A Novel Range Search Scheme Based on Frequent Computing for Edge-Cloud Collaborative Computing in CPSS

ZONGMIN CUI¹,², ZHIXING LU², HYUNHO YANG³, YUE ZHANG¹,³, AND SHUNLI ZHANG¹

¹School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China
²School of Information Management, Jiangxi University of Finance and Economics, Nanchang 330013, China
³School of Computer Information and Communication Engineering, Kunsan National University, Kunsan 54150, South Korea

Corresponding authors: Hyunho Yang (hhyang@kunsan.ac.kr) and Yue Zhang (zhangyue@jju.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61762055, in part by the Jiangxi Provincial Natural Science Foundation of China under Grant 20181BAB202014, and in part by the Key Scientific and Technological Research Project of Jiangxi Provincial Education Department of China under Grant GJJ190899.

ABSTRACT Due to the rapid advances of Information and Communication Technologies (ICT), especially 5G and Artificial Intelligence (AI), the Internet of Everything is gradually becoming a reality, and human beings living environments are becoming smarter and smarter. Every day there will be generated large amounts of data in Humans-Machines-Things hybrid space, which is also called Cyber-Physical-Social Systems (CPSSs). Today, the city we live in has become a data-driven society. However, how to effectively mine valuable information from these massive data to provide proactive and personalized services for human beings is a challenging problem. Thus, top-k search remains an important topic of ongoing research. In this paper, we focus on a basic problem of geo-tagged data: find the top-k frequent terms among the geo-tagged data in a specific region from the cloud. We first construct a Region Tree Index (RTI) for geo-tagged data. Then the list storage structure is proposed to Store Sorted Terms and Weights (SSTW) in RTI. And then an efficient kTermsSearch algorithm is presented to compute top-k frequent terms in a given region. Finally, extensive experiments verify the validity of the proposed scheme.

INDEX TERMS Cyber-physical-social system (CPSS), frequent computing, top-k search, edge-cloud collaborative computing.

I. INTRODUCTION

With incremental development of ICT, especially 5G and Internet of Things (IoT) technology, the Internet of Everything is gradually becoming a reality, and human beings living environments are becoming more and more smart and convenient [1]. The smart cities become a new hyperspace, i.e., CPSS [2].

Now more and more cyber-physical-social services appear in our daily life, such as Twitter, Instagram, Foursquare, Facebook, etc. As people use these services, huge amounts of data are generated from all aspects of the hyperspace. Nowadays, many people prefer to geo-tag their data on the Internet; for instance, social posts. In these geo-tagged data, people may talk about everything around them. For instance, what’s the decorative style of a new restaurant nearby, what are the community activities next week, etc. Spatio-temporal data, as an important data sources in CPSSs, which with great value in many commercial applications such as e-commerce, online education, sharing economy and so on. CPSSs users (e.g., marketers, government officers, etc.) can identify current trending topics within a region and adjust the products or plans accordingly. Nowadays the CPSSs have becoming data-driven society [3]. Meanwhile, blockchain technology is expected to solve the security and privacy issues of data sharing, which makes it more secure and feasible for human society to share and use the generated massive data in CPSSs.

The ultimate goal of CPSSs is to serve human beings and make our lives more and more convenient and intelligent by providing prospective and personalized services [2]. In CPSSs, how to fast and effectively mine valuable
information from these CPSS spatio-temporal data and provide proactive and personalized services for human beings is a challenging problem that needs to be solved urgently. While how to effectively and fast find the top-k frequent terms among the geo-tagged data in a specific region from the cloud, which is a basic problem in CPSSs.

As frequent terms in humans’ geo-tagged data can present the trending topics, the basic problem is that, given a user-specified region, how to find the top-k frequent terms from the geo-tagged data. Many researchers propose diverse methods to solve this problem [4]–[6]. However, there are still some limitations in existing methods. Such as [7]–[9] not consider the top-k terms search in a user-specified region, [10] has lower search accuracy, [4] with higher storage overhead and lower search efficiency.

In this paper, we propose an efficient Region Search scheme for top-k Frequent Terms (RS-kFT). In the scheme, we need to construct Region Tree Index (RTI) and list storage structure, which is named Store Sorted Term and Weight (SSTW). In RTI, each node corresponds to a specified region. The region of the parent node covers its children nodes regions. Each leaf node is associated with SSTW, where frequent terms and their weights are stored. As the index RTI is constructed based on regions, it is very applicable for the range search. When constructing a SSTW, first we extract frequent terms from the geo-tagged data in a specified region and compute the frequency for each term. Then we compute the weight of each term in the region using the frequency information. Finally, the term and its weight are stored in SSTW. As the frequency information of terms is stored in each node of the tree structure index [4], and the frequency information of the terms is only stored in the leaf nodes of RTI, the storage overhead is reduced. In the range search procedure, when a user sends a query (including a search region and an integer value k) to the cloud where stores a RTI and several SSTW. After receiving a query request, the cloud uses the proposed search algorithm to extract k most frequent terms.

In our search algorithm, to improve the efficiency of the search procedure, Zipf’s law [4] is used to calculate the number of the most frequent terms. Then the cloud extracts the most frequent terms from the STWs of the leaf nodes which are covered by the search region. Next, the cloud calculates k most frequent terms according to all the frequent terms extracted from the above leaf nodes. As (1) k most frequent terms are calculated according to the geo-tagged data information in the nodes (including internal nodes and leaf nodes) which are covered by the search region [4] and (2) k most frequent terms are only calculated according to the geo-tagged data information in the leaf nodes which are covered by the search region (our scheme RS-kFT), the search procedure in our scheme is more efficient. Additionally, as the weight information of terms in the SSTW of a leaf node is calculated by using all the frequency information of terms in the region of the leaf node, the search results are more accurate than the method in [4]. The contributions of this paper are listed as follows:

1. We design an edge-cloud cooperative computing system model of RS-kFT. Based on this model, we construct an index RTI and propose a list storage structure SSTW. As the address information of the geo-tagged data is not stored in SSTW, which can be greatly reduced the storage overhead.

2. We propose a new region search algorithm. This algorithm only needs to calculate the leaf nodes in the search area, which improves search efficiency compared with other methods.

3. We calculate the weight information of each term by using all frequency information of terms in the region of a leaf node. As the weight information is more accurate, the search results are more accurate.

4. We conduct extensive experiments to verify the proposed scheme in terms of storage overhead, search efficiency and accuracy. Experimental results show the proposed scheme has the best performance among the four methods.

