B. Mitschang et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2017),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 445

Quadtree-based Resource Description Techniques for Spatial Data in Distributed Databases

Stefan Kufer[1] Andreas Henrich[1]

Abstract: In the social media age, content creation and distribution of all sorts of digital media is of growing importance. The variety of media types, devices, and user groups results in the generated media items being stored in a very heterogeneous landscape which can be seen as a large distributed database consisting of personal computers, mobile devices, or web servers of social media sites, demanding for an appropriate overarching search system. Resource selection based on compact resource descriptions allows to efficiently determine resources maintaining relevant media items. Among other aspects, geospatial footprints have to be addressed for an effective media search.

We propose and evaluate quadtree-based techniques to summarize collections of geo-referenced media items in a distributed geographic IR context. Based on these summaries, resource selection decisions are made when searching for media items close to a given geographic location. The quadtree-based approaches are evaluated against previous work. The quadtree-based summaries prove to be very flexible and competetive.

Keywords: Geographic Information Retrieval, Distributed Information Retrieval, Resource Selection, Summarization, Spatial Data

1 Introduction

In distributed retrieval settings, two principal approaches can be distinguished. In the first setting, the data is assigned to the distributed resources (i.e. to the nodes in the network) based on its attribute values. Considering spatial attributes this could mean that each resource maintains data items spatially located in a certain area. Of course, this setting allows for an efficient processing for spatial nearest neighbor queries. However, the distribution of the data to the resources can be based only on one attribute and it might not be appropriate at all in certain application scenarios. This brings us to the second scenario of distributed retrieval, where the data is maintained in different resources simply because of its origin or other circumstances. An example could be geo-tagged media items—images for example—which are maintained in the personal media archives of people or organizations participating in a network.

Especially in the second scenario, effective and efficient resource description and selection techniques are important for distributed retrieval. Obviously, the media items usually have different attributes: low level image features, tags or short textual descriptions, timestamps, and a geographic position where the image was taken. Queries for media items can address all these aspects. Consequently, summaries (resource descriptions) for all these aspects should be distributed in the network (e.g. via rumor spreading [Cu03]) to allow for a targeted

[1] Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik, An der Weberei 5, D-96047 Bamberg, (stefan.kufer|andreas.henrich)@uni-bamberg.de

selection of the resources relevant to a certain query. Because of its two-dimensional nature and its descriptive power the spatial information can play an important role here. On the other hand, simple approaches for summarization, like bounding boxes, obviously are not appropriate. Assume for example a person usually taking pictures in Upper Franconia who in addition has some images in her collection taken at her last trip to the Olympic Games in Brazil. In this case, a bounding box would not be appropriate to describe these two spots.

In the paper at hand we present and evaluate quadtree-based approaches in comparison to previously published approaches based for example on multiple rectangles or Voronoi-like space partitionings. It turns out that quadtree-based approaches are well-suited especially when extremely compact resource descriptions are desired.

2 Problem Description and Preliminaries

In a multi-database model, the existence of multiple databases is explicitely modelled. It is scalable to large numbers, but introduces some additional complexity compared to the single-database model [Ca00, 127f]:

Resource Description Problem A brief description of the contents of a database or resource is required, facilitating the identification of resources providing relevant information.

Resource Selection Problem A subset of resources that (most likely) contain relevant information is selected (based on the resource descriptions) to contact while querying.

Result Merging The returned results of the selected subset of resources to be contacted have to be integrated and merged into an overall result.

For our distributed search scenario, we assume that every resource in the network maintains a set of geotagged media items, meaning each media item is enhanced with a single pair of latitude/longitude (lat/lon) coordinates. The geo-coordinates will be treated as plate-carrée-projected data points, resulting in lat/lon coordinates corresponding to x/y coordinates in a two-dimensional Cartesian coordinate system (with lon being x). Since the data points thus are encoded in a two-dimensional Euclidean space, we use the Euclidean distance for distance calculations. Note that former work [BH12] has investigated distance measures better suited for distance calculations on the earth's surface, namely the Vincenty distance and the Haversine distance, but did not result in relevant differences compared to the computationally less complex Euclidean distance.

Though the data is bounded (-90 to 90 in lat $= y$ and -180 to 180 in lon $= x$), the Date Line is taken into account for *all* distance calculations (point-to-point, point-to-rectangle, etc.), considering the data stems from lat/lon coordinates on the earth. Consequently, the distance between for example two points $d_1 = (179; 0)$ and $d_2 = (-179; 0)$ is 2 rather than 358. For this work, we take a static perspective on the database, meaning the number of data points per resource will not grow and shrink. In a 'real-world' scenario, the resource description techniques of course are qualified for dynamic data—by recalculating the respective resource description of a resource whose set of data points alters.

3 Resource Descriptions for Geospatial Data

For the Resource Description Task, the objective is to encode sets of two-dimensional data points effectively (accurate description) and efficiently (compact storage). Furthermore, a fast execution of the geometric search operations is also desirable. Previous work ([KBH12], [KBH13], [KH14], and on a more abstract level [BHK16]) has investigated several techniques distinguishable into three categories, which briefly are recapitulated in section 3.1. After clarifying some prerequisites for the use of quadtrees in our scenario (section 3.2), the quadtree-based techniques newly developed for this work are presented in section 3.3.

3.1 Basic and State of the Art Resource Description Techniques

Generally, in the context of Point Access Methods (PAMs), it is often distinguished between techniques which *organize the data* and techniques which *organize the embedding space* [Sa05, p. 2f]. We adapt this categorization and differentiate *Geometric Approaches* which organize the data and *Space Partitioning Approaches* which organize the embedding space. Furthermore, the properties of two arbitrary techniques can be combined into a new technique, which we classify as *Hybrid Approaches*.

Geometric Approaches Techniques of this category are using one or several bounding volumes to delimit a set of data points and thus *organize the data*. The two properties to be decided are the shape of the bounding volumes and the quantity of bounding volumes used.

For the shape, axis-aligned rectangles are most common: they are easy to compute, have a small memory footprint, and facilitate fast intersection and distance tests. Using a single shape for describing the data often is not an adequate solution, since the resource's data points can be accumulated at widely separated locations with a lot of 'dead space' inside the bounding volume. Thus, a resource's point cloud has to be divided into disjoint groups of spatially adjacent data points which then each are approximated by a bounding volume. Generally, clustering algorithms and algorithms for object decomposition of complex spatial objects are applicable for that. In the following, we will present the MBR as a basic single shape technique and the RecMAR$_{k,sl}$ as a technique utilizing several boxes.

MBR The Minimum Bounding Rectangle (MBR), also known as bounding box or envelope [Ca05, p. 1], is a rectangle whose sides are parallel to the coordinate axes of the space [Sa05, p. 195]. It is the smallest rectangle bounding a set of spatial features (the resource's data points in our case) and can be specified by the lower left corner and the upper right corner. The MBR is one of the most basic spatial data representations and will serve as a comparative scale for the other approaches. The MBR might cross the Date Line.[2]

For the resource description, the coordinates of the lower left corner and the upper right corner are captured in single-precision floating-point format, which occupies 32 bits for each value. The four necessary values for encoding the two-dimensional rectangle are linearly stored in a bit vector (see Figure 3 for an overview on the encoding schemes).

[2] Date Line crossing boxes can also occur for other Geometric Approaches or Hybrid Approaches utilizing bounding volumes as a foundation, since the computation of geometric shapes does not require bounded data spaces; thus the Date Line is not considered as a boundary for the x dimension for these techniques. This also applies for the polygons of the Voronoi diagrams' cells.

