Reverse Collective Spatial Keyword Querying (Short Paper)

Yang Wu(✉), Jian Xu, Liming Tu, Ming Luo, Zhi Chen, and Ning Zheng

School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China
{161050040,jian.xu,tuliming,luom,162050110,nzheng}@hdu.edu.cn

Abstract. Recently, Collective Spatial Keyword Querying (CoSKQ), which returns a group of objects that cover a set of given keywords collectively and have the smallest cost, has received extensive attention in spatial database community. However, no research so far focuses on a situation when the result of CoSKQ is taken as the input of a query. But this kind of query has many applications in location based services. In this paper, we introduce a new problem Reverse Collective Spatial Keyword Querying (RCoSKQ) that returns a region, in which the query objects are qualified objects with the highest spatial and textual similarity. We propose an efficient method which uses IR-tree to retrieve objects with text descriptions. To accelerate the query process, a pruning method that effectively reduces computing is proposed. The experiments over real and synthesis data sets demonstrate the efficiency of our approaches.

Keywords: Collective Spatial Keyword Querying · A set of query objects · Reverse

1 Introduction

Given a set of spatial-textual objects, a query object with a location and a set of keywords, Collective Spatial Keyword Querying (CoSKQ) is to find a set of objects such that it covers given keywords collectively and has the smallest cost. An example application is to find several places that can collaboratively provide drinking, singing and accommodation to a tourist. In previous works [1,8], the authors devised both exact and approximate solutions according to different spatial similarities. Then in [7], the authors studied CoSKQ on road network. However, these researches only finish the computation from the user's perspective. In this paper, we introduce a new problem Reverse Collective Spatial Keyword Querying (RCoSKQ) to find a region (e.g., influence zone), in which the total distance between query objects and a user is minimum, and the query keywords set of the user is a subset of all the keywords of query objects.

Reverse k Nearest Neighbor (RkNN) query has been extensively studied in spatial database community. Given a set of spatial objects O and a query object,

H. Gao et al. (Eds.): CollaborateCom 2018, LNICST 268, pp. 211–221, 2019.
https://doi.org/10.1007/978-3-030-12981-1_15

a RkNN returns objects in O which take query object as one of k nearest neighbors. In [2,10,11,13], only spatial similarity is considered. Then Reverse Spatial Textual k Nearest Neighbors (RSTkNN) query with both spatial similarity and textual relevance is proposed in [3,5,6,9,12]. However, different with RCoSKQ, only one query object is considered in these researches.

Imagine a case that the government wants to build a residential area in a business district, which includes three facilities, a cinema, a ktv and a Coffee House. In this case, if any user in this region intends to watch movies, sing a song and have coffee, these three facilities cover all the three activities collectively and minimize the accumulated distance to the user. In this example the result is a region in which query objects have the highest influence under combinations of their services. Such a combination consists of text descriptions of query objects. And both spatial proximity and textual similarity are considered in this situation. To the best of our knowledge, we are the first to explore this type of query in which a set of query objects are given in the spatial-textual database domain.

But we face two challenges in this study. The first challenge is how to identify all the keywords sets of query objects which collectively contained by them. The second is how to compute efficiently the influence region of the query objects.

To address these two challenges, we use an IR-tree to retrieve spatial-textual objects. The procedure is divided into two phases. The first phase is to find all the subsets of keywords. An incremental algorithm is presented to improve efficiency, in which query objects are visited one after another. For each existing subset during iteration, it is combined with the keywords of current visited object, and invalid subsets are deleted at the same time. Then in the second phase we use Half-space pruning technology to compute the influence region. Different with studies in [2,12], the input of RCoSKQ is a set of query objects. We develop an efficient spatial pruning method which utilizes both spatial and textual information of the objects.

Our contributions can be summarized as follows: (1) We propose a new query problem, Reverse Collective Spatial Keyword Querying RCoSKQ, and formalize it in the same way with CoSKQ. (2) We propose an efficient method based on Half-space pruning technology. Both textual similarity and spatial proximity are taken into consideration and the region is returned after examining each combination of keywords. An efficient and effective spatial pruning algorithm is proposed which can reduce redundant calculation of generating result region. (3) Extensive experiments are conducted on both real and synthesis data sets to evaluate the efficiency of the proposed algorithm.

The rest of this paper is organized as follows. Section 2 defines some basic concepts. Sections 3 and 4 illustrate our proposed approaches. Section 5 conducts the evaluation on different data sets and analyze of our experimental results. Finally we make a conclusion in Sect. 6.

