QDR-Tree: An Efficient Index Scheme
for Complex Spatial Keyword Query

Xinshi Zang, Peiwen Hao, Xiaofeng Gao[✉], Bin Yao, and Guihai Chen

Shanghai Key Laboratory of Scalable Computing and Systems,
Department of Computer Science and Engineering, Shanghai Jiao Tong University,
Shanghai 200240, China
{fei125,williamhao}@sjtu.edu.cn, {gao-xf,yaobin,gchen}@cs.sjtu.edu.cn

Abstract. With the popularity of mobile devices and the development of geo-positioning technology, location-based services (LBS) attract much attention and top-k spatial keyword queries become increasingly complex.It is common to see that clients issue a query to find a restaurant serving pizza and steak, low in price and noise level particularly.However, most of prior works focused only on the spatial keyword while ignoring these independent numerical attributes.

In this paper we demonstrate, for the first time, the *Attributes-Aware Spatial Keyword Query* (ASKQ), and devise a two-layer hybrid index structure called *Quad-cluster Dual-filtering R-Tree* (QDR-Tree). In the keyword cluster layer, a Quad-Cluster Tree (QC-Tree) is built based on the hierarchical clustering algorithm using kernel k-means to classify keywords.In the spatial layer, for each leaf node of the QC-Tree, we attach a Dual-Filtering R-Tree (DR-Tree) with two filtering algorithms, namely, keyword bitmap-based and attributes skyline-based filtering. Accordingly, efficient query processing algorithms are proposed.

Through theoretical analysis, we have verified the optimization both in processing time and space consumption. Finally, massive experiments with real-data demonstrate the efficiency and effectiveness of QDR-Tree.

Keywords: Top-k spatial keyword query · Skyline algorithm
Keyword cluster · Location-based service

1 Introduction

With the growing popularity of mobile devices and the advance in geo-positioning technology, location-based services (LBS) are widely used and spatial keyword

query becomes increasingly complex. Clients may have special requests on numerical attributes, such as price, in addition to the location and keywords.

Example 1. Consider some spatial objects in Fig. 1(a), where dots represent spatial objects such as restaurants, whose keywords and three numerical attributes are listed in Fig. 1(b). Dots with the same color own similar keywords, e.g., red dots share keywords about food. The triangle represents a user issuing a query to find a nearest restaurant serving pizza and steak with low level in price, noise, and congestion. At a first glance, o_8 seems to be the best choice for the close range, while o_1 surpasses o_8 in the numerical attributes obviously. This common situation shows that such complex queries deserve careful treatment.

(a) Geo-position of spatial objects

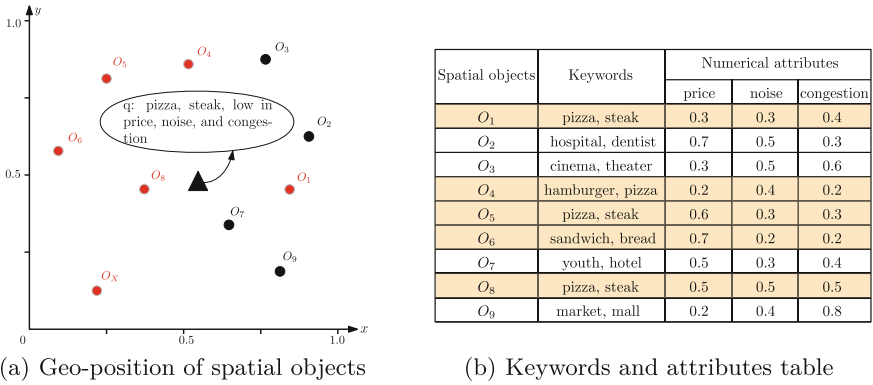| Spatial objects	Keywords	Numerical attributes		
		price	noise	congestion
O_1	pizza, steak	0.3	0.3	0.4
O_2	hospital, dentist	0.7	0.5	0.3
O_3	cinema, theater	0.3	0.5	0.6
O_4	hamburger, pizza	0.2	0.4	0.2
O_5	pizza, steak	0.6	0.3	0.3
O_6	sandwich, bread	0.7	0.2	0.2
O_7	youth, hotel	0.5	0.3	0.4
O_8	pizza, steak	0.5	0.5	0.5
O_9	market, mall	0.2	0.4	0.8

(b) Keywords and attributes table

Fig. 1. A set of spatial objects and a query (Color figure online)

Extensive efforts have been made to support spatial keyword query. However, prior works [7,9,15] mainly focused on the keywords of spatial objects but neglected or failed to distinguish independent numerical attributes. Recently, Sasaki [16] schemed out SKY R-Tree which incorporates R-tree with skyline algorithm to deal with the numerical attributes. However, it does not work well for multi keywords, which reduces their usage for various applications. Liu [10] proposed a hybrid index structure called Inverted R-tree with Synopses tree (IRS), which can search many different types of numerical attributes simultaneously. However, the IRS-based search algorithm requires providing exact ranges of attributes which is a heavy and unnecessary burden to the users. What's more, the exact match in in attributes can also lead to few or no query results to be returned.

Correspondingly, in this paper, we named and studied, for the first time, the *attributes-aware spatial keyword query* (ASKQ). This complex query needs to take location proximity, keywords' similarity, and the value of numerical attributes into consideration, that is respectively, the Euclidean spatial distance, the relevance of different keywords, and the integrated attributes of users' preference. Obviously the ASKQ has wide apps in the real world.