RecMAR$_{k,sl}$ The Recursive Minimum Area Rectangles (RecMAR$_{k,sl}$) approach is based on an algorithm developed by Becker et al. [Be92] for approximating a set of axes-parallel rectangles by two axes-parallel MARs, which—among all the pairs (s, t) of data point enclosing rectangles inside the overall MBR m—finds the pair for which the sum of the areas of s and t is minimal. The algorithm separately looks for different types of solutions which are distinguished depending on (a) the adjacency of the sides of s and the boundary of m and (b) the overlap between s and t. The best one of these solutions is selected. We adjusted the algorithm for the use with point data.

Furthermore, the algorithm can be applied recursively to compute up to k MARs, since it decomposes the arbitrary MBR m into the two MARs s and t (which contain all of m's data points). In the set $M = \{m_0, \ldots, m_n; n < k\}$ of already computed rectangles, the rectangle m_i whose area is largest is taken from M. m_i then is decomposed into the two MARs $m_{i.s}$ and $m_{i.t}$, which afterwards replace m_i in M. The recursion stops if either the maximum of k rectangles has been reached or the distance between the center of a rectangle and each of its associated data points is smaller than a threshold value sl for all rectangles. Conceptually, RecMAR$_{k,sl}$ is closely related to split strategies known from R-tree based approaches.

Space Partitioning Approaches The principle of methods describing the embedding space is to decompose the space into disjoint subspaces which then are exploited to identify regions (not) containing data points. Here, hierarchical and non-hierarchical methods can be distinguished. Hierarchical decomposition is based on the principle of recursive decomposition of the space [Sa05, p. XIX] that is represented by a corresponding tree structure which can be binary (such as for the kd-tree family) or multiway (such as the quadtree family). Non-hierarchical decomposition divides the space without the utilization of an underlying tree structure (for example grid-based methods).

Generally, the resulting subspaces are distinguished into regions containing data points (*occupied cells*) and regions not containing data points (*non-occupied cells*). Binary or quantitative information about cell occupancy can be associated with each cell. Another important distinction is whether the space decomposition is *the same for all resources* or if it may be *individual* for each resource. This depends on the question if the information necessary to (re)construct the subspace boundaries can efficiently be represented. If so, resource-individual space decomposition is applicable.

UFS$_{n,cc}$ The Ultra Fine-grained Summaries are using a Voronoi diagram to partition the underlying space. Generally, the Voronoi diagram of a given set $S = \{s_1, ..., s_n\}$ of n sites in R^d partitions the space of R^d into n regions, one per site. Each region includes all points in R^d with a common closest site in the given set S according to a distance metric D (Euclidean distance in our case) [Sa05, p. 346]. The Voronoi diagram is a non-hierarchical space decomposition which is not suited for resource-individual space decomposition, since either the coordinates of the sites or the corresponding polygons of the Voronoi cells are required to determine the locations of the subspaces. This information cannot be encoded efficiently in a resource's description given several hundred or thousand sites. Hence, the UFS$_{n,cc}$ approach utilizes a 'global' Voronoi diagram of a given set $S = \{s_1, ..., s_n\}$ which is the same for all resources. The distinctive quality of the space decomposition is dependent on the selection of appropriate sites [HB10], for which we randomly chose data points right

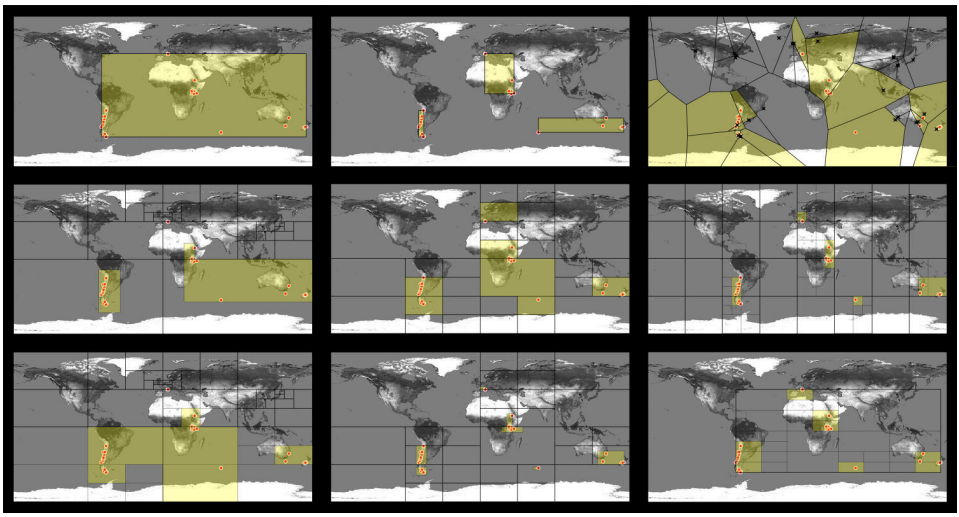

Fig. 1: Visualization of the techniques presented in this paper for the example resource with 2,186 data points. Left to right, top to bottom: MBR, RecMAR$_{3,0.1}$, UFS$_{32,x}$, KDMBR$^{3}_{32}$, QT$_{32,1.0}$, GridQT$^{8,1.0}_{4}$, KDQT$^{8,1.0}_{32}$, QTMBR$^{3}_{32,1.0}$, MBRQT$^{32,1.0}$. The red dots denote data points of the resources, the yellow surfaces indicate the regions containing data points which are described by the summaries.

out of the data collection. Not all sites have to be considered when ranking the resources. The parameter cc controls how many sites are taken into account (also see section 4).

Binary information about cell occupancy is captured. The resource description is represented by a bit vector with consecutive occupancy information for all n subspaces.

Hybrid Approaches Hybrid Approaches combine properties of two description methods for more effective results. The combination of one method each from the two aforementioned categories often achieves particular synergy, though it is also possible to combine two methods from the same class. Regardless of the specific methods combined, a two-step approach is used for all hybrid methods: In the first step, method `A` is used to build the foundation of the description. In the second step, method `B` is added as a refinement for the foundation.

KDMBR$^{b}_{n}$ This technique is a hybrid approach combining the organization of the embedding space known from kd-trees as a foundation with the utilization of an MBR as a bounding volume for the refinement of each occupied cell of the kd-structure.

A kd-tree is a binary search tree where the underlying space recursively is partitioned on the basis of the value of just one attribute at each level [Sa05, p. 49]. Generally, in a kd-tree, data points are organized in buckets which are split when an overflow occurs. With regard to the resource descriptions, we will examine a global kd space partition (i.e. it is the same for all resources). Thus, for the KDMBR$^{b}_{n}$ method, training data is used to learn the global space decomposition. Similarly to UFS$_{n,cc}$, the quality of the space decomposition is dependent on the selection of appropriate training data, for which we randomly chose data points right

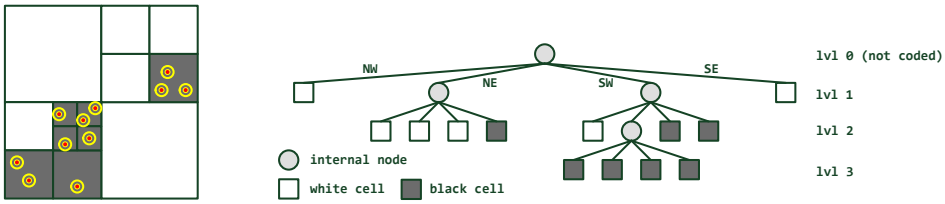

Fig. 2: Exemplary data point set depicted by a quadtree (left) and the matching tree structure (right).

out of the data collection, again. Since it is a bucket-based process, the universe initially consists of only one bucket. When a bucket-overflow occurs, the bucket is split into two. The split dimension is cyclically altered and the respective dimension is split into halves. After a split, the data insertion procedure is continued. The whole process continues until an amount of n buckets has been reached.