2 Preliminaries

Assume all objects are located in Euclidean space. Let **O** be a spatial data set consists of m spatial web objects, each object $o \in \mathbf{O}$ is associated with a tuple ($o.\lambda$,o.ψ), where $o.\lambda$ is a location and $o.\psi$ is a set of meaningful keywords that describe the object (e.g., the specialties of a restaurant or the sceneries of a scenic spot).

Definition 1. Collective Spatial Keyword Query (CoSKQ) [1]**:** Given a query q = $(q.\lambda, q.\psi)$, the *Collective Spatial Keyword Querying* (CoSKQ) problem is to find a set S of objects in **O** such that S covers $q.\psi$ and the *cost* of S is minimized.

In this paper, we consider the *sum cost* which is also adopted in [1]. That is,

$$Cost(q, S) = \sum_{k=1}^{N} dist(o_k, q) \tag{1}$$

where N is the number of objects in S, dist(o,q) is the Euclidean distance between q and o_k.

Definition 2. Collective Keyword Set (CKS): Given a set of query objects $Q = \{q_1, q_2, ..., q_n\}$, let Ψ be the set of keywords contained by all the objects in Q (e.g., $\Psi = \bigcup q_i.\psi$), *Collective Keyword Set* is a subset of Ψ contains $|Q|$ different keywords, each keyword is covered by an object of Q, respectively.

Since that in CoSKQ the query's keywords are collectively contained by a group of spatial web objects, we define that each query object covers one keyword and each *CKS* consists of $|Q|$ keywords. For example, there are two query points q_1 and q_2, $q_1.\psi = \{a,b\}$, $q_2.\psi = \{c\}$. The *CKSs* are {a,c} and {b,c}, respectively.

Definition 3. Collective Influence Region (CIR): Given a set of query objects $Q = \{q_1, q_2, ..., q_n\}$, *CKS* is a *Collective Keyword Set* of Q, *Collective Influence Region* is a region if any user in which takes *CKS* as query keywords set, Q will be the result of CoSKQ (Q) which contains $|Q|$ objects.

Definition 4. Reverse Collective Spatial Keyword Querying (RCo SKQ): Given a set of query objects Q, $CKS_Q = \{CKS_1, CKS_2, ..., CKS_k\}$ contains all the *CKSs* of Q. $CIR_Q = \{CIR_1, CIR_2, ..., CIR_k\}$ represents the corresponding *CIRs*. RCoSKQ returns a region R which is the union of all the *CIRs* (e.g., $R = \bigcup CIR_i$).

3 Collective Influence Region Based Algorithm

To the best of our knowledge, none of existing methods can be directly used for RCoSKQ. Most of the existing researches about RkNN and RSTkNN focus only on the situation of a single query object. Some existing researches like

Range Based Query and *Skyline Query* need one point in the result of RCoSKQ as query object, however, the result of RCoSKQ is a region and can not be known in advance. In this section, We propose Collective Influence Region Based Algorithm (CIRB) based on algorithm InfZone [2]. CIRB has two phases. In the first phase, all the *CKSs* of a set of query objects are enumerated. In the second phase, the *CIR* for each *CKS* is computed and all the *CIRs* are combined finally.

3.1 Index Structure

We use IR-tree to retrieve spatial-textual objects. The IR-tree [4] is an R-tree where each node has a reference to an inverted file. Entries of each leaf node are represented in the form (e.mbr,e.inv), where e refers to an object $o \in \mathbf{O}$, mbr is the bounding rectangle of a node, and $e.inv$ refers to the inverted file of a node. An inverted file contains a vocabulary of all distinct words and a posting list for each word t which means the objects contain the word t. For each non-leaf node, e refers to a child node, mbr is the minimum bounding rectangle of all entries of that child node and $e.inv$ is the corresponding inverted file which is the union of all the keywords of child nodes.

