Future Generation Computer Systems 94 (2019) 697–711

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Intelligent augmented keyword search on spatial entities in real-life internet of vehicles

Yanhong Li [a], Meng Wang [a], Rongbo Zhu [a,c,*], Ashiq Anjum [d], Xiaokun Du [a], Yuhe Feng [b], Changyin Luo [e], Shasha Tian [a]

[a] College of Computer Science, South-Central University for Nationalities, Wuhan, China
[b] Guangdong Intelligent Robotics Institute, Dongguan, China
[c] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China
[d] College of Engineering and Technology, University of Derby, UK
[e] School of Computer, Central China Normal University, Wuhan, China

HIGHLIGHTS

- This paper takes the first step towards studying intelligent augmented keyword search on spatial entities in real-life Internet of Vehicles (ASKQIV). ASKQIV distinguishes itself from existing work for spatial keyword querying since it considers keywords, expressions, and spatial locations, and weighted traffic network distances, intelligently and simultaneously.
- An efficient hybrid index called ASKTI is proposed. In ASKTI, the information of traffic network structure, and the keywords, the boolean expressions, and the spatial information of spatial entities are smartly organized. Moreover, several lemmas are proposed to prune huge amounts of irrelevant spatial entities for ASKQIV queries. Based on ASKTI, an efficient and intelligent algorithm for Top-k ASKQIV query processing is proposed.
- We extend our method to support boolean expressions in conjunctive normal form (CNF) and disconjunctive normal form (DNF), and with complex predicate operators. Moreover, we also discuss how to extend our method to handle boolean range ASKQIV queries and ranking ASKQIV queries.
- Experiments on one real traffic network of IoV environments and two spatial entity sets reveal the efficiency of our query processing method and the associated ASKTI index.

ARTICLE INFO

Article history:
Received 23 August 2018
Received in revised form 22 November 2018
Accepted 20 December 2018
Available online 28 December 2018

Keywords:
Intelligent query
Internet of vehicles
Boolean expression
Traffic network

ABSTRACT

Internet of Vehicles (IoV) has attracted wide attention from both academia and industry. Due to the popularity of the geographical devices deployed on the vehicles, a tremendous amount of spatial entities which include spatial information, unstructured information and structured information, are generated every second. This development calls for intelligent augmented spatial keyword queries (ASKQ), which intelligently takes into account the locations, unstructured information (in the form of keyword sets), structured information (in the form of boolean expressions) of 182MinzuAvespatial entities. In this paper, we take the first step to address the issue of processing ASKQ in real traffic networks of IoV environments (ASKQIV) and focus on Top-k ASKQIV queries. To support network distance pruning, keyword pruning, and boolean expression pruning intelligently and simultaneously, a novel hybrid index structure called ASKTI is proposed. Note in the real-life traffic networks of IoV environments, travel cost is not only decided by the network distance, but also decided by some additional travel factors. By considering these additional factors, a combined factor Cftc of each road (edge) in the traffic network of IoV environments is calculated, and weighted network distance is calculated and adopted. Based on ASKTI, an efficient algorithm for Top-k ASKQIV query processing is proposed. Our method can also be extended to handle boolean range ASKQIV Queries and ranking ASKQIV Queries. Finally, simulation experiments on one real traffic network of IoV environments and two synthetic spatial entity sets are conducted. The results show that our ASKTI based method is superior to its competitors.

1. Introduction

Internet of Things (IoT), which implies things connected to the Internet, is an important aspect of the new generation of

* Correspondence to: 182 Minzu Ave, Wuhan 430074, China.
E-mail addresses: liyanhong@mail.scuec.edu.cn (Y. Li), yizhihongli@sina.com (M. Wang), rbzhu@mail.scuec.edu.cn (R. Zhu), a.anjum@derby.ac.uk (A. Anjum), xiaokundu@mail.scuec.edu.cn (X. Du), fengyuhe@giriobot.com (Y. Feng), changyinluo@mail.ccnu.edu.cn (C. Luo), shashatian77@mail.scuec.edu.cn (S. Tian).

https://doi.org/10.1016/j.future.2018.12.051
0167-739X/

information technology. The IoT technology enables the interconnection between large volumes of distributed and heterogeneous smart things and allows them to communicate seamlessly with the users. Recently, Internet of Vehicles (IoV) as one of the important members of IoT has caught worldwide attention and has become a research hotspot. In IoV environments, the vehicle communicates with each other for data sharing and processing. Billions of Internet-enabled things, such as GPS and Web GIS, are deployed on the vehicles, and data containing both spatial and textual information is being generated at an unprecedented speed. For example, in Twitter, millions of social-media users access the web site from mobile devices and post billions of geo-tagged tweets. These user-generated content arrives at a relative high speed, and contains valuable information on both individual and business users. Thus, how to intelligently process and analysis the vast amounts of data to retrieve users's interests is important and imperative.

Spatial keyword query (SKQ), which uses a set of keywords and a space constraint to represent users' interests so as to explore useful information, has been discussed for years [1–7]. These work can be broadly categorized into two classes: SKQ in Euclidean space and SKQ in traffic networks, and the latter uses real traffic network distance rather than Euclidean distance, thus can better meet the requirement of real-time applications.

SKQ can also be categorized from another perspective. In particular, common SKQ, which uses a set of keywords and a space constraint to represent users' interests, mainly handles unstructured information (e.g., "iphone 7S", "on sale"); while SKQ with boolean expressions, which uses a boolean expression and a space constraint to represent users' interests, handles structured information (e.g., model = iphone 7S $\bigcap$ color = gold $\bigcap$ price $\leq$ 1000). However, in many novel applications, such as Twitter, Facebook, and Foursquare, a tremendous amount of spatial entities which include both unstructured information and structured information, are generated every day. For example, a venue in Foursquare includes a spatial location and a textual description which can be represented by a set of keywords. Moreover, a venue may also contain a collection of attribute-value pairs, such as average hotel price (the average price of a hotel room) and its value, hotel rating (generally falls into 5 categories - 1 star, 2 star, 3 star, 4 star, and 5 star) and its value, and user rating (the average user rating score of the hotel) and its value. To handle these spatial entities, a novel augmented spatial keyword search is proposed [8]. In particular, a query generally includes a spatial location, a set of keywords, and a boolean expression to precisely capture the user intention. For example, "beach hotel" $\bigcap$ hotel rating = 5 star $\bigcap$ user rating $\geq$ 4.5 $\bigcap$ price $\leq$ 1000, distance $\leq$ 5 km.

However, the existing work for augmented spatial keyword search is limited to Euclidean space, which is a simplification of the real situation. In real-life situations, the query users and entity objects are distributed on the edges of traffic networks in IoV environments, and the distance between a query user and an entity object is the length of the shortest path connecting them, which is determined by the connectivity of the real-life traffic networks. This paper takes the first step to fill the gap by studying intelligent augmented spatial keyword queries in real IoV environments (ASKQIV). In particular, we mainly focus on Top-k queries in real IoV, and our method can also be extended to handle boolean range queries and ranking queries.