As a refinement, for each occupied cell of the kd space partition, the set of data points located in this cell additionally is bounded by a cell-interior MBR which is quantized for storage efficiency. This strategy has originally been introduced with the buddy-tree [SK90]. Specifically, the rectangle representing a cell of the kd-structure is called the *potential* data region. The MBR containing all of a region's data points is called the *actual* data region. It is conservatively approximated by a multidimensional interval which is quantized into a grid of $2^{b \cdot d}$ cells, which is invoked onto the potential data region and thus exploits the potential data region's presence. These *coded actual* data regions are slightly bigger than the actual data regions but save storage space in the description—both dependent on the parameter b, which specifies the amount of bits spent per bound in each dimension (d is the dimensionality of the space, thus 2 in our case).

In the resource description, binary information about the occupancy of the cells of the kd-structure is captured consecutively in a bit vector. If a cell is occupied with at least one data point, the $4 \cdot b$ bits for the corresponding quantized MBR immediately follow.

3.2 Quadtree-based Resource Description Techniques

The quadtree is a generic name for all kinds of trees that are built by recursive division of space [Oo99, p. 391]. Although the term quadtree usually refers to the two-dimensional variant (which divides the space into four quadrants, typically referred to as NW, NE, SW, and SE quadrant), the basic idea applies to an arbitrary number of dimensions d. Basically, there are two types of quadtrees [Sa05, p. 28]: The point quadtree, where the subdivision lines are based on the values of the data points; in contrast, the trie-based type forms a regular decomposition of the embedding space from which the data points are drawn into congruent regions of equal size. In the following, we are interested in the trie-based type (see Figure 2).

For a concrete utilization of quadtrees, we require some consideration concerning the decomposition of the space into subspaces and the quadtree encoding. We will present the quadtree as a solitary technique plus several methods which utilize quadtree structures in Hybrid Approaches.

Rules for the Space Decomposition In general, we are interested in binary quadtrees: the single quadtree regions (also called cells or leaves of the quadtree) are labelled black (white) if they contain at least one data object (no data object). For binary quadtrees and *region data*, the space decomposition is applied until each resulting cell is homogeneous, that is the cells are fully black or white [Sa05, p. 211]. Since we are concerned with *point data*, we need to adapt and must define a sensible stopping criterion for the decomposition. For this work, the stopping criterion is primarily storage-oriented and is applied when a certain amount of cells c is reached. Initially starting with one cell for the whole data space, the biggest black cell is equally divided into four (assuming this is the area which can be further delimited most). If there are several equally sized black cells, a random choice is made. After the divide, the resulting sibling nodes are respectively labelled black or white. The space decomposition continues until an amount of c cells has been reached, but ends prematurely if the area of all black cells falls below a certain threshold area a. This serves as a second, selectivity-oriented stopping criterion.

Quadtree Encoding For a memory efficient representation, the linear storage of quadtrees has been proposed, where a list of values stores the hierarchical tree structure [MRJ02, p. 516]. Encoding schemes can be distinguished whether the values are encoded in depth-first-order (df-order) or breadth-first-order (bf-order) and if only black nodes (= black leaves of the quadtree structure) or if all leaf nodes and internal nodes of the quadtree are encoded. In this work, we evaluate two different encoding schemes: the linear quadtree (LQ) [Ga82] and the CBLQ code [Li97].

The linear quadtree (**LQ code**) is a df-order, only black nodes encoding. A black node is identified by a unique key derived from its ordered list of ancestors. The key of successive digits represents the quadrant subdivision from which the black node originates according to a df-traversing (digits: 0 for NW, 1 for NE, 2 for SW, 3 for SE). Keys consist of up to l digits, where l is the number of levels or depth of the quadtree. If a black node is at level i ($i < l$), then only i digits are obtained, followed by a marker X (this is an adaption versus the LQ code described in [Ga82], where $l - i$ markers would follow). A condensation of the quadtree can be applied: if four cells have the same code except for the last digit, the last digit is replaced by marker X and the four cells are pooled into one (for example, 310-311-312-313 becomes 31X, dashes inserted for readabilty). Surplus markers X are stripped off of the LQ code. The quadtree in Figure 2 would be represented by 13X-210-211-212-213-22X-23X or 13-21-22-23 after condensation. Since there are five different literals to encode (0-3 and X), 3 bits are needed for the binary representation of a literal.

The **CBLQ code** is a bf-order, all leaves and internal nodes encoding. The authors claim that on average, the required storage space is only 22.2% of the LQ code. In the CBLQ code, each black (white) leaf is coded by 1 (0). The numeral 2 encodes an internal node if at least one of its descendants is an internal node. If all descendants are leaves, the node is encoded by 3. The root node is not encoded. A condensation of the quadtree can be applied, too, that is four black siblings are merged into their father node. Hence, the quadtree in Figure 2 would be represented by 0320-0001-0311-1111 or 0330-0001-0111 after condensation (dashes only included for readability). Since there are four different literals (0-3), 2 bits are needed to encode one literal.

3.3 Novel Quadtree-based Techniques

Based on these fundamentals, we will now describe the quadtree-based resource descriptions newly introduced in this paper.

$\mathbf{QT}_{c,a}$ The 'solitary' quadtree ($\mathrm{QT}_{c,a}$) is a method conducting hierarchical, resource-individual space decomposition. The quadtree for describing the resource's regions which contain data points is built according to the general rules just discussed, that is dependent on the parameters c (specifying the maximum number of quadtree regions) and a (specifying the lower-bound threshold area for quadtree regions to not be partitioned any further). Condensation is applied for the solitary quadtree.

For the resource descriptions, the quadtree information is captured in a bit vector for both encoding schemes (LQ code and CBLQ code).

The LQ scheme requires additional metadata besides the raw quadtree data to facilitate an unambiguous decoding: (1) The number of levels (#lvl) or depth of the specific quadtree must be captured, since in the LQ scheme, there is no marker to tag the end of a black node's encoding when it is at the deepest level of the quadtree. The theoretical maximum level of a quadtree with c cells is $\lfloor c/3 \rfloor = l$. Therefore, $\lceil log_2\, l \rceil$ bits are required to unambiguously encode the depth of a quadtree. (2) The number of occupied quadtree regions (#oqr) is captured right after the level of the quadtree. This information is not stringently required for the solitary quadtree, but for the hybrid methods utilizing a quadtree. See the $\mathrm{GridQT}_r^{c,a}$ paragraph for further explanation. Nevertheless, it is captured for the solitary quadtree, too, since it only requires $\lceil log_2\, c \rceil$ bits and allows for a unified encoding and decoding procedure. Afterwards, the raw quadtree data is captured in the bit vector for the LQ scheme.

For the CBLQ scheme, only the raw quadtree information (bf-order, all leafs and internal nodes encoding) is captured.

$\mathbf{GridQT}_r^{c,a}$ The $\mathrm{GridQT}_r^{c,a}$ method is a hybrid technique combining a uniform grid as a foundation with a grid-cell-interior quadtree as a refinement for occupied cells.

A (uniform) grid is a non-hierarchical structure [Sa05, p. 10]: The space is subdivided into equal sized grid cells. It is imposed onto the universe with r rows and $2 \cdot r$ columns. For each grid cell occupied with at least one data point, a cell-interior quadtree is built, dependent on the parameters c and a. For each quadtree, condensation is applied if possible.

In the bit vector for the resource description, binary information about the cell occupancy is consecutively captured. For an occupied cell, the quadtree information immediately follows. Beside the raw quadtree information, some additional information must be captured to unambiguously determine the tail of the quadtree data and the continuation of the grid cell occupancy information:

For the LQ code, the level/depth and the number of occupied quadtree regions (#oqr) of the cell-interior quadtrees must be captured. Again, the level or depth of the quadtree is encoded with $\lceil log_2\, l \rceil$ bits. The maximum number of occupied leaf nodes in a quadtree where condensation is applied is $c - 1$, (if all quadtree regions are occupied, the quadtree would

be condensed into its root node). Therefore, it requires $\lceil log_2\, c - 1 \rceil$ bits to unambiguously encode the actual number. Nevertheless, we utilize $\lceil log_2\, c \rceil$ bits to encode the actual number of occupied leaf nodes, since it is only a marginal disparity but allows for a unified encoding and decoding process with techniques where no condensation is applied (that is hybrid techniques that use a quadtree as a foundation).

