Open Access

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

www.ronpub.com/ojdb
ISSN 2199-3459

Quasi-Convex Scoring Functions in Branch-and-Bound Ranked Search

Peter Poensgen, Ralf Möller

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

ABSTRACT

For answering top-k queries in which attributes are aggregated to a scalar value for defining a ranking, usually the well-known branch-and-bound principle can be used for efficient query answering. Standard algorithms (e.g., Branch-and-Bound Ranked Search, BRS for short) require scoring functions to be monotone, such that a top-k ranking can be computed in sublinear time in the average case. If monotonicity cannot be guaranteed, efficient query answering algorithms are not known. To make branch-and-bound effective with descending or ascending rankings (maximum top-k or minimum top-k queries, respectively), BRS must be able to identify bounds for exploring search partitions, and only for monotonic ranking functions this is trivial. In this paper, we investigate the class of quasi-convex functions used for scoring objects, and we examine how bounds for exploring data partitions can correctly and efficiently be computed for quasi-convex functions in BRS for maximum top-k queries. Given that quasi-convex scoring functions can usefully be employed for ranking objects in a variety of applications, the mathematical findings presented in this paper are indeed significant for practical top-k query answering.

TYPE OF PAPER AND KEYWORDS

Short Communication: *top-k queries, query answering, top-k ranking, ranking, quasi-convex functions, scoring functions, branch-and-bound, Branch-and-Bound Ranked Search, BRS*

1 INTRODUCTION

A naive way for answering top-k queries is to consider a complete dataset of N tuples of a given relation, and compute the value of a scoring function for each tuple (whiich is seen as a data point), while maintaining and finally returning the k highest-ranked (or lowest-ranked) tuples. This algorithm has the computational complexity of $\mathcal{O}(N \log k)$, and since k is small and fixed, the procedure is called *sequential search*. For reducing search efforts one can hierarchically partition the set of data points (or the data space itself) in a preprocessing step. The resulting partitions can be assigned to different levels of an index tree (e.g., an *r-tree* or a *k-d-tree*), with which a top-k solution can then be determined for various queries with different scoring functions.

A well-known method for efficiently processing maximal (minimal) top-k queries based on tree indexes is the branch-and-bound principle. For this purpose, it must be possible to identify maximal (minimal) bounds in the search space to enable goal-oriented branching and bounded search. An associated procedure in the context of top-k queries is called Branch-and-Bound

Ranked Search (BRS) (see [14]). In order to be able to effectively apply BRS, it must be ensured that firstly for each partition there exist appropriate score values and secondly can be computed efficiently. In general, for arbitrary multivariate scoring functions used in applications the determination of search bounds based on scores for search space partitions requires considerable effort.

Only for monotone functions, maximal (minimal) bounds can easily be determined by considering the score of the top-right (lower-left) partition corner. For computing arbitrary (non-monotonic) BRS partition scoring functions, currently, there is no efficient algorithm known, and therefore, efficient top-k query answering cannot be provided up to now in the case of non-monotonic scoring functions. In this paper, we investigate the class of *quasi-convex point scoring functions* and provide an appropriate BRS *partition scoring function*, such that the BRS framework becomes applicable to large datasets for the first time in the context of quasi-convex scoring functions used in *maximizing top-k queries*. Given that quasi-convex scoring functions are used to allow for ranking domain objects in a variety of applications, the mathematical findings presented in this paper are significant for practical maximum top-k query answering.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces related background of Branch-and-Bound Ranked Search (BRS). Section 4 presents the class of quasi-convex functions and investigates the maximum principle to be used in bounded search for top-k solutions in BRS. Section 5 summarizes the key aspects developed in this paper.

2 RELATED WORK

Efficient top-k processing in domains such as the Web, multimedia search and database systems has shown a great impact on performance. A common way to identify the top-k objects is scoring all objects based on some ranking function. The class of the ranking function determines its use and the design of the respective top-k processing techniques. Regardless of its class, a ranking function is by definition a real multivariate function for which different fundamental theorems may apply. Many techniques have been proposed in literature for answering top-k queries.

A still valid classification is presented in [7]. This paper classifies top-k processing techniques based on the restrictions they impose on the underlying ranking function into:

- Monotone ranking functions
- Non-monotone ranking functions

The majority of top-k techniques assumes monotone ranking functions (see Definition 2 in Section 3.2.). Using monotone ranking functions is common in many practical applications. For example, many top-k processing scenarios involve linear combinations of multiple scoring predicates or maximum/minimum functions, which are all monotone. Monotone ranking functions have special properties that can be exploited for efficient processing of top-k queries. Several top-k techniques exploit the geometrical properties of linear functions to efficiently retrieve the top-k answers. All these methods are not applicable to non-monotone functions, because they have to assume monotonicity and its special features. For more details we refer readers to Section 3.2, 4.1.2 and 6.1 in [7].