Different from the traffic networks discussed in existing works, where the travel cost is only decided by the network distance, in the real-life traffic networks of IoV, addition to the network distance, the travel cost is also decided by some additional travel factors, such as weather condition (f_{wc}), crowd congestion (f_{cc}), lane condition (f_{lc}), road alignment (f_{ra}), and road intersection type (f_{it}). By considering these additional factors, a factor Cftc (Combined factor of travel cost) of each edge is calculated, and thus the travel cost (in the form of weighted network distance) is calculated and adopted.

Our ASKQIV problem presents four main challenges. The first challenge lies in that there are millions of spatial entities in the system, and a large volume of keywords, attributes and values need to be considered. Secondly, many users may initiate queries at the same time, the match processing method proposed should be efficient and intelligent enough, such that the matched entities can be sent to the relevant query users as soon as possible. Thirdly, keywords, boolean expressions, and spatial requirements are all considered which enlarges the difficult of query processing. Furthermore, weighted network distance, which depends on traffic network structure, the positions of spatial entities and query users, and the additional traffic factors, is considered. Thus the proposed scheme should take into account query keywords, boolean expressions, location information, and traffic network structure intelligently and synchronously, in both indexing and matching.

To this end, we propose a novel hybrid index called ASKTI. Moreover, several lemmas are proposed to prune huge amounts of unqualified spatial entities for the user queries, thus to greatly improve the matching intelligence and efficiency of our method. Based on ASKTI index and the pruning lemmas, we present our Top-k ASKQIV query processing method, which follows the pruning-and-verifying schema. In the pruning step, according to the location and textual information of the entities and the query, the traffic network structure, and the traffic condition (in the form of additional travel factors), a large part of irrelevant regions, edges, and spatial entities are intelligently and efficiently pruned, and a set of regions where the Top-k result entities may reside, together with the candidate entities, are obtained. In the verification step, candidate entities are efficiently verified, and the final query result is retrieved.

The key contributions are as follows.

1. This paper takes the first step towards studying ASKQIV problem. ASKQIV distinguishes itself from existing works for spatial keyword querying since it considers keywords, expressions, and spatial locations, and weighted traffic network distances, intelligently and simultaneously.

2. To address ASKQIV problem, an efficient hybrid index, called ASKTI, is designed. In ASKTI, the information of traffic network structure, and the keywords, the boolean expressions, and the spatial information of spatial entities are smartly organized. Moreover, several lemmas are proposed to prune huge amounts of irrelevant spatial entities for ASKQIV queries. Based on ASKTI, an efficient and intelligent algorithm for Top-k ASKQIV query processing is proposed.

3. We extend our method to support boolean expressions in conjunctive normal form (CNF) and disconjunctive normal form (DNF), and with complex predicate operators. Moreover, we also discuss how to extend our method to handle boolean range ASKQIV queries and ranking ASKQIV queries.

4. Simulation experiments on one real traffic network of IoV environments and two spatial entity sets are conducted to evaluated the performance of our ASKTI index and query processing algorithm.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 gives problem descriptions and preliminaries. In Section 4, the index structure is proposed in detail. Section 5 presents Top-k ASKQIV query processing method. Section 6 discusses how to extend our method to support complex boolean expressions and handle other types of ASKQIV queries. Section 7 shows the experimental study, and finally, Section 8 concludes this paper.

2. Related work

2.1. Related research in IoV

With the development of IoV in recent years, lots of research work has been presented in this area [9–18]. Yu et al. [10] introduced cloud computing into vehicular networks and proposed a new architecture which includes a vehicular cloud, a roadside cloud, and a central cloud. Under this architecture, the vehicle can share several kinds of resources, such as computation resources, storage resources, and bandwidth resources, with each other, so as to maximize resource utilization. Wang et al. [11] discussed the issue of efficient content distribution among the vehicles in IoV, and proposed a distributed game theory-based method. Zhang et al. [13] addressed the challenges in designing a scalable IoV system by describing CarStream, an industrial system of big data processing for chauffeured car services. CarStream collected and processed multiple types of driving data including vehicle status, driver activity, and passenger-trip information. Song et al. discussed the foundations, principles, and applications in cyber–physical systems [14] and smart cities [19]. Moreover, the security and privacy in cyber–physical systems have also been studied [20]. Recently, Artificial intelligent technologies were introduced into the IoV [16,21–23]. Chen et al. [16] proposed a probabilistic model called CPGN for decision-making in the rear-end collision avoidance system, targeting modeling the impact of important influential factors of collisions on the occurring probability of possible accidents in IoV. Chen et al. [23] proposed an innovative paradigm called Cognitive Internet of Vehicles (CIoV) to provide a more intelligent vehicular network. Different from existing works, which mainly focus on communication technologies, CIoV enhances transportation safety and network security by mining effective information from both physical and network data space. Recently, Chen et al. [24] discussed the issue of how to maintaining cloud intelligence while reducing network traffic in view of the requirements of AI-based applications on cloud intelligence and real-time performance. By fully using the computing and communication resources of the edge cloud, they proposed a traffic control algorithm (LLTC). This algorithm only offloads the useful data to the cloud by the labeling and selecting of unlabeled data, thus can not only reduce the amount of data transmission, but also have little influence on cloud intelligence. Moreover, Secure enforcement in CIoV has also been discussed [25].

2.2. Spatial keyword query in Euclidean space

To handle SKQ queries in Euclidean Space, Zhou et al. [26] proposed a spatial–textual index by combining inverted indexes with R-trees, and proposed three different combining schemes. Felipe et al. [1] designed IR^2-tree that integrates an R-tree and signature files. Cong et al. [27] designed IR-tree by combining inverted lists and R-tree. later, some other famous indexing techniques, such as S2I, I^3 [2] and IL-Quadtree [28], were also proposed. Chen et al. [29] provided an all-around survey of 12 state-of-the-art geo-textual indices and query processing techniques. Huang et al. [30] first discussed moving Top-k SKQ processing, and proposed an efficient query processing method. Then, Li et al. [31] explored direction-aware SKQ processing. It returns the k nearest neighbors of the query which satisfy both direction and keyword constraints of the query. Cao et al. [32] studied closet keywords search (Keyword Cover), which finds a set of objects that cover all the input keywords and have the minimum inter-objects distance. Moreover, some variants of SKQ, such as interactive Top-k SKQ [3], approximate keyword query of semantic trajectory [33], reverse spatial and textual k nearest neighbor query [34], keyword search over distributed graphs [35], and augmented spatial keyword search [8], have also been discussed.

2.3. Spatial keyword query in traffic networks

Recently, SKQ processing on traffic networks has also been studied. Carlsson et al. [4] discussed SKQ processing in traffic networks, and proposed an index called RNI. However, they only studied a simple case to obtain the object which has the shortest network distance to query q among all the objects sharing at least one common keyword with query q. Guo et al. [36] discussed distributed SKQ processing on traffic networks. They proposed a novel distributed index that enables each machine to independently evaluate the search operation in a distributed way. Gao et al. [5] explored reverse Top-k Boolean spatial keyword (RkBSK) retrieval on the traffic network, and discussed how to answer RkBSK queries with arbitrary k and no any pre-computation.