The remainder of this paper is organized as follows. Section II discusses the related works. Section III presents the system model of the proposed scheme RS-kFT and the preliminaries. Section IV shows the construction process of RS-kFT, including the index RTI and the storage structure SSTW. Section V presents the search algorithm of top-k terms. Section VI presents the experiments and Section VII concludes this paper.

II. RELATED WORKS

A. TOP-K RESEARCH

Ahmed et al. [4] present a theoretical model for studying basic analytics queries on geo-tagged data, namely: for a given spatio-temporal region, how to find the most frequent terms among the social posts in the region. Using this model, the authors propose an index structure and algorithms to efficiently solve the problem of top-k spatiotemporal range queries. The index structure employs R-tree augmented by top-k Sorted Term Lists (STLs). By further optimizing the index, the storage overhead can be alleviated. Still, as the information about ordered terms must be stored all the nodes (including the leaf nodes and internal nodes) in the index, the storage overhead needs to be further reduced. Additionally, as the search algorithm for top-k terms is executed from the leaf nodes to the root, multiple rounds of calculations result in the low efficiency of the search algorithm. In Ahmed et al.’s research, the authors theoretically study and experimentally validate the ideal length of the stored term lists, and perform detailed experiments to study how to select the size of the index to achieve faster execution and smaller space requirements.

To provide error-tolerance search experiences, Hu et al. [7] study Location-Based Services (LBS), which can find the relevant Point of Interests (POIs) having similar keywords with the query. In [7], the authors introduce a function to
quantify the relevance between POIs and the query, by considering the similarity between keywords and physical distance. Then, the authors devise an effective index structure to organize the POIs and develop an efficient search algorithm based on the index. Felipe et al. [6] regard that the problems of nearest neighbor search on spatial data and keyword search on text data have been extensively studied separately. Thus, the authors expect to solve how to respond quickly and efficiently to spatial keyword queries. In [6], the authors construct an indexing structure called Information Retrieval R-tree, which combines R-tree with superimposed text signatures. Then, the authors present the algorithm that constructs and maintains the indexing structure and use it to answer top-k physical keyword queries. In the procedure of index construction, as text signature is required, the index construction is very complex and inefficient.

Web page rank is an inherently subjective matter, which depends on the readers’ interests, knowledge, and attitudes. However, there are also many objective factors in the web page rank. In [11], the authors proposed PageRank method to effectively measuring the people’s interest and attention devoted to them. PageRank method can only be used to search and rank web pages.

In the related study of top-k query, reverse spatial term queries [12], time-aware spatial keyword cover query (TSKQCQ) [13], and top-k spatio-temporal keyword query [14] all consider the location information of the text. Mamoulis et al. [15] and Zhang et al. [16] reduce the time and space consumption by strict reduction of processing objects. Meanwhile, they also reduce the consumption of network bandwidth [17]. On this basis, the access index of stored data [18] becomes a hot spot. Consequently, a kind of indexed spatio-temporal graph [19] is built based on geographical knowledge. The graph enables users to search and query spatio-temporal catalog. Liu et al. [20] proposed PruningKOSR algorithm and StarkOSR algorithm. The two algorithms combine the shortest path query index technology to find the optimal sorting route.

Our work differs from the aforementioned research in the following respects. Firstly, we strictly consider the temporal and spatial attributes of keywords. Meanwhile, we classify keywords according to the temporal and spatial information. Secondly, the keywords and related information we provide are directly placed on the list after pretreatment. When the user query, they don’t have to do too many complicated calculations, which saves keyword return time. Finally, we sort keywords according to the weight algorithm.

**B. RESEARCH ON SPATIAL PREFERENCE**

Chen et al. [21] study the top-k term search method over massive volumes of geo-tagged streaming data. The method can be widely used in discovering the most frequent nearby terms from tremendous stream data. Given a query location and a set of geo-tagged data within a sliding window, the authors propose top-k terms search approach via considering term frequency, spatial proximity, and term freshness. The authors develop a mechanism to construct quad-tree based indexing structure, indexing update technique, and best-first based searching algorithm. This method can handle streaming data well. However, the method does not consider how to handle these data which are collected from the Internet and stored in the cloud.

Li et al. [22] gave a specific spatial acquisition object approach. The system assigns certain attributes to the user who meets the specific requirements according to the spatial relationship between certain objects. However, this approach is not suitable for massive data acquisition. The massive data seriously slow down the acquisition speed. Moreover, this approach has high requirements for user devices. This problem has also been studied in the context of physically annotated web objects, where the goal is to combine both the textual content and the geolocation of web pages.

Chen et al. [23] fixed the loss of user expectation objects and simplified the optimal refinement problem into a general linear programming problem. The goal is to optimize the direction-aware direction perception of keywords in the query. Long et al. [24] propose a HOC-Tree index structure, which is based on OC-Tree and Hilbert curve. Spatio-temporal and physical attributes of data are also taken into account. Uncertain Top-k Query (UTK) is introduced to satisfy users’ preference needs in [25]. Mouratidis and Tang [25] adopt a linear scoring function to organize the data owned by users. Users who initially make precise queries may make inconclusive queries.

Flexible multi-standard application queries, points of interest, Moving Top-k Physical Keyword (MkSK) all belong to the physical preference query. Yiu et al. [26] formally define physical preference query based on existing explanations. The corresponding index technology and search algorithm are given by them. Besides considering users’ POI (Points of Interest), Tian et al. [27] also consider the scores of POI and users’ location, which makes user queries more appropriate and effective. On the other hand. Reverse top-k Geo-Social Keyword (RkGSK) query [28] has also become an important research direction of physical preference query. The query identifies various information of the object to find potential customers (such as diners) to provide system users (such as restaurant managers).

Compared with the above spatial preference queries, we only provide users with current popular and effective keywords based on the use of keywords and spatio-temporal information. In other words, we are not just for one aspect or one person. Different users can query keywords that meet their requirements according to the different query length k. Therefore, our query algorithm is not only applicable to the specific population, but also all users.

**C. KEYWORD RELATED RESEARCH**

From a statistical point of view, Aizawa [10] give the unusual word a larger frequency and increase its weight. That is, the frequency of frequency words is equal to the number of words appearing in the object divided by the total words in the
object, and then multiplied by the frequency of the inverse document. The research of [29] is LSA (Latent Semantic Analysis). In the direct analysis of word frequency, Blei [29] holds that objects are composed of topics. The words in objects are selected from topics, which have a certain probability. Goranci et al. [30] study the facility location problem of client operation to optimize the time overhead of query algorithm by maintaining the optimal solution constant factor.