Using non-monotone ranking functions is common in many practical applications as well. Example 1 and 2 given in Section 4 are representative for a variety of top-k processing scenarios using (quasi-) convex ranking functions. Many distance measures (see Section 4) are also convex functions. The problem of an efficient answering of top-k queries in the context of quasi-convex ranking functions has not been addressed properly yet. The following five papers deal with non-monotone ranking function: [16], [15], [10], [8] and [9]. However, none of these methods is suitable for the BRS algorithm.

In [7] the essentials of [16] and [15] are summarized as follows: The technique proposed in [16] supports arbitrary ranking functions by modelling top-k query as an optimization problem. The optimization goal function consists of a Boolean expression that filters tuples based on query predicates, and a ranking function that determines the score of each tuple. The goal function is equal to zero whenever a tuple does not satisfy the Boolean expression, and it is equal to the tuple's score otherwise. The answer to the top-k query is the set of k tuples with the highest values of the goal function.

The methodology developed in [15] presents an index-merge framework that performs a 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.

In the paper [10] a so called SD-Query is presented, which aggregates similarity and distance into a single function. The proposed function class measures the distance between a given (query-) point. The paper [8] presents a top-k procedure for a family of ranking functions allowing the use of distance functions among others. The idea of [9] is to decompose the ranking function as a supremum of a certain set of functions where an efficient top-k retrieval procedure can be

easily applied. The ranking functions used in [10], [8] and [9] are not quasi-convex. The work presented in this paper demonstrates in two ways why the class of quasi-convex functions in the context of branch-and-bound is so important:

- By introducing quasi-convex function as an upper set of convex function we can find top-*k* items by only controlling the vertices of each minimal bounding rectangle: Quasi-convexity fits with grid partition strategies using convex and compact partition sets.
- Quasi-convex functions generalize the maximum principle of monotone functions in the context of the BRS algorithm, which makes this method available to a wide range of practically relevant (non-monotonic) applications.

3 PRELIMINARIES

Branch-and-bound processing of ranking queries was introduced by Tao and colleagues as Branch-and-Bound Ranked Search, BRS [14]. The proposed 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. 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 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).

In geometry, an MBR (also called hyper rectangle) is a d-dimensional analog of a line $(d = 1)$, of a rectangle $(d = 2)$, or of a cuboid $(d = 3)$. The one-dimensional hyper rectangle is a line segment between two different points. The two-dimensional hyper rectangle has four 1-dimensional sides, each of which is a copy of a 1-dimensional hyper rectangle. The rectangle is formed by the joining two copies of dimension 1 by connecting corresponding points with a line segment (see Figure 1). The 3-dimensional hyper rectangle is the usual cuboid. Its six sides are 2-dimensional rectangles. The cuboid is formed by joining two copies (surfaces) of the dimension 2. This method can be generalized. The d-dimensional unit hyper rectangle is formed by joining two copies of the dimension $d-1$. In other words: The boundary of a d-dimensional hyper rectangle consists of a number of hyper rectangles of dimension $i = 0, \dots, d-1$, for each i we have $\binom{d}{i}2^{d-1}$ hyper rectangles.

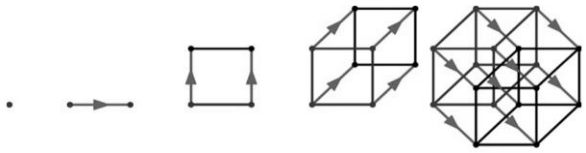


Figure 1: Hyper rectangles of dimension 0 to 4 ([4])

Definition 1. Hyper Rectangle

An *axis-parallel hyper rectangle* $M \subset \mathbb{R}^d$ is defined as the finite Cartesian product $M = I_1 \times I_2 \times \dots \times I_d$ of closed intervals $I_j = [l_j, h_j]$ with $l_j \leq h_j$ for any $j = 1, \dots, d$. The point $h = (h_1, \dots, h_d)$ is the upper right corner of M, $l = (l_1, \dots, l_d)$ the lower left corner. Thus, one can represent a d-dimensional hypercube by the vector $v_M = (h_1, \dots, h_d, l_1, \dots, l_d) \in \mathbb{R}^{2d}$. All vertices of M, *vertices(M)*, can be derived from v_M by combining the corresponding coordinates of the different dimensions. An *MBR* is the smallest enclosing axis-parallel hyper rectangle for a set of data points. A *general hyper rectangle* is a hyper rectangle that does not have to be axis-parallel. □

To ensure that quasi-convex functions $f: M \to \mathbb{R}$ are bounded on their domains it is sufficient to assume that a hyper rectangle M is compact. 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 Heine-Borel Theorem (see [5]). 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 hyper rectangle is also compact.

We use the iterative construction method for the proof of the main claim of this paper in Section 4.