3.2 The Collective Keyword Set

A straightforward method to acquire all the *CKSs* is to enumerate all possible combinations of keywords of query objects Q and then filter out those are not *CKS*. This method is time consuming and might yield an exponential time complexity in terms of the number of keywords of Q. In order to solve this problem, we propose an incremental enumeration method. In each of the subsequent iterations, for each existing subset, it is combined with the current visited object to generate a new subset. At the same time, the invalid subsets are removed. For example, assume that p_1, p_2, p_3 are a group of query objects, and $\{t_1, t_5\}$, $\{t_2, t_4\}$, $\{t_4, t_6\}$ are the corresponding keywords of query objects. Table 1 illustrates the entire iteration process. Initially, the candidate set $\widetilde{C}$ is empty. Then every query object is visited iteratively and each keyword is used to combine with the existing combinations. In step 4, p_3 is visited and the keywords of p_3 are t_4 and t_6. For the keyword t_4, four combinations are formed (e.g., $\{t_1, t_2, t_4\}$, $\{t_1, t_4, t_4\}$, $\{t_5, t_2, t_4\}$, $\{t_5, t_4, t_4\}$). However, $\{t_1, t_4, t_4\}$ and $\{t_5, t_4, t_4\}$ are not *CKSs* because the duplication of t_4. And at last, there are six qualified sets which consist of three distinct keywords.

Table 1. Optimized enumeration processing

Step	Action	Candidate set
1	Initialization	{}
2	Insert p_1	$\{t_1\}$,$\{t_5\}$
3	Insert p_2	$\{t_1, t_2\}$,$\{t_1, t_4\}$,$\{t_5, t_2\}$,$\{t_5, t_4\}$
4	Insert p_3	$\{t_1, t_2, t_4\}$,$\{t_5, t_2, t_4\}$,$\{t_1, t_2, t_6\}$,$\{t_1, t_4, t_6\}$,$\{t_5, t_2, t_6\}$,$\{t_5, t_4, t_6\}$

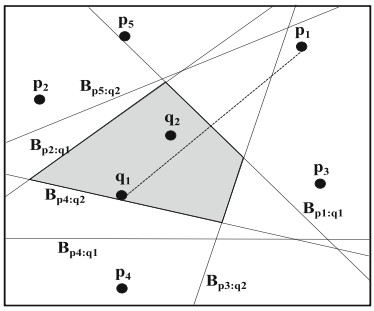


Fig. 1. The Collective Influence Zone

3.3 The Collective Influence Zone

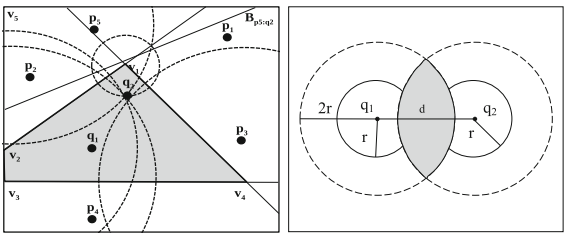
In this phase, the *CIR* for each *CKS* is achieved. In order to compute the result region of RCoSKQ, we compute *CIR* for each *CKS* and then combine all returned regions. The detail procedure of calculating *CIR* is similiar to the basic idea of InfZone [2]. But there are some differences, first we iteratively compute the influence zone *IZone* for each query object. Then, not all of the non-leaf nodes need to be verified whether they will affect *IZone*, only those non-leaf nodes need to be considered whose inverted files contain keywords of query object. After computing *IZone* for each query object, the *CIR* is formed by computing the intersection of *IZones* and the result of RCoSKQ is returned.

Figure 1 shows an example of *CIR*. The *CKS* is {*vegetable,beer*} and the shadow area is the corresponding *CIR*. The formation of the region is based on Half-space pruning. For example, $B_{p1:q1}$ is the perpendicular bisector of the dotted line between q_1 and p_1. The Half-space that contains q_1 is represented with $H_{q1:p1}$ and the half-space $H_{p1:q1}$ is an area for each user in which p_1 is taken as its nearest neighbor with the query keyword *vegetable*. Given q_1 as a query object, $B_{p1:q1}$, $B_{p2:q1}$ and $B_{p4:q1}$ are corresponding perpendicular bisectors because p_1, p_2 and p_4 all cover *vegetable*. In the same way, we get a region formed by $B_{p3:q2}$, $B_{p4:q2}$ and $B_{p5:q2}$. The lemma below shows that the intersection of the two regions is a *CIR*.

Lemma 1. *Given a group of query objects* $Q = \{q_1, q_2, ..., q_n\}$ *and a corresponding* $CKS = \{CKS_1, CKS_2, ..., CKS_m\}$, $\widetilde{R} = \{r_1, r_2, ..., r_n\}$ *is a set of regions produced by each* q_i *and all the spatial objects cover* w_i. *The intersection of elements of* $\widetilde{R}$ *is the CIR of* Q.

