IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 7, 2019, accepted October 28, 2019, date of publication November 8, 2019, date of current version November 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952390

SMPKR: Search Engine for Internet of Things

JINE TANG[1], ZHANGBING ZHOU[2,3], LEI SHU[4,5], AND GERHARD HANCKE[6,7]

[1]School of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China
[2]School of Information Engineering, China University of Geosciences, Beijing 100083, China
[3]Computer Science Department, TELECOM SudParis, 91000 Évry, France
[4]College of Engineering, Nanjing Agricultural University, Nanjing 210031, China
[5]Lincoln Joint Research Center of Intelligent Engineering, Nanjing Agricultural University, Nanjing 210031, China
[6]Computer Science Department, City University of Hong Kong, Hong Kong
[7]Electronic and Computer Engineering Department, University of Pretoria, Pretoria 0002, South Africa

Corresponding author: Jine Tang (tangjine2008@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702232 and Grant 61772479, and in part by the Science and Technology Planning Project of Guangdong Province under Grant 2017A050506057.

ABSTRACT The Internet of Things (IoT) has become the infrastructure to widely support ubiquitous applications. Due to the highly dynamic context setting, IoT search engines have attracted increasing attention from both industrial and academic field to crawl and search heterogeneous data sources. Today, a large amount of work on IoT search engines is devoted to finding a particular mobile object device, or a group of object devices satisfying the constraint on query terms description. However, it still lacks studies on enabling so-called spatial-temporal-keyword-aware query. Only a few research work simply applies a keyword or spatial-temporal matching to identify object devices. In this case, it is insufficient to simultaneously consider the spatial-temporal-keyword aspect in order to satisfy the user request. To address this challenge, we develop a new search mechanism over PKR-tree (denoted *SMPKR*), in which PKR-tree unifiedly integrates spatial-temporal-keyword proximity with the help of a coding enabled index. Efficient algorithms are developed for answering range and (enhanced) KNN queries. Extensive experimental results demonstrate that our SMPKR search engine promotes the efficiency of searching for object devices with spatial-temporal-keyword constraints in comparison with the state of arts.

INDEX TERMS Internet of Things, spatial-temporal-keyword query, PKR-tree, SMPKR search engine, range and (enhanced) KNN queries.

I. INTRODUCTION

The Internet of Things (IoT) is one of the topical research disciplines nowadays, and has become the infrastructure to widely support ubiquitous applications [1]–[3]. As expected, an IoT system connects and manages huge numbers of sensors and/or monitoring devices, denoted ''object device'', which are capable of continuously providing real-time heterogeneous status data of physical objects like forests, airports, vehicles, traffic etc [4]–[6]. This setting promotes the moving object device search with spatial-temporal-keyword constraints to be a promising service in the IoT [7]–[9].

Due to increasing application demands and rapid technological advances in IoT systems, a lot of effort from both industry and academia has been put in IoT search engine [10]–[13]. Without loss of generality, an IoT search engine should have the capability of near-realtimely returning moving object devices of high relevance with respect to a certain query in spatial-temporal and keywords aspects. Until now, there are many existing works that adopt hierarchical architectures to support construction of IoT search engine. However, these works are still limited by query constraints, only supporting searches with relatively simple keyword constraint [14]–[16], static location constraint [15]–[17], or current state constraint [18]. Snoogle [15] and Microsearch [10], construct a two-tier distributed hierarchical architecture for searching with keywords constraints. Keywords are extracted from textual descriptions, which are then organized in the form of inverted files responsible for searching sensors at a local. Dyser [18], builds a two-tier centralized hierarchical architecture in support of the search for physical entities with the constraints on both keywords and user-specified current state. Lately, some researchers focus on constructing IoT search engine for query request with both spatial-temporal and keywords constraints. Serving as the core of

The associate editor coordinating the review of this manuscript and approving it for publication was Xijun Wang.

this kind of search engines, index structures apparently are the foundation. IoT-SVKSearch [19], a multimodal search engine, constructs a two-tier distributed architecture in the form of *index master server - index node server*. Each index node server consists of a set of hierarchical trees which aim to index full-text keywords or the timeslot-based dynamically moving locations of object devices. Thus, the index structure cannot fully process the spatial-temporal-keyword search of moving object devices. Though current research achieves good results for simple types of query constraints, there are still limitations to be addressed for retrieving the object devices measurements with dynamically changed content in real time and fully considering spatial-temporal-keyword searching constraints simultaneously.

To address this challenge, this paper develops a new search mechanism over PKR-tree (denoted *SMPKR*). To be specific, we present the construction procedure of SMPKR as follows. First, we give an analysis on the system structure. Second, we devise effective algorithms to construct a safe region based on the frequency of query regions. Third, we focus on designing an index structure, named PKR-tree, to resolve the problem of how to simultaneously integrate spatial-temporal-keyword proximity for object devices moving within safe regions. Finally, we devise algorithms to support range and (enhanced) KNN search of moving object devices. Comprehensive performance analysis is presented afterwards. The contributions of this paper are presented as follows:

- A three-tier hierarchical search architecture is established in IoT system, where a key-based index tree is constructed for moving object devices to promote query and communication.
- Predictive safe regions where object devices will move are determined through structuring the frequent query regions and traveling time in order to reduce the amount of sensory data to be transmitted within IoT.
- An index structure is developed to improve the search performance through simultaneously concatenating spatial-temporal-keyword binary value of moving object devices. Our query algorithms, namely range query and (enhanced) KNN query, are implemented upon this index structure.
- Extensive experiments are conducted, and evaluation results demonstrate that SMPKR search engine performs better in the efficiency of query processing for spatial-temporal-keyword-based object devices than the state of arts.

The remainder structure of this paper is given as follows. In Section II, we present the components and architecture of our search system. Section III presents how to construct the predictive safe regions. Section IV present our PKR-tree index and search algorithms. Section V reports experimental results. Section VI reviews related works. Section VII summarizes this work.

TABLE 1. Overview of notations.

Notations	Description
OD	object device
SIN	sub-index node
PIN	primary index node
HD	head nodes
FQT-tree	frequent query region mining tree
gc	spatial grid
$MP(t_{fr})$	the predictive object position at the future reference time t_{fr}
PKR-tree	index tree constructed for predictive locations of object devices
PKR_{key}	the key value indexed in STK-tree
TID	the time partition
HLV	Hilbert-Curve value
KY	keywords encoding value
PKR_{ckey}	index key value of a non-leaf node
PI	a pointer pointing to the object device's specific information
PC	a pointer pointing to child nodes
KNN	the k nearest neighbors
D_k	the estimated distance between the query location and its k'-th nearest neighbor
SK	the SMPKR search engine based on PKR-tree
CS	the Chained Structure searching scheme
SL	the Snoogle searching scheme
IS	the IoT-SVK searching scheme
FRQ	frequent range queries
KNNQ	the k nearest neighbors query

II. SYSTEM DESIGN

In this section, we present our system components, then introduce the hierarchical search architecture. The notations used in this paper are listed in Table 1.