3.1 Branch-and-Bound Ranked Search

The strategy to answer a maximum (or minimum) top-k query with BRS is described as a bounded search through an r-tree (see Listing 2). 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.

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 $score\_p$. 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

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)))
               // pq with (obj, score) entries
    result = {}
3   n = 0
4   object
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-Algorithm (based on [14])

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 $score\_p$, and point scores are used to insert each point into *pq* (Line 12 and 13). Otherwise, a non-leaf r-tree node is found (Line 14), and the respective children are inserted into *pq* (Line 15 and 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. The result points are returned and represent the *k* top-scored objects. We can safely assume that $score\_p$ is an operation in $\mathcal{O}(1)$, whereas $score\_r$ is only 'simple' for monotonic $score\_p$ functions. For non-monotonic functions, $score\_r$ can impose considerable overhead, and indeed can be implemented effectively only for specific classes of functions, as we will see below. Once $score\_r$ works effectively, bounding the search via *pq* and *rt* is effective, and the BRS algorithm can be applied to large datasets.

The actual call to BRS is given as follows: BRS(*data_rtree*, *k*, *max*, *scorePoint*, *scoreMBR*), where *k* is the expected number of results, the function *max* indicates maximum top-k, $scorePoint$ is a query-specific function that maps a data object (point) to a score being used for ranking the object, and $scoreMBR$ determines the largest value that the function $scorePoint$ can return when applied to the points associated with an MBR specified as a parameter. We will give an efficient implementation of $scoreMBR$ for quasi-convex $scorePoint$ functions below.

3.2 Maximum Principle of Monotonic Ranking Functions

Definition 2. Monotonic Multivariate Function

A function $f: D \to \mathbb{R}$; $(x_1, \dots, x_d) \mapsto y, D \subset \mathbb{R}^d$ of d real variables is *monotonic increasing* if and only if $f(x_1, \dots, x_d) \leq f(x'_1, \dots, x'_d)$ if $x_i \leq x'_i \; \forall \; i = 1, \dots, d$. It is *strictly* monotonic increasing if strict inequality holds. Similarly, we can define (strictly) *monotonic decreasing* $(\geq)$ functions. An increasing or decreasing function is called a monotonic function. Otherwise f is called non-monotonic. For example each affine linear function $f(x_1, x_2) = ax_1 + bx_2 + c$ for each $a, b, c \in$

$\mathbb{R}^+$is monotonic, whereas $g(x_1, x_2) = \frac{x_1}{x_2}$ is obviously non-monotone. □

Let $x^0 = (x_1^0, \dots, x_n^0) \in D$ be an arbitrary but firmly selected point of the domain of a multivariate function $f: D \to \mathbb{R}, x \mapsto f(x_1, \dots, x_n)$. If you just change the i-th component and leave all others then the mapping $x_i \mapsto f_i(x_1^0, x_2^0, \dots, x_{i-1}^0, x_i, x_{i+1}^0, \dots x_n^0)$ defines a one-dimensional function, which is called the i-*th component function* of f. f is increasing (decreasing) *in the i-th component* if f_i is (independent from x^0) an increasing (decreasing) function of x_i. If a function is increasing (decreasing) we denote it as a monotone function in the i-*th* component.

Monotonicity implies that any real-valued monotonically increasing function of several variables $f: M \to \mathbb{R}$ takes its maximum at the upper right vertex of the MBR, i.e., the vertex with the largest coordinates. Simplified, we will speak of a *maximum principle* of a function, assuming its maximum values on the edges of its domain (see [12]). A monotonic function takes its maximum at a vertex of an MBR, which is the upper right (lower left) one if f is increasingly (decreasingly) monotone. If a function f is strictly monotonic in all dimensions, but not necessarily increasing or decreasing in all dimensions the function also assumes it's maximum in a vertex, which does not necessarily have to be the upper right or lower left one. It is called the dominating vertex (see [14]). If an MBR M is spanned by the two opposite vertices $l = (l_1, \dots, l_d)$ and $h = (h_1, \dots, h_d)$ where l is the upper left, h the upper right one, then the dominating vertex $e = (e_1, \dots, e_d)$ can be specified by $e_i = l_i$,, if f is strictly decreasing on the i-th dimension and $e_i = h_i$,, if f is strictly increasing on the i-th dimension for each $i \in \{1, \dots, d\}$. Figure 2 shows the maximum points of the monotonically increasing function f_2, the monotonically decreasing function f_3 and the maxima of two component-wise monotonic functions f_1, f_4.

However, for non-monotonic functions, this kind of simple maximum principle does not hold, and the above method to compute the dominating point is not easily adaptable to non-monotonic functions. In general, for a function f in several variables one has to use complex methods of mathematical analysis to compute the bound (MBR score) $f_{max}(M)$. Next, we will show that quasi-convex functions take their maxima on some of the (finitely many) vertices of a (d-dimensional) MBR. Thus, in this respect, quasi-convex functions represent a generalization of the class of monotonic functions, the latter of which has the property that the maximum is found on one known vertex (top-right vertex).