Rose et al. [31] propose a Rapid Automatic Keyword Extraction (RAKE) algorithm. The algorithm can extract keywords independently with unsupervision. Tang et al. [32] automatically extract top-\(k\)’s insights from multidimensional data to meet the needs of non-professionals and analysts. Wei et al. [33] propose a top-\(k\) query algorithm to solve the problem of excessive space and time consumption in network search. Meanwhile, they also ensure the accuracy of the return results. Inches et al. [34] compare the traditional TREC collections with the user-generated data. In addition, they study their similarities and differences.

Method TextRank [35] is an unsupervised algorithm that uses only the information of the document itself for keyword extraction. First, it divides a given natural language text into a set of words or phrases. Second, it adds edges between cells based on the symbiotic relationship between cells. Third, it sorts the cells based on their scores. It does not consider the importance of some keywords to the geo-tagged data. It reduces the number of invalid keywords in the recommended keyword by deleting some wordless keywords that are not required. Therefore, TextRank certainly undermines the integrity of the geo-tagged data. Our scheme RS-kFT considers the importance of some keywords to the geo-tagged data. We guarantee the integrity of the geo-tagged data. Thus, RS-kFT has better performance than TextRank.

Method KSMT [36] divides the text into several constituent words. According to the order of the words, it sets a fixed window size. It records their co-existing relationships (i.e., two different words appear at the same time). KSMT focuses only on the importance of keywords under a window. The invalid words with a big number of co-existing relationships may appear with valid words frequently. Thus, it is easier to extract invalid words than TextRank and RS-kFT. Our scheme considers the weight of different positions of different keywords, not just counting word frequency. Thus, we have fewer invalid words than KSMT. That is, our query is more accurate than KSMT.

The traditional keyword extraction algorithms are difficult to deal with the complex and changeable data. In this paper, the proposed keyword weight algorithm synthetically considers the spatio-temporal information generated by keywords and the use of the keywords. It ensures the validity of the returned keywords without consuming too much time and storage space. Moreover, the algorithm also adapts to the complex situation of current data by considering the value of keyword weight in multi-region and multi-text.

III. SYSTEM MODEL AND AND PRELIMINARIES
A. SYSTEM MODEL
The system model of RS-kFT is shown in Figure 1. In RS-kFT, the geo-tagged data on networks are collected and stored in the cloud (step ①). The geo-tagged data are classified into several regions according to their location information. For each region, the geo-tagged data in the region should be carefully handled. The handling process is as follows. For each region, the cloud administrators first extract all the frequent terms from the geo-tagged data. The administrator then calculates the weights for all the frequent terms in a region. And then the administrator stores the frequent terms and their weights in a storage structure SSTW (step ②, see Section IV A). Finally, after all the STWs of regions have

![FIGURE 1. The edge-cloud cooperative computing system model of RS-kFT.](image-url)
been built, the administrator constructs a tree index RTI (see Section IV B) and associates each SSTW to a leaf node of the RTI. When a user submits a query which includes the search region \(R_q\) and the integer value \(k\) (step 3), the cloud executes the search algorithm (step 4) and returns \(k\) where \(k\) is the most frequent terms in the region \(R_q\) (step 5) to the user as the search results (step 6, see Section V). Moreover, if the users are not willing to pay a fee for the cloud and are willing to wait a long time for a search, then these users can also select to employ edge servers or edge end devices to perform corresponding operations.

**B. PRELIMINARIES**

For ease of exposition, we summarize the notations of this paper in Table 1.

| Notation | Explanation |
|----------|-------------|
| \(u\)    | a user      |
| \(R\)    | a region    |
| \(o\)    | an object, such as a geo-tagged social post etc. |
| \(o_R\)  | an object in a region \(R\) and \(o_R\) can be denoted by a tuple \(<\text{data}_{o_R}, \text{loc}_{o_R}\>\), where \(\text{data}_{o_R}\) is the posts, textual content etc., and \(\text{loc}_{o_R}\) is the location of the object |
| \(\text{data}_{o_R}\) | the contents in an object |
| \(\text{data}_{o_R}\) | the contents in the object \(o_R\) |
| \(\text{loc}_{o_R}\) | the location information of an object |
| \(\text{loc}_{o_R}\) | the location information of the object \(o_R\) |
| \(\text{term}_{o_R}\) | a term extracted from the data, e.g. a popular topic, a big news etc. |
| \(\text{term}_{o_R}\) | a term in the data of the object \(o_R\) |
| \(\text{SSTW}\) | a data structure which store sorted terms and weights |
| \(\text{RTI}\) | region tree index |
| \(q\)    | a query     |
| \(q_u\)  | a query submitted by a user and \(q_u\) can be denoted by a tuple \(<\text{data}_{q_u}, \text{loc}_{q_u}\>\), where \(\text{data}_{q_u}\) is the queried region by \(u\) and \(k\) is the number of most frequent terms that \(u\) expected to know |
| \(k\)    | a positive integer which is set by a user before a query |

**Definition 1 (Zipf’s Law):** In the ordered frequency list of a dataset that contains \(r\) terms, let \(p\) be the rank of a term and \(\text{freq}(p, t)\) be the frequency of the \(p\)th term. The frequency of a term is inversely proportional to its rank in the frequency list. Thus, the Zipf parameter \(c\) (which is collection specific) is given by \(c = \frac{p \cdot \text{freq}(p, t)}{t}\) [4].

According to Zipf’s law, the frequency \(\text{freq}(p, t)\) of a term at any arbitrary rank \(p\) can be computed by \(\frac{t}{p} \cdot \text{freq}(p, t)\).

**IV. CONSTRUCTION OF SSTW AND RTI**

**A. GENERATION OF SSTW**

Given a region \(R\), we show how to construct a SSTW. For convenience, we suppose that there are many objects in \(R\) and an object \(o\) can be arbitrary geo-tagged data. First, the administrator of the cloud analyzes all the objects in \(R\). The administrator extracts all the terms in these objects and calculates the frequencies for all these terms. Many methods can be used to extract terms (how to extract terms from the objects is out of the scope of this paper). Then the administrator calculates the weights for these terms. Next, the administrator stores these terms and their weights in a SSTW of \(R\). Finally, all the terms in the SSTW of \(R\) are sorted from large weight to small weight. The calculation steps for calculating the weights of terms in \(R\) are listed as follows.