For the CBLQ scheme, the number of inner nodes (#in) is captured, which is $\lfloor c/3 \rfloor = i$ at a max. Therefore, $\lceil log_2\, i \rceil$ bits are required for encoding the actual number of inner nodes.

KDQT$_n^{c,a}$ The KDQT$_n^{c,a}$ method is a hybrid technique organizing the embedding space with a kd space partitioning just as KDMBR$_n^b$ but utilizing a cell-internal quadtree for the refinement of the occupied cells of the kd-structure. Therefore, a performance comparison between these two methods evaluates which refinement is more efficient for kd-structures.

Analogous to GridQT$_r^{c,a}$, the cell-interior quadtrees are built dependent on the parameters c and a and possibly with condensation. Also, the resource descriptions are structurally identical to the GridQT$_r^{c,a}$ resource descriptions.

QTMBR$_{c,a}^b$ The QTMBR$_{c,a}^b$ method is a hybrid technique organizing the embedding space with a quadtree structure—which is individual for each resource and hence a local space partitioning—as a foundation and a quantized MBR as a bounding volume for the refinement of each occupied quadtree region.

The basic quadtree is built dependent on the parameters c and a. The quantized MBR for refining quadtree regions occupied with at least one data point is built just as for KDMBR$_n^b$, the parameter b controlling the accuracy and the storage requirements. Since occupied quadtree regions are further refined, *no* quadtree condensation is applied.

The resource descriptions require incorporating some meta data beside the raw quadtree and MBR data to unambiguously decode the bit vectors:

For the LQ code, the bit vector first captures the number of levels (#lvl) or depth of the subsequent quadtree. The maximum depth or number of levels of a non-condensed quadtree with c regions is $\lfloor c/3 \rfloor = l$. Therefore, $\lceil log_2\, l \rceil$ bits are required to encode the depth of the quadtree. Afterwards, the number of occupied quadtree regions (#oqr) is captured using $\lceil log_2\, c \rceil$ bits. The raw quadtree data succeeds. Finally, the MBR data for the occupied quadtree regions is consecutively encoded (see Figure 3).

For the CBLQ scheme, the number of inner nodes (#in) of the quadtree is required as meta data and occupies the first $\lceil log_2\, i \rceil$ bins of the bit vector. Afterwards, the raw quadtree data and MBR data for the occupied quadtree regions is captured consecutively.

MBRQT^{c,a} The hybrid methods presented so far utilize approaches organizing the embedding space as a foundation. Of course, it is also practicable to utilize (a) bounding volume(s) as a foundation. Methods organizing the embedding space first need to describe the whole universe U, partitioning it into regions (not) containing data points. By using (a) bounding volume(s) as a foundation, the space partitioning can be restricted to the 'regions of interest' (that is the regions delineated by the bounding volume(s)) and does not have to take regions

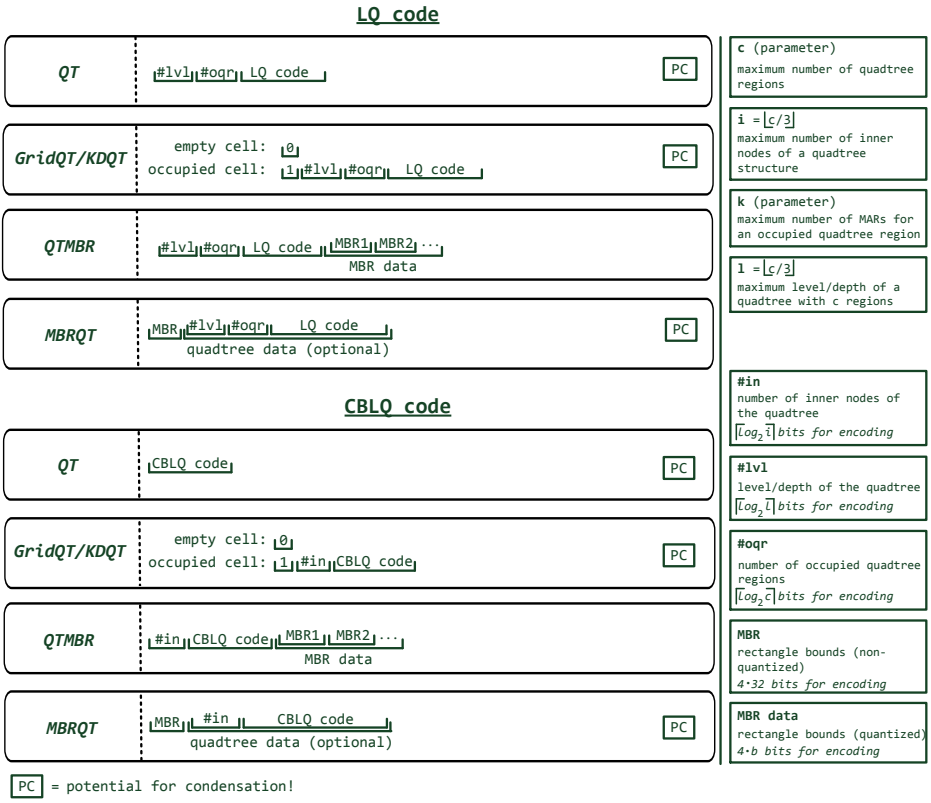

Fig. 3: Composition of the bit vectors for the different techniques and both encoding schemes.

not containing data points at all into account. The MBRQT^{c,a} method is a hybrid method combining a basic MBR as a foundation with an MBR-interior quadtree as a refinement.

First, the 'region of interest' is defined by determining the MBR of the resource's data points. The MBR-interior quadtree then is built dependent on the parameters c and a, only being bounded by the MBR and not by the universe U. Note that therefore, single quadtree regions might cross the Date Line, which is not the case for universe-related approaches such as QT$_{c,a}$ or hybrid methods using a quadtree as a foundation. The MBR-interior quadtrees can be condensed. In case the exterior MBR is a point or a very small rectangle with an area $\leq a$, no MBR-interior quadtree is built and the resource's data points are solely represented by the MBR.

The bit vector descriptions first capture the MBR bounds in single-precision floating-point format (that is $4 \cdot 32$ bits). In case the MBR area is bigger than a, they are followed by the data of the MBR-interior quadtree, which is encoded akin the cell-interior quadtrees for the GridQT$_r^{c,a}$ and KDQT$_n^{c,a}$ resource descriptions for both schemes—only missing the leading 1 for indicating an occupied cell which is unnecessary for MBRQT^{c,a}.

4 Resource Selection

With the resource descriptions presented in section 3, the problem of representing the 'content' of a resource has been elaborated. The next task is the resource selection problem.

The resource selection process has to be oriented towards the query type supported by the search system. For our search scenario and our data collection (see section 5.1), we assume precise kNN queries, i.e. the actual k nearest neighbors within the resource network shall be retrieved for a query point. This kind of resource selection process involves a ranking of the resources followed by requesting the closest data points from the resources while pruning irrelevant resources until the k nearest neighbors have been unambiguously determined.

Resource Ranking Aside from $\text{UFS}_{n,cc}$, all other resource description methods, in the very end, depict one or several rectangular areas to describe the data point cloud of a resource. Therefore, for all of them, the same ranking algorithm is applied, which utilizes the lower-bound distances from the query point to the rectangular areas. It is verbally described in the following.