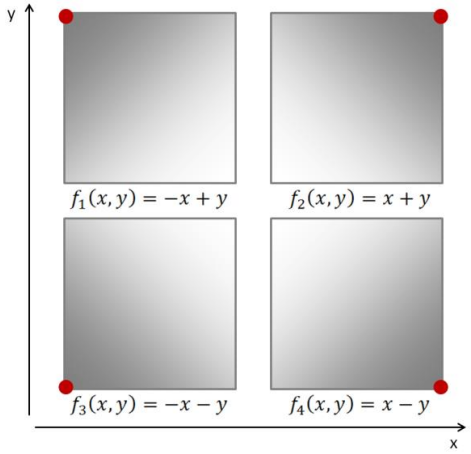


Figure 2: Maximum principle of (component-wise) monotonic functions for 2-dimension

4 QUASI-CONVEX FUNCTIONS

In this section, we characterize quasi-convex functions in terms of hyper rectangles, which are always convex and compact subsets of the Euclidian space. The formal definition of convex and quasi-convex functions is presented as follows: (see Section 3.4.1 of [3]).

Definition 3. **Convex and Quasi-Convex Function**

Let $f: C \to \mathbb{R}$ be a function defined on a convex subset C in $\mathbb{R}^d$. Recall that a set C is said to be convex, if for all x, y ∈ C and all $t \in [0,1]$, the line segment $tx + (1-t)y$ between x and y also belongs to C.

A function f is called convex, if

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \qquad (1)$$

for all $x, y \in C$ and all $t \in [0,1]$. Geometrically, this inequality means that the line segment between $(x, f(x))$ and $(y, f(y))$, which is the chord from $f(x)$ to $f(y)$, lies above the graph of f.

A function f is called *quasi-convex*, if

$$f(tx + (1-t)y) \leq max\{f(x), f(y)\} \qquad (2)$$

for all $x, y \in C$ and all $t \in [0,1]$. This means that on a line segment the maximum of a quasi-convex function is always attained in one of its end-points. □

Note that quasi-convexity is a generalization of convexity. Any norm is convex (see [1]), for example

- the max-norm $f(a_1, \dots, a_d) = \max_i\{|a_i|\}$,
- the Euclidian norm $f(a_1, \dots, a_d) = \sqrt{\sum_{i=1}^d a_i^2}$,
- the p-Norm: $f_p(x) = \left(\sum_{i=1}^d |a_i|^p\right)^{\frac{1}{p}}, p \geq 1$.

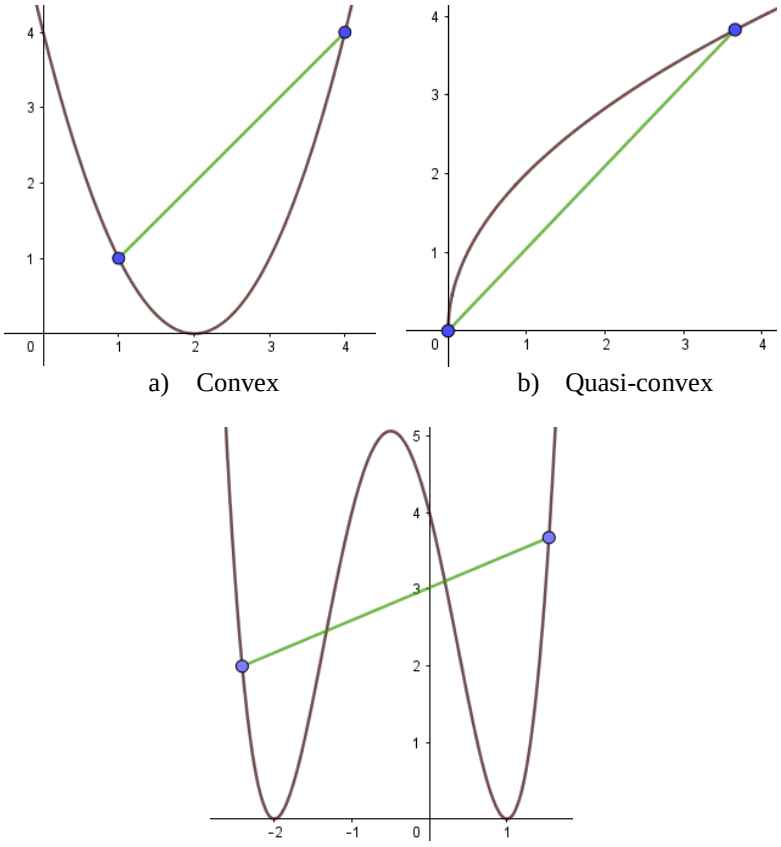


Figure 3: Convex, quasi-convex and non-convex functions in 1-dimension

Thus, any top-k query whose ranking function is a norm and which is to be maximized can be applied to the BRS algorithm. The following function $f: \mathbb{R}^d \to \mathbb{R}$ given by $f(a_1, \ldots, a_d) = \sqrt{\max_i\{|a_i|\}}$ is quasi-convex but not convex (see [2]). The main difference between convex, quasi-convex, and non-convex functions for dimension $d = 2$ is illustrated in Figure 3.