A. SYSTEM COMPONENTS

The SMPKR search engine typically consists of three different kinds of components: object device (OD), sub-index node (SIN) and primary index node (PIN). We proceed by giving the description for these components as follows:

- *An object device* denotes a sensor or monitoring device attached to a real-world moving object. It can continuously sample the states of physical object and store the textual description about the object. Apart from that, object devices can be regarded as spatial objects and organized into several index tree structures according to their spatial-temporal-textual features, thus supporting the retrieval of spatial-temporal-keyword-based searching conditions.
- *A SIN* is in charge of maintaining sensory data gathered from object devices in its vicinity in the form of index trees. Thus, it is usually a static sensor or monitoring device placed within an area, for instance, a certain location in a forest. A SIN is typically powered by batteries, and equipped with a micro-controller and holds a great

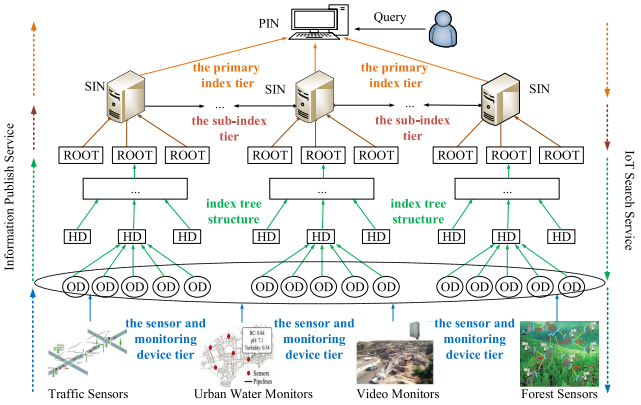

FIGURE 1. Architecture of SMPKR search engine.

deal of memory. A collection of SINs comes into being a homogeneous sensor network.

- *A PIN* has enough storage and computational capacities, and takes charge of collecting data from different SINs in the network.

B. SYSTEM ARCHITECTURE

Figure 1 shows the three-tier hierarchical searching architecture. Intuitively, three tiers in this figure establish two types of services for (i) information publish, which exploits ubiquitous sensors and monitoring devices to sense the status of physical objects, and (ii) IoT search, which facilitates the searching of physical objects when he/she initiates a query request. Through the information exchange among these three tiers, the system can achieve the searching task efficiently and accurately. The architecture and its tiers are presented as follows:

- *Sensor and monitoring device tier.* This tier is in the bottom of the hierarchy. It involves different kinds of object devices, which can continuously sense the states of physical objects.
- *Sub-index tier.* Each SIN is responsible for managing object devices located in a specific geographical region. Meanwhile, such object devices send their textual information to the specific SIN with the help of head nodes (HD) in each tree hierarchy. The corresponding SIN handles these object devices' spatial-temporal and textual information in a single tree structure so as to promote the searching at the local. Usually, the object' owner preloads or incrementally uploads its description to SIN.
- *Primary index tier.* This tier is on the top of the hierarchy. PIN gathers the aggregated object information from SINs so that it can return a group of SINs that are most likely to answer the user request. Hence, PIN can be deemed as the base station of the network, holding all the objects' information. As for the traffic, either SINs transmit information among themselves or SINs behave as relay nodes between PIN and ODs.

Without loss of generality, the system search performance depends upon two predominant costs: (i) the wireless communication cost coming from location updates among OD, SIN, PIN, and (ii) the query evaluation cost at SIN, PIN. To reduce the communication and evaluation costs, we propose an innovative approach to let the PIN maintain a predictive safe region, i.e., a rectangular area, on each moving object device. The aim of computing predictive safe region of each object device is to guarantee the current results of all query requests remain valid as long as all object devices are residing inside their respective predictive safe regions. In the following, we propose a solution to find the predictive safe regions of object devices and construct an index for them to support range and KNN queries.

III. SAFE REGION CONSTRUCTION

In this section, we first introduce frequent query regions mining. Next, we present how to construct predictive safe regions of moving object devices.

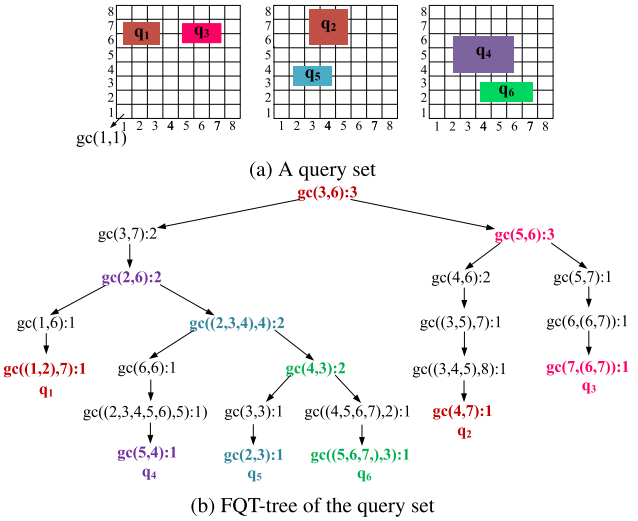

FIGURE 2. An example of FQT-tree. (a) A query set. (b) FQT-tree of the query set.

A. FREQUENT QUERY GRID CELL MINING

To facilitate the mining of frequent query region, we first divide the spatial area in terms of equal or similar grid cells, denoted as gc (e.g., the $gc(1, 1)$ as shown in Figure 2a). Then we use a FP-tree [20], [21] based method to build FQT-tree for mining frequent query region. Traditionally, FP-tree represents all information relevant to compact frequency in a database. In the following, we give the mining procedure in detail: (i) Order all frequent query grid cells according to their query times; (ii) Scan each query region and insert the corresponding frequent query grid cells into FQT-tree as a branch. Figure 2b gives an example of FQT-tree for the query set $\{q_1, \cdots, q_6\}$ shown in Figure 2a. Each node in FQT-tree is represented in the form of $\{gc(i, j) : qt\}$, where qt is the query times of grid cell $gc(i, j)$. Each branch in FQT-tree denotes

the traversal path (→) for a query region. As can be seen in Figure 2b, the start node and end node in the traversal path of each query region are marked with the same color.

Now, the constructed FQT-tree includes all the frequent query grid cells. At this case, we can transfer to mine upon FQT-tree for achieving the frequent query set. It turns out that all single-branch FQT-trees can produce the full frequent query set. Hence, given a query grid cell i', all branches that contain i' can be visited by following the linked branch passing through i'. These branches via i' compose the query set $Q \bigcup i'$, therefore the traversal obtains all frequent query grid cells in $Q \bigcup i'$. Based on the foregoing discussion, we give the main steps to build FQT-tree for $Q \bigcup i'$ at each iteration: (i) finding the frequent query grid cells; (ii) revisiting the branches of Q along the linked branch of i' and inserting the corresponding query grid cells in $Q \bigcup i'$. This iteration terminates when only one branch is contained in the new constructed FQT-tree.