First, for a term of \(o_i\) in the region \(R\) \((i = 1, 2, \ldots, N^R_o, N^R_o\) is the total number of objects in \(R\)), we use Formula (1) to calculate the average frequency of the term according to the frequency information of all terms in \(R\).

\[
\text{TNF} = \sum_{i=1}^{N^R_o} \frac{\text{count}(o_i, \text{term})}{\text{count}(o_i, \text{term})^2} \cdot \frac{N^R_o}{N^R_o}
\]

In Formula (1), \(N^R_o\) is the total number of objects in \(R\), term\(^*\) could be any term in \(R\), \(\text{count}(o_i, \text{term})\) returns the total number of term in \(o_i\) and \(\text{count}(o_i, \text{term})^*\) returns the total number of all the terms in \(o_i\).

Next, we calculate value Inverse Document Frequency (IDF) by using Formula (2) of the theory of inverse document frequency [9]. IDF denotes the importance of the term in \(R\). The IDF of term can be presented as the logarithm of the ratio of the total number of objects in \(R\) to the total number of objects which contain the term.

\[
\text{IDF} = \log\left(\frac{N^R_o}{|\{o | o \in R \land \text{term} \in o\}|}\right)
\]

In Formula (2), \(N^R_o\) is the total number of objects in \(R\), \(|\{o | o \in R \land \text{term} \in o\}|\) is the total number of the objects which are in \(R\) and contains the term.

Finally, the weight of term can be calculated by using Formula (3).

\[
\text{weight}(\text{term}) = \text{TNF} \times \text{IDF}
\]

**B. GENERATION OF RTI**

To build a RTI, the administrator should (i) give a method to divide a whole region into several small regions and (ii) construct a tree structure to organize all these regions together. The construction procedure of RTI is as follows.

First, the administrator constructs the root node of RTI. The root node is associated with the whole region. Second, the administrator constructs the child nodes under the root of RTI. Each child node of the root is associated with a small region. The small regions must be the sub-regions covered by the region of the root, and all the small regions can not intersect with each other. In the same way, the administrator can iteratively construct a whole RTI from the root node to the leaf nodes in a top-down manner. For each leaf node of RTI, it is associated with a region. In the region, there are many objects. Thus, a SSTW can be constructed according to the objects in the region of the leaf node. Finally, after all the STWs of leaf nodes have been constructed, we place STWs under the corresponding leaf nodes of RTI. For ease of exposition, we give Example 1 to illustrate the structure of RTI with SSTW and show how to compute the weight of each term.
RTI with SSTWs is finally constructed.

which are associated with

the node

is associated with

there are two sub-regions

VOLUME 8, 2020

Thus, the frequency of term

appears twice in the object

there are 4 objects

Objects

R

two regions

R

3

and

R

4

are in

R

1

is the whole region. In

R

1

, there are two sub-regions

R

2

and

R

3

. Thus, the root of RTI

is associated with

R

1

and two child nodes of the root

are associated with

R

2

and

R

3

, respectively. For the same reason,

the node

R

2

is the parent of

R

4

and

R

5

which are associated with

R

2

. Similarly, the node

R

3

is the parent of

R

6

and

R

7

which are associated with

R

3

. As shown in Figure 3, the whole

RTI with SSTWs is finally constructed.

Example 1: As shown in Figure 2, all the geo-tagged data

are in

R

1

, R

1

contains two regions

R

2

and

R

3

. R

2

contains two regions

R

4

and

R

5

. R

3

contains two regions

R

6

and

R

7

. Objects

o

1

, o

2

, o

3

and

o

4

are in

R

4

. Objects

o

5

, o

6

, o

7

and

o

8

are in

R

5

. Objects

o

9

, o

10

, o

11

and

o

12

are in

R

6

. Objects

o

13

, o

14

, o

15

and

o

16

are in

R

7

.

As shown in Figure 2, R

1

is the whole region. In

R

1

, there are two sub-regions

R

2

and

R

3

. Thus, the root of RTI

is associated with

R

1

and two child nodes of the root

are associated with

R

2

and

R

3

, respectively. For the same reason,

the node

R

2

is the parent of

R

4

and

R

5

which are associated with

R

2

. Similarly, the node

R

3

is the parent of

R

6

and

R

7

which are associated with

R

3

. As shown in Figure 3, the whole

RTI with SSTWs is finally constructed.

In Example 1, there are 7 regions and 16 objects. We suppose each object contains several terms as shown in Table 2.

For the sake of illustration, we detailed how to compute the weight of term

3

and term

4

in the region

R

4

. In Example 1, there are 4 objects

(o

1

, o

2

, o

3

and

o

4

) in

R

4

. The term

3

appears twice in the object

o

2

(note that

o

2

contains 3 terms).

Thus, the frequency of term

3

in

o

2

is

2

3

. As the term

3

appears only once in the object

o

4

, the frequency of term

3

in

o

4

is

1

7

. According to the formula (1), we can compute the TNF of term

3

in

R

4

. The TNF of term

3

in

R

4

is equal to

1

+ 3

2

. As the region

R

4

contains 4 objects and the term

3

appears twice among them, we can compute the IDF of term

3

in

R

4

by using the formula (2), which is equal to

log(4

3

)

. Finally, we can compute the weight of term

3

according to the formula (3), weight(term

3

) = 2

3

+ 1

2

×

log(4

3

) ≈ 0.1756.

Similarly, term

4

appears only once in

o

3

and there are 3 terms in

o

3

. The TNF of term

4

in

o

3

is

1

3

. As there are 4 objects in

R

4

and term

4

only appears once in these objects, the IDF of term

4

in

R

4

is

log(4

3

).

Finally, we can compute the weight of term

4

in the region

R

4

as follows

weight(term

4

) = 1

3

×

log(4

3

) ≈ 0.2006.

According to the appearing times of each term in a given region, we can compute their weights by using Formula (3) and finally construct SSTWs. In Example 1, the SSTW

R

4

, SSTW

R

5

, SSTW

R

6

and SSTW

R

7

can be calculated (as shown in Table 3 and 4).