> At first, the representation of each resource is constructed as a list of rectangular areas, which are built from the resource description. These representations can be cached to avoid having to rebuild them for each query. Afterwards, for each rectangular area, an `R-Entry` is built. An `R-Entry` captures the minimum distance of the rectangular area from the query point and the size of the surface area of the rectangle. The `R-Entries` of a resource are ascendingly sorted by (a) minimum distance from the query point and (b) minimum surface area (in case of equal distances). With each resource being represented by their sorted list of `R-Entries`, the actual resource ranking commences.
> A total ordering of the resources is defined by a pairwise comparison of two resources r_a and r_b, each represented by their list of `R-Entries`. The smaller list of `R-Entries` is filled with dummy entries which are most unfavorable for the ranking to avoid having unequal list sizes. To decide on the ranking between r_a and r_b, the respective `R-Entries` r_{a_i} and r_{b_i} at the same list index i are compared one after another, starting with $i = 0$. If the minimum distance of r_{a_i} is smaller than the minimum distance of r_{b_i}, r_a is ranked higher. In case of equal minimum distances, r_a is ranked higher if the surface area of r_{a_i} is smaller than the surface area of r_{b_i}. If no decision can be made by comparing the `R-Entries` at index i, i is incremented and the `R-Entries` at the next index position are compared alike. If the comparison of the entire lists of `R-Entries` does not lead to a decision, r_a is ranked higher than r_b if r_a is 'bigger' than r_b, that is administers more data points. If r_a and r_b are of the same size, a random ranking decision is made.

For $\text{UFS}_{n,cc}$, the ranking algorithm basically is the same aside from being based on the polygonial areas of the Voronoi cells instead of rectangular areas. Therefore, the Voronoi diagram has to be constructed explicitely from the sites. Only the cc closest Voronoi cells (with respect to the query point) are considered for the ranking.

The precise kNN algorithm The next step in the resource selection process is to efficiently determine the actual nearest neighbors in the resource network based on the resource ranking. In the following, we will verbally describe an algorithm for a precise kNN search, which is

implemented as an iterative range query where the query radius decreases every round. The parallelism of the search scenario is exploited by contacting n_{rp} resources per round.

At start, the query radius r is set to infinity and the `topk[]` array of the (current) k nearest neighbors is empty. First, the resources are ranked by the respective ranking algorithm for the different resource descriptions methods described above. The sorted list L_r determines the order in which the resources should be queried for their data points. The list L_r is processed until all resources from the list have been queried or successfully have been pruned, and thus the list is empty.

In each round, the resource descriptions of the next n_{rp} resources of the list L_r are examined. When the resource res cannot be pruned, the top k data points of res are requested and the global `topk[]` array is updated by replacing the (with respect to the query point q) most distant data points in `topk[]` with closer data points of the local result array of res, and finally ascendingly sorting `topk[]` by the distance from the query point again. The pruning of resources is based on lower-bound distances provided by the resource description information. If the lower-bound distance of a resource's description from the query point q is greater than the current query radius r, the resource can be safely pruned from search. For $\text{UFS}_{n,cc}$, the Voronoi cells are reconstructed in order to check which cells overlap the 'query ball' (and thus may contain relevant data points). After querying or pruning, the n_{rp} resources examined are removed from L_r and the query radius r is set to the distance of the current kth nearest neighbor for the next round. When the algorithm ends, the k nearest neighbors unambiguously have been determined. Note that after the initial ranking, no re-ranking of resources is needed.

5 Evaluation

In the evaluation, the main focus will be on two criteria: (1) The *resource description sizes* (rds). Small rds are preferable since the amount of data to be transferred in the network and the general storage space requirements are reduced. We will concentrate on the resource description sizes averaged over the set of resources. (2) The *resource fraction contacted* (rfc) in order to retrieve the top k data points in the resource network. The fewer resources need to be queried for their data points, the better the resource descriptions.

Obviously, the optimization of the two criteria is conflicting: the more storage space is spent, the more accurate the resource descriptions can be and fewer resources typically need to be queried; analogous the reverse. Varying the parameterization of the different techniques influences the accuracy of the resource descriptions and hence the rds and the rfc values. We apply the Skyline operator [BKS01] to filter out the 'interesting' parameterizations for a specific technique. Considering the measurements for the different techniques and their parameterizations as sets of two-dimensional points, with rds being the x-dimension and rfc being the y-dimension, a point is 'interesting' if it is not dominated by any other point. A point dominates another point if it is as good or better in all dimensions and better in at least one dimension [BKS01, p.1]. Thus, for a specific technique, a point p_x representing the parameterization x dominates a point p_y representing a parameterization y if ($p_x.rfc \leq p_y.rfc$ and $p_x.rds < p_y.rds$) or ($p_x.rfc < p_y.rfc$ and $p_x.rds \leq p_y.rds$).

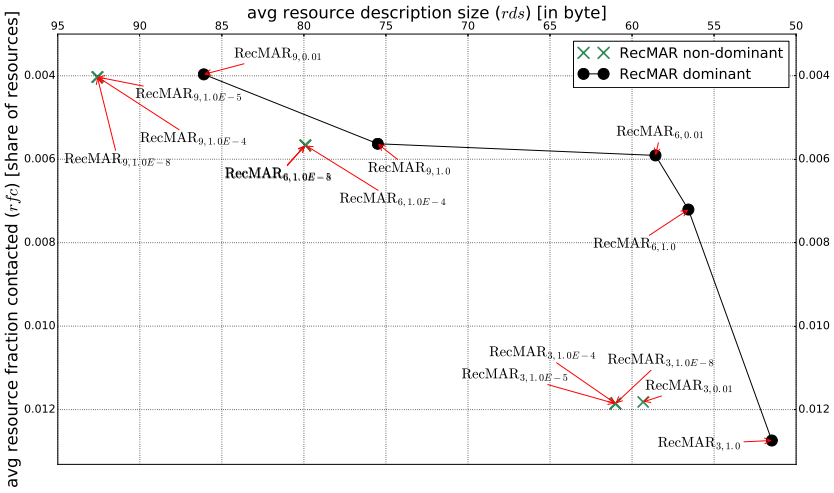

Fig. 4: Exemplary Skyline determination for RecMAR$_{k,sl}$. The black dots connected by the black lines forge the Skyline of the dominant parameterizations for RecMAR$_{k,sl}$.

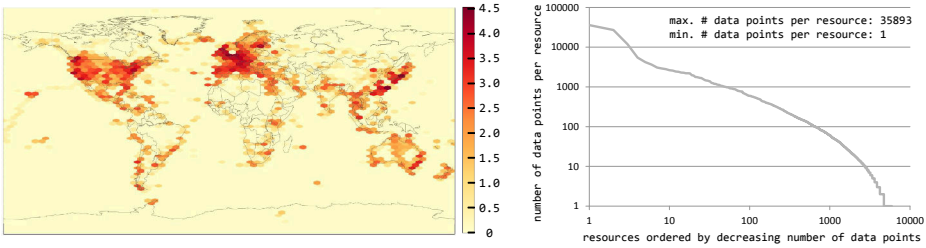

Fig. 5: Distribution of the data points in the data space (left) and number of data points per resource (right). Note that the values on the left legend are $log_{10}(x+1)$ scaled, so the number of data points per bin is calcluated by $x = 10^n - 1$. For example, $n = 4.0$ results in $x = 9,999$.

Figure 4 illustrates the Skyline determination for the RecMAR$_{k,sl}$ technique. Note that both axes are inverted, i.e. the more northeast-bound a point, the more dominant it is. The Skylines determined for the respective techniques will be used for inter-technique comparisons.

5.1 Experimental Setup

The data collection is based on 406,450 geo-referenced images uploaded to Flickr by 5,951 different users. The images were crawled in 2008. The corresponding spatial data points are assigned to resources via user ID. Hence, it is assumed that every user in the network operates an own resource for spatial data. The spatial distribution of the data points and the distribution of the amount of data points maintained by the resources are depicted in Figure 5. The spatial distribution is very patchy and the distribution of data points to resources is skewed: few resources maintain a lot of data points and a lot of resources maintain only a few data points. The skewed distribution is typical for Peer-to-Peer (P2P) data [Cu03, p. 5].

