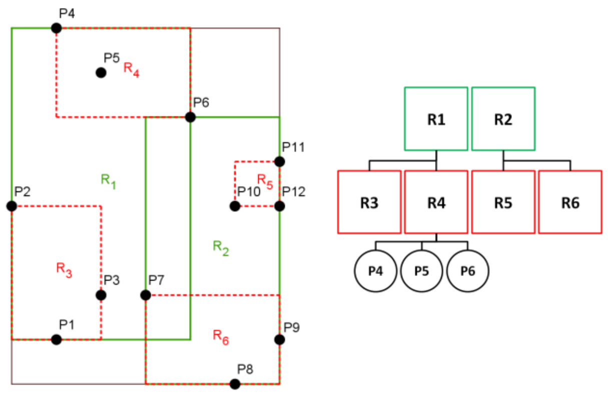

Figure 1: Partitioning of data room (left) with corresponding R-Tree (right)

Algorithm BRS (*rt, k, type, score_p, score_r*)

```
   // rt is an R-tree on the dataset
   // k denotes the number of data points to return
   // type can be either 'min' or 'max'
   // score_p is a point scoring function
   // score_r computes a score for an MBR
1  let pq = build_priority_queue(
       type,
       map(lambda(obj). (obj, score_r(mbr(obj),score_p)),
       root(rt)))
       // create pq with (obj, score) entries using
       // the anonymous lambda function
2  result = {}
3  n = 0
4  object // can be either a data point or an MBR
5      while n < k and not empty?(pq) do
6          object := delete_next(pq)
7          if point?(object) then
8              result := result ∪ {object}
9              n := n + 1
10         else
11             if leaf?(object) then
12                 for p ∈ points(mbr(object)) do
13                     insert((p, score_p(p)) , pq)
14             else
15                 for e ∈ children(object) do
16                     insert((e, score_r(e, score_p)), pq)
17     result
```

Listing 1: BRS (based on [18])

nodes of the higher level. R-trees for top-k queries on tuples of a given relation have the special property that leaf nodes consist of multiple data points defined by the tuples of the relation (cardinalities depend on partition sizes). An MBR $M \subset \mathbb{R}^d$ is defined as the finite Cartesian product $M = I_1 \times I_2 \times \ldots \times I_d$ of closed intervals $I_j = [a_j, b_j]$ with $a_j \leq b_j$ for any $j = 1, \ldots, d$ [16]. We denote M as vector $v_M = (a_1, b_1, \ldots, a_d, b_d) \in \mathbb{R}^{2d}$.

Figure 1 demonstrates a part of a two-dimensional point dataset and the corresponding R-Tree. The root consists of two MBRs (R1, R2) and the four MBRs R3, R4, R5 and R6 are internal nodes. For simplicity only the data points P4, P5 and P6 assigned to the MBR R4 are shown in the figure.

The strategy to answer a minimum top-k query with BRS is described as a bounded search through an R-tree (see Listing 1). The algorithm uses the R-tree to partition and index the dataset of a given relation, and for bounding the search for top-k result points BRS maintains a priority queue *pq* of R-tree entries or points (see [4]). Initially the algorithm loads the root of the R-tree, i.e., a set of MBRs, into the priority queue *pq* (Line 1). Actually, pairs of objects and scores are inserted into *pq* for determining the ranking of objects. The score of an MBR M is determined by applying score_r to two parameters, namely M and the point scoring function f. Both functions need to be provided to BRS. Afterwards objects from *pq* are considered in a loop.

In each iteration the node with the highest ranking (highest or lowest scored object, depending on type) is retrieved from *pq* (Line 5 and 6). If a point is found (Line 7), it is added to the result. If a leaf node is retrieved from the queue (Line 11), its points are extracted, the score of data points p are determined with the function f, and point scores are used to insert each point into *pq* (Lines 12, 13). Otherwise a non-leaf R-tree node is found (Line 14), and the respective children are inserted into *pq* (Line 15, 16). Each child MBR is scored with score_r in the same way as in the initialization of *pq* with root nodes. The algorithm terminates when k data points are found (collected in result) or the queue is empty (see [13]).