Tackling with the ASKQ in Example 1, common search algorithms [7,9,15] ignoring numerical attributes may retrieve finally o_1, o_5, o_8 indiscriminately, and SKY R-Tree-based algorithm may return o_4 as one of results, and IRS-Tree-based algorithm may retrieve no objects when the query predicate is set as "*price* < 0.3 & *noise* < 0.3 & *congestion* < 0.4". Apparently, none of these algorithms can satisfy the users' need. These gaps motivate us to investigate new approaches that can deal with the ASKQ efficiently.

In this paper, we propose a novel two-layer index structure called *Quad-cluster Dual-filter R-Tree* (QDR-Tree) with query processing algorithms. In the first layer we deal with keyword specifically. Considering numbers of keywords share the similar semanteme and clients tend to query objects of the same class, we cluster and store the keywords in a Quad-Cluster Tree (QC-Tree) by hierarchical clustering algorithm using kernel k-means clustering [6]. With keyword relaxation operation and Cut-line theorem to avoid redundance, QC-Tree can balance search time and space cost well.

In the second layer we deal with spatial objects with numerical attributes. At each leaf node of the first layer, a Dual-filter R-Tree (DR-Tree) is attached according to two filtering algorithms, namely, keyword bitmap-based filtering and attributes skyline-based filtering, which effectively reduce the false positives.

Moreover, we also propose a novel method to measure the relevance of one spatial object with the query keywords. We measure the similarity of different keywords from both textual and semantic aspects. For the latter one, the *term vectors* that are obtained by word2vec [12] are applied to represent every keywords, and therefore, the similarity can be quantified. Note that both queries and spatial objects usually own several keywords, a bitmap of keywords is used to measure the relevance between two lists of keywords lightly and efficiently.

Table 1 compares the current index with QDR-Tree in three aspects. Apparently, QDR-Tree outperform existing methods in tackling with the ASKQ, and can achieve great improvements in query processing time and space consumption. This will be demonstrated in both theoretical and experimental analysis. Massive experiments with real-data also confirm the efficiency of QDR-Tree.

Table 1. Comparisons among current indexes and QDR-tree

Index	From	Location proximity	Muti-keywords	Fuzzy attributes
IR-Tree	TKDE (2011) [9]	✓	✓	✗
IL-Quadtree	ICDE (2013) [18]	✓	✓	✗
SKY R-Tree	DASFAA (2014) [16]	✓	✗	✓
IRS-Tree	TKDE (2015) [10]	✓	✓	✗
QDR-Tree	DEXA (2018)	✓	✓	✓

To sum up, the main contributions of this paper are summarized as follows:

– We formulate the attributes-aware spatial keyword query, which takes spatial proximity, keywords' similarity and numerical attributes into consideration.

– We design a novel hybrid index structure, i.e., QDR-Tree which incorporates Quad-Cluster Tree with Dual-filtering R-Trees and accordingly propose the query processing algorithm to tackle the ASKQ.
– We propose a novel method to measure the relevance of one spatial objects with query keywords based on word2vec and bitmap of keyword.
– We conduct an empirical study that demonstrates the efficiency of our algorithms and index structures for processing the ASKQ on real-world datasets.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 3 formulates the problem of ASKQ. Section 4 presents the QDR-Tree. Section 5 introduces the query processing algorithm based on the QDR-Tree. Three baseline algorithms are proposed in Sect. 6 and considerable experimental results are reported. Finally, Sect. 7 concludes the paper.

2   Related Work

Existing works concerning the ASKQ include spatial keyword search, keyword relevance measurement, and the skyline operator.

Spatial Keyword Search. There are many studies on spatial keyword search recently [7,17,18]. Most of them focus on integrating inverted index and R-tree to support spatial keyword search. For example, IR2-tree [7] combines R-trees with signature files. It preserves objects spatial proximity, which is the key to solve spatial queries efficiently, and can filter a considerable portion of the objects that do not contain all the query keywords. Thus it significantly reduces the number of objects to be examined. SI-index [18] overcomes IR2-trees' drawbacks and outperform IR2-tree in query response time significantly. [17] proposes inverted linear quadtree, which is carefully designed to exploit both spatial and keyword-based pruning techniques to effectively reduce the search space.

Keyword Relevance Measurement. The traditional measurement on keyword relevance includes textual and semantic relevance. The textual relevance can be computed using an information retrieval model [2,4,5]. They are all TF-IDF variants essentially sharing the same fundamental principles. The semantic relevance is measured by many methods. [13,14] apply the Latent Dirichlet Allocation (LDA) model to calculate the topic distance of keywords. Gao [3] proposed an efficient disk-based metric access method which achieves excellent performance in the measurement of keywords' similarity.

The Skyline Operator. The skyline operator deals with the optimization problem of selecting multi-dimension points. A skyline query returns a set of points that are not dominated by any other points, called a skyline. It is said that a point o_i dominates another point o_j if o_i is no worse than o_j in all dimensions of attributes and is better than o_j at least in one dimension. Borzsonyi et al. [1] first introduced the skyline operator into relational database systems and introduced three algorithms. Geng et al. [11] propose a method which combines the spatial information with non-spatial information to obtain skyline results. Lee [8] et al. focused on two methods about multi-dimensional subspace skyline computation and developed orthogonal optimization principles.