2.4. Query processing in IoVs and intelligent query processing

Wang et al. [37] explored traffic information query strategy in VANETs, and proposed an urban area-oriented traffic information query processing mechanism, which can acquire the realtime traffic information of multiple paths from source to destination in relatively fast and accurate manner, and help users to determine an optimal route. Anakavej et al. [38] presented a new framework and feature set for vehicle model query system. By giving model names or manufacturer names as keywords, the desired vehicle images can be queried from target videos or vehicle image databases using internet-vision approach. Li et al. [39] addressed the issue of processing Spatial Keyword Queries (SKQ) in IoV. A novel hybird index called SKIV was proposed as an important part of the algorithm design to improve the performance of SKQ query processing. Recently, intelligent query processing has also been studied. Wang et al. [40] discussed the issue of digital intelligent query optimization of digital library. They adopted data association rules to mine and extract digital library data from the perspective of user experience, and extract association rules based on the basic idea of relational database, so as to achieve intelligent query optimization. Kolomvatsos et al. [41] proposed a learning scheme where a query controller assigns each query to the available processor. They used the q-learning algorithm to calculate the reward of QC for each allocation between query and processor, thus they got an efficient model to derive the optimal allocation for the incoming queries. Wang et al. [42] explored intelligent image search in traffic images, and proposed a novel image search interface-Semantic Visual Query Builder (SeVQer) to work as a non-verbal interface, which allows user to drag and drop from the provided image data to formulate a user query.

2.5. Boolean expression matching

Boolean expression matching has been studied extensively, and many research approaches have been proposed in this field [43–47]. Sadoghi et al. [44] explored how to efficiently index boolean expressions over high-dimensional discrete space, and proposed a novel index called BE-tree. BE-tree uses a two-phase space-cutting technique and organizes the boolean expressions in a hierarchical index. Moreover, it employs self-adjustment policies to dynamically modify the tree structure to adjust to a variable workload. To efficiently organize e-commence products, which are sparse and with very-high dimensions, Zhang et al. [45] proposed a novel index called Opindex. In particular, Opindex uses a two-level partition scheme: subscriptions are first divided by their pivot attributes into subscription lists, and then the predicates in each subscription list are further partitioned by operators. However, those methods discussed above ignore location information. Later, Guo et al. [48] explored the issue of spatial matching and boolean expression matching, and took into account continuous moving

Table 1
Symbols and definitions.

Notation	Definition
E	A spatial entity with a set of keywords, a collection of attribute-value pair, and a location on the traffic network of IoV environments
Q	An intelligent augmented spatial keyword query with a set of keywords, a boolean expression, and a location on the traffic network of IoV environments
$Q.B$	The boolean expression for intelligent augmented spatial keyword query Q which includes a set of predicates
$E.V$	A collection of attribute-value pairs for spatial entity E
$E.L(Q.L)$	The location of entity E (query Q)
$d_N(c, Q)$	The network distance between cell c and query Q
$d_E(c, Q)$	The Euclidean distance between cell c and query Q
$d_N^{min}(c, Q)$	The minimum network distance between cell c and query Q
$Cftc$	Combined factor of travel cost
$d_{WN}^{min}(c, Q)$	The minimum weighted network distance between cell c and query Q
$d_{WN}^{max}(c, Q)$	The maximum weighted network distance between cell c and query Q

queries over dynamic event streams. They employed the concept of safe region to reduce communication overhead, and proposed a novel index, BEQ-Tree, to handle spatial boolean expression matching. Recently, Zhao et al. [49] also discussed this issue and proposed an RP-tree index. RP-tree is a hybrid index which combines R-trees with a boolean expression index, and it can efficiently filter subscriptions with boolean expressions and locations.

However, most related studies have at least one of the following limitations: (1) They focus on unstructured keywords or structured boolean expressions, and cannot handle spatial entities with both keywords and boolean expressions; (2) only consider keyword match and boolean expression match, while ignoring spatial constraint requirement; (3) are limited to Euclidean space, and cannot handle intelligent augmented keyword search in real traffic networks of IoV environments, which is a more realistic scenario. Thus, it is important and necessary to propose an efficient method to deal with intelligent augmented keyword search on spatial entities in real IoV. This paper fills the gap by developing a novel hybrid index which supports spatial constraint, keywords, and boolean expressions to achieve high intelligence and efficiency in indexing and matching in real IoV environments.

3. Problem definitions

In real-life situations, the query users and entity objects are distributed on the roads (edges) of the traffic network in IoV, and the distance between a query user and an entity object is the length of the shortest path connecting them, which is determined by the connectivity of the traffic network. This paper takes the first step towards studying ASKQIV query processing. In particular, we mainly focus on Top-k ASKQIV queries, and our method can be extended to handle boolean range ASKQIV queries and ranking ASKQIV queries. Some frequently used symbols and their definitions are summarized in Table 1.

3.1. System model

IoV is a convergence of vehicular communication systems and the IoT, and includes all kinds of vehicles, either fitted or integrated with two-way radio frequency (RF) equipments. The vehicles are typically a class of remote mobile objects and can communicate with each other or smart phones via RF equipments. Our system consists of cloud servers, road side units (RSUs), and cellular user equipments (CUEs). CUEs include vehicles and smart phones, and can communicate with each other to exchange information. Moreover, CUEs can communicate with RSUs by orthogonal spectrum resource blocks.

3.2. A traffic network of IoV environments

In many existing works, a traffic network is usually modeled as an undirected weighted graph $G = (V, \bar{E})$, where V is a set of vertices, and $\bar{E}$ is a set of edges. A vertice $v \in V$ represents a road intersection or an end-point in the traffic network. An edge $e(v_i, v_j, l) \in \bar{E}$, represents the road segment between two vertices v_i and v_j $(i \neq j)$, and l represents the length of the road segment.

However, in the real situations, the travel time for the vehicle in IoV environments is not only decided by the length and the speed limitation of the roads (edges), but also depended on the additional traffic factors of the edges, such as weather condition (f_{wc}), crowd congestion (f_{cc}), lane condition (f_{lc}), road alignment (f_{ra}), and road intersection type (f_{it}). Thus, for an edge $e \in \bar{E}$, these five factors are also kept, besides its start vertice, end vertice, and length. Next, we discuss these additional traffic factors in detail.

Weather condition (f_{wc}). Travel time may be greatly influenced by the weather which will deteriorate the roadway network performance. Driving from home to work takes a longer time on a rainy or snowy day than on a sunny day, and the impact of rain and snow is a function of their intensity [50]. The experiment results in [50] show that, the ranges of the total travel time increase due to light, moderate and heavy rain are: 0.1 to 2.1%, 1.5 to 3.8%, and 4.0 to 6.0%, respectively; Light and heavy snow result in travel time increase of 5.5 to 7.6% and 7.4 to 11.4%, respectively. Thus, we define five different weather conditions which are clear, light rain, moderate rain, heavy rain/light snow, and heavy snow, and assign values from 0 to 4, respectively.

Crowd congestion (f_{cc}). Road congestion also has a great impact on travel time [51]. Travel time for the same distance will be much shorter in smooth road than commuting in heavy traffic. As a result, we define three different levels of crowd congestion which are smooth, light traffic, and heavy traffic, and assign values 0, 2, and 4, respectively.