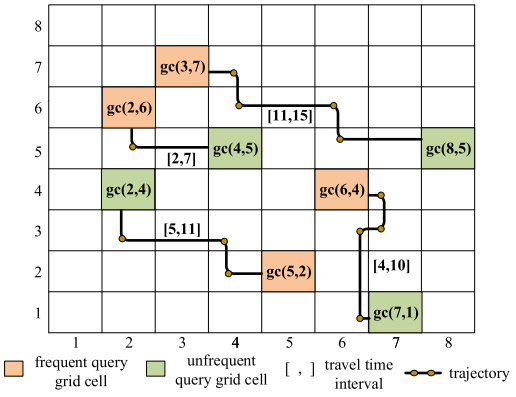

FIGURE 3. Travel time structure for frequent query grid cells.

B. PREDICTIVE SAFE REGION CONSTRUCTION

The safe region measures how far an object device can move without affecting the results of any query request. Figure 3 depicts an example about the travel time intervals from frequent query grid cells to grid cells reachable to them at time slot t, which is the underlying data structure used in achieving predictive safe regions. For instance, the travel time from $gc(2, 4)$ to frequent query grid cell $gc(5, 2)$ takes 5 to 11 minutes, while it takes 11 to 15 minutes to travel from $gc(8, 5)$ to frequent query grid cell $gc(3, 7)$ at time slot t. Given the frequent query regions and a certain time interval, we can easily find out the grid cells that are reachable to them by visiting this travel time structure [22]. The construction of *travel time structure* is done offline when moving object devices are registered. Thus, it can be fully preloaded into SMPKR search engine.

It is worth noting that, given the frequent query regions and a certain time interval, for the purpose of locating predictive safe regions of object devices in grid cells reachable to them, the key is in support of predictive location calculation for moving object devices in a future reference time relevant to this time interval. Let us step through the formalization of predictive location calculation as follows.

Definition 1 (Predictive Location Calculation): Given a moving object device represented by its location $ML(t_{ref})$ at reference time t_{ref}, its velocity ML_V, as well as a travel time interval $[t_{min}, t_{max}]$, the aim is to calculate the object location $ML(t_{fr}) = ML(t_{ref}) + ML_V(t_{fr} - t_{ref})$ at the future reference time $t_{fr} \in [t_{ref} + t_{min}, t_{ref} + t_{max}]$.

Given a travel time interval $[t_{min}, t_{max}]$ within which SQ_{rb} is a set of grid cells that are reachable to frequent query regions, we can obtain the predictive safe region for each object device $OD \in SQ_{rb}$ through calculating the minimum bounding rectangle of predictive object locations.

IV. STRUCTURE AND ALGORITHM

This section presents the procedure to construct PKR-tree for predictive safe regions, and gives algorithms of predictive range and KNN queries afterwards.

A. INDEX STRUCTURE

It is worth noting that given the object devices within the range of a SIN, constructing PKR-tree of predictive safe regions for these object devices can be converted to index their predictive locations. In this article, we aim to build PKR-tree based on B^x-tree [23], which in turn is based on B^+-tree. To make PKR-tree easy to implement in real SMPKR search systems, the PKR-value is proposed to unvaryingly support B^+-trees in the following concatenation ($\oplus$) form:

$$PKR_{key} = [TD]_2 \oplus [HV]_2 \oplus [KC]_2$$

TD is the time partition in PKR-tree. HV indicates the Hilbert-Curve [24] value of object device's location in TD. KC is the encoding value of object device' keywords (see Section IV-B).

Now, we formally give the expression form of leaf node and non-leaf node based on PKR_{key} as follows.

- *Leaf node:* $\langle PKR_{key}, ID_{OD}, gc, v_x, v_y, t, PI \rangle$. ID_{OD} is the identity of object device. gc denotes the grid cell where the object device is. (v_x, v_y) is the velocity at time instant t. PI is a pointer pointing to the specific information of object device.
- *Non-leaf node:* $\langle PKR_{ckey}, ID_{HD}, PC \rangle$. PKR_{ckey} involves all the index key values of child nodes. ID_{HD} is the identity of head node that is chosen from object devices in child nodes. PC is the pointer pointing to child nodes.

Notice that, it is similar in spirit to B^+-trees algorithms for insertion and deletion of object devices in PKR-tree.

B. KEYWORDS ENCODING

For the purpose of efficiently constructing tree index for query, we strive to encode keywords into an n-dimension binary vector through a context-free text mapping method. Then we map the vector from n-dimension to 1-dimension so as to achieve another numeric component for index construction. In this article, we adopt Hilbert Spatial Filling Curve for this mapping and obtain KC.

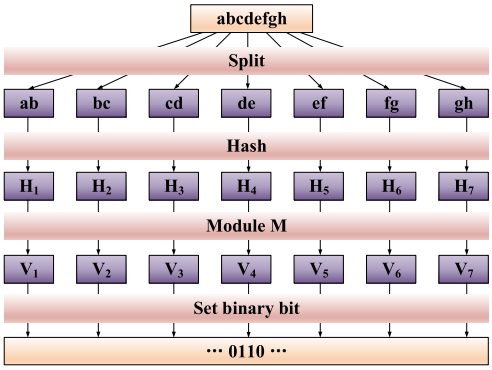

FIGURE 4. An example of pair coding.

Definition 2 (Pair Coding Function): Given a keyword $S_1 = c_1c_2\cdots c_n$, and a binary vector $S_2 = b_0b_1\cdots b_{m-1}$ $(n < m)$ where $b_i (i \in [0, m-1])$ is initialized as 0, $S_1 \rightarrow S_2$ is a pair encoding function such that $b_i = 1$ only when $H(c_j, c_{j+1}) = i$, where $H(c_j, c_{j+1}) \in [0, m-1])(j \in [1, n-1])$ is the hash value of (c_j, c_{j+1}).

We use an example in Figure 4 to explain the pair coding procedure: (i) split keyword *abcdefgh* into seven pairs as *ab, bc, cd, de, ef, fg, gh*, according to the principle that each pair of adjacent characters is joined together; (ii) set the corresponding hash values of *ab, bc, cd, de, ef, fg, gh* as $H_1, H_2, H_3, H_4, H_5, H_6, H_7$, respectively; (iii) set the value of dual encoding function as D and suppose that the binary bit length of D is M, which is initialized as 0; (iv) modulo with regards to M for $H_1, H_2, H_3, H_4, H_5, H_6, H_7$, and set the corresponding binary bit as 1 according to the model value. For example, if $M = 6$, then the initial $D = 000000$. If the model values are 1, 2, 5, 6, then $D' = 110011$.

Summary: It is observed that the storage of keywords incurs data redundancy and an n-bit string may be transmitted by a less number of bits without any information loss. Consequently, keywords can be encoded and compressed to achieve better transmission efficiency and lower processing overhead, which is more desirable for extremely resource constrained sensor nodes. Pair coding algorithm adopts lossy compression method, corresponding to setting binary bit digits of keywords. Due to the bit number of keywords being the specific value obtained through data compression, it is not changed with the length of keywords. Although lossy compression consumes a certain amount of accuracy, conforming to bit matching is only a necessary condition for keywords matching. Hence, the search algorithm, which takes a filtering approach that filters most of object devices whose keywords are not consistent with query conditions, can reduce the complexity of keywords matching.