The actual call to BRS is given as follows: BRS(data_rtree, k, 'min', score_p, score_r), where k is the expected number of results, the function *min* indicates minimum top-k, *score_p* is a query-specific function that maps a data object (point) to a score being used for ranking the object, and *score_r* determines the lowest value that the function *score_p* can return when applied to the points associated with an MBR specified as a parameter. We will give an efficient implementation of *score_r* for a parabolic polynomial *score_p* functions below.

4 PARABOLIC POLYNOMIALS

In general for d numeric attributes of a given relation and for any even exponents of the component functions, the scoring function of Example 1 results in a special class of polynomials. Polynomial functions are common in minimizing problems (see [12] and [19]).

Definition 1: Parabolic Polynomials

A real-valued function $f\colon \mathbb{R}^d \to \mathbb{R}$

$$(x_1, \dots, x_d) \mapsto \sum_{i=1}^{d} \omega_i \, (x_i - q_i)^{2n_i}$$

with $n_i \in \mathbb{N}, \omega_i \in \mathbb{R}\backslash 0$ for each $i \in \{1, \dots, d\}$ is a polynomial with even degree. The d-dimensional data point $q = (q_1, \dots, q_d)$ is denoted as "center" or "midpoint" of the graph of the function. $\omega = (\omega_1, \dots, \omega_d)$ is the weighting vector. For a fixed i the one-dimensional function $x_i \mapsto \omega_i(x - q_i)^{2n_i}$ is called the i-th component function of f, which depending on the weighting (positive or negative) is an upwardly or downwardly opened parabola. We define such a function as a *parabolic polynomial*.

4.1 Minimum Principle of Parabolic Polynomials

The Extreme Value Theorem (see [9]) of analysis ensures that each continuous function on a compact set obtains its (finite) minimum. It forms the basis for the minimum principle and is also fundamental for our *MinScore* algorithm.

Theorem 1: Minimum Principle of Parabolic Polynomials

Let $f\colon M \to \mathbb{R}$ be a parabolic polynomial, $M \subset \mathbb{R}^d$ a minimal bounding rectangle and $x_{min} \in M$ the minimum of f on M. Then x_{min} lies on the edge of M or $f_{min}(M) \coloneqq f(x_{min}) = 0$.

Proof: From real analysis it is known that a subset of the Euclidian space $\mathbb{R}^d$ is compact if and only if it is closed and bounded, which is the statement of the well-known Heine-Borel Theorem (see [3]). Since each interval $[a, b] \subset \mathbb{R}$ is closed and bounded and every finite Cartesien product of closed intervals is closed and bounded as well, each minimal bounding rectangle is also compact. Thus, the continuity of parabolic polynomials ensures that f attains its minimum on M.

Next we show that if $f(x) = \sum_{i=1}^{d} f_i(x)$ is the sum of d component functions, then its minimum can be calculated by the sum of the minima of its component functions. In case of parabolic polynomials this means that the following mathematical equation is satisfied:

$$\min_{(x_1,\dots,x_d)\,\in M} \sum_{i=1}^{d} \omega_i \, (x_i - q_i)^{2n_i} = \sum_{i=1}^{d} \min_{x_i \,\in I_i} (\omega_i (x_i - q_i)^{2n_i}) \qquad (1)$$

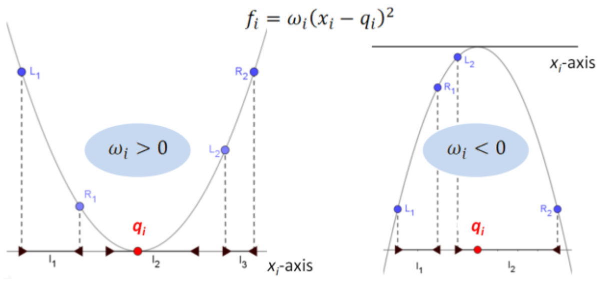

Figure 2: One-dimensional parabolic functions

Proof: Let be $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ a data point where the parabolic polynomial is minimal, then mathematical equation is satisfied:

$$f(x) = \sum_{i=1}^{d} f_i(x_i) \qquad (2)$$

Let be $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_n) \in \mathbb{R}^d$ a point on which the component functions $f_i(\tilde{x}_i)$ is minimal for each $i \in \{1, \dots, d\}$, then by definition and because of (2),

$$f(\tilde{x}) = \sum_{i=1}^{d} f_i(\tilde{x}_i) \le \sum_{i=1}^{d} f_i(x_i) = f(x). \qquad (3)$$