Alternatively, quasi-convexity can be defined by sublevel sets. Thus, a function f is quasi-convex if and only if all its sublevel sets $N_f(\alpha) = \{x \in D \mid f(x) = \alpha, \alpha \in \mathbb{R}\}$ are convex (see [3], 3.4.1 and Figure 4). Many requirements from practice can be modeled by quasi-convex ranking functions [3].

Example 1. **Real Estate Database**

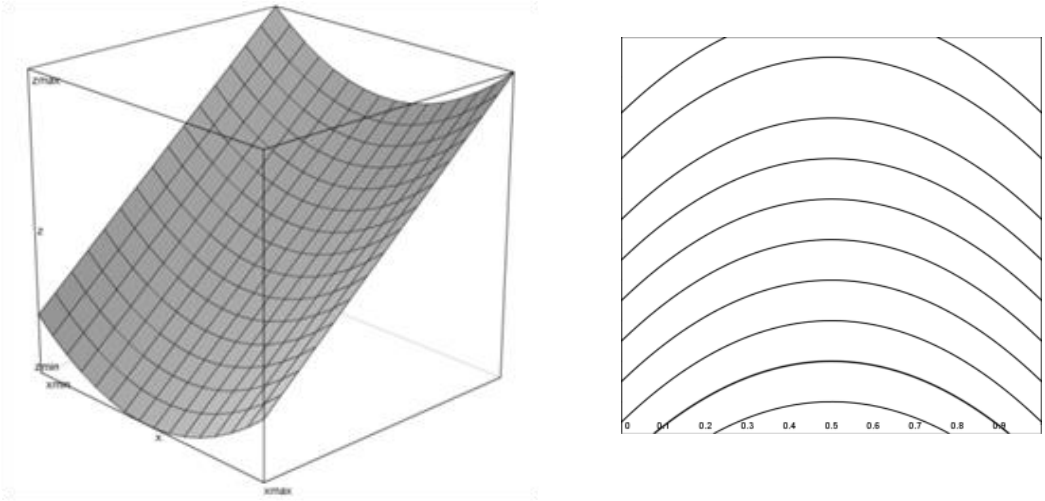
As an example, we consider an online real-estate information system for buildings with the following attributes: latitude (*la*), longitude (*lo*) as well as post code (*zip*) information. Consider the case of data stored in a relational database table *RealEstate*. We are interested in the top ten objects which are located in the northern outskirts of Munich (north/eastern or north/west), as close as possible to the airport (north) of Munich. Without loss of generality, we consider a projection of both attributes to the unit interval which maps the center (48.13, 11.58) of Munich to (0.5, 0) (see [3]). The corresponding maximal top-k query over this database is given by the following SQL query Q1:

```
SELECT TOP 10 *
FROM RealEstate
WHERE zip BETWEEN 80331 AND 81929
ORDER BY (la − 0.5)² + lo DESC
```

Figure 4 shows the plot of the function $f(x_1, x_2) = (x_1 - 0.5)^2 + x_2$ over its two attributes and its sublevel sets, which are obviously convex. Hence, as stated above f is (quasi-)convex but obviously not monotone. As a second example, we consider a patient database with a *Patient* table that provides the *age* and blood pressure values *sys* and *dia* of each patient.