Lane condition (f_{lc}). Traffic flow can be divided and undivided. As for undivided traffic flow, all vehicles (motor vehicles and non-motor vehicles) are organized on one roads. Since motor vehicles and non-motor vehicles are intermingled, the interference between them is bound to be great, and the driving capacity will be greatly affected. What is more, drivers have a sense of insecurity, which makes it difficult for them to improve their driving capacity. If the traffic flow is divided, the interference between the motor vehicles and the non-motor vehicles, and between the vehicles on two different directions can be avoided, so that the vehicles can run faster. Lane width is another important factor that determines the speeds of vehicles on the road [52], and the reduction factor of vehicle capacity should be considered when the lane width is less than 3.5 m. As a result, we define four different lane conditions which are divided & wide, divided & narrow, undivided & wide, and undivided & narrow, and assign values 0, 1, 2, and 3, respectively.

Road alignment (f_{ra}). The roads of the traffic network in IoV environments consist of straight road segments and curves. Roads with curves will need more travel time comparing to straight road segments. Besides, the longitudinal profiles of roads can be divided into horizontal, the straight line of the upper slope, the straight line of the lower slope, and the vertical curve. In the last three cases, the speed of driving will be affected. We refer to them as acclive. As a result, we define four different road alignments which are straight & horizontal (the maximum longitude gradient is smaller

than 5%), curve & horizontal, straight & acclive, and curve & acclive, and assign values 0, 1, 2, and 3, respectively.

Road intersection type (f_{it}). The forms of road intersections are usually divided into: plane intersections and three-dimensional intersections. For plane intersections, there exist many collision points and interlacing points, and large blind area of vehicle drivers, thus the road capacity is obviously affected. While three-dimensional intersections can reduce or eliminate the impact points of intersections, thus obviously improve the road capacity. As a result, we define two different road intersection types which are plane intersection and three-dimensional intersection, and assign values 2 and 0, respectively.

To calculate the realistic travel cost, we define a factor Cftc (Combined factor of travel cost) as follows, and for each edge $e(v_i, v_j)$ in the real-world traffic network of IoV environments, its Cftc is calculated and adopted.

$$Cftc_{(v_i,v_j)} = \sum_{k=1}^{5} \frac{f_k}{Max_{f_k}} \times w_k \tag{1}$$

where, f_1 to f_5 represent five additional travel factors, i.e., f_{wc} to f_{it}, across edge (v_i, v_j), Max_{f_1} to Max_{f_5} indicate the maximum values of these factors, and w_1 to w_5 give the weights of these factors. The value of each weight varies from 0 to 1, and the sum of the weights of these five factors equals 1.

Definition 1 (*Traffic Network*)**.** A traffic network in IoV environments is an undirected weighted graph G, which consists of a finite set of vertices N and a set of edges $\bar{E}$.

Definition 2 (*Edge*)**.** An edge $e(v_i, v_j, l, F) \in \bar{E}$ is the road segment between two vertices v_i and v_j ($v_i, v_j \in N$ and $i \neq j$), l is the length of edge e, and F is the set of additional traffic factors of e discussed before.

Our model can be extended to support directed weighted graph, which represents unidirectional traffic, by simply allowing the length and additional factors of $e(v_i, v_j)$ be set different from that of $e(v_j, v_i)$.

3.3. Predicate

In a spatial–textual query system with keywords and boolean expressions, a query is usually represented by a set of keywords, a boolean expression, a location, and a spatial region or a value k to represent the number of result objects returned. A boolean expression is a combination of user-specified predicates, which is determined by three elements: an attribute A, an operator f_{op} and a value v. That is, $P_{(A,f_{op},v)}$ denotes a predicate as follows.

Definition 3 (*Predicate*)**.** A predicate is defined as $P_{(A,f_{op},v)}$, where A is an attribute, f_{op} is an operator, and v is a value. $P_{(A,f_{op},v)}$ accepts an input value x and outputs a boolean value indicating whether or not the operator constraint is satisfied:

$$P_{(A,f_{op},v)}(x) = \{0, 1\}.$$

This paper adopts a general and expressive data model. It can support numerical, categorical, and string attribute domains with standard relational operators ($<, \leq, =, \neq, >, \geq$), set operator ($\in_s$), and interval operator ($\in_i$).

3.4. Boolean expression

For ease of presentation, we only handle a simple situation where a boolean expression is a conjunction of predicates, that is in the simple CNF form, and leave the discussion of how to handle more general forms of boolean expressions in Section 6.1.

Definition 4 (*Boolean Expression*)**.** A boolean expression B is a combination of user-specified predicates in CNF which is modeled as follows:

$$B : P_{1(A,f_{op},v)}(x) \bigcap P_{2(A,f_{op},v)}(x) \bigcap \ldots \bigcap P_{n(A,f_{op},v)}(x).$$

3.5. Spatial entity and intelligent augmented spatial keyword query

Definition 5 (*Spatial Entity*)**.** A spatial entity E in the traffic network of IoV environments is defined as $E = (E.tags, E.V, E.L)$, where $E.tags$ is the associated descriptive tags which includes a set of keywords, $E.V$ is a collection of attribute-value pairs, and $E.L$ is the entity location which is a spatial point on an edge of the traffic network. $E.V$ is modeled as a conjunction of equality predicates which is used to match the augmented user intention in the form of a boolean expression, and the size of $E.V$ is the number of attribute-value pairs denoted by n. Thus, a spatial entity E can be denoted as follows:

$$E = \{tags, P_1 = v_1 \bigcap P_2 = v_2 \bigcap \ldots \bigcap P_n = v_n, E.L\}.$$

Definition 6 (*Intelligent Augmented Spatial Keyword Query in Traffic Networks of IoV Environments (ASKQIV)*)**.** An ASKQIV query Q is defined as $Q = (Q.W, Q.B, Q.L)$, where $Q.W$ is the associated descriptive keywords, $Q.B$ is a combination of user-specified predicates in CNF, and $Q.L$ is a spatial point on an edge of the traffic network. Thus, an ASKQIV query Q can be denoted as follows:

$$Q = \{Q.W, P_{1(A,f_{op},v)}(x) \bigcap P_{2(A,f_{op},v)}(x) \bigcap \ldots \bigcap P_{n(A,f_{op},v)}(x), Q.L\}.$$

3.6. Match semantics

Given an ASKQIV query Q and a spatial entity E, Q matches E if and only if three requirements, which are keyword match, boolean expression match and spatial match, are all satisfied. In particular, boolean expression match means each of the predicates in $Q.B$ has a match in $E.V$

Definition 7 (*Keyword Match*)**.** For ASKQIV query Q and spatial entity E, if for any keyword occurring in Q, it also appears in E, i.e., $Q.W \subseteq$ E.tags, there exists a keyword match between Q and E.

In particular, $Q \sim_W E$ is used to denote keyword match.

Definition 8 (*Attribute Match*)**.** For ASKQIV query Q and spatial entity E, if for any attribute $Q.B_i.A$ in Q, it is also contained in E.V, i.e., $Q.B_i.A \in$ E.V, there exists an attribute match between Q and E.