MBR		no parameters
$RecMAR_{k,sl}$	k	3|6|9
	sl	1.0|0.01|1.0E-4|1.0E-5|1.0E-8
$UFS_{n,cc}$	n	512|2048|8192
	cc	16|64|256|n
$KDMBR^{b}_{n}$	n	512|2048|8192
	b	3|4|6|8
$QT_{c,a}$	c	256|512|1024|2048|8192
	a	0.001|1.0E-5|1.0E-7|1.0E-8
$GridQT^{c,a}_{r}$	r	16|32|64
	c	16|32|64|256|512
	a	0.001|1.0E-5|1.0E-7|1.0E-8
$KDQT^{c,a}_{n}$	n	512|2048|8192
	c	16|32|64|256|512
	a	0.001|1.0E-5|1.0E-7|1.0E-8
$QTMBR^{b}_{c,a}$	c	64|256|512|1024
	a	0.1|0.05|0.01|0.001
	b	3|4|6|8
$MBRQT^{c,a}$	c	16|64|256|512|1024|2048|8192
	a	0.001|1.0E-5|1.0E-7|1.0E-8

a = stopping area

b = # of bits

c = # of cells

cc = # of centroids considered

k = # of MARs

n = number of subspaces

r = # of rows

sl = threshold distance rectangle center vs. associated data points

Note that for the hybrid techniques, the parameters of the foundation are always subscript whereas the parameters for the refinement are always superscript.

Fig. 6: Listing of the tested parameter variations for the different techniques.

For the evaluation, we assume a kNN search for the k nearest spatial data points—associated with media content such as images—with respect to a spatial query location (or query point) in a resource network. It is assumed that every resource knows the resource description of every other resource in the network. The query points are determined as follows: At first, a random resource is selected; then, a random data point from the resource is chosen as a query point. This simulates that all resources—irrespective of the size—have the same probability to issue a query. For the 50 selected query points, 7.74 resources contribute to the top 50 data points on average. This results in an rfc of $7.74/5,951 = 0.0013$ for an ideal resource selection technique which only contacts relevant resources for their data.

The parameters of the techniques are varied in certain windows which seem suitable in terms of run time for the summary computation and general storage space requirements (see Figure 6). Also, the GPS accuracy is taken into account for the quadtree-based techniques so that the stopping area parameter matches the GPS accuracy limits on the highest parameterization. All possible parameter combinations are tested.

Some resource description techniques rely on randomness; for example $UFS_{n,cc}$ (selection of the sites) or the quadtree-based techniques (selection of the next quadtree region to split when there are several equally sized black regions). For these techniques, we conducted four runs with different seeds to minimize the effect of outliers for each parameterization (using the same set of 50 query points in each run).

5.2 Optimizations

To augment the capability of the search system, we conduct several optimizations which affect the space efficiency of the resource descriptions.

The resource descriptions must be serialized to be distributed in the network. This requires 27 byte serialization overhead, which are included in the measurements (see section 5.3). We applied two optimizations to reduce the storage requirements of the resource descriptions:

Since all resource descriptions in the very end are encoded as bit vectors, we attempt to compress them by using `Javas`'s `gzip` implementation with default parameters. In case the compression turns out to result in a reduced amount of data, the compressed bit vector will be distributed in the network.

For 'small' resources—and depending on the technique and its parameterization—the resource descriptions might need more space than transferring the few data points directly. A direct transfer of course is also more precise, since the summarizations of the resource data only describe approximated areas containing data points. Therefore, in case the size of the resource descriptions equals or exceeds the storage space needed to encode the data points themselves, the resource 'directly' is represented by the data points it maintains rather than by an approximated description of the areas where the data points are located.

All in all, for non-quadtree-based methods, there are four ways of transferring the data representing the resource (the *resource description*), depending on if the resource is described by approximating areas (the resource *summary*) or directly by its data points (the *direct representation*), and whether the data is compressed or not. For quadtree-based *summaries*, it additionally has to be differentiated if the quadtrees are encoded by the LQ scheme or the CBLQ scheme, resulting in six different resource description types. The information on the resource description type has to be incorporated into the resource description. For a unified encoding scheme, we utilize 3 bits for all resource descriptions. Additionally, the resource sizes which are used in the ranking algorithms are encoded with 5 bit, overall resulting in 1 extra byte in addition to the serialization and the resource description data itself.

Note that the resource ranking is not affected by the compression but by the direct transfer: resources, for which the data points directly have been transferred, are not ranked since the exact locations of their data points already are known. The possibility of transferring the data points itself rather than a summary also has consequences for the query processing and the determination of the rfc values. The query is processed in a two-step approach:

In the first step, the coordinates of the top k closest data points w.r.t. the query point are determined. The coordinates of the data points which have been directly transferred are already known (therefore, the corresponding resources do not have to be contacted in this step); for the coordinates of the remaining data points (approximated by the summaries), the respective resources need to be contacted.

In the second step, all resources contributing to the top k data points are contacted for the media data associated with these data points.

Hence, the rfc value arises as result from: $(res_{sum} + res_{dt})/res$, with res_{sum} being the number of ranked resources (i.e. resources which transferred their summaries as resource descriptions) contacted while the kNN algorithm is applied, res_{dt} being the number of resources which directly transferred their data points and contribute to the top k result and res being the cardinality of the resource set.

5.3 Experimental Results

The listed baseline in Figure 7 is the minimum number of resources that need to be contacted. The results show that it is possible to eminently surpass the selectivity of an MBR while expending only little extra storage. With only 1.7 byte additional storage on average (rds value), $\text{QTMBR}^3_{64,0.1}$ achieves a more than five times better selectivity (lower rfc value) (MBR: rds = 42.34 byte, rfc = 0.0471; $\text{QTMBR}^3_{64,0.1}$: rds = 44.04 byte, rfc = 0.0087).

The uniform grid obviously is not a suitable foundation, since it is the worst hybrid technique by quite a margin. Even $\text{MBRQT}^{c,a}$, despite being based on the subpar MBR as foundation, completely dominates $\text{GridQT}^{c,a}_r$ by its Skyline. For the Space Partitioning Approaches, $\text{UFS}_{n,cc}$ outperforms the solitary $\text{QT}_{c,a}$. Comparing the Skylines, the gap seems to grow with increasing rds. Considering the additional complexity (the need of a selection mechanism for the sites and their deployment within the network) plus the considerably lower entropy of the $\text{UFS}_{n,cc}$ summaries due to longer zero sequences in the bit vectors (meaning the `gzip` entropy compression works better since quadtree data generally is of high entropy), the results for $\text{QT}_{c,a}$ are still decent in comparison, though.

The assessment of the results for $\text{QTMBR}^b_{c,a}$ (all necessary information already contained in the resource descriptions) versus state-of-the-art KDMBR^b_n (the fundamental space partition needs to be spread separately) is similar. For rds values of about 55 byte, $\text{QTMBR}^b_{c,a}$ even is superior to KDMBR^b_n. Assuming the typical 'knee-like' pathway of the Skylines, this would also hold for smaller rds values, but cannot be confirmed due to the delayed advent of the KDMBR^b_n Skyline. The KDMBR^b_n Skyline completely dominates the $\text{KDQT}^{c,a}_n$ Skyline. Therefore, in case the foundation is already decently accurate, quantized MBRs are a more efficient means of refinement compared to internal quadtrees. For coarse foundations like the uniform grid, we expect internal quadtrees to be more suitable. $\text{QTMBR}^b_{c,a}$ almost completely dominates its 'mirror-technique' $\text{MBRQT}^{c,a}$—especially for low rds—on a large margin. Combining (adaptive) space partitioning with quantized bounding volumes seems to be more promising than utilizing full accuracy bounding volumes, be it hybrid or solitary. This also backed by $\text{RecMAR}_{k,sl}$ being clearly distanced by the other techniques from about 62 bytes rds on, only dominating $\text{GridQT}^{c,a}_r$ before. Generally, at the latest from about 90 bytes rds on, only marginal differences between the different techniques can be observed (all of them conducting an asymptotic approximation towards the rfc baseline).