Example 2. Hypertension or Hypotension

In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. It can cause serious



a) 3-dimensional Graph

b) 2-dimensional Sublevel sets

Figure 4: Graph and sublevel sets of the function of Example 1

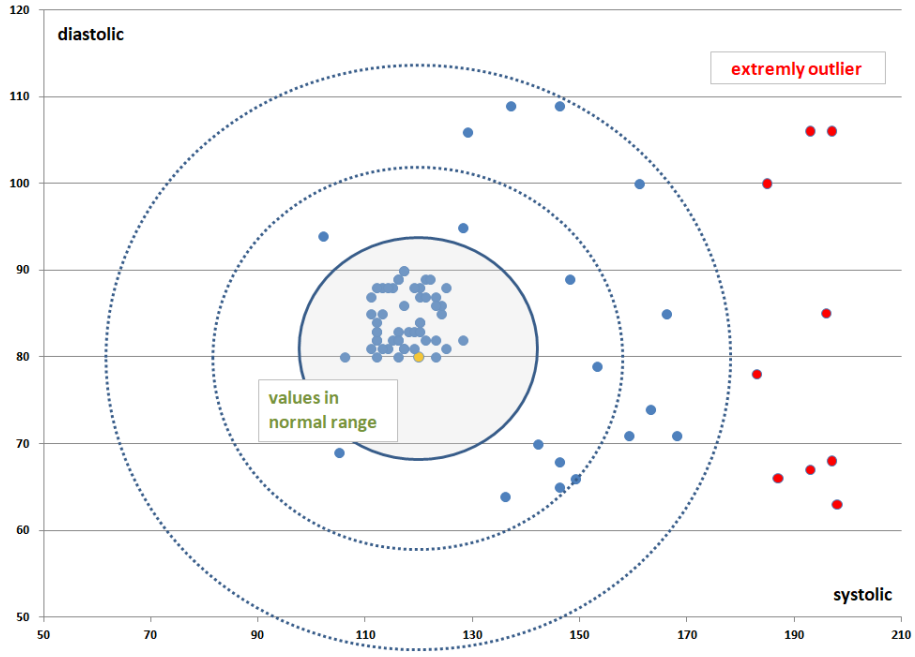
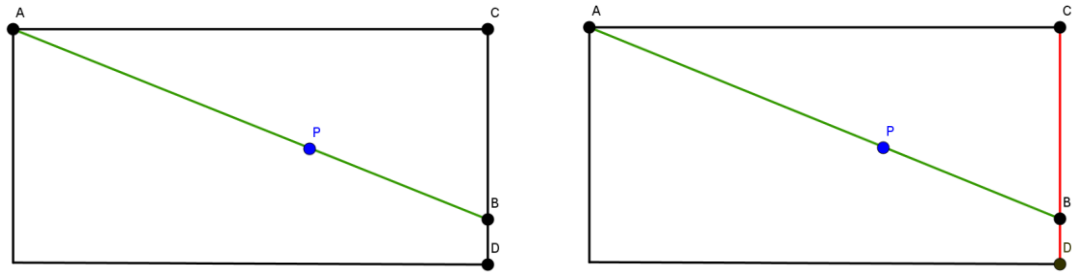


Figure 5: Patient sample with blood pressure readings

problems in statistical and business analyses. Sometimes, as in the following example, outliers are just the result set of a problem (See the red points in Figure 5). The definition for outliers is not unique and varies with the application. For our example an outlier is a data point that is very different in one or more coordinates from the coordinates of a given query point.

We would like to generate a sample of 10 patient aged 40 to 50 with high (Hypertension) or low blood pressure (Hypotension). The goal is to find the k highest scoring points by maximizing the following SQL query Q2:

```
SELECT TOP 10 *
FROM Patient
WHERE age BETWEEN 40 AND 50
ORDER BY ω1(sys − 120)² +
         ω2(dia − 80)² DESC
```



a) Straight line intersects the boundary b) Boundary with maximum point

Figure 6: 2-dimensional Hypercube

The positive real numbers ω_1, ω_2 are weighting parameters. In both examples, the ranking functions are not monotonic, and thus the known monotonic frameworks (See [7]) are not applicable.

Both examples, like many other practical applications, use a convex ranking function, which is generally not monotonic (See [13]). In order to be able to derive a criterion for the determination of a score for each MBR of an r-tree, however, we only need the more general property of quasi-convexity, which holds for convex functions as well. For Q1 in our Example 1 we use $pointScore = lambda(p).(sqr(la(p) - 0.5) + lo(p))$ as a parameter in the call to BRS (see Listing 2). The second parameter of our top-k query example, lo (longitude), is used in a linear fashion here just for demonstration purposes, it could be handled in the same way as *la* (latitude). In the following section, we will show that a quasi-convex function is characterized by being maximal on the vertices of each MBR.

4.1 Maximum Principle for Quasi-Convex Ranking Functions

Quasi-convex functions, unlike convex ones, need not be automatically continuous. The floor function $\mathbb{R} \to \mathbb{R}; x \mapsto \lfloor x \rfloor$ is an example of a quasi-convex function that is neither convex nor continuous. To ensure the existence of a maximum, it is sufficient to assume compactness of the domain, the continuity of the function is not necessary. If compactness is given, i.e., if the domain is a hyper rectangle, we will show that the maximum of quasi-convex functions is to be found on the boundary of that domain. This principle is called maximum principle of a function [12]. We are now ready to formulate our main result as a theorem. Note that the assumptions of the theorem as well as its claim correspond to properties of the hyper rectangle.