In particular, $Q \sim_A E$ is used to denote attribute match.

Definition 9 (*Predicate Match*)**.** Let $E.V_i$ be an attribute-value pair in $E.V$, then $E.V_i.A$ is the attribute of $E.V_i$ and $E.V_i.v$ is the value. For spatial entity E and ASKQIV query Q, if there exists a predicate $Q.B_j$ meeting the condition that $Q.B_j.A = E.V_i.A$ and $P_{(Q.B_j.A,Q.B_j.op,Q.B_j.v)}(E.V_i.v) = 1$, there is a predicate match between Q and E. In short, $Q.B_j$ predicate matches $E.V_i$.

In particular, $Q \sim_P E$ is used to denote attribute match.

Definition 10 (*Boolean Expression Match*)**.** For each predicate $Q.B_j$ in $Q.B$, if there exists an attribute-value pair in $E.V$ matches $Q.B_j$, then $Q.B$ boolean expression matches $E.V$.

In particular, $Q \sim_{BE} E$ is used to denote boolean expression match.

Definition 11 (*Match Between ASKQIV Query Q and Spatial Entity E*)**.** Given an ASKQIV query Q and an entity E, Q matches E, denoted by $Q \sim E$, if there exist keyword match, boolean expression match, and spatial match between Q and E.

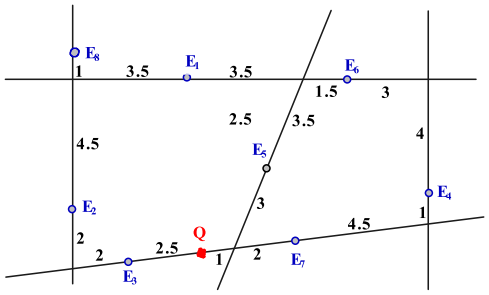
3.7. Problem statement

By using the weighted road network distance $dist_{WN}$ to measure the spatial proximity between the query and the entities, we can formally define Top-k ASKQIV query as follows.

Definition 12 (*Top-k ASKQIV Query*)**.** Given a spatial entity data set $\widehat{E}$, a Top-k ASKQIV query $Q = (Q.W, Q.B, Q.L, k)$ retrieves a set of entities $O \subseteq \widehat{E}$ such that $\forall E \in O$ and $E' \in \widehat{E} - O$, (1) $Q \sim E$, (2) if $Q \sim E'$, then $dist_{WN}(Q, E) < dist_{WN}(Q, E')$.

Example 1. Fig. 1 illustrates an example of Top-k ASKQIV query, and 8 spatial entities and 1 ASKQIV query are located on the edges of a traffic network of IoV environments.[1] Each entity has a set of keywords and a set of attribute-value pairs to give its descriptive message. Moreover, it also has a spatial location which is a spatial point on an edge of the traffic network of IoV. The query user Q contains four components: a set of query keywords {*Italian*, *coffee*}, a boolean expression "rating $\geq 4.3 \bigcap$ pcc (per capita consumption) $\leq$60", a spatial point l_Q indicating the current position of query Q, and a value $k = 2$ indicating the top-2 relevant entities will be wanted.

We intelligently prune entity E_3, E_7, and E_2 which are close to query Q since they dissatisfy the keyword requirement and/or expression requirement of Q, and entity E_5 and E_4 are keep as the top-1 and top-2 entities since they are the closest two from query Q among those which include all the query keywords and meet the boolean expression requirement of Q. By using our ASKTI index, we can intelligently calculate the travel cost of the remaining entities (in the form of weighted network distance limitation) according to the locations of entities and the query, the traffic network structure, and the traffic condition. Since all the remaining entities are more distanced from query Q than E_4, they are all safely pruned, and we can intelligently terminate the process as early as possible.



Spatial entities:
E_1:({fish,pizza,bread},{rating=4.5,pcc=70,no_smoking=yes},l_1)
E_2:({pizza,Italian,bread},{rating=4.6,pcc=65,no_smoking=yes},l_2)
E_3:({coffee,Italian},{rating=4.2,pcc=78},l_3)
E_4:({coffee,Italian},{rating=4.6,pcc=55,no_smoking=yes},l_4)
E_5:({pizza,Italian,coffee},{rating=4.6,pcc,no_smoking=yes},l_5)
E_6:({coffee,fish,bread},{rating=4.3,pcc=45,no_smoking=yes},l_6)
E_7:({Italian,fish,bread},{rating=4.6,pcc=52},l_7)
E_8:({Italian,pizza,coffee},{rating=4.5,pcc=56,no_smoking=yes},l_8)

Network distances

E	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
d_N(E,Q)	11	6.5	2.5	8.5	4	9	3	12

Fig. 1. An example of Top-k ASKQIV query.

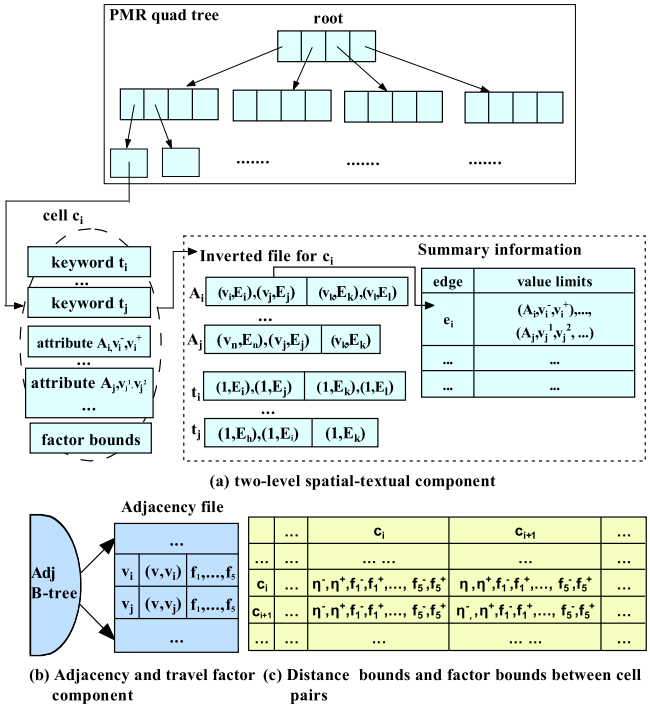


Fig. 2. The index segment of ASKTI.

4. Index for intelligent augmented keyword search on spatial entities in iov

Our work focuses on processing ASKQIV queries. This section mainly discusses the data structures and the index used in our system. As discussed in Section 3.2, a traffic network of IoV is represented by a weighted graph, consisting of a set of vertices and edges. Note for an edge $e \in E$, besides its start vertice, end vertice, and length, five additional traffic factors, which are weather condition (f_{wc}), crowd congestion (f_{cc}), lane condition (f_{lc}), road alignment (f_{ra}), and road intersection type (f_{it}), are also kept. Moreover, a set of spatial entities and a set of ASKQIV queries distributed on the edges of the traffic network are maintained.