C. DISCUSSION

Some one may concern whether PKR_{key} is reasonably designed for index construction. The foregoing discussion indicates PKR_{key} assigns higher priority to sequence values than to location mapping values. This design is attractive since the object devices related to user request are usually much less than unrelated object devices. Using PKR_{key}, object devices that have spatial-temporal-keyword related to each other will tend to be stored closely, which reduces the cost of processing spatial-temporal-keyword-aware queries. In general, the closer the PKR_{key} values, the more similar the object devices, and thus, the more efficiently the matched nodes are accessed. In light of advantages existed in our SMPKR search engine, predictive range and KNN query algorithms are developed upon PKR-tree to improve the efficiency of object devices searching task.

D. PREDICTIVE RANGE QUERY

A *Predictive Range Query* aims to retrieve all object devices predicted to be inside a query region Q_r after a time period Q_t. Broadly speaking, when a predictive range query comes, the following steps are executed on SMPKR search engine:

- The user first queries PIN, that returns SINs which may include object devices relevant to query request.
- Upon the returned list of SINs, the user performs a distributed range query along PKR-tree to find a qualified answer (see Algorithm 1).

Algorithm 1 RangeQuery

Require: RT: the root of PKR-tree
Q_r: the query range
Q_{ky}: the query keywords
Q_t: the query time interval
Ensure: QR: the query result

1: $QKC \leftarrow$ the encoding value for Q_{kC}
2: $[QTD_{min}, QTD_{max}] \leftarrow$ the time partition interval in Q_t
3: $\{[QHV_{sk}, QHV_{ek}]\} \leftarrow$ HLVconvert(Q_r)
4: **foreach** $QTD \in [QTD_{min}, QTD_{max}]$ **do**
5: **foreach** $\vartheta \in \{[QHV_{sk}, QHV_{ek}]\}$ **do**
6: $[Q_{skey}, Q_{ekey}] \leftarrow [QTD \oplus \vartheta \oplus QKC]$
7: **end for**
8: **end for**
9: Queue.push(RN_{set}, RT)
10: **while** $RN_{set} \neq \emptyset$ **do**
11: $rn \leftarrow$ Queue.pop(RN_{set})
12: **if** rn is a leaf node and passes query conditions evaluation **then**
13: $QR \leftarrow QR \cup rn$
14: **else if** $[Q_{skey}, Q_{ekey}] \cap rn.STK_{ckey} \neq \emptyset$ **then**
15: $CL_{set} \leftarrow$ the child nodes of rn
16: Queue.push(RN_{set}, CL_{set})
17: **end if**
18: **end while**

Algorithm 1 presents main steps of the range query in each SIN local region. Initially, we find the encoding value for query keywords detailed in Section IV-B (line 1). For each query time partition, we obtain the interval range of query key through the aggregation of binary values from time partition (line 2), a set of 1-dimensional Hilbert value intervals (line 3),

and keyword encoding (line 4-8). The node key range is used by the head nodes in each level of PKR-tree to carry on judgment and pruning (line 14), until leaf nodes satisfying the range query requirement are found (line 12-13). The time complexity of traversing PKR-tree is $O(n_{qp} \times n_{ql} \times n)$, where n_{qp} is the number of query time partitions, n_{ql} is the number of query range intervals, and n is the number of object devices. Usually, n_{qp} and n_{ql} are constant ($n_{qp} \ll n, n_{ql} \ll n$). Therefore, the time complexity of Algorithm 1 can be derived as $O(n)$.

E. PREDICTIVE KNN QUERY

A *predictive KNN query* aims to retrieve k nearest object devices predicted to be with the highest probability to show up around query point Q_l after a time period Q_t. Similar to predictive range query, when a predictive KNN query comes, the following steps are executed on SMPKR search engine:

- The user first queries PIN, that returns a ranked list of SINs which may include KNN object devices according to distance constraints.
- Then, the user performs a distributed KNN query from the returned list of SINs to retrieve k qualified answers (see Algorithm 2).

Algorithm 2 KNNQuery

Require: RT: the root of PKR-tree
Q_l: the query location
Q_{ky}: the query keywords
Q_t: the query time interval
k: the number of nearest neighbors
Ensure: QR: the query result
1: $D_k = \frac{2}{\sqrt{\pi}}[1 - \sqrt{1 - (\frac{k}{N})^{\frac{1}{2}}}$
2: $r_q = \frac{D_k}{k}$
3: $flag \leftarrow 1$
4: **while** $flag$ **do**
5: $R \leftarrow GetRange(Q_l, r_q)$
6: $nu \leftarrow$ RangeQuery(RT, R, Q_{ky}, Q_t)
7: $QR \leftarrow QR \cup nu$
8: **if** k neighbors are found **then**
9: $R \leftarrow GetRange(Q_l, r_k)$
10: $nu \leftarrow$ RangeQuery(RT, R, Q_{ky}, Q_t)
11: $QR \leftarrow QR \cup nu$
12: $flag \leftarrow 0$
13: **else**
14: $r_q \leftarrow Next_{radius}()$
15: $flag \leftarrow 1$
16: **end if**
17: **end while**

It is noteworthy that KNN query is a gradual extended range query. It starts with an initial search radius r and extends r by an increment until KNN results are returned. Both r and increment are set to D_k/k as that in [23], where D_k is the estimated distance between the query location and its k'-th nearest safe region. We have the following equation to estimate D_k with N as the total number of locations in a unit space:

$$D_k = \frac{2}{\sqrt{\pi}}[1 - \sqrt{1 - (\frac{k}{N})^{\frac{1}{2}}} \quad (1)$$

To answer KNN query, HV and KC values for each time partition TD are incorporated to form search space. We also explore the search order in search space so as to obtain the query result as soon as possible.

- *Search Space.* Suppose it requires n rounds of range enlargement, for each round i, we denote starting and ending points of the corresponding one-dimensional search interval by HV_{si} and HV_{ei}, respectively. Then, the search space of n rounds can be given by:

$$\{TD \oplus KC \oplus [HV_{si}, HV_{ei}], i \in [1, n]\}$$

- *Search Order.* Observe that $[HV_{si}, HV_{ei}]$ with smaller i has a shorter spatial distance to the query location. Therefore, $TD \oplus KC \oplus [HV_{si}, HV_{ei}]$ with smaller i has the priority to search firstly.

On the basis of Algorithm 1, Algorithm 2 presents the KNN query procedure in each local SIN region. Initially, we obtain the query radius through computing the estimated distance between the query location and the k'-th nearest neighbor (line 1-2). In each time partition, the funtion *GetRange()* constructs the search range centered at the query location (line 5). *RangeQuery()* is used to retrieve query results within the search range R (line 6). If KNN results are returned, the query range will be refined by calculating the distance between the query location and the k-th nearest neighbor found so far (line 8-11). In case less than k neighbors are found, the query radius is enlarged to start a new round of search (line 14). The time complexity of Algorithm 1 is $O(n)$, since the time complexity of traversing the PKR-tree for k nearest neighbors is $O(n_{qnd} \times n)$, where n_{qnd} is the number of query rounds and n is the number of object devices. Usually, n_{qnd} is a constant and $n_{qnd} \ll n$. Therefore, the time complexity of Algorithm 2 can be derived as $O(n)$.