3   Problem Statement

Given an geo-object dataset O in which each object o is denoted as a tuple $\langle \lambda,$ K, A$\rangle$, where $o.\lambda$ is a location descriptor which we assume is at a two dimensional geographical space and is composed of latitude and longitude, $o.K$ is the set of keywords, and $o.A$ represents the set of numerical attributes. Without loss of generality, we assume the attributes $o.a_i$ in $o.A$ are numeric attributes and normalize each $o.a_i \in [0, 1]$. We assume that smaller values of these numercial attributes, e.g., price and noise, are preferable. As for other numerical attributes' values which are better if higher, such as the rating and health score, we convert them decreasingly as $o.a_i = 1 - o.a_i$. The query q is represented as a tuple $\langle \lambda, K, W \rangle$, where $q.\lambda$ and $q.K$ represent the location of the user and the required keywords respectively, and $q.W$ represents the set of weight for different numerical attributes and user's different preference on these attributes. $\forall q.w_i \in q.W, q.w_i \geq 0$ $(i = 1, \ldots, |q.W|)$ and $\sum_{i=1}^{|q.W|} q.w_i = 1$. The reason for assigning weight to each attribute instead of qualifying exact range of attributes is to prepare for the fuzzy query on numerical attributes. In order to elaborate the QDR-Tree , we firstly define the keyword distance and the keyword cluster as follows.

Definition 1 (Keyword Distance). *Given two keywords k_1, k_2, their keyword distance, denoted as $d(k_1, k_2)$, includes both textual distance and semantic distance. The textual similarity between two keywords is denoted as $d_t(k_1, k_2)$ which is measured by the Edit Distance. The semantic distance between two keywords denoted as d_s is measured by the Euclidean distance of the term vector generated by word2vec. With a parameter $\delta(\in [0, 1])$ controlling their relative weights, Eq. (1) describes the formulation of $d(k_1, k_2)$.*

$$d(k_1, k_2) = \delta d_t(k_1, k_2) + (1 - \delta)d_s(k_1, k_2) \tag{1}$$

Definition 2 (Keyword Cluster). *A keyword cluster (C_i) is formed by similar keywords. The cluster diameter is defined as the maximum keyword distance within the cluster. One keyword can be allocated into the cluster if the diameter after adding it does not exceed the threshold τ, i.e. $\forall k_i, k_j \in C_i, d(k_i, k_j) < \tau$. Each cluster has a center object denoted as $C_i.cen$. All the keyword clusters (C_i) make up the set of keyword clusters ($\mathbb{C}$).*

Definition 3 (Attributes-Aware Spatial Keyword Query). *Given a geo-object set O and the attributes-aware spatial keyword query q, the result includes a set of $Top_\kappa(q)$,[1] $Top_\kappa(q) \subset O$, $|Top_\kappa(q)| = \kappa$ and $\forall o_i, o_j : o_i \in Top_\kappa(q), o_j \in O - Top_\kappa(q)$, it holds that $score(q, o_i) \leq score(q, o_j)$.*

As for the evaluation function, $score(q, o)$ in Definition 3, it is composed of three aspects, including the location proximity, the keywords similarity, and the value of numerical attributes, and will be discussed at large in the Sect. 5.

---

[1] Hereafter, Top-k is denoted as Top-κ to avoid confusion with the k-means algorithm.

4   QDR-Tree

In this section, we introduce a new hybrid index structure QDR-Tree, which is a new indexing framework for efficiently processing the ASKQ. The QDR-Tree can be divided into two layers, the keyword cluster layer and the spatial layer where the QDR-Tree can be split up into two sub-trees, named as Quad-Cluster Tree (QC-Tree) and Dual-filtering R-tree (DR-Tree) respectively.

4.1   Keyword Cluster Layer

The keyword cluster layer deals with keyword search with both textual and semantic similarities. Neither appending an R-Tree to each keyword with a huge space redundancy, nor just clustering all keywords into k groups with a high false positive ratio during query search, QC-Tree smartly splits keyword set into hierarchical levels and link them by a Quad-Tree.

   To improve the searching efficiency, we propose a new hierarchical quad clustering algorithm based on the *kernel k-means* [6]. Compared with the traditional k-means clustering, kernel k-means will have better clustering effect even the samples do not obey the normal distribution and is more suitable to cluster the keywords. Moreover, different from the common clustering, hierarchical clustering can form a meaningful relationship between different clusters, which is helpful to allocate a new sample and decrease the cost of misallocation. After the clustering process finishs, a quad-cluster tree (QC-Tree) is used to arrange all of these clusters, which is the core composition of the keyword cluster layer. In Algorithm 1, the critical part is applying the kernel k-means to each keyword cluster per level, with k fixed as 4. Furthermore, when the diameter of the keyword cluster is smaller than the $\tau_{cluster}$, the duplication operation is executed, which is presented in Algorithm 2 and will be discussed later.