There are millions of spatial entities in the system, and many users may issue ASKQIV queries at the same time. Thus, how to intelligently prune irrelevant entities for an ASKQIV query as many as possible, so as to improve the performance in the match operation between the query and spatial entities, is a challenge. To support weighted network distance pruning, keyword pruning, and boolean expression pruning intelligently and simultaneously, a novel hybrid index structure called ASKTI is designed, which is shown in Fig. 2.

ASKTI consists of three components, two-level spatial–textual component, adjacency and travel factor component, and distance bound and factor bound component. Next, we discuss these three components in detail.

Two-level spatial–textual component. The index structure should take into account the relative invariance of the network topology, and the dynamic nature and life cycle of spatial entities and queries. Thus, the method of static-dynamic separation and layered construction is proposed.

Firstly, construct the static part (the first level) of the component, i.e., a spatial index SI on the network edges. Due to the morphological diversity of actual traffic network structures, the distribution of vertices and edges on a traffic network of IoV environments may be very uneven. Thus, an adaptive partition method is needed to balance the information of edges in each region.

[1] For ease of presentation, the additional traffic factors of the edges of the traffic network are omitted in this example.

Moreover, given the coordinates of an entity (or query), the edge where it lies should be rapidly located. For these two purposes, a PMR quadtree is designed. In this quadtree, each leaf quad contains the IDs of the edges intersecting it. The tree is built by iteratively inserting the network edges. If the number of edges in a leaf quad exceeds a threshold, the quad is split into four new ones and becomes their predecessor in the tree.

Secondly, construct the dynamic part (the second level) of the component, i.e., a textual index on the spatial entities. For each leaf quad (cell) c_i, we keep the following items: (1) a set of keywords which is the union of keyword sets for all entities in c_i; (2) a set of attributes which include all the attributes of the entities in c_i. For numeric attribute A_i, we also keep the maximum value v_i^+ and minimum value v_i^- of this attribute for entities in c_i; For categorial attribute A_j, the possible value set, pvset $= \{v_j^1, v_j^2, \dots\}$, of this attribute for entities in c_i is also kept; (3) the set of inverted lists containing the entities within c_i. Specifically, one list per different keyword or attribute in the descriptions of the entities. The content of the inverted list for each attribute A_i includes the IDs of the entities containing A_i and the values of A_i for the entities, and the list is in ascending order of attribute value. While the content of inverted list for each keyword t_i only includes entity IDs and the list is in ascending order of entity ID. To support group pruning of entities within a leaf cell, each posting list in inverted file is further partitioned into groups based on the edges where the entities locate. Furthermore, for each edge e_i, some summary information, such as the maximum and minimum values of each numeric attribute for entities in e_i, and possible values of each categorial attribute for entities in e_i, are materialized. To efficiently search entities which meet the textual requirement of query q, for each cell, its inverted lists are arranged in non-decreasing order of the number of entities in each list.

Adjacency and travel factor component. To allow traversing the network from vertex to vertex, the adjacency and travel factor component (as shown in Fig. 2(b)) is designed. For each vertex, it gives the pointer pointing to its adjacent vertices (traffic network vertices). In particular, a B-tree is built to point to the block in the adjacency file where the adjacent vertices of a given vertex v are. Note that the additional traffic factors of each edge are also kept.

Distance bound and factor bound component. In order to use network distance to prune network space, we calculate and keep the effective distance bounds between each pair of cells similar to [53]. In particular, for each cell pair c_i and c_j, we define a pair of parameters, namely η^- and η^+, as shown in Eqs. (2) and (3):

$$\eta^-(c_i, c_j) = \min_{E_k \subseteq c_i, E_l \subseteq c_j} \frac{d_N(E_k, E_l)}{d_E(E_k, E_l)} \tag{2}$$

$$\eta^+(c_i, c_j) = \max_{E_k \subseteq c_i, E_l \subseteq c_j} \frac{d_N(E_k, E_l)}{d_E(E_k, E_l)} \tag{3}$$

Based on the positions of entity E and query Q, the minimum and maximum network distances between E and Q can be computed as follows (suppose that E locates at c_i, and Q locates at c_Q).

$$d_N^{min}(E, Q) = \eta^-(c_i, c_Q) \times d_E(E, Q) \tag{4}$$

$$d_N^{max}(E, Q) = \eta^+(c_i, c_Q) \times d_E(E, Q) \tag{5}$$

Similarly, the minimum and maximum network distances between cell c_i and query Q can also be computed.

To compute the realistic travel cost of each edge e, the additional travel factors of e are also maintained. Moreover, to facilitate the pruning of unrelated cells based on the real traffic cost, for each cell pair c_i and c_j, the maximum and the minimum value of each additional factor of the edges passing through the shortest path between one vertice v_i in c_i and one vertice v_j in c_j are calculated and kept, which are shown in Fig. 2(c). Note for each cell c_i, the maximum and the minimum value of an additional factor $f_j(c_i)$ of the edges within the cell, equal $f_j^-(c_i, c_i)$ and $f_j^+(c_i, c_i)$ in Fig. 2(c). Thus the weighted minimum and maximum network distances between E and Q can be revised as follows (suppose that E locates at c_i, and Q locates at c_Q).

$$d_{WN}^{min}(E, Q) = \eta^-(c_i, c_Q) \times d_E(E, Q) \times Cftc^-(c_i, c_Q) \tag{6}$$

$$Cftc^-(c_i, c_Q) = \sum_{j=1}^{5} \frac{f_j^-(c_i, c_Q)}{Max_{f_j}} \times w_j \tag{7}$$

$$d_{WN}^{max}(E, Q) = \eta^+(c_i, c_Q) \times d_E(E, Q) \times Cftc^+(c_i, c_Q) \tag{8}$$

$$Cftc^+(c_i, c_Q) = \sum_{j=1}^{5} \frac{f_j^+(c_i, c_Q)}{Max_{f_j}} \times w_j \tag{9}$$

Similarly, the minimum and maximum weighted network distances between cell c_i and query Q can be computed as follows (suppose that Q locates at c_Q).

$$d_{WN}^{min}(c_i, Q) = \eta^-(c_i, c_Q) \times d_E^{min}(c_i, Q) \times Cftc^-(c_i, c_Q) \tag{10}$$

$$d_{WN}^{max}(c_i, Q) = \eta^+(c_i, c_Q) \times d_E^{max}(c_i, Q) \times Cftc^+(c_i, c_Q) \tag{11}$$

5. Intelligent augmented keyword search processing on spatial entities in iov

Based on ASKTI index, we propose our Top-k ASKQIV query processing method, which includes two steps — pruning and verification. In the pruning step, according to the location and textual information of the entities, the traffic network structure, and the traffic condition, the query space is intelligently reduced by region, group (in the form of the edge where entities reside) and single entity. As a result, a large part of irrelevant regions, edges, and spatial entities are safely pruned, and a set of regions where the Top-k result entities may reside, together with the candidate entities, are obtained. Different from traditional spatial keyword queries which mainly use Euclidean distance or network distance as the travel cost, our travel cost is intelligently calculated by the locations of entities and the query, the traffic network structure, and the traffic condition. In the verification step, by using the data determined in the first step, the final query result is calculated and retrieved.