F. ENHANCED KNN QUERY

Notice that KNN query results are the object devices sorted based on their spatial-temporal-keyword relevance scores. At this case, Algorithm 2 is not much enough when ranking object devices by seamlessly combining their spatial, temporal and keywords features. In this article, we consider to compute the temporal-keyword relevance [25], [26] along with the spatial relevance to achieve the overall relevance score and return the k most relevant object devices.

Keywords Relevance: For simplicity, we consider the following ones to calculate keywords relevance: (i) The inverse document frequency (idf). It is quantified by measuring the inverse of frequency of keyword kwd appearing in object devices' documents; (ii) The term frequency (tf). It is quantified by the raw frequency of kwd inside an object device's document d; (iii) The document length. It is quantified by

measuring the total number of keywords in the document. In this article, we adopt a simple and very common formula to evaluate the keywords relevance between an object device's document d and a query q as follows.

$$SMI_{d,q} = \frac{\sum_{kwd} W_{d,kwd} W_{q,kwd}}{W_d W_q} = \frac{\sum_{kwd} \ln(1+\overline{f_{d,kwd}}) \ln(1+\frac{n}{f_{kwd}})}{\sqrt{\sum_{kwd} W_{d,kwd}^2} \sqrt{\sum_{kwd} W_{q,kwd}^2}} \quad (2)$$

where n is the number of object devices' documents. $f_{d,kwd}$ indicates the frequency of kwd in d. $\overline{f_{d,kwd}} = \frac{f_{d,kwd}}{max(f_{d,kwd})}$ is the normalized $f_{d,kwd}$. f_{kwd} is the number of documents containing kwd. $W_{d,kwd}$ captures the tf score while $W_{q,kwd}$ captures the idf score. W_d represents document length and W_q stands for query length.

Temporal Relevance: Each timespan in the object device's document can be associated with a series of time cells. As such, we can use the analogous ideas used in keywords relevance to determine the temporal relevance between an object device's document d and a query q as

$$SMI'_{d,q} = \frac{\sum_c W_{d,c} W_{q,c}}{W'_d W'_q} = \frac{\sum_c \ln(1+\overline{f_{d,c}}) \ln(1+\frac{n}{f_c})}{\sqrt{\sum_c W_{d,c}^2} \sqrt{\sum_k W_{q,c}^2}} \quad (3)$$

where n is the number of object devices' documents. $f_{d,c} = \frac{T_d \cap c}{c}$ indicates the frequency of time cell c in d, which is measured by the area of overlap between the document timespan T_d and c divided by the area of c. $\overline{f_{d,c}} = \frac{f_{d,c}}{max(f_{d,c})}$ is the normalized $f_{d,c}$. f_c is the number of documents containing c. $W_{d,c}$ captures the tf score while $W_{q,c}$ captures the idf score. W'_d represents document length and W'_q stands for query length.

Spatial Relevance: The spatial relevance of an object device's document d is represented by the inverse of the distance between the center location of d and that of spatial scope s, given as follows.

$$SMI''_{d,q} = \frac{1}{dist(Center_d, Center_s)} \quad (4)$$

Spatial-Temporal-Keyword Relevance: Based on Eq. 2, Eq. 3 and Eq. 4, the aggregation function F can be used to calculate the spatial-temporal-keyword relevance entirely.

$$F = \alpha \cdot SMI_{d,q} + \beta \cdot SMI'_{d,q} + (1-\alpha-\beta) \cdot SMI''_{d,q} \quad (5)$$

G. PERFORMANCE ANALYSIS

This section gives a cost model to predict the performance of our IOT search engine leveraging PKR-tree. Notice that index I/O cost comes from the node traversal to visit nodes that may contain satisfactory object devices. To facilitate this discussion, we consider a total search range A as a constant that includes $|L|$ locations, the tree fanout f as a constant, as well as a query scope Q_s. The cost function is presented as follows:

$$IO_{tree} = O(\frac{|Q_s|}{A} \sum_{h=0}^{\log_f |L|-1} f^h) \approx O(|Q_s||L|) \quad (6)$$

Eq. 6 indicates that the tree traversal cost is mainly dependent on the size of Q_s and the size of PKR-tree, which, in turn, is affected by $|L|$.

TABLE 2. Parameters and values.

Parameters	values
keywords number	10
keywords length	20
frequent queries number	100
object devices number	1000, 3000, 5000, 7000, 9000
query range length	10, 20, 30, 40, 50, 60
k	100, 200, 300, 400, 500

V. IMPLEMENTATION AND EVALUATION

In this section, we evaluate the performance of our proposed SMPKR search engine, including query response time, query accuracy, and message complexity. We mainly focus on object devices and SIN interaction because object devices, SIN and head nodes communicating between them are power constrained and computationally challenged devices, while PIN is a resource-rich device. This makes the performance of object devices, SINs and head nodes crucial for the validity of the searching system.

A. EXPERIMENTAL ENVIRONMENT

Synthetic datasets of 10,000 moving object devices are produced from the Network-based Generator [27]. We extract a road network map from check-in records over 10,000 Beijing taxi cabs and divide the corresponding spatial region into $N \times N$ squared grid cells. The grid width is set according to the minimum and maximum step length it takes from moving object devices randomly distributed. Each grid cell is indicated by a Hilbert value and stores the object devices updated in it. Therefore, all record information about an object device formulate a trajectory in a sequence of grid cells series. Notice that the check-in records of each taxi cab contain its ID, grid coordination, as well as the check-in time. We can obtain the travel time interval mentioned in Section III-B by calculating the average minimum and maximum time that is taken by the object devices moving among the grid cells at time interval t. Hence, we can efficiently make full the travel time data structure before starting experiments offline. Besides, we randomly choose documents from a real dataset such as China Daily [28] and attach them to each object device as keywords. All the index structures and algorithms are implemented by Java. All experiments are conducted on a laptop with an Intel(R) Core(TM) i5-5200U CPU 2.20GHZ, 8G memory, 64 bits operating system. The parameters applied in the experiment are listed in Table 2.

To evaluate the SMPKR (SK) search performance, the following searching methods are chosen as the comparative benchmarks:

- *Chained Structure* (*CS*): A set of object devices connected in the form of chained structure.
- *Snoogle* (*SL*) [15]: An information retrieval system constructed on sensor networks for the real world.
- *IoT-SVK* (*IS*) [19]: A real-time multimodal search engine mechanism implemented for the Internet of Things.

B. PREPROCESSING TIME

Initially, we study the preprocessing time of SMPKR search engine, including, (i) keywords encoding time, (ii) safe region construction time, and (iii) PKR-tree construction time. This one-time processing is done offline when moving object devices and users' queries are registered. The experimental result is shown in Figure 5. Generally, keywords encoding time increases linearly along with the number of keywords. We also observe that the safe region and PKR-tree construction time are very efficient, because it takes only about 7 seconds and 2200 seconds respectively for the construction purpose.