Algorithm 1. Hierarchical quad clustering algorithm

Input: keyword set K, cluster number k
Output: Quad-Cluster Tree: T_{qc}
1 T_{qc}.add(K)
2 Insert K into a priority queue U /* instert as a set */
3 **while** $U \neq \emptyset$ **do**
4 $\quad$ $S \leftarrow U$.Pop()   /* pop the whole set */
5 $\quad$ $\{S_1, S_2, S_3, S_4\} \leftarrow$ Kernel$k$Means (k, S) /* k=4 by default */
6 $\quad$ **foreach** $S_i \in \{S_1, S_2, S_3, S_4\}$ **do**
7 $\quad\quad$ **if** $S_i.diameter < \tau_{cluster}$ **then**
8 $\quad\quad\quad$ Duplication (S_1, S_2, S_3, S_4)
9 $\quad\quad$ **else**
10 $\quad\quad\quad$ insert S_i in to U
11 $\quad\quad\quad$ T_{qc}.add(S_i)   /* S_i are children of S */

Algorithm 2. Duplication

Input: Four keyword sets: S_1, S_2, S_3, S_4
Output: Duplicated keyword sets: S'_1, S'_2, S'_3, S'_4

1 **for** $\forall k_i \in S_1 \bigcup S_2 \bigcup S_3 \bigcup S_4$ **do**
2   **if** $\sigma(d(k_i, S_j.cen)) < \tau_{dup}$ **then**   /* Variance */
3 $S'_j \leftarrow k_i \bigcup S_j$, if $k_i \notin S_j$ with $j \in \{1, 2, 3, 4\}$

4 $\{S_1, S_2, S_3, S_4\} \leftarrow \{S'_1, S'_2, S'_3, S'_4\}$

Figure 2(a) illustrates the hierarchical clustering in Algorithm 1, where each dot represents a keyword and different aggregation of these dots presents different keyword clusters. The dots marked in different color are the centroid of these clusters, and moreover, same color denotes their clusters stay in the same level.
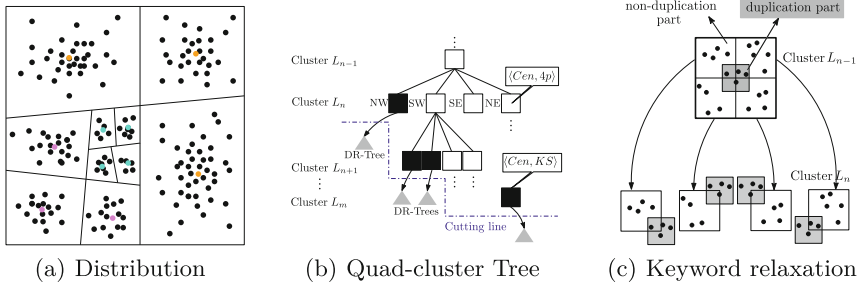

(a) Distribution   (b) Quad-cluster Tree   (c) Keyword relaxation

Fig. 2. Overview of the keyword cluster layer

Notice that, the main target of QC-Tree is to improve the pruning effect of keywords while making the future query keyword set located in only one keyword cluster. As is shown in both Algorithm 1 and Fig. 2(a), with the cluster level growing, the cluster will be more centralized and compact. That means the possibility of one query being allocated to different clusters increases layer by layer. It is necessary to decide an optimal $\tau_{cluster}$ to terminate the hierarchical cluster proceeding, if not, there would only be a single keyword in each cluster finally. The basic structure of QC-Tree is displayed in Fig. 2(b), where each internal node keeps the centroid keyword (cen) and four pointers (4p) to its four descendants nodes, and each leaf node will keep the keyword set in this cluster and the pointer to a new DR-Tree. Additionally, a cut-line is drawn to emphasize the shift of index structure, which is mainly dependent on the value of $\tau_{cluster}$.

As is analyzed above, the leaf cluster is where a query would most likely be scattered into different clusters. We will take a keyword-relaxation operation by duplicating some keywords among the four clusters sharing the same parent node. In Fig. 2(c), for a keyword cluster, its keywords are grouped into four sub-clusters and the duplication operation need to be executed. The dots in the shadow represent the keywords that will be duplicated and allocated to all of

these four sub-clusters because they are closed to all of the sub-clusters. Here, we introduce another threshold (τ_{dup}) to decide whether to execute the duplication operation. Although this keyword-relaxation operator will cause redundancy of keywords and extra space consumption, it will largely improve the time efficiency, which will also be demonstrated in the experimental verification.

4.2 Spatial Layer

Under each keyword cluster in the bottom of QC-Tree, we build a DR-Tree based on dual-filtering technique to organize the spatial objects in this cluster.

In Fig. 3, a basic structure of DR-Tree is shown in the spatial layer. Each internal node N records a two-element tuple: $\langle SP, KB \rangle$. The first element SP stands for the skyline points of the numerical attributes of all objects in the subtree rooted at the node. The second element is a bitmap of the keywords included in this cluster, which uses 1 and 0 to denote the existence of keywords.

Keyword Bitmap Filter Algorithm: In the DR-Tree, each node just records the keyword bitmap, and then the specific keywords list is kept only in the leaf keyword cluster. Then, the keyword relevance can be calculated just by Bitwise AND within the pair of bitmaps, which can decrease the storage consumption and increase the query efficiency.