Since (3) holds, $f(x) \ge f(\tilde{x})$ and thus $f(x) = f(\tilde{x})$. Since (1) holds, the calculation of the minimum of f over M can be reduced to the one-dimensional case. In $\mathbb{R}^1$ two types of component functions have to be distinguished. Figure 2 illustrates the procedure. If the weighting of the function f is positive (Figure 2 left side) the minimum value of f is zero if and only if the interval that defines the edge of M contains the coordinate of the global minimum (I_2). Otherwise, if the coordinate of the minimum is outside of the interval (I_1 or I_3) the minimum is the lowest function value of the left and right boundary of the interval. If the weighting of the function is negative (Figure 2 right side) the minimum is always the lowest function value of the left and right boundary of the interval (I_1 or I_2).

After these theoretical and geometrical preparations we can introduce our MinScore algorithm, which is fundamental for the BRS algorithm. The algorithm *min_i* calculates the minima of all component functions, which, as mentioned above, are then summed up in the MinScore algorithm *score_r*.

4.2 The Min-Score Algorithm

Since Theorem 1 holds, the algorithm for determining the minimum of a parabolic polynomial can be reduced to the one-dimensional case. In $\mathbb{R}^1$ two types of component functions have to be distinguished. Figure 1

```
Algorithm min_i $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\omega}, \boldsymbol{q}, \boldsymbol{d})$
    // $a$ lower, $b$ upper interval limit
    // $\omega \in \mathbb{R}\backslash\{0\},\ q \in \mathbb{R}, d \in \mathbb{N}$
1   let $p = lambda(x)\omega(x-q)^{2d}$
        $min\_i$
2     if $\omega < 0$ then
3       $min\_i := min\{p(a), p(b)\}$
4     else // if weighting positive
5       if $q \notin [a, b]$ then
6           $min\_i := min\{p(a), p(b)\}$
7       else
8           $min\_i := 0$
9     $min\_i$
```

Listing 2: Calculating the minimum of one-dimensional parabolic polynomials on [a,b]

```
Algorithm score_r (M, score_p)
    // $M$ is an MBR
1   let $(a_1, b_1, \dots, a_d, b_d) = interval(M)$
    // the interval function generates the interval
    // vector $v_M = (a_1, b_1, \dots, a_d, b_d) \in \mathbb{R}^{2d}$
    // of a $d$-dimensional MBR $M$
2   let $((q_1, \dots, q_d), (\omega_1, \dots, \omega_d)(n_1, \dots, n_d)) =$
                $get\_parameters(score\_p)$
    // $(q_1, \dots, q_d)$ the mid-point of score_p
    // $(\omega_1, \dots, \omega_d)$ the weighting vector
    // $(n_1, \dots, n_d)$ the vector of exponents
3   $\sum_{i=1}^{d} min\_i(a_i, b_i, \omega_i, q_i, n_i)$.
    // $\omega_i, q_i, n_i$ the respective $i$-th component
```

Listing 3: MinScore algorithm

illustrates the procedure. Listing 2 describes the corresponding algorithm for calculating the minimum for one-dimensional polynomials. In this case, the function is defined in the first step by transferring the parameters q, ω and d (Line 1). The distinction of weighting is made in the IF-Statement in Line 2 i.e. the parameter ω decides whether the minimum is located on the edge or inside the minimal bounding rectangles (MBR) and its function value is null.

The minimum $f_{min}(M)$ results from the sum of all the minima determined for the component functions f_i on I_i for each $1 \leq i \leq d$ (see Listing 3).