V. SEARCH ALGORITHM FOR TOP-k TERMS

In general, if a user wants to search top-k terms in a region, the cloud must compare the weights of all terms in the region. After multiple comparisons, the cloud can finally find out k most frequent terms and send them to the user as the search results. However, the comparisons of all the terms are very time-consuming.

| TABLE 2. Objects and corresponding terms. |
| object | terms in an object |
|--------|-------------------|
| o

1

 | \{term

1

, term

2

, term

3

\} |
| o

2

 | \{term

1

, term

3

, term

4

\} |
| o

3

 | \{term

2

, term

4

, term

5

\} |
| o

4

 | \{term

2

, term

3

\} |
| o

5

 | \{term

2

, term

4

, term

5

\} |
| o

6

 | \{term

2

, term

5

, term

6

\} |
| o

7

 | \{term

1

, term

2

, term

3

\} |
| o

8

 | \{term

4

, term

5

\} |

| TABLE 3. The weights of terms in SSTW. |
| terms in SSTW | weight |
| term

4

 | 0.2006 |
| term

3

 | 0.1756 |
| term

1

 | 0.1305 |
| term

2

 | 0.1234 |
| term

5

 | 0.1003 |

| TABLE 4. The weights of terms in SSTW. |
| terms in SSTW | weight |
| term

5

 | 0.4013 |
| term

4

 | 0.2006 |
| term

3

 | 0.1756 |
| term

2

 | 0.1305 |
| term

1

 | 0.1234 |
In RS-kFT, each leaf node of RTI is associated with a SSTW which contains dozens of terms. If a user wants to search \( k \) most frequent terms in a search region, the cloud first calculates an integer value \( \lambda \) by using Zipf’s law (\( \lambda \) is much less than the total number of terms in the search region). Then, the cloud finds out all the leaf nodes covered by the search region. Next, the cloud only extracts \( \lambda \) terms in the STWs of the leaf nodes which have been picked out, and compares the weights of \( \lambda \) terms. Finally, utilizing the results of weight comparison and the search algorithm, the cloud can obtain \( k \) most frequent terms. As the compared total number of terms is decreased, the search efficiency is improved. However, as Zipf’s law is a theorem in statistics, there may be some false positive values and some true negative values in the search results. A false positive value means that three is a term in the search result, but it is not expected term by the user. A true negative value means a term is not in the search result but it is just the term expected by the user. A false positive value means that there is a term in the search result, but it is not the term that the user expects.

**A. CALCULATE \( \lambda \)**

First, we calculate the frequency of a term in search region \( R \). Assume that \( N^R_o \) is the total number of objects in \( R \) and the average object has \( x \) terms. We can compute the number of total terms in \( R \) is \( xN^R_o \). Let \( p \) be an integer variable \((p = 1, 2, \ldots) \) and \( \text{freq}(p, xN^R_o) \) be a function which returns the frequency of the \( p \)th term in \( R \). (Suppose all the terms in \( R \) are ranked from high frequency to low frequency). Let the value \( C \) be the constant in Zipf’s law. Thus, we have Formula (4).

\[
C = p \cdot \frac{\text{freq}(p, xN^R_o)}{xN^R_o} \quad (4)
\]

Then we can compute the frequency of the \( p \)th term in the region \( R \) which is equal to \( \text{freq}(p, xN^R_o) = \frac{C \cdot xN^R_o}{p} \).

As each node of RTI is associated with a region, we can suppose that a node \( A \) of RTI is associated with the search region \( R \). (In our scheme RS-kFt, the search region must be the region of a node in RTI). For ease of exposition, we suppose that there is \( h \)-layer and there are \( L_i \) nodes on the \( i \)th layer. Thus, the RTI frequency of the \( p \)th term which is a node \( L_i \) on the \( i \)th layer in the region is \( \frac{C \cdot xN^R_o}{L_i p} \). (In Zipf’s law, the constant is not totally the same for a different layer of the tree structure [4]).

Let \( B \) be a node on the \( i \)th layer in RTI and \( Z_i \) be the total number of leaf nodes under \( B \). The expectation of the \( p \)th term can be calculated as \( Z_i \frac{C \cdot xN^R_o}{L_i p} \). Under the node \( A \) (contain \( A \)), the expectation of the \( p \)th term can be calculated as

\[
\theta(p, R, xN^R_o, Z_1, Z_2, \ldots, Z_h) = \sum_{i=1}^{h} Z_i \frac{C \cdot xN^R_o}{L_i p}. \quad (5)
\]

Suppose that there are \( Z_R \) leaf nodes in the region \( R \) and set \( p \) to \( k \). Thus, the frequency of the \( k \)th term is equal to \( Z_R \frac{C \cdot xN^R_o}{k} \). If the frequencies of terms in \( R \) is equal to or bigger than the value of \( Z_R \frac{C \cdot xN^R_o}{k} \), the cloud can extract \( k \) most frequent terms in the region \( R \). The top-\( k \) search condition can be expressed as Formula (6)

\[
\lambda = \min\{\theta(p, R, xN^R_o, Z_1, Z_2, \ldots, Z_h) \leq Z_R \frac{C \cdot xN^R_o}{k} \}. \quad (6)
\]

Namely, \( \lambda \) is equal to the smallest \( p \) which satisfies the condition

\[
\theta(p, R, xN^R_o, Z_1, Z_2, \ldots, Z_h) \leq Z_R \frac{C \cdot xN^R_o}{k}.
\]

**B. SEARCH ALGORITHM**

The search algorithm (Algorithm 1) is given in this section. The algorithm first calculates the integer value \( \lambda \) for the search region \( R \). Then, the algorithm extracts \( \lambda \) terms (The weights of these terms are bigger than the weights of the other terms). Next, the algorithm merges the same terms and recalculate the weights of these terms. Finally, the search algorithm finds out \( k \) terms from the extracted terms as the search results.

To illustrate clearly, we show how to perform range search for most frequent top-\( k \) terms in Example 1. Suppose that, a user \( u \) wants to search most frequent 2 terms (It means that \( k \) is equal to 2) in the region \( R_2 \). According to Zipf’s law, the constant value \( C_2 \) of the second layer of RTI is equal to 0.1 and \( C_3 \) of the third layer of RTI is also equal to 0.1 (The value can be obtained by using the statistical data from the network). According to the formula (5), we can compute the value \( \theta(p, R, xN^R_o, Z_1, Z_2, \ldots, Z_h) = \frac{C_2 \cdot xN^R_o}{L_2 p}Z_2 + \frac{C_3 \cdot L_3 x}{L_3 p}Z_3 \). In Figure 3, we can see that, there are two nodes \( (L_3 = 2) \) on the third layer of RTI and each node contains only one leaf node on average \((Z_3 = 1)\). There is only one node \( (L_2 = 1) \) on the second layer of RTI and the node contains two leaf nodes on average \((Z_2 = 2)\). Thus, \( \theta(p, R, xN^R_o, Z_1, Z_2, \ldots, Z_h) = \frac{C_2 \cdot L_2 x}{L_2 p}Z_2 + \frac{C_3 \cdot L_3 x}{L_3 p}Z_3 \). According to the formula (6) \( \theta(p, R, xN^R_o, Z_1, Z_2, \ldots, Z_h) \leq Z_R \frac{C \cdot xN^R_o}{k} \), it is easy to know that \( Z_R \frac{C \cdot xN^R_o}{k} \). Finally, we have the inequality \( \frac{L_2 x}{L_2 p} \leq \frac{L_3 x}{L_3 p} \). Thus, the smallest integer \( p \) is 3. Namely, the value \( \lambda \) is 3. So, we extract 3 terms from SSTW_{R_4} (These terms have the biggest weights in SSTW_{R_4}) and 3 terms from SSTW_{R_5} (These terms have the biggest weights in SSTW_{R_5}). Then, we sort these extracted terms by their weights. Note that, for the same term, their weights should be added and let the sum of their weights to be the new weight of the term. Then, we sort these extracted terms from SSTW_{R_4} and SSTW_{R_5} according to their weights (as shown in Table 5).