For the data transfer option chosen, the share of directly transferred data points as well as the share of zipped summaries increases with increasing rds values, which was to be expected. In Table 1, for each technique, we display the margin in which different key figures vary for the Skyline-forming parameterizations. Hybrid techniques using global space partitioning as a foundation and also $\text{UFS}_{n,cc}$ benefit a lot from compression, their share of zipped summaries generally is high. The foundation seems to be decisive regarding the entropy, since the share of zipped summaries is high even when internal quadtrees are utilized as a refinement. In contrast, hybrid techniques using quadtrees as a foundation as well as the solitary $\text{QT}_{c,a}$ yield a significantly lower share of zipped summaries. Thus, both overhead for distributing the information about the global space partitioning as well as for (de-)compression can be avoided when using these techniques. In general, the share of

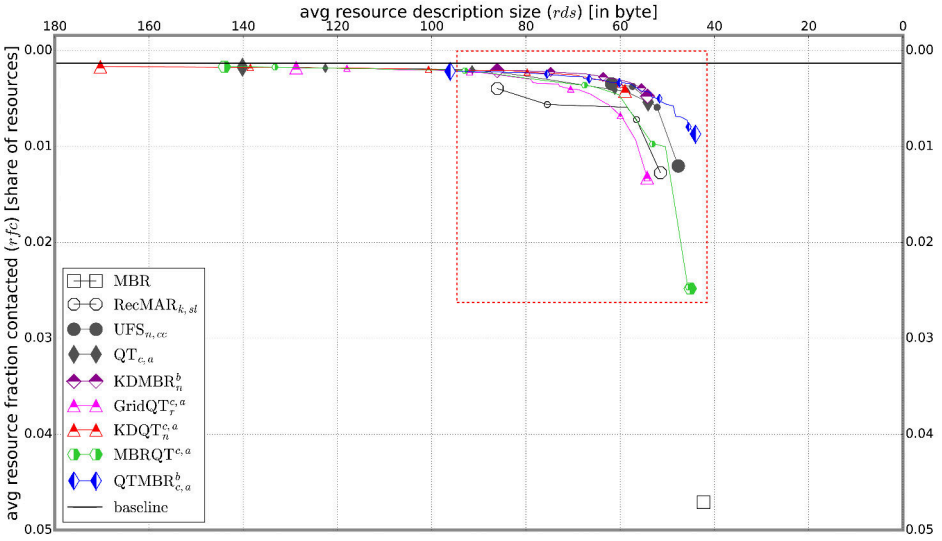

Fig. 7: Overview of the Skyline comparison for the different techniques. Markers on the Skylines are sampled, i.e. not all rds/rfc data points which constitute a Skyline are depicted by a marker.

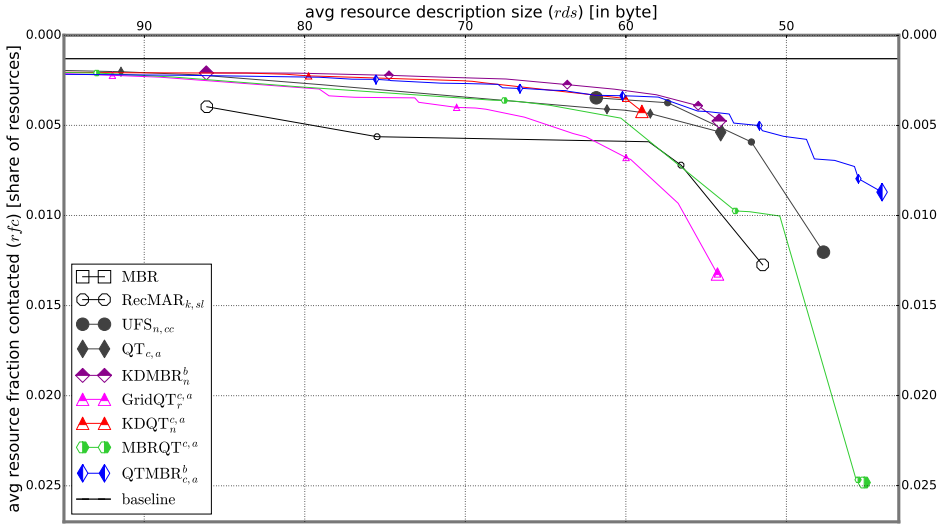

Fig. 8: Detailled view on the experimental results (magnification of the red rectangle in Figure 7).

CBLQ-coded quadtrees is higher the more the parameterization fosters the formation of large quadtrees. Thus, the LQ scheme is more effective for encoding small quadtrees but becomes inefficient for larger quadtrees, which is intuitive since the LQ scheme encodes the path for every black node separately (which imposes redundancy for larger quadtrees).

Tab. 1: Spans in which several key figures vary for the parameterizations forming the Skyline of the respective techniques. Remember that for techniques not utilizing quadtrees, there is no choice between the LQ scheme and the CBLQ scheme for the summary. Thus, the only option is to (not) zip them; the respective shares are displayed in the `lqz/sumz` or `lqnz/sumnz` rows. `lq(n)z`: (non-)zipped LQ scheme summaries, `sum(n)z`: (non-)zipped 'non-quadtree' summaries, `dt(n)z`: (non-)zipped directly transferred data points, `cblq(n)z`: (non-)zipped CBLQ scheme summaries.

technique → key figures ↓	MBR	RecMAR$_{k,sl}$	UFS$_{n,cc}$	KDMBR$_n^b$	QT$_{c,a}$	GridQT$_r^{c,a}$	KDQT$_n^{c,a}$	QTMBR$_{c,a}^b$	MBRQT^{c,a}
lqz/ sumz	0%	0.05% - 15.2%	47.0% - 63.5%	48.5% - 59.4%	0% - 0.48%	12.5% - 55.5%	18.9% - 52.5%	0%	0% - 1.1%
lqnz/ sumnz	70.7%	40.1% - 67.1%	0.26% - 23.2%	1.0% - 18.6%	26.2% - 69.6%	0% - 2.3%	0.48% - 16.1%	42.3% - 82.7%	38.2% - 51.4%
dtz	0%	0.05% - 0.72%	0% - 1.4%	0.02% - 2.1%	0% - 0.15%	0.03% - 8.1%	0.03% - 7.0%	0% - 0.03%	0% - 0.18%
dtnz	29.2%	32.8% - 44.0%	29.8% - 37.8%	30.6% - 41.3%	0% - 40.9%	35.5% - 52.7%	32.3% - 51.5%	0% - 29.6%	29.2% - 41.1%
cblqz	-	-	-	-	0% - 25.1%	8.2% - 27.3%	5.8% - 18.4%	0% - 1.0%	0% - 15.8%
cblqnz	-	-	-	-	7.6% - 30.4%	0% - 0.75%	0.07% - 3.8%	17.3% - 28.8%	3.8% - 22.6%
avg. res. desc. size (in byte)	42.3	51.5 - 86.1	47.7 - 61.8	54.2 - 86.1	54.1 - 140.2	54.3 - 128.8	59.0 - 170.3	44.0 - 96.1	45.2 - 144.0
max res. desc. size (in byte)	44	76 - 172	90.25 - 307	202 - 2299.5	113 - 1983.5	216.75 - 5439.5	369 - 9934.5	82.75 - 2079	50 - 2071.25

The RecMAR$_{k,sl}$ Skyline is built by parameterizations of $sl = 1$ or 0.01. Apparently, dividing the point sets into very narrow groups and allocating a full precision rectangle to bound each of them is not worthwhile. The dominant UFS$_{n,cc}$ parameterizations all consider a small amount of cells in the ranking process ($cc = 16$). The cell occupancy of distant cells thus is negligible concerning the relevance of a resource for the 'region of interest', the size of the resources proves more relevant (low cc values benefit bigger resources, see section 4). For KDMBR$_n^b$, almost all parameterizations are located on the Skyline, solely the use of $b = 3$ for the quantized rectangles is mostly not sufficient. The Skyline of GridQT$_r^{c,a}$ is composed of few parameterizations with $r = 64$. Hence, also for an intra-GridQT$_r^{c,a}$ consideration, the adaptivity of the space partitioning should be added in time. For KDQT$_n^{c,a}$, the share of non-zipped summaries is 15% at least when $n = 512$. For more cells, the shares rapidly drop. The maximum rds is much bigger compared to GridQT$_r^{c,a}$. This is because the adaptivity will result in very many cells being occupied in the extreme case—with each cell featuring an internal quadtree for refinement. The Skyline of MBRQT^{c,a} misses a lot of parameterizations with 'middle-sized' internal quadtrees ($c = 256$ to 1024); hence, the use of a small or a large amount of cells seems to be way to go.