Example 2. We illustrate an example of the MinScore algorithm with a scoring function of Example 1 on a three-dimensional MBR. The function is given by

$$f: [50{,}100]^3 \to \mathbb{R};$$
$$(x_1, x_2, x_3) \mapsto (x_1 - 120)^2 + (x_2 - 80)^2 - (x_2 - 50)^2$$

We get $\omega = (1, 1, -1), q = (120, 80,\ 50)$ and the interval vector $v_M = (50, 100, 50, 100, 50{,}100) \in \mathbb{R}^6$ The min_i algorithm calculates the minimum at the associated intervals for each component function.

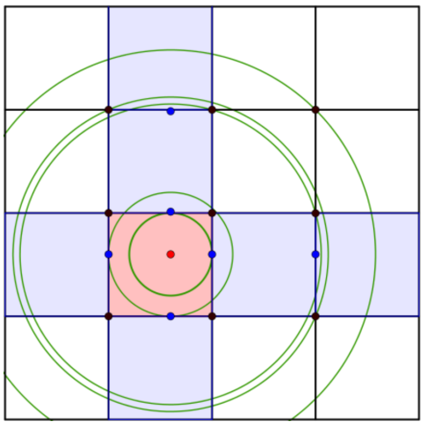

Figure 3: Minimum positions of a parabolic polynomial of order 2 with positive weights

The minimum of $f_1(x) = (x_1 - 120)^2$ on $I_1 = [50, 100]$ is the lowest function value of the left and right boundary of the interval, because the weighting is positive and the first component $q_1 = 120$ of the query point q is not included in I_1. As result we get $Min(f_1(50), f_1(100)) = 400$. The minimum of $f_2(x) = (x_1 - 80)^2$ on $I_2 = [50, 100]$ is 0, because the interval I_2 contains $q_1 = 80$. Since the weighting of f_3 is negative the minimum of f_3 on $I_2 = [50, 100]$ is the lowest function value of the left and right boundary. As result we get $Min(f_3(50), f_3(100)) = -2500$. The total minimum value, i.e. the result of the *MinScore* algorithm is the sum of these three values, i.e. -2100. The minimum of f is on an edge of the MBR $M = I_1 \times I_2 \times I_3$ with the coordinates $(100, 80, 50)$.

Geometrically, three different minimum types of parabolic polynomials can be distinguished. Those that are inside an MBR, those that lie on the $(d-1)$-dimensional edges, but not on a vertex, and lastly, those that lie on a vertex. For parabolic polynomial holds a minimum principle on a minimum bounding rectangles or the minimum $f_{min}(M) = 0$.

Figure 3 shows an example of the minima and their geometric position using the positively weighted function $f(x_1, x_2) = (x_1 - 0{,}4)^2 + (x_2 - 0{,}4)^2$ for dimension 2 in a grid partition of the unit rectangle. The local minimum of f on the red rectangle is on the red data point inside the MBR. The function value is null at this point. On the blue rectangles the function is minimal on the corresponding blue data points, i.e. on the edges. The other black points correspond to the minimum points of the other rectangles and lie on one of the vertices of the same.

To compute the lower bound on a d-dimensional MBR $M = I_1 \times I_2 \times \dots \times I_d$, the *MinScore* algorithm *score_r* requires, without case distinctions, exactly d computational steps by summarizing the minimum of the component function f_i of its interval $I_i = [a_i, b_i]$ for each $i \in [1, \ldots, d]$. This makes the algorithm linear and efficient. Thus, the BRS algorithm using score_r solves the minimization problem of top-k queries with parabolic polynomials as the evaluation function efficiently.

At the end of this part we specify the time complexity of the BRS method. If the priority queue is implemented as a heap, (see [5]) if N is the number of data points, g the granularity of the partitioning, and thus the number of MBRs at the lowest inner level of the R-tree from which c must be visited, then the steps of the BRS algorithm can be described as follows:

- Find the MBR in the priority queue with the lowest lower bound.
- Determine the top-k elements from an MBR with $\frac{N}{g}$ data points. We assume an equal distribution of points in all MBRs.
- Merge the set of already existing (examined) k data points with the k newly determined points.
- Determine the top-k elements from a list of 2k data points.