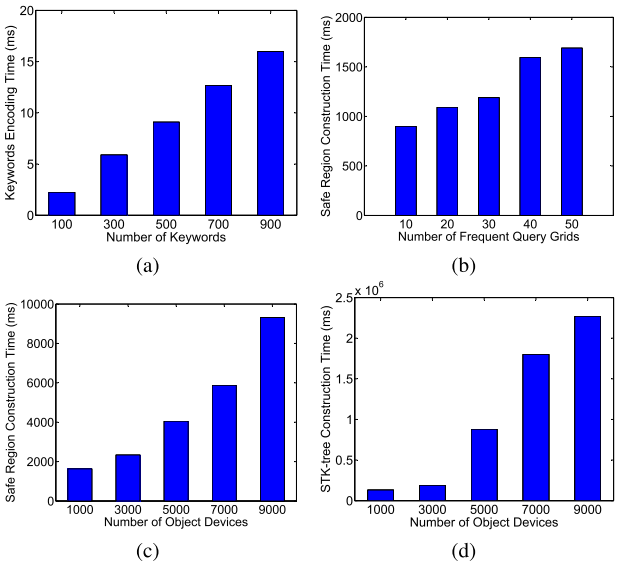

FIGURE 5. Preprocessing time.

C. SMSTK QUERY PERFORMANCE

Figure 6a and 6b illustrate the range query time. Generally, it increases along with the number of object devices and query range length. As presented in the construction procedure of PKR-tree, the object devices are organized according to their spatial-temporal-keyword proximity. Therefore, it requires to retrieve all object devices inside searching key ranges. This induces the increase in query response time.

Figure 6c and 6d illustrate the KNN query time. Generally, it tends to increase with the dataset of object devices and the k value. This is due to the fact that KNN queries are dependent

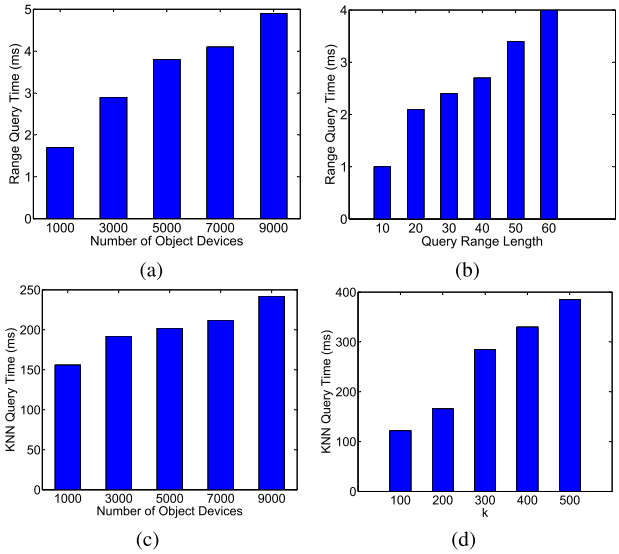

FIGURE 6. SMSTK query time.

on their searching rounds, i.e, the size of k, and the range query time in each round.

D. QUERY PERFORMANCE COMPARISON

1) FREQUENT QUERY

Figure 7 shows the comparison of query time and query accuracy for these four searching schemes when query ranges are conducted for 100 times and an average value is adopted. Generally, SMPKR performs the best on query processing time as shown in Figure 7a. This is because constructing and indexing safe regions from frequent query set can guarantee higher query stability and query accuracy.

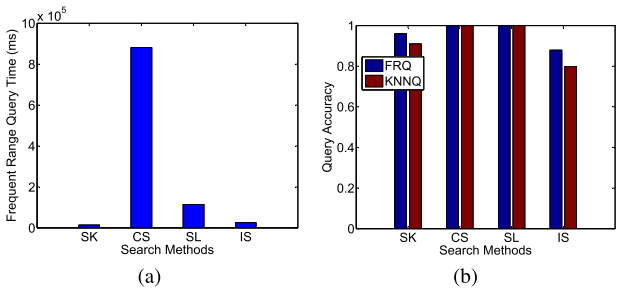

FIGURE 7. Query performance.

In this article, we estimate the query accuracy for the frequent range query and KNN query as follows:

$$QueryAccuracy = \frac{|QueryResult - ActualResult|}{ActualResult}$$

We set the object devices number as 1000 and k as 10 to test query accuracy. Figure 7b shows the estimated query accuracy of 100 frequent range queries (FRQ) and KNN query (KNNQ). The main observations are as follows: (i) *CS* and *SL* perform the best in query accuracy; (ii) Next only to them is our SMPKR searching scheme; (iii) *IS* does

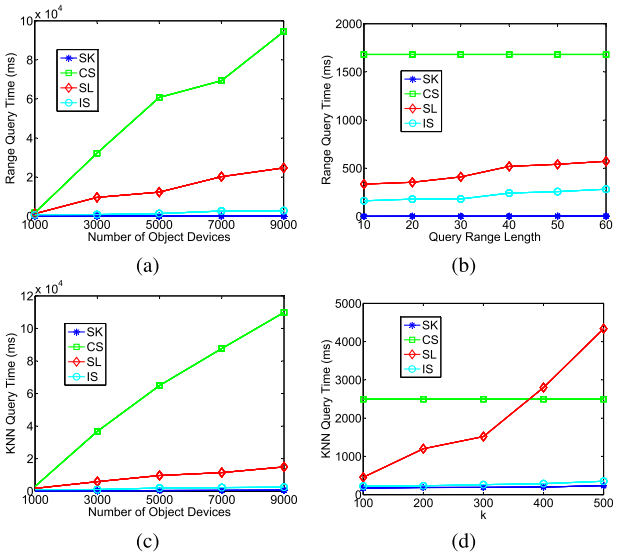

FIGURE 8. Query time.

the worst. The reason is that (i) safe region is constructed from the estimated prediction positions in SMPKR and (ii) index is constructed from the mapped grid centers in *IS*. Experimental results show that SMPKR and *IS* generate less query time at the cost of losing some query accuracy. Generally, our SMPKR searching scheme outperforms other searching schemes due to a better trade-off between query accuracy (almost one hundred percent) and query time (more than two seconds).

2) QUERY WITH VARIOUS PARAMETER SETTINGS

Figure 8 gives range and KNN query time performance for these four searching schemes with different parameter settings. Without loss of generality, we assume that query and object device have a high matching degree on keywords.

This figure shows that SMPKR searching scheme performs the best, benefiting from the index and pruning of concatenated key value on spatial, temporal and keywords. Generally, the pruning performance is the dominant factor for the query process, and hence, has more impact on the query performance. We also notice that a better searching scheme consists of two components: (i) the number of pruning judgement conditions, named as NPC (including spatial, temporal and keywords constraints), and (ii) the strength of pruning hierarchy, named as SPH. Consequently, we present pruning performances of these four searching schemes in Table 3. Without loss of generality, the less number of the pruning judgment conditions a searching scheme sets up, the stronger the pruning hierarchy a searching scheme constructs, and the better the query performance that the searching scheme can achieve. Besides, these four searching schemes take more time for range and KNN queries as the number of object devices increases. The reason is similar to what we have presented for previous experiments. In addition, the cost of these four searching schemes except *CS* tends to increase

TABLE 3. Pruning performance.

Searching scheme	NPC	SPH
STK-tree	1	strong
Chained Structure	3	weak
Snoogle	2	medium
IoT-SVK	2	strong

when the query range length and k value increase, whereas *CS* maintains a constant performance. In fact, *CS* needs to search all the object devices no matter how long the query range length is and how many ranked object devices are accessed.