5.1. Pruning techniques

Before discussing our algorithm in detail, we first introduce several lemmas to prune the irrelevant traffic network space intelligently and efficiently.

Lemma 1. *Given a Top-k ASKQIV query $Q = (Q.W, Q.B, Q.L, k)$ and a cell c, c can be safely pruned if $d_E^{\min}(c, Q) > d_{wk}$ or $d_N^{\min}(c, Q) > d_{wk}$ or $d_{WN}^{min}(c, Q) > d_{wk}$. In particular, E_k is the kth nearest neighbor of Q, and d_{wk} equals $\sum_{e_i \in path(E_k, Q)} (d_N(e_i) \times Cftc(e_i))$, where $path(E_k, Q)$ is the shortest path from E_k to Q.*

Proof. For any entity E in c, there are (1) $d_{WN}(E, Q) = \sum_{e_i \in path(E,Q)} (d_N(e_i) \times Cftc(e_i)) \geq d_N^{\min}(c, Q) \times Cftc^-(c, c_Q) \geq d_N^{\min}(c, Q) \geq d_E^{\min}(c, Q)$; (2) $d_{WN}(E, Q) = \sum_{e_i \in path(E,Q)} (d_N(e_i) \times Cftc(e_i)) > d_N^{\min}(c, Q)$

$\times Cftc^{-}(c, c_Q) \geq \eta^{-}(c, c_Q) \times d^{E}_{min}(c, Q) \times Cftc^{-}(c, c_Q) = d^{min}_{WN}(c, Q)$. By $d^{\min}_{E}(c, Q) > d_{wk}$ or $d^{\min}_{N}(c, Q) > d_{wk}$ or $d^{min}_{WN}(c, Q) > d_{wk}$, we know that $d_{WN}(E, Q) > d_{wk}$ must hold. Therefore, c can be safely pruned. ■

Lemma 2. *Given a Top-k ASKQIV query $Q = (Q.W, Q.B, Q.L, k)$ and a cell c, if there exists an item $t \in Q.W \cup Q.B$ such that the inverted list for t does not exist or is empty, then c can be safely pruned.*

Proof. If the invert listed for t does not exist or is empty, it means c does not include any entity E containing $t \in Q.W \cup Q.B$, which also means cell c does not contain any result entity of query Q. Hence, c can be safely pruned. ■

Lemma 3. *Given a Top-k ASKQIV query $Q = (Q.W, Q.B, Q.L, k)$ and a cell c, if there exists an attribute $A_i \in Q.B$ such that $Q.B.P_{A_i} \cap c.A_i.[v_i^-, v_i^+] = \varnothing$ (A_i is a numeric attribute) or $Q.B.P_{A_i} \cap c.A_i.pvset = \varnothing$ (A_i is a categorial attribute), then c can be safely pruned.*

Proof. For cell c, if there exists a numeric attribute $A_i \in Q.B$ such that $Q.B.P_{A_i} \cap c.A_i.[v_i^-, v_i^+] = \varnothing$, it means that c does not include any entity E, which contains an attribute-value pair $E.v_j$ such that $E.V_j.A = A_i$ and $E.V_j.v$ conforms to predicate $Q.B.P_{A_i}$ of query Q. Thus cell c does not contain any result entity of query Q, and hence can be safely pruned. Similarly, for cell c, if there exists a categorial attribute $A_i \in Q.B$ such that $Q.B.P_{A_i} \cap c.A_i.pvset = \varnothing$, c can be safely pruned. ■

Lemma 4. *Given a Top-k ASKQIV query $Q = (Q.W, Q.B, Q.L, k)$ and an edge e in cell c, e can be safely pruned if $Q \notin e$ & $d^{min}_{WN}(e.start, Q) > d_{wk}$ & $d^{min}_{WN}(e.end, Q) > d_{wk}$.*

Proof. For the start point $e.start$ of edge e, there is $d_{WN}(e.start, Q) = \sum_{e_i \in path(e.start, Q)} (d_N(e_i) \times Cftc(e_i)) > d^{\min}_{N}(e.start, Q) \times Cftc^{-}(c, c_Q) \geq \eta^{-}(c, c_Q) \times d^{min}_{E}(e.start, Q) \times Cftc^{-}(c, c_Q) \geq d^{min}_{WN}(e.start, Q)$. By $d^{min}_{WN}(e.start, Q) > d_{wk}$, we know that $d_{WN}(e.start, Q) > d_{wk}$ must hold. Similarly, if $d^{min}_{WN}(e.end, Q) > d_{wk}$, then $d_{WN}(e.end, Q) > d_{wk}$ must hold. Since Q does not locate on edge e, for any entity E on edge e, the shortest path from E to query Q must pass through one of its two end points, i.e., $e.start$ or $e.end$. Thus, we have $d_{WN}(E, Q) \geq d_{WN}(e.start, Q)$ (or $d_{WN}(e.start, Q)) > d_{wk}$, Therefore, e can be safely pruned. ■

Lemma 5. *Given a Top-k ASKQIV query $Q = (Q.W, Q.B, Q.L, k)$ and an edge e, if there exists an item $t \in Q.W \cup Q.B$ such that the (sub) inverted list of t for e does not exist or is empty, then e can be safely pruned.*

Proof. Similar to the proof of Lemma 2. ■

Lemma 6. *Given a Top-k ASKQIV query $Q = (Q.W, Q.B, Q.L, k)$ and an edge e, if there exists an attribute $A_j \in Q.B$ such that $Q.B.P_{A_i} \cap e.A_j.[v_j^-, v_j^+] = \varnothing$ (A_j is a numeric attribute) or $Q.B.P_{A_i} \cap e.A_j.pvset = \varnothing$ (A_j is a categorial attribute), then e can be safely pruned.*

Proof. Similar to the proof of Lemma 3. ■

5.2. Pruning phase

Algorithm 1 gives the pseudo-code for the pruning step of Top-k ASKQIV query processing. Set $\widehat{E}_{cand}$ is used to keep the candidate entities, and the key of a candidate entity E in $\widehat{E}_{cand}$ is the minimal weighted network distance from E to query Q, i.e., $d^{min}_{WN}(E, Q)$. We also keep a float d_{Wk}, which is initialized to be infinity, to keep the maximum weighted distance value of the kth nearest neighbor of query Q. Moreover, a heap *Hnode*, which is initialized to be empty, is used to organize the nodes (cells) encountered during network expansion. In particular, heap *Hnode* is an ordered structure and $d^{min}_{WN}(c, Q)$ is served as the key of a node (cell) c in the heap. Note that several pruning techniques proposed in subsection IV-A will be used to intelligently prune a large number of irrelevant cells and edges, together with huge numbers of unpromising entities, thus greatly improving the efficiency of our method. We now introduce this step in detail.