Because bitwise AND within bitmaps need an exact keywords matching, in order to support similar keyword matching, we also implement the relaxation in each query process. In Fig. 3, as is highlighted in blue, the bitmap of query keywords performs a search-relaxation by switching some 0-bits to 1-bits based on the keyword similarity The search-relaxation algorithm will be proposed in Algorithm 4 in Sect. 4.2.

Multidimensional Subspace Skyline Filter Algorithm: In order to satisfy the needs of user's intention on multiple attributes, a filter called Multidimensional Subspace Skyline Filter, which is inspired by [1,8], is employed to amortize the query false positive and the cost of computation. We use the Evaluate() algorithm proposed in [8] to gain the multidimensional skyline points efficiently, and then let every QC-Tree node record the skyline points of its descendants. Furthermore, in order to reduce the complexity of recording multidimensional skyline points, we will take the point-compression operation by merging the closed skyline points in the attributes space. We calculate the cosine distance between skyline points' attributes to measure the similarity, and then merge these closed points when cosine distance is larger than a threshold.

5 QDR-Based Query Algorithm

In this section, we will introduce the ASKQ processing algorithms based on QDR-Tree. The process includes finding the Leaf Cluster, making search-relaxation and searching in the DR-Tree.
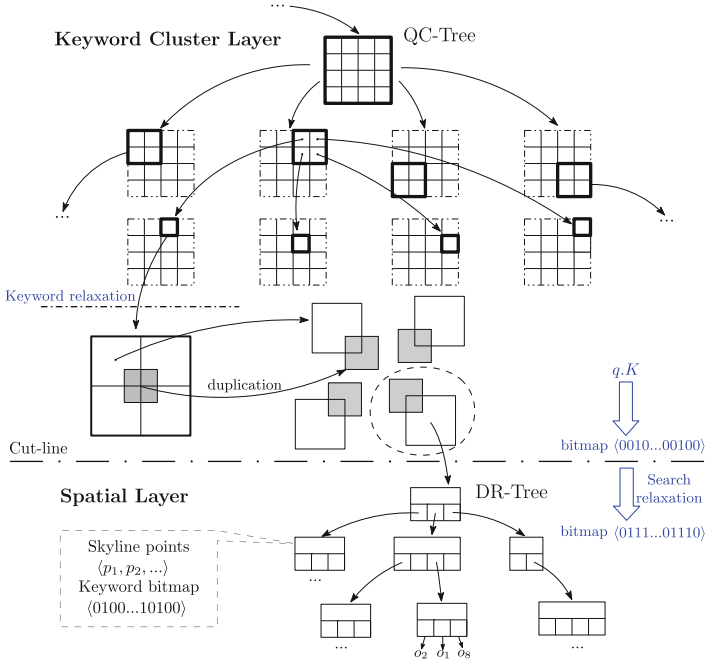
Fig. 3. Structure of QDR-Tree

Find the leaf cluster. The leaf keyword cluster that is best-matched with q can be obtained by iteratively comparing q with the four sub-clusters in each cluster level. If the combination of keywords in the query is typical and can be allocated into the same cluster, only one keyword cluster will be found. Otherwise, more than one keyword cluster may be returned.

Search-Relaxation. As is stated in Sect. 4.2, by means of executing search-relaxation, bitmap-based filter can support similar keyword matching. In Algorithm 4, a bitmap of relaxed query keyword is obtained by switching 0-bit to 1-bit if their keyword distance is under a threshold. By adopting a rational threshold, we can make a good trade-off between time cost and space occupation.

Algorithm 3. FindLeafCluster

Input: q, QC-Tree T_{qc}
Output: the leaf cluster: LC
1 $LC \leftarrow \emptyset$
2 **foreach** $k \in q.K$ **do**
3 $\quad$ $lc \leftarrow T_{qc}.root$
4 $\quad$ **while** lc *is not leaf cluster* **do**
5 $\quad\quad$ $ls \leftarrow lc.sub_i$, with $d(k, lc.sub_i.cen)$ is minimum among 4 $lc.subs$
6 $\quad$ $LC \leftarrow LC \cup lc$

Algorithm 4. Search relaxation

Input: bitmap of query keyword: *bmq*, bitmap of keyword cluster: *bmc*
Output: bitmap of relaxed query keyword: *bmr*

1 **for** $i \leftarrow 1$ to /bmq/ **do**
2   **if** $bmq[i] = 1$ **then**
3 $bmr[i] \leftarrow 1$
4   **for** $j \leftarrow 1$ to /bmc/ **do**
5   **if** $d(k_i, k_j) < \tau$ **then**
6 $bmr[j] \leftarrow 1$

Algorithm 5 illustrates the query processing mechanism over QDR-Tree. Given a query q, the object retrieval is carried out firstly by traversing the QC-Tree to locate the best-matched keyword cluster. Secondly, after executing search-relaxation, it will traverse the DR-Tree in the ascending order of the scores and keep a minimum heap for the scores. Notice that, if more than one keyword cluster is located, it will traverse all of them. At last the Top-κ results can be returned.