Therefore, the time complexity for these steps is:

$$T_{BRS} \in \mathcal{O}\left(c\left[T_{delMin}(g) + T_{Topk}\left(\frac{N}{g}\right) + T_{Merge}(2k) + T_{Topk}(2k)\right]\right).$$

Because the pq is a heap, we get

$$T_{BRS} \in \mathcal{O}\left(c\left[log(g) + \frac{log(k)N}{g} + 1 + log(2k)\right]\right) = \mathcal{O}\left(c \cdot \frac{log(k)N}{g}\right).$$

Thus, the BRS differs from the sequential search by the factor $c\ /\ g$ which is much smaller than 1, because c is generally very small in relation to g. If one assumes the appropriate storage capacity and chooses the granularity depending on the set of data points $g \coloneqq N\ /\ log(N)$, we obtain

$$\mathcal{O}(log(N)log(k)).$$

The storage space is well invested, as in comparison to the sequential search we get an improvement from N to $log(N)$.

5 Analysis and Discussion

Parabolic polynomials satisfy a minimum principle on minimal bounding rectangles (MBRs), or the minimum lies inside the MBR and is null. The position of the minima and thus also their function values (bounds)

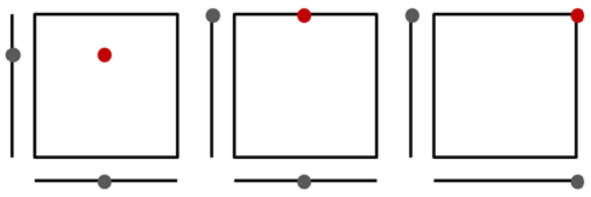

Figure 4: Minimum types of parabolic functions

can be determined by considering the individual component functions of a parabolic polynomial on the respective intervals by case distinction. Thus, the problem of determining a lower bound for all possible function values of data points of a multi-dimensional hyper rectangle is reduced to the one-dimensional case. Figure 4 shows schematically the different cases for the dimension $d = 2$, which can be analogously transferred to each higher dimension.

If all component functions of the parabolic polynomial are minimal inside the respective interval (Figure 4, left), then the associated parabolic polynomial is also minimal inside the MBR and the function value is null. If all component functions are minimal at the end point of the interval (Figure 4, right), the parabolic polynomial takes its minimum on an edge of the MBR. In all other cases (Figure 4, mittle), the parabolic polynomial is minimal on one of the $(d-1)$-dimensional faces of the MBR.

6 Conclusion

In this paper we addressed the problem of answering non-monotonic top-k queries by minimizing parabolic polynomials as ranking functions. We introduced the minimum principle for this class of functions that enables an efficient method for calculating lower bounds for the Branch-and-Bound Ranked Search algorithm. Parabolic polynomials are useful as shown in this paper for e.g. the determination of the objects, which in certain attributes should be as close as possible and at the same time in others as far away as possible from the components of a query point. This class of functions fits well with partition strategy as long as the clusters are hyper cubes like minimal bounding rectangles in R-trees. Parabolic polynomials solve the minimization problem in the context of the Branch-and-Bound Ranked Search method.

References

[1] J. Clausen, "Branch and Bound Algorithms-Principles and Examples", Technical Report, *Department of Computer Science, University of Copenhagen*, 1999.

[2] G. Das, D. Gunopulos and N. Koudas, “Answering Top-k Queries Using Views,” *Proc. of the 32nd international conference on Very Large Data Bases,* pp. 451-462, September 2006.

[3] S. Drapeau, A. Jamneshan, M. Karliczek and M. Kupper, "The Algebra of Conditional Sets, and Concepts of Conditional Topology and Compactness," *J.Math. Anal. Appl.,* vol. 1, no. 437, pp. 561-589, 2016.

[4] M. L. Fredman, "A Priority Queue Transform," *WAE: International Workshop on Algorithm Engineering LNCS,* vol. 1668 , pp. 243-257, 1999.

[5] M. L. Fredman and R. E. Tarjan, "Fibonacci Heaps and Their Uses in Improved Optimization Algorithms," *Journal of the ACM,* vol. 34, pp. 596-615, 1987.

[6] A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching,” *Proc. ACM SIGMOD Conference Boston,* pp. 47-57, 1984.