**TABLE 5. Extracted terms from SSTW.**

| Extracted terms from SSTW_{R_4} and SSTW_{R_5} | weight |
|-----------------------------------------------|--------|
| term_1                                       | 0.5016 |
| term_2                                       | 0.2884 |
| term_3                                       | 0.1756 |
| term_4                                       | 0.1505 |

(These terms have the biggest weights in SSTW_{R_4}) and 3 terms from SSTW_{R_5} (These terms have the biggest weights in SSTW_{R_5}). Then, we sort these extracted terms by their weights. Note that, for the same term, their weights should be added and let the sum of their weights to be the new weight of the term. Then, we sort these extracted terms from SSTW_{R_4} and SSTW_{R_5} according to their weights (as shown in Table 5).
Finally, the $k = 2$ terms term4 and term1 are returned as the search results.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

In our experiments, we compare the proposed RS-kFT scheme with the methods STL [4], TextRank [35], and KSMT [36] in terms of storage overhead, search efficiency, and search accuracy. As far as we know, methods STL, TextRank, and KSMT are currently the mainstream of top-$k$ search methods.

Our experiments run on a win10 computer with dual-core 1.99GHz CPU and 32G RAM. The tested geo-tagged data are collected from the Internet by using the programming language Python.

A. COMPARISON OF STORAGE OVERHEAD

We test the storage overhead among STL, TextRank, KSMT and RS-kFT, when the depth of index is 2. The size of the tested geo-tagged data is 10M, 20M, 30M, 40M and 50M, respectively. As shown in Figure 4 (a), the storage overhead of RS-kFT is less than STL, TextRank and KSMT. This is because that the location information must be stored in the method STL, but is not need a store in our scheme RS-kFT. In STL, when the size of tested geo-tagged data increases, large amounts of location information need to be stored. In TextRank and KSMT, lots of addresses and weights need to be stored for keywords. Our scheme RS-kFT saves a lot of storage overhead by dividing the address area. For a single region, only the weight information of that area needs to be stored. Thus, RS-kFT scheme has the best advantage when it is used to handle large amounts of geo-tagged data.

We also compare the storage overhead when the index depth of STL, TextRank, KSMT and RS-kFT is 1, 2, 3, 4, 5, respectively. The size of tested geo-tagged data is 10M. For the same number of data, the deeper the index, the more nodes. The existing methods store addresses of keywords in almost all nodes. However, we do not store any address. Therefore, as shown in Figure 4 (b), we can see that the storage overhead of RS-kFT is less than STL, TextRank and KSMT.

B. COMPARISON OF SEARCH ACCURACY

STL only considers the number of appeared keywords. TextRank certainly undermines the integrity of the geo-tagged data. KSMT only considers the impact of each keyword under a small window. None of them considers the importance of a single keyword to the geo-tagged data. We consider it and guarantee the integrity of the geo-tagged data. Thus, RS-kFT has the best accuracy among the four methods in Figure 5 (a).

As shown in Figure 5 (b), our true positive terms in the returned results are about 59% on average when the integer value $k$ is set to 20. Thus, the search results of RS-kFT are more accurate than STL, TextRank and KSMT. This is because that the weight information of a term is calculated by utilizing all the frequency information of terms in a region in RS-kFT. However, the other three methods only focus on keyword weights in an object.

Figure 5 (c) displays the comparison of search accuracy among STL, TextRank, KSMT and RS-kFT when the integer value $k$ is different. When the integer value $k$ of user query increases, our scheme RS-kFT is relatively stable. Under our test datasets, the number of effective keywords recommended

---

Algorithm 1 kTermsSearch

Require: $R$, a search region queried by a user $k$, the number of frequent terms expected by a user.

Ensure: $k$ most frequent terms in the region $R$.

1: set $A$ to the root node of RTI;
2: if the users are not willing to pay for the cloud and are can wait a long time for query then
3: the system will select the appropriate edge services or end devices for these users;
4: else
5: send the query request to the cloud;
6: end if
7: for all each node in RTI do
8: if the region of $A$ is not equal to $R$ then
9: set $A$ as its children in turn;
10: end if
11: if the region of $A$ is equal to $R$ then
12: calculate the integer value $\lambda$ for $R$ by using the formula $\lambda = \min_{p}(\theta(p, R, xN_o^R, Z_1, Z_2, \ldots, Z_o) \leq Z_R C_o xN_o^R \frac{1}{k})$, in which $C_o xN_o^R \frac{1}{k}$ is the frequency of the $k$th term in the region $R$ and $Z_R$ denotes the total number of leaf nodes under the node $A$;
13: for all each child of $A$ do
14: extract $\lambda$ terms from the SSTW of the leaf node (the weights of these $\lambda$ terms are bigger than the weights of the other terms);
15: put all the extracted terms in the set $S$ and re-calculate their weights by using the following steps;
16: if a term appears in the set $S$ only once then
17: the weight of the term is equal to the original weight and does not need to be re-calculated;
18: else
19: the new weight of the term needs to be re-calculated and its new weight is equal to the sum value of its $N$ weights (suppose the term appears $N \geq 2$ times, which means the term has $N$ weights);
20: end if
21: find out the $k$ terms in the set $S$ that the weights of these $k$ terms are bigger than the other terms in the set $S$;
22: end for
23: end if
24: end for
25: return $k$ terms.
to users remains at the highest level. While the other three methods are not effective enough when the number of users' queries is small. This is because they may put a large number of invalid keywords in front of the list of frequency words recommended to the users. However, in our scheme, when the number of users' queries increases continuously, the proportion of effective keywords gradually increases, which leads to the number of invalid keywords are very limited.

C. COMPARISON OF SEARCH EFFICIENCY

We compare the search efficiencies of RS-kFT, STL, KSMT and TextRank in Figures 6 (a) and 6 (b). In these experiments, the size of the tested geo-tagged data is 10M.