The ranking score of an object o for ASKQ is calculated by Eq. (2). Here, $\alpha, \beta \in [0,1]$ are parameters indicating the relative importance of these three factors. $\psi(q,o)$ is the Euclidian distance between q and o. The D_s^{max} is the maximal spatial distance that the client will accept. $\phi(q,o)$ which represents the keyword relevance between q and o is determined by the result of Bitwise AND between their keyword bitmaps. The smaller the score, the higher the relevance.

$$score(q,o) = \alpha\beta \times \frac{\psi(q,o)}{D_s^{max}} + (1-\beta) \times \frac{1}{\phi(q,o)} + (1-\alpha)\beta \times \sum_{i=1}^{|q.W|} q.w_i \times o.a_i \quad (2)$$

What is more, the score for non-leaf node N can also been measured to represent the optimal score of its descendant nodes, which is defined as Eq. (3)

$$score(q,N) = \alpha\beta \times \frac{\min \psi(q,N.MBR)}{D_s^{max}} + (1-\beta) \times \frac{1}{\phi(q,N)}$$
$$+ (1-\alpha)\beta \times \min_{\forall p \in N.sp} \sum_{i=1}^{|q.W|} q.w_i \times p.a_i \quad (3)$$

where the $\min \psi(q, N.MBR)$ represents the minimum Euclidian distance between the N's MBR and the $\phi(q, N)$ is can also be calculated by the bitmap of keywords kept in this node. We can prove that $Top_\kappa(q)$ is an exact result by the Theorem 1. If the score of the internal node dose not satisfy the ASKQ, there is no need to search its descendant nodes. Hence, the final Top-κ objects will have the least κ scores.

Theorem 1. *The score of an internal node N is the best score of its descendant object o to the query q.*

Proof. the score factors in location proximity, keyword relevance and non-spatial attributes' value. First, the MBR of the N encloses all of its descendant objects, then $\forall o_i \in$ descendant objects of N, $\min \psi(q, N.MBR) \leq \psi(q, o_i)$. Second, the keyword bitmap includes all of the keywords existing in the descendant objects of N. Obviously, $\phi(q, N) \geq \phi(q, o)$. Finally, the skyline points dominate or are equal to all of descendent objects concerning the value of attributes, i.e., $\min_{\forall p \in N.SP} \sum_{i=1}^{|q.W|} q.w_i \times p.a_i \leq \sum_{i=1}^{|q.W|} q.w_i \times o.a_i$. All these inequalities contribute to that $score(q, N) \leq score(q.o)$. □

Algorithm 5. QDR-Search algorithm

Input: a query q, Top$_\kappa$ results κ, and a QDR-Tree T_{qdr}
Output: Top$_\kappa(q)$

1 $LC = $ FindLeafCluster (q, T_{qc});
2 **for** $i \leftarrow 1$ to $|LC|$ **do**
3    $q.$bitmap $\leftarrow$ SearchRelaxation $(q.$bitmap$, LC[i].$bitmap$)$
4   Minheap.insert$(LC[i].$root$, 0)$
5   **while** *Minheap.size() $\neq$ 0* **do**
6     $N \leftarrow$ Minheap.first()
7   **if** N *is an object* **then**
8   Top$_\kappa(q).$insert(N)
9   **if** *Top$_\kappa(q).$size() $\geq k$* **then**
10   **break**
11   **else**
12   **for** $n_i \in N.entry$ **do**
13   **if** *Number of objects with smaller score than score(q, n_i) in Minheap $< (\kappa - $ Top$_\kappa(q).$size())* **then**
14   Minheap.insert$(n_i,$ score$(q, n_i))$

---

6   Experiment Study

6.1   Baseline Algorithm

In this section, we propose three baseline algorithms which are based on the three existing indexes listed in Table 1, including IR-Tree [9], SKY R-Tree [16] and IRS-Tree [10]. As is discussed in Sect. 1, none of these existing indexes can be qualified for the ASKQ due to different drawbacks. The specific algorithm designs will be respectively explained in detail as follows.

 Because the IR-Tree pays no attention on the value of numerical attributes, all spatial objects containing the query keywords and numerical attributes will be extracted. After that they will be ranked by the comprehensive value of

numerical attributes. Eventually, the top-κ spatial objects are just the result of the ASKQ.

Different from the IR-Tree, the SKY R-Tree fails to support multi-keywords query because one SKY R-Tree can only arrange one keyword and its corresponding spatial objects, such as restaurant. In order to deal with the ASKQ, all of the SKY R-Trees containing the query keywords will be searched and merged to obtain the final top-κ results.

The last baseline algorithm is proposed based on the IRS-Tree which is originally intended to address the GLPQ. Unlike ASKQ, the GLPQ requires specific range of attributes to leverage the IRS-Tree. To copy with the ASKQ, we will firstly set some different suitable ranges of each attributes as the input, which insures that enough spatial objects can be returned. Afterwards, we will further to select top-κ objects from the results in the first stage. Apparently, in our experiments, the IRS-Tree will not make much sense anymore.

Notice that, all of these three baseline algorithms cannot solve the ASKQ directly at a time and need subsequent elimination of redundancy, which determines their inefficiency in the ASKQ.

In the experiment section, we conduct extensive experiments on both real and synthetic datasets to evaluate the performance of our proposed algorithms.

6.2   Experiment Setup

The real dataset is crawled from the famous location-based service platform, Foursquare. After information cleaning, the dataset has about 1M objects consisting of geographical location, the keyword list written in English, and the normalized value of attributes. Each spatial object contains the keywords such as steak, pizza, coffee, etc. and four numerical attributes, including price, environment, service and rating.