3) MESSAGE COMPLEXITY

To further compare these four searching schemes, this set of experiment implements our range and KNN query schemes to test message complexity. In what follows, we explain how to determine message complexity: (i) a query sent to SIN, or transmitted to PIN or to object devices is counted as one message unit; (ii) an answer with m retrieved object devices is counted as m message units because the message length increases along with m.

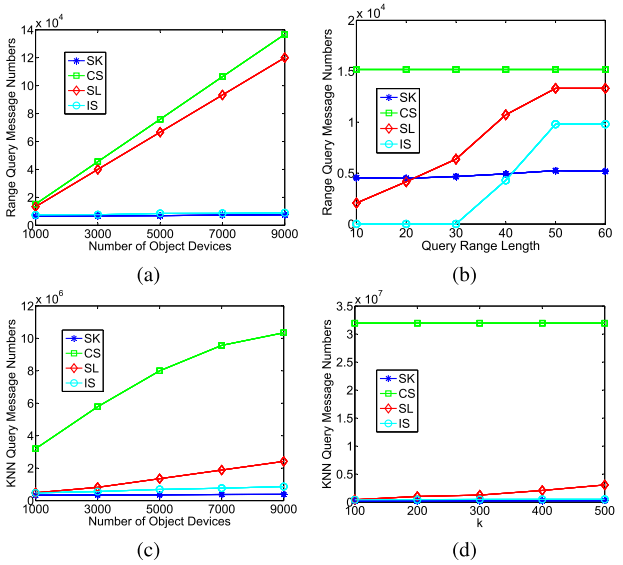

FIGURE 9. Message complexity.

Figure 9 shows message complexity of these four searching schemes. We find that all these schemes show an increasing cost in message complexity by varying object devices number, as the number of PINs and object devices overlapping or within the same query range grows with object devices number. We also observe that our SMPKR search engine performs the best, mainly because of the better pruning performance, which contributes to less messages to be forwarded. Next, we evaluate query message complexity by changing the query range length and k. The performance trend of these four searching methods except *CS* increases with the query range length and k. Nonetheless, *CS* performs

the worst and the cost increment is almost zero since it needs to query all the object devices regardless the increasing in the query range length and k. Further, *SL* and *IS* perform better than our SMPKR searching scheme for smaller query range length. In fact, a smaller query range includes less or no object devices, and thus less messages are to be forwarded. However, our SMPKR searching scheme needs less forwarding messages as the increasing of query range length, since more object devices may be included in a bigger query range, and thus *Sl* and *IS* do worse when object devices provide a larger query result set for a larger query range length. In terms of k impact, our SMPKR searching scheme performs the best while *CS* does the worst, due to the same reason for why the query time is dependent on the pruning performance, shown in Table 3. Generally, the efficient pruning ability promotes the capability of our SMPKR searching scheme to achieve better query performance.

VI. RELATED WORK AND COMPARISON

In this section, we give a review of existing research works in IOT search engine and spatial-temporal-keyword query.

A. IOT SEARCH ENGINE

Yap *et al.* [16] design MAX system, which uses tags instead of sensors to sense the physical entities and store their textual information. MAX system is well-suited for queries that are frequently changed because of the pull mode adopted. However, a high communication overhead occurs since it requires to broadcast the messages to each tag and sub station. Wang *et al.* [15] propose Snoogle for pervasive environments search, in which the entity is described by a set of keywords and saved in sensor nodes. A two-tier hierarchical architecture is constructed for querying matched entities via keywords. However, it is difficult to apply Snoogle in large-scale network environment since the data transmission mode is inefficient. In general, the search processes of these systems exploit keyword matches for sensors static information. They are not well suitable to retrieve the dynamically sampled sensory values in large-scale network environments. Ostermaier *et al.* [18] propose a real-time search engine, namely Dyser, which presents the real-world objects and sensors by Web pages. The predictive mechanism proposed, working well in searching efficiency, makes the search engine applied in networking environment that is resource constrained. However, the index is still relevant to keywords and can only retrieve the latest states of physical objects. Ding *et al.* [19] propose a hybrid real-time search engine, called IoT-SVK, upon spatial-temporal, and keyword conditions. IoT-SVK tends to use the same grid regions to denote the original curve path, thus having the capability of supporting dynamic location changes of sensors. However, it is still a combination structure by separately considering two aspects (i) spatial-temporal feature and (ii) keywords, which has an influence on searching efficiency in certain extent.

Generally, current proposals can hardly facilitate mobile object devices search in IoT. This paper proposes SMPKR searching engine that can well improve the searching efficiency, since it adopts an integrated key value in building the spatial index.

B. SPATIAL-TEMPORAL-KEYWORD QUERY

Li *et al.* [26] propose a spatial index called IR-tree, which supports document retrievals leveraging the unified representation of textual and spatial relevances. However, it does not consider the time factor. Khodaei *et al.* [25] consider the time factor and develop a hybrid index structure along with textual aspect of documents in a unified inverted list manner. However, it does not consider the spatial factor. Nepomnyachiy *et al.* [29] further consider both the time and spatial-keyword aspects. They propose to shrink temporal searching space by time-stamped data. Thereafter, they employ a shallow tree to search over spatial and textual dimensions. Mehta *et al.* [30] propose two hybrid indexes to support spatial-temporal and spatial-keyword queries, which further incorporate keyword information and temporal dimension to realize spatial-temporal-keyword query. Unfortunately, the two methods proposed are still required to treat the three factors into two structures. Hoang-Vu *et al.* [31] handle keywords, time and space features within a single index structure for efficiently answering spatial-temporal-keyword queries. However, mapping the keywords into numbers is strictly monotone for constructing spatial partitioning structure.

In general, most of current index technologies adopt *combined structures* to treat one or several aspects of space, time and text issues. Instead, we propose a *single index structure* named PKR-tree, which explores the concatenated key value of spatial, temporal and textual components. Therefore, our SMPKR can improve the retrieve performance significantly by key pruning.

VII. CONCLUSION

In this paper we have proposed an innovative IOT search engine SMPKR over mobile object devices. Generally, a PKR-tree index structure is constructed through applying safe region and keyword encoding strategy to establish concatenated spatial-temporal-keyword key values. Predictive range and (enhanced) KNN query algorithms are developed over our SMPKR search engine, which also supports frequent query operations. Extensive experiments have been conducted, and the results show that this SMPKR search engine improves the search efficiency for spatial-temporal-keyword-aware object devices in comparison with the state of arts.

REFERENCES

[1] S. Li, L. Xu, and S. Zhao, "The Internet of Things: A survey," *Inf. Syst. Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[2] B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitous data accessing method in IoT-based information system for emergency medical services," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1578–1586, May 2014.