In Figure 6 (a), we can see that our scheme RS-kFT is more efficient than the other three methods when the integer value $k$ is set to 200, 400, 600, 800 and 1000, respectively. In this case, the depth=1. That is, the length of keyword interception
in these four methods is consistent. As STL needs to filter the address information of the keyword, it takes more time than TextRank, KSMT, and RS-KFT. Meanwhile, TextRank and KSMT take more time than our scheme to compare addresses.

Moreover, when the depth=5, RS-KFT is more stable than the other three methods. In STL, KSMT and TextRank, more depth cases more iteration, which leads to more comparisons of addresses in the existing methods. Therefore, compared to RS-KFT, the existing methods need to consume more system time. In addition, the consumed system time may further increase with the increase of the number of user queries (i.e. k).

In summary, our scheme significantly outperforms the existing method in almost all aspects. Therefore, our scheme may be applied in many scenarios with range query, such as cloud computing [37], [38], recommendation system [39], real-time graph stream [40], ciphertext query [41], [42], privacy-preserving for location-based services [43], continuous optimization [44], energy scheduling and optimization [45], etc.

VII. CONCLUSION

In this paper, we propose an efficient spatio-temporal range search scheme for top-k frequent terms that is suitable for edge-cloud collaborative computing. We first design an edge-cloud cooperative computing system model of RS-KFT. Based on the proposed model, we construct an efficient and accurate index. Under the leaf nodes of the index, we build the RS-KFT storage structure for alleviating the storage overhead of a large amount of spatio-temporal data. Then, we optimize the method for calculating the frequency of spatio-temporal data by adding the weight parameter in the STL. In the proposed algorithm, as the size of $\lambda$ has been shortened, the amount of expected searched data is less, which can enhance search efficiency and accuracy.

REFERENCES

[1] F. Piccialli, S. Cuomo, V. S. D. Cola, and G. Casolla, “A machine learning approach for IoT cultural data,” J. Ambient Intell. Humanized Comput., vol. 2–3, pp. 1–12, Sep. 2019.
[2] X. Wang, L. T. Yang, Y. Wang, X. Liu, Q. Zhang, and M. J. Deen, “A distributed tensor-train decomposition method for cyber-physical-social services,” ACM Trans. Cyber-Phys. Syst., vol. 3, no. 4, pp. 1–15, Oct. 2019.
[3] S. Zhang, L. T. Yang, L. Kuang, J. Feng, J. Chen, and V. Piru, “A tensor-based forensics framework for virtualized network functions in the Internet of Things: Utilizing tensor algebra in facilitating more efficient network forensic investigations,” IEEE Consum. Electron. Mag., vol. 8, no. 3, pp. 23–27, May 2019.
[4] P. Ahmed, M. Hasan, A. Kashyap, V. Hristidis, and V. J. Tsotras, “Efficient computation of Top-k frequent terms over spatio-temporal ranges,” in Proc. ACM Int. Conf. Manage. Data (SIGMOD), 2017, pp. 1227–1241.
[5] L. Chen, G. Cong, C. S. Jensen, and D. Wu, “Spatial keyword query processing: An experimental evaluation,” Proc. VLDB Endowment, vol. 6, no. 3, pp. 217–228, 2013.
[6] I. De Felipe, V. Hristidis, and N. Rishe, “Keyword search on spatial databases,” in Proc. IEEE 24th Int. Conf. Data Eng., Apr. 2008, pp. 656–665.
[7] J. Hu, J. Fan, G. Li, and S. Chen, “Top-k fuzzy spatial keyword search,” Chin. J. Comput., vol. 35, no. 11, p. 2237, 2012.
[8] A. Magdy, A. M. Aly, M. F. Mokbel, S. Elnikety, Y. He, S. Nath, and W. G. Aref, “GeoTrend: Spatial trending queries on real-time microblogs,” in Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst. (GIS), 2016, p. 7.
[9] A. Skovgaard, D. Sidlauskas, and C. S. Jensen, “Scalable top-k spatio-temporal term querying,” in Proc. IEEE 30th Int. Conf. Data Eng., Mar. 2014, pp. 148–159.
[10] A. Aizawa, “An information-theoretic perspective of tf-idf measures,” Inf. Process. Manage., vol. 39, no. 1, pp. 45–65, Jan. 2003.
[11] P. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the Web,” Stanford InfoLab, vol. 98, pp. 161–172, Nov. 1999.
[12] S. Farazi and D. Rafiei, “Top-K frequent term queries on streaming data,” in Proc. IEEE 35th Int. Conf. Data Eng. (ICDE), Apr. 2019, pp. 1582–1585.
[13] Z. Chen, T. Zhao, and W. Liu, “Time-aware spatial keyword cover query,” Data Knowl. Eng., vol. 122, pp. 81–100, Jul. 2019.
[14] X. Liu, C. Wan, N. N. Xiong, D. Liu, G. Liao, and S. Deng, “What happened then and there: Top-k spatio-temporal keyword query,” Inf. Sci., vol. 453, pp. 281–301, Jul. 2018.
[15] N. Mamoulis, M. L. Yu, K. H. Cheng, and D. W. Cheung, “Efficient top-k aggregation of ranked inputs,” ACM Trans. Database Syst., vol. 32, no. 3, p. 19, Aug. 2007.
[16] D. Zhang, Y. Li, C. Jao, I. Shao, and H. T. Shen, “Augmented keyword search on spatial entity databases,” VLDB J., vol. 27, no. 2, pp. 225–244, Apr. 2018.
[17] P. Cao and Z. Wang, “Efficient Top-k query calculation in distributed networks,” in Proc. 23rd Annu. ACM Symp. Princ. Distrib. Comput. (PODC), New York, NY, USA, 2004, pp. 206–215.
[18] P. Zellweger, “The Aleph data relation in structured data, a tree within a tree visualization,” Electron. Imag., vol. 2016, no. 1, p. 1, Feb. 2016.
[19] S. Neumaier and A. Polleres, “Enabling spatio-temporal search in open works,” in Proc. 23rd Annu. ACM Symp. Princ. Distrib. Comput. (PODC), New York, NY, USA, 2004, pp. 206–215.
[20] S. Neumaier and A. Polleres, “Enabling spatio-temporal search in open works,” in Proc. 23rd Annu. ACM Symp. Princ. Distrib. Comput. (PODC), New York, NY, USA, 2004, pp. 206–215.
[21] M. Li, Y. Gu, M. Chen, and G. Yu, “Efficient processing method for reverse top-k spatial preference Queries,” J. Softw., vol. 28, no. 2, pp. 310–325, 2017.
[22] L. Chen, Y. Li, J. Xu, and C. S. Jensen, “Towards why-not spatial keyword top-k queries: A direction-aware approach,” IEEE Trans. Knowl. Data Eng., vol. 30, no. 4, pp. 796–809, Apr. 2018.
[23] J. Long, L. Zhu, C. Zhang, S. Lin, Z. Yang, and X. Yuan, “HOC-Tree: A novel index for efficient spatio-temporal range search,” in Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining, 2018, pp. 93–107.
[24] K. Mouratidis and B. Tang, “Exact processing of uncertain top-k queries in multi-criteria settings,” Proc. VLDB Endowment, vol. 11, no. 8, pp. 866–879, Apr. 2018.
[25] M. Lung Yiu, X. Dai, N. Mamoulis, and M. Vaitis, “Top-k spatial preference queries,” in Proc. IEEE 23rd Int. Conf. Data Eng., Apr. 2007, pp. 1076–1085.
[26] Y. Tian, P. Jin, S. Wan, and L. Yue, “Group preference queries for location-based social networks,” in Proc. Asia–Pacific Web Age Inf. Manage. Joint Conf. Web Big Data. Berlin, Germany: Springer, 2017, pp. 556–564.
[27] J. Zhou, Y. Gao, G. Chen, C. S. Jensen, R. Chen, and D. Cai, “Reverse top-k geo-social keyword queries in road networks,” in Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE), Apr. 2017, pp. 387–398.
[28] D. M. Blei, “Probabilistic topic models,” Commun. ACM, vol. 55, no. 4, pp. 77–84, Apr. 2012.
[29] G. Goranci, M. Henzinger, and D. Leniowski, “A tree structure for dynamic facility location,” in Proc. 26th Annu. Eur. Symp. Algorithms, Aug. 2018, pp. 1–23.
[30] S. Rose, D. Engel, N. Cramer, and W. Cowley, “Automatic keyword extraction from individual documents,” Text Mining, Appl. Theory, vol. 1, pp. 1–20, Mar. 2010.
[31] B. Tang, S. Han, M. L. Yiu, R. Ding, and D. Zhang, “Extracting Top-k insights from multi-dimensional data,” in Proc. ACM Int. Conf. Manage. Data (SIGMOD), New York, NY, USA, 2017, pp. 1500–1524.
[32] Z. Wei, X. He, X. Xiao, S. Wang, S. Shang, and J.-R. Wen, “TopPPR: Top-k personalized PageRank queries with precision guarantees on large graphs,” in Proc. Int. Conf. Manage. Data (SIGMOD), New York, NY, USA, 2018, pp. 441–456.
[33] G. Inches, M. J. Carman, and F. Crestani, “Statistics of online user-generated short documents,” in Proc. Eur. Conf. Inf. Retr. Berlin, Germany: Springer, 2010, pp. 649–652.
[35] R. Mihalcea and P. Tarau, “Textrank: Bringing order into text,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2004, pp. 404–411.