Theorem 1. Maximum principle of quasi-convex functions

Let $f: D \to \mathbb{R}, D \subseteq \mathbb{R}^d$ be a quasi-convex (ranking) function. For each hyper rectangle $M \subseteq D$ generated by its 2^d vertices $e_1, \dots, e_{2^d} \in \mathbb{R}^d$ the following holds:

$$f_{max}(M) = max\{f(e_1), \dots, f(e_{2^d})\}. \quad (3)$$

In other words: If one limits the domain of a quasi-convex function f to a (general) hyper rectangle, then f takes its (not necessarily unique) maximum in some vertex of the hyper rectangle.

Proof by Induction: Basis: We show that the statement holds for $d = 2$. In this case M is a two-dimensional rectangle of the Euclidian plane which is mapped by f to a three-dimensional surface illustrated in Figure 7. If the maximum point P is on the edge of the rectangle our claim is proven, because a quasi-convex function of a one-dimensional hyper rectangle (line-segment), assumes its maximum on its vertices. That is precisely the definition of a quasi-convex function. We assume the maximum point P is in the interior of the rectangle. A straight line through an arbitrarily chosen vertex A and P intersects the boundary of the rectangle in a further point B (see Figure 6 a). Since f is quasi-convex $f(P) \leq max\{f(A), f(B)\}$ holds. If $f(A) \geq f(B)$ our claim is proven. If this does not hold, we consider the straight line through B and a second vertex C. The second straight line is a sub-set of the edge of the rectangle. Therefore, it intersects the box in a third vertex D (see Figure 6 b).

Inductive step: We have to show, that if the claim holds for dimension $d - 1$, then it also holds for dimension d. The statement follows immediately from the (geometric) construction of a d-dimensional hyper rectangle M (see Section 3) that is composed of $2d$ $(d - 1)$-dimensional hyper rectangles, so that each

Algorithm: `scoreMBR` $(M, score\_p)$

```
// M is an MBR
// score_p is a point scoring function
1   {e_1, ..., e_2^d} := vertices(M)
    // the vertices function generates the set
    // of vertices of a d-dimensional MBR M
2   max({score_p(e_1), ..., score_p(e_2^d)})
```

Listing 2: Algorithm for determining the MBR max score for quasi-convex *score-p* functions

straight line through a vertex A and a point P of the interior of M intersects an i-dimensional hyper rectangle in the boundary for at least one $i \in \{0, \dots, d-1\}$. □

This theorem is particularly important because the class of quasi-convex functions fits well to the BRS algorithm, since one can calculate the bounds of the MBRs only by considering their bounding vertices, similarly to the monotone case in which the bounding vertex is directly determined. Monotonicity implies that any real valued monotonic increasing function of several variables $f: M \to \mathbb{R}$ takes its maximum at the upper right vertex of the hyper rectangle, i.e., the vertex with the largest coordinates.

This principle is not adaptable in general, but for quasi-convex functions one can determine a score by computing scoring function values for all vertices and then finding the largest one. The algorithm for computing the max score for a quasi-convex function to be used as an argument to BRS is given in Listing 2. For the actual call see line 16 in Listing 1.

4.2 Characterization of Quasi-Convex Functions on Hyper Rectangles

The following theorem characterizes quasi-convex functions on hyper rectangles. It shows that also the backward direction of Theorem 1 holds. This means that a quasi-convex function is already uniquely determined by the maximum principle on general hyper rectangles. Theorem 1 has shown that and how the BRS can be applied to non-monotone functions. Theorem 2 makes clear why quasi-convexity is a generalization of monotonicity in the context of top-k queries answered with the BRS algorithm.

Theorem 2. Characterization of quasi-convex functions

Let $f: D \to \mathbb{R}$ a function, $D \subseteq \mathbb{R}^d$ a convex domain, then the following holds: If f is maximal on at least one vertex of each hyper rectangle $M \subseteq D$ the function is quasi-convex.

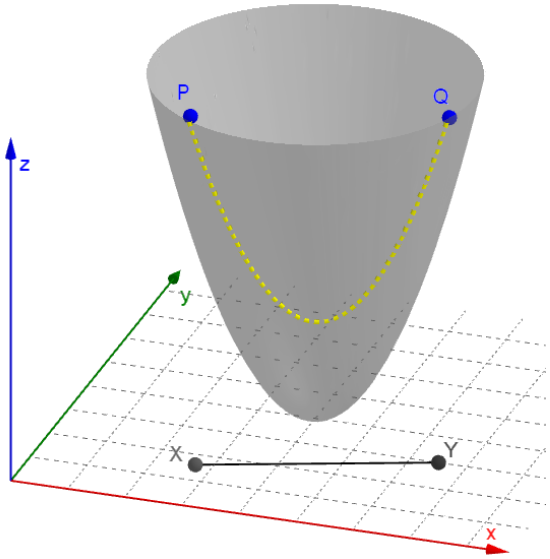


Figure 7: Maximum values of a one-dimensional hyper rectangle

Note: The term "each" means, that M is a general hyper rectangle.