In the synthetic dataset, each object is composed of coordinates, various keyword, and multi-dimensional numerical attributes. The size of the synthetic dataset varies in the experiments. The coordinates are randomly generated in (0, 10000.0), and the average number of keywords per object is decided by a parameter r which denotes the ratio of the number of object's keywords to the cluster's. Without loss of generality, the values of each numercial attribute are randomly and independently generated, following a normal distribution.

We compare the query cost of proposed algorithms with different datasets respectively. The experimental settings are given in Table 2. The default values are used unless otherwise specified. All algorithms are implemented in Python and run with Intel core i7 6700HQ CPU at 2.60 GHz and 16 GB memory.

6.3   Performance Evaluation

In this section, we campare different baseline algorithms proposed in Sect. 6.1 with our framework. We evaluate the processing time and disk I/O of all the proposed methods by varying the parameters in Table 2 and investigate their effects. In the first part we study the experimental results on the real dataset.

Table 2. Default value of parameters

Parameter	Default value	Descriptions
κ	10	Top-κ query
$|o.A|$	4	No. of attributes' dimension
δ	0.5	Weight factor of Eq. (1)
α	0.5	Weight factor of Eq. (2)
β	0.67	Weight factor of Eq. (2)
$\tau_{cluster}$	0.3	Threshold of quad clustering
τ_{dup}	0.05	Threshold of duplication
$|O|$	1M	Number of objects
M	25	Maximum number of DR-tree entries

Index Construction Cost: We first evaluate the construction costs of various methods. The cost of an index is measured by its construction time and space budget. The costs of various methods are shown in Fig. 4, where IRS refers to the baseline structure IRS-Tree, the SKY-R, IR refer to SKY R-Tree, IR-Tree respectively, and the QDR represents our design. We can see that, SKY R-Tree and IR-Tree exceed both in time consumption and space cost, because they are short of attention of either attributes or multi-keywords. Moreover, since we employ bitmap and skyline points to measure numercial attributes, QDR is more lightweight than IRS.

Effect of κ: We investigate the effect of κ on the processing time and disk I/O of the proposed algorithms by randomly generate 100 queries. Here, considering that the SKY R-Tree and IR-Tree do not take into account either attributes or keywords, we add a filter operation after their query process. For example, the SKY R-Tree returns Top-κ results of each keyword and merges them in the second stage. Obviously, this redundancy of result is the main reason of the high time cost. As shown in Fig. 5, with the increase of κ, IRS and QDR have the same smoothly increasing trend on query time and disk I/O. QDR exceeds in query time cost with different parameters. It indicates that we can effectively receive the Top-κ results from one branch to another.

Effect of $|O|$: Parameter $|O|$ denotes the number of objects in the QDR-Tree, We increase the number of objects in the synthetic dataset from 0.1 to 5 M. It can be seen from Fig. 7(a) and (b), that, SKY-R and IR have more obvious increases in query time cost and disk I/O when the data size increases, which can be explained because of their larger redundancy along with the larger dataset. On the other hand, the QDR is more stable and surpasses another three indexes.

Effect of $|o.A|$: Parameter $|o.A|$ denotes the number of attributes the object o covers. As shown in Fig. 7(c) and (d), the query time and Disk I/O of the IR-Tree based on synopses tree has distinct increase, because it fails to consider the attributes, while another three frameworks are more stable, which is mainly because of either the skyline filter algorithm or the synopses tree.
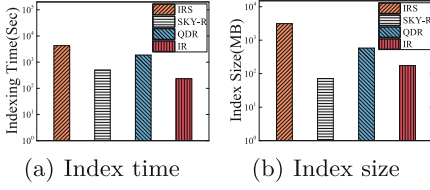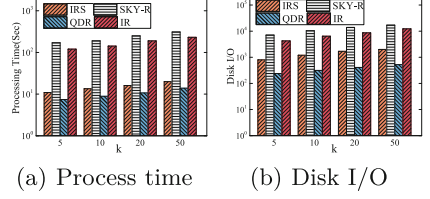
(a) Index time   (b) Index size

Fig. 4. Construction cost

(a) Process time   (b) Disk I/O

Fig. 5. Effect of κ

Effect of $\tau_{cluster}$ & τ_{dup}: $\tau_{cluster}$ and τ_{dup} are the crucial parameters in our QDR-Tree, which are analyzed theoretically in Sect. 4. The experimental results also verify their effect on the Processing Time and Index Size. Figure 6(a) and (c) show that both $\tau_{cluster}$ and τ_{dup} have an optimal value to minimize the processing time. Smaller or larger value will both increase the processing time because of keyword scattring or redundancy. In Fig. 6(b) and (d), index size decreases as the $\tau_{cluster}$ becomes larger and reach saturation at some point, while it increases along with the τ_{dup} because of the increase of keyword redundancy.
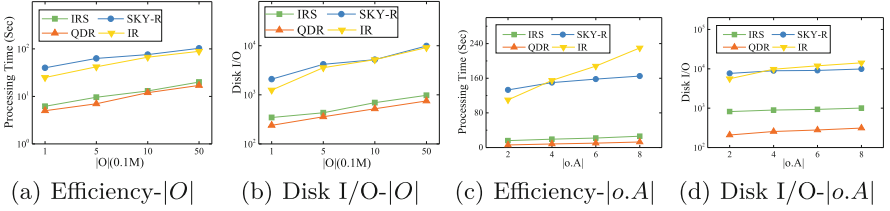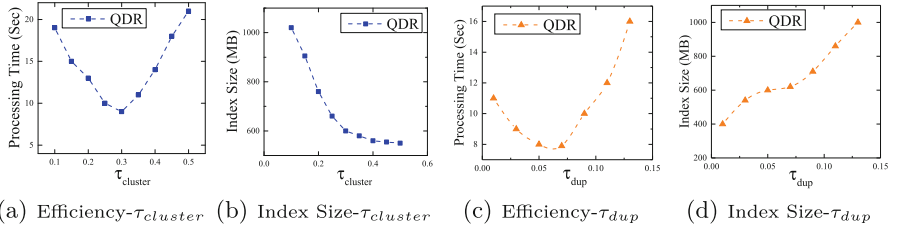