[3] F. A. Turjman and S. Alturjman, "Context-sensitive access in industrial Internet of Things (IIoT) healthcare applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2736–2744, Jun. 2018.
[4] F. Zhang, M. Liu, Z. Zhou, and W. Shen, "An IoT-based online monitoring system for continuous steel casting," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1355–1363, Dec. 2016.
[5] T. Qiu, K. Zheng, M. Han, C. L. P. Chen, and M. Xu, "A data-emergency-aware scheduling scheme for Internet of Things in smart cities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2042–2051, May 2018.
[6] E. Park, Y. Cho, J. Han, and S. J. Kwon, "Comprehensive approaches to user acceptance of Internet of Things in a smart home environment," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2342–2350, Dec. 2017.
[7] A. Shemshadi, Q. Z. Sheng, and Y. Qin, "ThingSeek: A crawler and search engine for the Internet of Things," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2016, pp. 1149–1152.
[8] P. Barnaghi and A. Sheth, "On searching the Internet of Things: Requirements and challenges," *IEEE Intell. Syst.*, vol. 31, no. 6, pp. 71–75, Nov./Dec. 2016.
[9] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha, "An Internet of Things framework for smart energy in buildings: Designs, prototype, and experiments," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 527–537, Dec. 2015.
[10] C. C. Tan, B. Sheng, H. Wang, and Q. Li, "Microsearch: A search engine for embedded devices used in pervasive computing," *ACM Trans. Embedded Comput. Syst.*, vol. 9, no. 4, 2010, Art. no. 43.
[11] H. Ma and W. Liu, "A progressive search paradigm for the Internet of Things," *IEEE MultimediaMag.*, vol. 25, no. 1, pp. 76–86, Jan./Mar. 2018.
[12] Y. Zhou, S. De, W. Wei, and K. Moessner, "Search techniques for the Web of Things: A taxonomy and survey," *Sensors*, vol. 16, no. 5, p. 600, 2016.
[13] S. Pattar, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "Searching for the IoT resources: Fundamentals, requirements, comprehensive review, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2101–2132, 3rd Quart., 2018.
[14] K. Aberer, M. Hauswirth, and A. Salehi, "Infrastructure for data processing in large-scale interconnected sensor networks," in *Proc. Int. Conf. Mobile Data Manage.*, 2007, pp. 198–205.
[15] H. Wang, C. C. Tan, and Q. Li, "Snoogle: A search engine for pervasive environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 8, pp. 1188–1202, Aug. 2010.
[16] K.-K. Yap, V. Srinivasan, and M. Motani, "MAX: Human-centric search of the physical world," in *Proc. 3rd Int. Conf. Embedded Netw. sensor Syst.*, 2005, pp. 166–179.
[17] A. Kansal, S. Nath, J. Liu, and F. Zhao, "SenseWeb: An infrastructure for shared sensing," *IEEE MultimediaMag.*, vol. 14, no. 4, pp. 8–13, Oct. 2007.
[18] B. Ostermaier, K. Römer, F. Mattern, M. Fahrmair, and W. Kellerer, "A real-time search engine for the Web of Things," in *Proc. IEEE Conf. Internet Things*, Nov./Dec. 2010, pp. 1–8.
[19] Z. Ding, Z. Chen, and Q. Yang, "IoT-SVKSearch: A real-time multimodal search engine mechanism for the Internet of Things," *Int. J. Commun. Syst.*, vol. 27, no. 6, pp. 871–897, 2014.
[20] J. Han, J. Pei, and Y. Yiwen, "Mining frequent patterns without candidate generation," *ACM SIGMOD Rec.*, vol. 29, pp. 1–12, Jun. 2000.
[21] G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-trees," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 10, pp. 1347–1362, Oct. 2005.
[22] A. M. Hendawi, M. Ali, and M. F. Mokbel, "Panda: A generic and scalable framework for predictive spatio-temporal queries," *GeoInformatica*, vol. 21, no. 2, pp. 175–208, 2017.
[23] C. S. Jensen, L. Dan, and B. C. Ooi, "Query and update efficient B+-tree based indexing of moving objects," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2004, pp. 768–779.
[24] R. Uddin, C. Ravishankar, and V. Tsotras, "Indexing moving object trajectories with Hilbert curves," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2018, pp. 416–419.
[25] A. Khodaei, C. Shahabi, and A. Khodaei, "Temporal-textual retrieval: Time and keyword search in Web documents," *Int. J. Next-Gener. Comput.*, vol. 3, no. 3, pp. 288–312, 2012.
[26] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D.-C. Lee, and X. Wang, "IR-tree: An efficient index for geographic document search," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 4, pp. 585–599, Apr. 2011.
[27] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.
[28] [Online]. Available: http://www.chinadaily.com.cn/
[29] S. Nepomnyachiy, B. Gelley, W. Jiang, and T. Minkus, "What, where, and when: Keyword search with spatio-temporal ranges," in *Proc. Workshop Geograph. Inf. Retr.*, 2014, pp. 2:1–2:8.
[30] P. Mehta, D. Skoutas, and A. Voisard, "Spatio-temporal keyword queries for moving objects," in *Proc. 23rd SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2015, Art. no. 55.
[31] T.-A. Hoang-Vu, H. T. Vo, and J. Freire, "A unified index for spatio-temporal keyword queries," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 135–144.

JINE TANG received the Ph.D. degree from the China University of Geosciences, Beijing, China, in 2014. She is currently an Associate Professor with the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China. Her research interests include process-aware information systems, spatial-temporal database, sensor network middleware, and data security.

ZHANGBING ZHOU received the Ph.D. degree in computer science from the Digital Enterprise Research Institute (DERI), Galway, Ireland, in 2010. He worked as a Software Engineer with Huawei Technologies Company Ltd., Beijing, China, for one year. He served as a member for the Technical Staff at Bell Laboratories, Lucent Technologies, Beijing, for five years. He is currently a Professor with the China University of Geosciences, Beijing, and an Adjunct Professor with TELECOM SudParis, Évry, France. He has authored over 100 refereed articles. His research interests include process-aware information systems and sensor network middleware. He has served as an Associate or Guest Editor of over ten journals.

LEI SHU received the B.Sc. degree in computer science from South Central University for Nationalities, China, in 2002, the M.Sc. degree in computer engineering from Kyung Hee University, South Korea, in 2005, and the Ph.D. degree from the Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland, in 2010. He is currently a Distinguished Professor with Nanjing Agricultural University, China, and a Lincoln Professor with the University of Lincoln, U.K. He is also the Director of the NAU-Lincoln Joint Research Center of Intelligent Engineering. He has published over 400 articles in related conferences, journals, and books in the area of sensor networks. His H-index is 46 and i10-index is 172 in Google Scholar Citation. He was recently elected as a member of the EU Academy of Sciences. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the *IEEE Communications Magazine*, the IEEE NETWORK, the IEEE SYSTEMS JOURNAL, IEEE ACCESS, the IEEE/CAA JOURNAL OF AUTOMATIC SINICA, and *Sensors*.

GERHARD HANCKE received the B.Eng. and M.Eng. degrees from the University of Pretoria, South Africa, in 2002 and 2003, respectively, and the Ph.D. degree in computer science from the Security Group, the University of Cambridge's Computer Laboratory, in 2008. He is currently an Associate Professor with the City University of Hong Kong, Hong Kong. His research interests are system security, embedded platforms, and distributed sensing applications for the Internet of Things.