Proof: Let s be a line segment between any two selected points $P = (X, f(X))$ and $Q = (Y, f(Y))$. By assumption, f takes its maximum value on the vertices on each M. Thus, this function is also maximal on the corners of all one-dimensional hyper rectangles, the line segments. So, in particular, f takes its maximum on the one-dimensional hyper rectangle which is spanned by the two vertices X and Y (Figure 7 illustrates the situation for dimension 2). □

5 CONCLUSIONS

In contrast to monotonic functions used for ranking functions, for general functions no direction can be specified for value increase. The algorithm of monotonic functions for determining the dominant vertex given in [14] is useless in case of quasi-convex functions. Therefore, in order to determine the maximum vertex or corners and, thus, the upper bound, unless specific additional properties can be exploited for the function, one must generally calculate all 2^d function values and select the largest one.

The procedure call $max\{f(e_1), \dots, f(e_{2^d})\}$ to identify the max score $f_{max}(M)$ of a given MBR M is exponential in the number of dimensions. However, for a fixed dimension d, which will usually be rather small in practical applications, the overall BRS (Branch-and-Bound Ranked Search) procedure is indeed average-case-logarithmic in the number of points (depending on the r-tree structure), and therefore efficient in practice. The procedure *scoreMBR* as defined in this paper is preferable to uninformed elaborate and inefficient analytical methods of mathematical analysis (partial derivatives, Hesse matrix, etc.) for determining maximal MBR score. BRS computation for quasi-convex ranking functions is considered to be a fixed-parameter tractable problem on d (FPT for short). The class of quasi-convex functions is the class of functions that take their maxima on some of the finitely vertices of each (general) hyper rectangle.

The main results of this paper are that we can find the key values for each r-tree node (MBR) only by calculating the function values of the vertices of the MBRs. The findings show that in this case more general but costly methods of mathematical analysis for maximum finding are not required, and thus we make the BRS algorithm effective not only for monotone ranking functions, but also for quasi-convex ranking functions in maximum top-k queries. Therefore, the results of the experimental evaluation in [14] can be transferred to quasi-convex functions, since only the *getMaxScore* function that calculates the upper bounds for the BRS has changed.

REFERENCES

[1] A. A. Ahmadi, E. de Klerk and G. Hall, "Polynomial Norms," *SIAM Journal on Optimization*, vol. 29(1), p. 399–422., 2019.

[2] H. H. Bauschke and P. L. Combettes, Convex Analysis and Monotone Operator Theory in Hilbert Spaces, Second ed., Canada: Springer, 2017, pp. 177-187.

[3] S. Boyd and L. Vandenberghe, "Convex Optimization," *Cambridge University Press*, 2004.

[4] H. S. M. Coxeter, Regular Polytopes, New York: Dover Publications, Inc., 1973.

[5] J. Dieudonné, Grundzüge der modernen Analysis, Bd. 1, Braunschweig: Vieweg, 1985.

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

[7] I. F. Ilyas, G. Beskales and M. A. Soliman, "A Survey of Top-k Query Processing Techniques in Relational Database Systems," *ACM Computing Surveys 40(4), Article 11*, 2008.

[8] N. Madrid and U. Straccia, "On Top-k Retrieval for a Family of Non-monotonic Ranking Funktions," *Flexible Query Answering Systems*, pp. 507-518, 2013.

[9] N. Madrid and P. Rusnokb, "A Top-K Retrieval Algorithm Based on a Decomposition of Ranking Functions," *Information Sciences*, vol. 474, pp. 136-153, 2019.

[10] 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.

[11] Y. Saad and M. Schultz, "Topological Properties of Hyper-Cubes," *IEEE Transactions on computers*, no. 37, pp. 867-872, 1988.

[12] R. Sperb, Maximum Principles and their Appications, Academic Press, Inc., 1981.

[13] V. Sverák, "New Examples of Quasiconvex Functions," *Archive for Rational Mechanics and Analysis 119.4*, pp. 293-300, 1992.

[14] Y. Tao, V. Hristidis, D. Papadias and Y. Papakonstantinou, "Branch-and-Bound Processing of Ranked Queries," *Information Systems, 32(3)*, pp. 424-445, 2007.

[15] D. Xin, J. Han and K. Chang, "Progressive and Selective Merge: Computing Top-k with Ad-hoc Ranking Functions," *Proc. ACM SIGMOD*, pp. 103-114, June 2007.

[16] Z. Zhang, S. Hwang, K. Chang, M. Wang, C. A. Lang and Y. Chang, "Boolean + Ranking: Querying a Database by K-Constrained Optimization," *Proc. ACM SIGMOD*, pp. 359-370, 2006.

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 started his professional career as IT-consultant (database and software development), an area in which he was working for 5,5 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 a 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”.