Proof. Assume $Q = \{q_1, q_2\}$ *is a set of query objects,* r_1 *is a region which is produced by all the half-spaces of* q_1 *between* q_1 *and competitors which may have greater influence than* q_1. *According to Half-space pruning, for any object in* r_1, q_1 *must be its nearest neighbor compared with competitors. And the region of* q_2 *is* r_2. *Let* R *be the intersection of* r_1 *and* r_2. *So each object* p *in* R *will take* q_1 *or* q_2 *as nearest neighbor under different keywords. Let* $Q' = \{p_1, p_2\}$ *which*

represents another combination. Thus there must be Cost(p,Q') > Cost(p,Q), which proves the region is CIR.



(a)Space pruning (b)The optimized space

Fig. 2. Optimized pruning method

4 The Optimized Algorithm

The efficiency of the algorithm CIRB decreases dramatically as problem size increases. There are two drawbacks, first each time we compute *CIR* for every *CKS*, *CIR* is initialized with the whole data set. Second, it is time consuming to combine all the *CIRs*.

In this section, we introduce an effective pruning method which can simplify the CIRB. Same with the algorithm InfZone [2], C_p denotes a circle centered at p with radius equal to *dist(p,q)*, where *dist(p,q)* is the distance between p and q. The query object q may not be the nearest neighbor of p if an entry intersects with C_p because there may be objects closer to p than q in the entry. In InfZone, it is proved that only the convex vertexes of the unpruned polygon are used to judge whether an entry may influence the region. In Fig. 2(a), only v_1, v_2, v_3 and v_4 are used to construct circles C_{v1}, C_{v2}, C_{v3} and C_{v4}. Any entry intersects with these circles will be used to prune the unpruned polygon. In CIRB, an unpruned polygon is returned for each query object. *CIR* is returned after intersecting all these unpruned polygons. So we should find that the regions pruned when dealing with the former query objects must not be *CIR*. However, lots of objects will be used to prune the region which has been pruned before in CIRB.

In our Optimized CIRB algorithm (OCIRB), the unpruned polygon is set to the initial region of current query. When the last query object is processed, the region returned is the *CIR* of *CKS*. Figure 2(a) shows an example and the shadow area depicted is the unpruned polygon of q_1. When q_2 is processed, take object p_5 as an example, p_5 will not be visited because no C_v intersects with p_5. In CIRB, the initial region is the whole data set, p_5 intersects with C_{v5} so it is used to prune the region. However, $B_{p5:q2}$ not intersects with the unpruned polygon of q_1 which means the visit of p_5 is redundant.

Now, we analyse the number of facilities used to compute the CIR. We assume that the facilities are uniformly distributed in a unit space and the number of facilities is |F|. The expected area of a randomly chosen facility point is $1/|F|$(the sum of the areas of all the influence zones is 1 [2]). We approximate the influence zone to a circular shape having the same area. In Fig. 2(b), there are two query objects q_1 and q_2 both with radius r. A facility can be ignored if it lies at a distance greater than $2r$ from q_1 or q_2. In CIRB, the area of facilities to be considered is $S_1 = 2\pi r^2$. In OCIRB, for q_2, only objects located in the shadow area will be traversed to compute the CIR. The area needs to be considered is $S_2 = 2r^2 \arccos(d/2r) - 1/2d\sqrt{4r^2 - d^2}$, d is the distance between q_1 and q_2. Note that $d > 4r$ means that there is no CIR of query set $\{q_1,q_2\}$. Thus, for query set $\{q_1,q_2\}$, the amount of calculations for $(S_1 - S_2)|F|$ facilities are pruned off.

5 Experiments

5.1 Setup

Here we use two datasets, namely, GN (extracted from geonames.usgs.gov) and Hotel (extracted from www.allstays.com). Table 2 shows some properties of the two data sets. Several synthesis data sets which are randomly generated. Each object has a location and a set of words.

Table 2. Dataset statistics

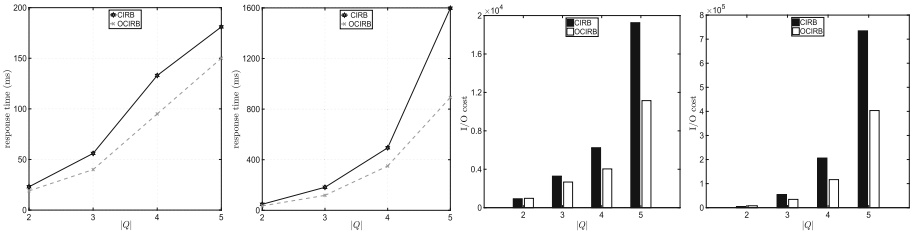
Statistics	Hotel	GN
Number of objects	20,790	627,773
Total unique terms	602	102,037
Avg unique terms per object	3.9	3.3

We evaluate the first algorithm CIRB from Sect. 3 which is based on Half-space pruning and the optimized algorithm OCIRB from Sect. 4. Two metrics I/O cost and running time are used to evaluate the performance of our algorithm. The I/O cost is measured as the number of index nodes accessed from the disk. The running time is measured as the time duration from the beginning of the algorithm to the end.