Small quadtrees can be encoded very efficiently: for $QT_{256,0.001}$, not a single resource directly transfers its data points (69.6% LQ encoded, 30.4% CBLQ encoded). Generally, for $QT_{c,a}$, small values for the stopping area a only become efficient with a high value for c; for quadtrees with a low amount of cells (small c value), a bigger stopping area is more suitable. Only 4 out of 34 parameterizations on the $QTMBR^b_{c,a}$ Skyline utilize $b = 8$; the use of $b = 4$ or $b = 6$ seems to be most suitable for $QTMBR^b_{c,a}$. $QTMBR^b_{c,a}$ displays the lowest share of zipped summaries and therefore benefits least from compression. This is most remarkable, since $QTMBR^b_{c,a}$, even for bigger rds, is very competitive to techniques strongly benefitting from compression.

6 Related Work

The techniques presented in this paper are based on or strongly related to multidimensional access methods supporting search operations in centralized databases. Within these, it is distinguished between Point Access Methods (PAMs), for searching sets of points in two or more dimensions, and Spatial Access Methods (SAMs), which handle spatially extended objects. See [Sa05] and [GG98] for an extensive overview on these topics. Both PAMs and SAMs are applied in rather low-dimensional data spaces which are coordinate-based. For high-dimensional data spaces not based on coordinate systems, Metric Access Methods (MAMs) have been developed. See [He09] for an extensive tutorial on MAMs. Hierarchical data structures akin the quadtree are used in numerous application fields. See [Sa84] for a fundamental survey. Manouvrier et al. illustrate several possibilities for the linear storage of quadtrees in [MRJ02].

7 Conclusion

Hybrid Approaches for spatial resource description show higher potential for describing the geospatial footprint of resources compared to solitary techniques (both Geometric Approaches and Space Partitioning Approaches)—a suitable selection of techniques to be combined presumed. Within these, quadtree-based techniques are very competitive and—in particular a combination of a quadtree and quantized cell-interior MBRs ($QTMBR^b_{c,a}$)—provide similar performance to the state-of-the-art ($KDMBR^b_n$). Furthermore, they simultaneously encode all necessary information within the resource descriptions, therefore superseding the need to separately sample information about the data collection as a whole plus processing the collected data and distributing the result in the network afterwards, considerably reducing the complexity of the search system.

References

[Be92] Becker, B.; Franciosa, P. G.; Gschwind, S.; Ohler, T.; Thiemt, G.; Widmayer, P.: Enclosing many boxes by an optimal pair of boxes. In: *Proc. of STACS 92: 9th Ann. Symp. on Theor. Aspects of Comp. Sc. Cachan, France*. Springer Berlin Heidelberg, pp. 475–486, 1992.

[BH12] Blank, D.; Henrich, A.: Describing and Selecting Collections of Georeferenced Media Items in Peer-to-Peer Information Retrieval Systems. In: *Discovery of Geospatial Resources: Methodologies, Technologies, and Emergent Applications*. Information Science Reference, pp. 1–20, 2012.

[BHK16] Blank, D.; Henrich, A.; Kufer, S.: Using Summaries to Search and Visualize Distributed Resources Addressing Spatial and Multimedia Features. Datenbank-Spektrum 16/1, pp. 67–76, 2016.

[BKS01] Börzsönyi, S.; Kossmann, D.; Stocker, K.: The Skyline Operator. In: *Proc. of the 17th Int. Conf. on Data Engineering*. IEEE Computer Society, Washington, DC, USA, pp. 421–430, 2001.

[Ca00] Callan, J.: Distributed Information Retrieval. In: *Advances in Information Retrieval*. Kluwer Academic Publishers, pp. 127–150, 2000.

[Ca05] Caldwell, D. R.: Unlocking the Mysteries of the Bounding Box. A/2, pp. 1–20, Aug. 2005.

[Cu03] Cuenca-Acuna, F. M.; Peery, C.; Martin, R. P.; Nguyen, T. D.: PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In: *12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12 '03)*. IEEE Press, Seattle, Washington, pp. 1–11, 2003.

[Ga82] Gargantini, I.: An Effective Way to Represent Quadtrees. *Commun. ACM* 25/12, pp. 905–910, Dec. 1982.

[GG98] Gaede, V.; Günther, O.: Multidimensional Access Methods. *ACM Comput. Surv.* 30/2, pp. 170–231, 1998.

[HB10] Henrich, A.; Blank, D.: Description and Selection of Media Archives for Geographic Nearest Neighbor Queries in P2P Networks, 2010.

[He09] Hetland, M. L.: The Basic Principles of Metric Indexing. In: *Swarm Intelligence for Multi-objective Problems in Data Mining*. Springer Berlin Heidelberg, pp. 199–232, 2009.

[KBH12] Kufer, S.; Blank, D.; Henrich, A.: Techniken der Ressourcenbeschreibung und -auswahl für das geographische Information Retrieval. In: *Proc. of the IR Workshop at LWA 2012*. Dortmund, Germany, pp. 1–8, 2012.

[KBH13] Kufer, S.; Blank, D.; Henrich, A.: Using Hybrid Techniques for Resource Description and Selection in the Context of Distributed Geographic Information Retrieval. In: *Advances in Spatial and Temporal Databases: 13th Intl. Symp., SSTD 2013, Munich, Germany*. Springer Berlin Heidelberg, pp. 330–347, 2013.

[KH14] Kufer, S.; Henrich, A.: Hybrid Quantized Resource Descriptions for Geospatial Source Selection. In: *Proc. of the 4th Int. Workshop on Location and the Web*. LocWeb '14, ACM, Shanghai, China, pp. 17–24, 2014.

[Li97] Lin, T.-W.: Set Operations on Constant Bit-length Linear Quadtrees. *Pattern Recogn.* 30/7, pp. 1239–1249, July 1997.

[MRJ02] Manouvrier, M.; Rukoz, M.; Jomier, G.: Quadtree representations for storage and manipulation of clusters of images. *Im. Vis. Comp.* 20/7, pp. 513–527, 2002.

[Oo99] Oosterom, P. V.: Spatial Access Methods. In. Vol. 1, *Geographical Information Systems*, chap. 27, pp. 385–400, 1999.

[Sa05] Samet, H.: Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[Sa84] Samet, H.: The Quadtree and Related Hierarchical Data Structures. *ACM Comput. Surv.* 16/2, pp. 187–260, June 1984.

[SK90] Seeger, B.; Kriegel, H.-P.: The Buddy Tree: An Efficient and Robust Access Method for Spatial Data Base. In: *Proc. of the Sixteenth Intl. Conf. on VLDB*. Morgan Kaufmann Publishers Inc., Brisbane, Australia, pp. 590–601, 1990.