[36] Y. Chen, J. Wang, P. Li, and P. Guo, “Single document keyword extraction via quantifying higher-order structural features of word co-occurrence graph,” Comput. Speech Lang., vol. 57, pp. 98–107, Sep. 2019.

[37] B. Wu, Z. Zhao, Z. Cui, Z. Mei, and Z. Wu, “Secure and efficient adjacency search supporting synonym query on encrypted graph in the cloud,” IEEE Access, vol. 7, pp. 133716–133724, 2019.

[38] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, and L. Liu, “Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds,” IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 12, pp. 3501–3517, Dec. 2016.

[39] J. Yu, S. Na, and Z. Cui, “Maximum recommendation in Geo-social network for business,” Tech. Gazette, vol. 26, no. 2, pp. 433–440, 2019.

[40] L. Chi, B. Li, X. Zhu, S. Pan, and L. Chen, “Hashing for adaptive real-time graph stream classification with concept drifts,” IEEE Trans. Cybern., vol. 48, no. 5, pp. 1591–1604, May 2018.

[41] J. Shi, C. Huang, K. He, and X. Shen, “ACS-HCA: An access control scheme under hierarchical cryptography architecture,” Chin. J. Electron., vol. 28, no. 1, pp. 52–61, Jan. 2019.

[42] Z. Mei, H. Zhu, Z. Cui, Z. Wu, G. Peng, B. Wu, and C. Zhang, “Executing multi-dimensional range query efficiently and flexibly over outsourced ciphertexts in the cloud,” Inf. Sci., vol. 432, pp. 79–96, Mar. 2018.

[43] Z. Wu, R. Wang, Q. Li, X. Lian, G. Xu, E. Chen, and X. Liu, “A location privacy-preserving system based on query range cover-up for location-based services,” IEEE Trans. Veh. Technol., early access, Mar. 18, 2020, doi: 10.1109/TVT.2020.2981633.

[44] H. Peng, C. Deng, and Z. Wu, “Best neighbor-guided artificial bee colony algorithm for continuous optimization problems,” Soft Comput., vol. 23, no. 18, pp. 8723–8740, Sep. 2019.

[45] M. Qiu, L. T. Yang, Z. Shao, and E. H.-M. Sha, “Dynamic and leakage energy minimization with soft real-time loop scheduling and voltage assignment,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 3, pp. 501–504, Mar. 2010.

ZONGMIN CUI received the B.E. degree from Southeast University, in 2002, and the M.S. and Ph.D. degrees from the Huazhong University of Science and Technology, in 2006 and 2014, respectively. He is currently an Associate Professor with the School of Information Science and Technology, Jiujiang University. His research interests include privacy protection, personalized recommendation, information security, and data query.

ZHIXING LU received the B.E. degree from Jiujiang University. He is currently pursuing the M.S. degree with the School of Information Management, Jiangxi University of Finance and Economics. His research interests include personalized recommendation, CPSS computing, and so on.

HYUNHO YANG is currently a Professor with the School of Computer Information and Communication Engineering, Kunsan National University, Kunsan, South Korea. His research interests include deep learning, machine learning, ubiquitous/pervasive computing, and big data.

YUE ZHANG is currently pursuing the Ph.D. degree with the School of Computer Information and Communication Engineering, Kunsan National University, Kunsan, South Korea. She is also a Lecturer with the School of Information Science and Technology, Jiujiang University, Jiujiang, China. Her research interests include pervasive computing, big data, transfer learning, and network forensics.

SHUNLI ZHANG is currently a Lecturer with the School of Information Science and Technology, Jiujiang University, Jiujiang, China. His research interests include pervasive computing, big data, and CPSS design.