In each experiment, we generate 30 groups of query sets and then report the average I/O cost and average running time. The default numbers of query objects (e.g., $|Q|$) and terms per object (e.g., $|p.\psi|$) are set to 3 and 4, respectively. We first randomly choose an object in database space and then search a set of objects near to the object, a set of query objects is selected from these objects. Both algorithms are implemented in Java and ran on a PC equipped with an Intel 2.1 GHZ Xeon E5-2620 CPU and 16 GB RAM.

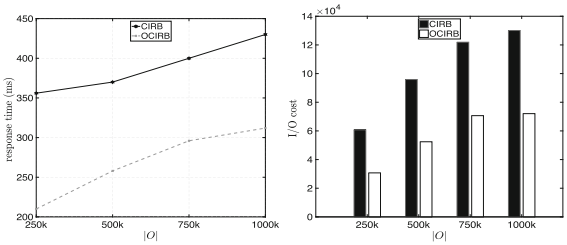
5.2 Performance Evaluation

Effect of $|Q|$: In the first set of experiments, we sought to analyze how response time and I/O accesses are impacted by the number of query objects. The experiments are conducted on both real data sets GN and Hotel. The results are reported in Fig. 3. Figure 3(a) and (b) show the effect on response time when varying the number of query objects. As the number of query objects increases, the possible combinations of keywords also increase and more region needs to compute. As OCIRB avoids a lot of unnecessary regions pruning, it outperforms CIRB in both data sets. Figure 3(c) and (d) shows the result of I/O accesses when varying the number of query objects. The I/O accesses increase proportionally with the increase of $|Q|$, as the number of keywords in CKS is directly related to the number of query objects which leads to more relevant nodes visited. As a great number of nodes which will not affect the unpruned polygon are ignored, the I/O accesses of OCIRB are much less than CIRB.



(a)Hotel (b)GN (c)Hotel (d)GN

Fig. 3. The response time and I/O cost on two real data sets

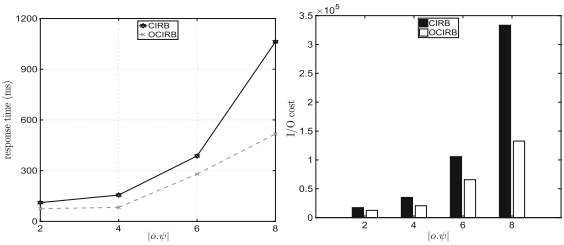


(a)Synthesis data (b)Synthesis data

Fig. 4. Effect of $|O|$

Effect of datasize $|O|$: This experiment is to evaluate the performance of algorithms both on response time and I/O accesses. Four data sets with cardinalities of 250k, 500k, 750k and 1000k (i.e., 1 million) are randomly generated and the keywords are randomly extracted from the unique terms of GN. The number of

query objects in a query is fixed at three. Figure 4(a) shows the result of response time when varying $|O|$. As more related objects are found and may be used to prune the region, the response time of both algorithms increases rapidly. But the OCIRB performs much better than CIRB, the reason is that computing a new region is time consuming and OCIRB eliminates a lot of useless computation. In Fig. 4(b), as the nodes of IR-tree increase when $|O|$ increases, the number of nodes visited augments and more time is spent to traverse IR-tree. Less objects accessed makes OCIRB handle fewer nodes.