[7] G. Hjaltason and H. Samet, “Distance Browsing in Spatial Databases,” *ACM TODS, 24(2),* pp. 265-318, Juni 1999

[8] A. Land und A. Doig, “An Automatic Method for Solving Discrete Programming Problems,” *Econometrica,* pp. 497-520, 1960.

[9] J. E. Martinez-Legaz, "On Weierstrass Extreme Value Theorem," *Optimization letters,* pp. 391-393, 2014.

[10] D. R. Morrison, S. H. Jacobson, J. J. Sauppe und E. C. Sewell, “Branch-and-bound algorithms: A Survey of Recent Advances in Searching, Branching, and Pruning,“ *Discrete Optimization,* Bd. 19, pp. 79-102, February 2016.

[11] D. Papadias, Y. Tao, G. Fu and B. Seeger, “An Optimal and Progressive Algorithm for Skyline Queries,” *ACM SIGMOD,* pp. 467-478, 2003

[12] P. Parrilo and B. Sturmfels, "Minimizing Polynomial Functions," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS 60,* pp. 83-99, 2003.

[13] P. Poensgen and R. Möller, “Quasi-Convex Scoring Functions in Branch-and-Bound Ranked Search,” *Open Journal of Databases (OJDB),* 7(1), Pages 1-11, 2020, [Online] http://nbn-resolving.de/urn:nbn:de:101:1-2019092919333113374958

[14] S. Ranu and A. Singh, “Answering Top-k Queries Over a Mixture of Attractive and Repulsive Dimensions,” *Proc. of the VLDB Endowment 5(3),* pp. 169-180, 2011

[15] N. Roussopoulos, S. Kelly and F. Vincent, “Nearest Neigbor Queries,” *Proc. ACM SIGMOD Conference New York,* pp. 71-79, 1995

[16] Y. Saad and M. Schultz, "Topological properties of hyper-cubes," *IEEE, Transactions on computers,* no. 37, pp. 867-872, 1988.

[17] R. Sperb, “Maximum Principles and their Appications,” New York: Academic Press, Inc., 1981.

[18] Y. Tao, H. Vagelis, D. Papadias and Y. Papakonstantinou, "Branch-and-Bound Processing of Ranked Queries," *Information Systems,* pp. 424-445, 2007.

[19] H. Vui and P. Son, "Minimizing Polynomial Functions," *ACTA Mathematica Vietnamica,* vol. 32, no. 1, pp. 71-82, 2007.

[20] D. Xin, J. Han and K. Chang, “Progressive and Selective Merge: Computing Top-k with Ad-hoc Ranking Functions,” *Proc. ACM SIGMOD International Conference on Management of Data,* pp. 103-114, June 2007.

AUTHOR BIOGRAPHIES

Peter Poensgen is an IT-coordinator at Talanx AG, a European insurance group based in Hannover and Cologne. He received a diploma in Mathematics and received the degree Dr. rer. nat. from the University of Lübeck. He started his professional career as IT-consultant (database and software development), an area in which he was working for more than five years. Peter also worked in the finance industry in various business areas (business intelligence and analytics, data management and software development). His research interests mainly focus on data mining, query processing and optimization as well as algorithms for solving convex optimization problems. Peter provides courses in these areas at FOM University of Applied Sciences in Cologne.

Ralf Möller is Full Professor of Computer Science at University of Lübeck and heads the Institute of Information Systems. He was Associate Professor of Computer Science at Hamburg University of Technology from 2003 to 2014. From 2001 to 2003 he was Professor at the University of Applied Sciences in Wedel/Germany. In 1996 he received the degree Dr. rer. nat. from University of Hamburg. Prof. Möller was a co-organizer of international workshops and is the author of numerous workshop and conference papers as well as several books and journal contributions (h-index=35 according to Google Scholar). He served as reviewer for all major journals and conferences in knowledge representation and reasoning research areas, and he has been PI in several EU and DFG projects. Professor Möller is spokesperson of the Research Unit “Data Linking” in the DFG-funded Cluster of Excellence “Understanding Written Artefacts”.