(a) Efficiency-$|O|$ (b) Disk I/O-$|O|$ (c) Efficiency-$|o.A|$ (d) Disk I/O-$|o.A|$

Fig. 6. Synthetic dataset

(a) Efficiency-$\tau_{cluster}$ (b) Index Size-$\tau_{cluster}$ (c) Efficiency-τ_{dup} (d) Index Size-τ_{dup}

Fig. 7. Effect of $\tau_{cluster}$ & τ_{dup}

7   Conclusion

In this paper, we formulated the attributes-aware spatial keyword query (ASKQ) and proposed a novel index structure call Quad-cluster Dual-filtering R-Tree (QDR). QDR-Tree is a two-layer hybrid index based on two index structures and two searching algorithms. We also proposed a novel method to measure the relevance of spatial objects with query keywords, which applies keyword-bitmap and search-relaxation to achieve exact and similar keyword match. Moreover, by

employing the keyword-relaxation, we greatly improve the time efficiency at the sacrifice of a little space consumption. Finally, massive experiments with real datasets demonstrate the efficiency and effectiveness of QDR-Tree.

References

1. Borzsonyi, S., Stocker, K., Kossmann, D.: The skyline operator. In: IEEE International Conference on 2002 Data Engineering (ICDE), pp. 421–430 (2002)
2. Cao, X., Cong, G., Jensen, C.S.: Retrieving top-k prestige-based relevant spatial web objects. Int. Conf. Very Large Data Bases (VLDB) **3**, 373–384 (2010)
3. Chen, L., Gao, Y., Li, X., Jensen, C.S., Chen, G.: Efficient metric indexing for similarity search. IEEE Trans. Knowl. Data Eng. (TKDE) **29**(3), 556–571 (2017)
4. Cong, G., Jensen, C., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. Int. Conf. Very Large Data Bases (VLDB) **2**(1), 337–348 (2009)
5. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. Int. Conf. Very Large Data Bases (VLDB) **2**(1), 337–348 (2009)
6. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 551–556 (2004)
7. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: IEEE International Conference on Data Engineering (ICDE), pp. 656–665 (2008)
8. Lee, J., Hwang, S.: Toward efficient multidimensional subspace skyline computation. Int. Conf. Very Large Data Bases (VLDB) **23**(1), 129–145 (2014)
9. Li, Z., Lee, K.C., Zheng, B., Lee, W.C., Lee, D., Wang, X.: Ir-tree: an efficient index for geographic document search. IEEE Trans. Knowl. Data Eng. (TKDE) **23**(4), 585–599 (2011)
10. Liu, X., Chen, L., Wan, C.: LINQ: a framework for location-aware indexing and query processing. IEEE Trans. Knowl. Data Eng. (TKDE) **27**(5), 1288–1300 (2015)
11. Ma, G., Arefin, M.S., Morimoto, Y.: A spatial skyline query for a group of users having different positions. In: Third International Conference on Networking and Computing, pp. 137–142 (2012)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Computer Science (2013)
13. Qian, Z., Xu, J., Zheng, K., Sun, W., Li, Z., Guo, H.: On efficient spatial keyword querying with semantics. In: Navathe, S.B., Wu, W., Shekhar, S., Du, X., Wang, X.S., Xiong, H. (eds.) DASFAA 2016. LNCS, vol. 9643, pp. 149–164. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32049-6_10
14. Qian, Z., Xu, J., Zheng, K., Zhao, P., Zhou, X.: Semantic-aware top-k spatial keyword queries. World Wide Web (WWW), pp. 1–22 (2017)
15. Ray, S., Nickerson, B.G.: Dynamically ranked top-k spatial keyword search. In: ACM International Conference on Management of Data (SIGMOD), pp. 6–18 (2016)
16. Sasaki, Y., Lee, W.-C., Hara, T., Nishio, S.: Sky R-tree: an index structure for distance-based top-k query. In: Bhowmick, S.S., Dyreson, C.E., Jensen, C.S., Lee, M.L., Muliantara, A., Thalheim, B. (eds.) DASFAA 2014. LNCS, vol. 8421, pp. 220–235. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05810-8_15
17. Tao, Y., Sheng, C.: Fast nearest neighbor search with keywords. IEEE Trans. Knowl. Data Eng. (TKDE) **26**(4), 878–888 (2014)
18. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: efficient top k spatial keyword search. In: IEEE International Conference on Data Engineering (ICDE), pp. 901–912 (2013)