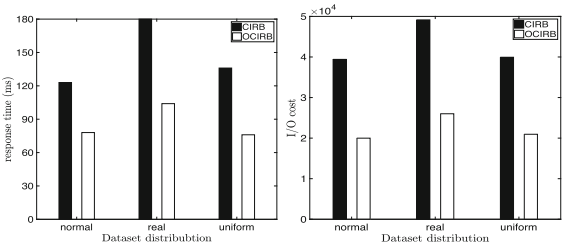


(a)Synthesis data (b)Synthesis data

Fig. 5. Effect of $|o.\psi|$

Effect of $|o.\psi|$: We further evaluate the response time and I/O accesses of the two methods on synthesis data sets under different number of keywords per object. The sizes of data sets are fixed at 500k and the number of query objects is fixed at three. The $|o.\psi|$ is roughly 2, 4, 6 and 8. Figure 5(a) shows the response time when varying $|o.\psi|$. As $|o.\psi|$ increases, the number of combinations of keywords augments dramatically. In Fig. 5(b), the I/O accesses increase obviously because the number of combinations increases exponentially. The performance is obviously improved in OCIRB because the number of objects processed is reduced.

Effect of data distribution: In Fig. 6, we study the effect of data distribution on both algorithms. U, R and N correspond to Uniform, Real, and Normal



(a)Synthesis data (b)Synthesis data

Fig. 6. Effect of data distribution

distributions, respectively. The synthetic data sets contain the same number of points as GN, which contains 627,773 points. The keywords of each data set are randomly extracted from the unique terms of GN. Figure 6 demonstrates that OCIRB performs better than CIRB.

6 Conclusion

In this paper, we identified a new problem Reverse Collective Spatial Keyword Querying, named RCoSKQ. A RCoSKQ returns a region in which the query objects will be the best group of objects which have the highest spatial and textual similarity. An efficient algorithm CIRB which based on Half-space pruning is developed and an IR-tree is constructed to retrieve spatial-textual objects. All the *CKSs* of query objects are enumerated firstly, then we compute *CIR* for each *CKS* and combine them together. We also adopt a spatial pruning method, which reduces the number of objects visited, to improve the efficiency of CIRB algorithm. Extensive experiments on both synthetic and real data sets demonstrate the effectiveness of our algorithm.

Acknowledgment. This work is supported by the National Natural Science Foundation of China (No. 61572165), the Natural Science Foundation of Zhejiang Province (No. LZ15F 020003).

References

1. Cao, X., Cong, G., Jensen, C.S., Ooi, B.C.: Collective spatial keyword querying. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 373–384. ACM (2011)
2. Cheema, M.A., Lin, X., Zhang, W., Zhang, Y.: Influence zone: efficiently processing reverse k nearest neighbors queries. In: 2011 IEEE 27th International Conference on Data Engineering (ICDE), pp. 577–588. IEEE (2011)
3. Choudhury, F.M., Culpepper, J.S., Sellis, T., Cao, X.: Maximizing bichromatic reverse spatial and textual k nearest neighbor queries. Proc. VLDB Endowment **9**(6), 456–467 (2016)
4. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. Proc. VLDB Endowment **2**(1), 337–348 (2009)
5. Fang, H., et al.: Ranked reverse boolean spatial keyword nearest neighbors search. In: Wang, J., et al. (eds.) WISE 2015. LNCS, vol. 9418, pp. 92–107. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26190-4_7
6. Gao, Y., Qin, X., Zheng, B., Chen, G.: Efficient reverse top-k boolean spatial keyword queries on road networks. IEEE Trans. Knowl. Data Eng. **27**(5), 1205–1218 (2015)
7. Gao, Y., Zhao, J., Zheng, B., Chen, G.: Efficient collective spatial keyword query processing on road networks. IEEE Trans. Intell. Transp. Syst. **17**(2), 469–480 (2016)
8. Long, C., Wong, R.C.W., Wang, K., Fu, A.W.C.: Collective spatial keyword queries: a distance owner-driven approach. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pp. 689–700. ACM (2013)

9. Lu, J., Lu, Y., Cong, G.: Reverse spatial and textual k nearest neighbor search. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 349–360. ACM (2011)
10. Tao, Y., Papadias, D., Lian, X.: Reverse kNN search in arbitrary dimensionality. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30, pp. 744–755. VLDB Endowment (2004)
11. Wei, W., Yang, F., Chan, C.-Y., Tan, K.-L.: FINCH: evaluating reverse k-nearest-neighbor queries on location data. Proc. VLDB Endowment **1**(1), 1056–1067 (2008)
12. Xie, X., Lin, X., Xu, J., Jensen, C.S.: Reverse keyword-based location search. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 375–386. IEEE (2017)
13. Yang, S., Cheema, M.A., Lin, X., Zhang, Y.: SLICE: reviving regions-based pruning for reverse k nearest neighbors queries. In: 2014 IEEE 30th International Conference on Data Engineering (ICDE), pp. 760–771. IEEE (2014)