Firstly, by using the PMR quad tree of the index, the cell c_Q containing query Q can be obtained. Then it en-heaps the foot node of PMR quad tree together with distance 0 into *Hnode*. Next, we iteratively de-heap a node c from *Hnode* and process c in the following steps until *Hnode* is empty: (1) If $d^{\min}_{E}(c, Q)$ (or $d^{\min}_{N}(c, Q)$ or $d^{\min}_{WN}(c, Q)$) is larger than d_{Wk}, cell c can be pruned according to Lemma 1 (lines 7-8). Note that the cells are list in ascending order of $d^{\min}_{WN}(c, Q)$, and c together with all the cells following it in the heap can all be ignored. In other words, the iteration can be terminated; (2) If c is not pruned, we further check whether c is a non-leaf node; (3) If c is a non-leaf node, then for each non-empty child node c_i of c, whose $d^{min}_{WN}(c_i, Q)$ is no larger than d_{Wk}, we insert $(c_i, d^{min}_{WN}(c_i, Q))$ into *Hnode* in ascending order of distance value (lines 10–13); (4) Otherwise, for each inverted list of $t_i \in Q.W \cup Q.B$ for cell c (denoted by $IL_{c_{t_i}}$), (a) if $IL_{c_{t_i}}$ does not exist or is empty, cell c can be pruned (by Lemma 2); (b) if t_i is a numeric attribute such that $Q.B.P_{t_i} \cap c.t_i.[v_i^-, v_i^+] = \varnothing$, or t_i is a categorial attribute such that $Q.B.P_{t_i} \cap c.t_i.pvset = \varnothing$, cell c cannot contain any result entity of query Q and can be safely pruned (by Lemma 3). otherwise, cell c may contain result entities, and needs further processing.

Then, each edge e in cell c are processed in ascending order of $d^{min}_{WN}(e.start, Q) + d^{min}_{WN}(e.end, Q)$ as follows. Similar to lines 6–8 and 16–19, edge e is checked whether contains candidate entities according to Lemmas 4–6 (lines 21–28). If false, edge e is skipped and the iteration for edge e breaks. Otherwise, (1) for each inverted list of t_i for edge e, i.e., $IL_{e_{t_i}}$ ($t_i \in Q.B$), the sublist which includes all the entities whose $V_{t_i}.v$ conforms to $Q.B.P_{t_i}$ is selected (lines 29–30); (2) all the relevant (sub) inverted lists $IL_{e_{t_i}}$ ($t_i \in Q.W \cup Q.B$) are sorted according to the number of entities on them.

We use $\widetilde{E}$ to represent a full set, and each $IL_{e_{t_i}}$ is intersected with $\widetilde{E}$ orderly. If $\widetilde{E}$ becomes $\emptyset$ after certain intersecting step, the following intersecting steps can be ignored. When all the intersecting operations of e are finished, the entities in $\widetilde{E}$ meet the textual requirement of query Q, and are inserted into $\widehat{E}_{cand}$ in ascending order of $d^{max}_{WN}(E, Q)$. Note that the value of d_{Wk} is modified correspondingly (lines 36–39).

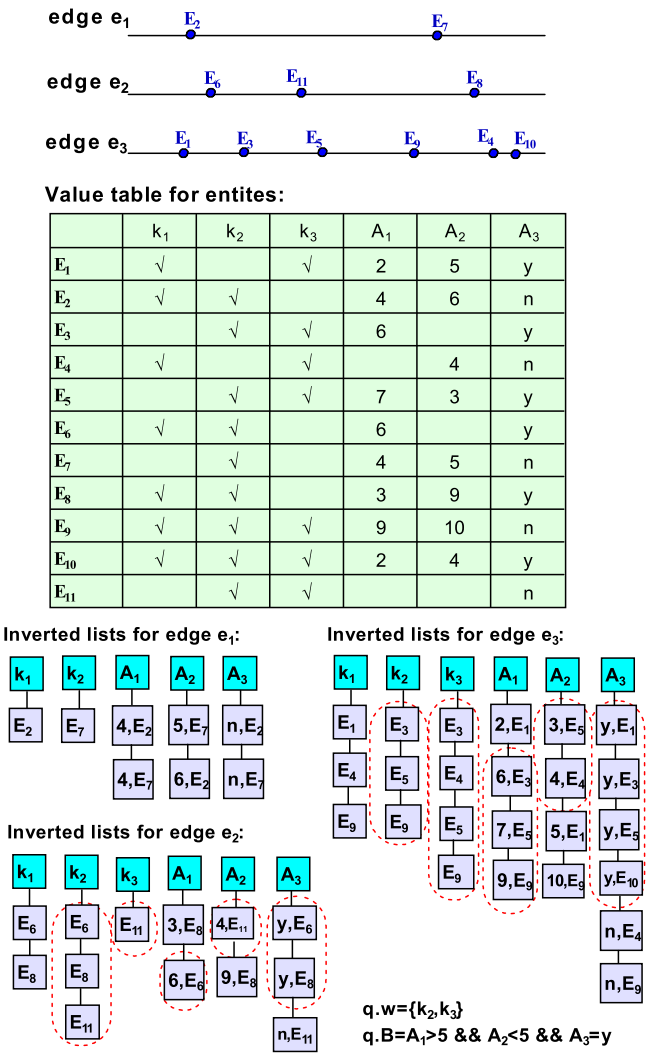
Example 2. Fig. 3 gives an example of how to process the entities on an edge to obtain candidate entities. As shown in the figure, 11 entities are distributed on three edges, e_1, e_2, and e_3, and the keywords and attribute-value pairs of the entities are shown in the value table. For example, entity E_1 locates on edge e_3, its keyword set is $\{k_1, k_2\}$, and it includes a set of attribute-value pairs which are $A_1 = 2 \cap A_2 = 5 \cap A_3 = y$. As discussed before, the inverted list of each item t for each edge is organized as follows: (1) if t is a keyword, the inverted list is arranged in ascending order of entity ID; (2) if t is an attribute, the inverted list is arranged in ascending order of attribute value, with ties resolved in favor of entities with smaller ID. For example, in the inverted list of A_3 for edge e_3, the entities whose value of A_3 equals 'y' (i.e., E_1, E_3, E_5, and E_{10}) are listed before those with 'n' value (i.e., E_4 and E_9). For E_1 and E_3, E_1 is listed before E_3 since it has a smaller ID.

Assume that $Q.W = \{k_2, k_3\}$ and $Q.B = A_1 > 5 \cap A_2 < 5 \cap A_3 = y$. Firstly, edge e_1 is safely pruned since it does not include

Algorithm 1: Pruning phase

```
1  begin
2      Ẽ, Ê_cand = ∅; float d_Wk = ∞; Hnode = ∅;
3      Locate the cell c_Q where query Q locates;
4      Push the root node of PMR quad tree together with its distance
       0 into Hnode;
5      WHILE:while Hnode ≠ ∅ do
6          c=de-heap(Hnode);
7          if d_E^min(c, Q) > d_Wk or d_N^min(c, Q) > d_Wk or d_WN^min(c, c_Q) > d_Wk)
           then
8              Break; //Lemma 1;
9          if c is a non-leaf node then
10             for each non-empty child node c_i of c do
11                 Calculate the minimum weighted distance
                   d_WN^min(c_i, Q) by using Equation 10;
12                 if d_WN^min(c_i, Q) ≤ d_Wk then
13                     Insert c_i and its distance d_WN^min(c_i, Q) into Hnode
                       in ascending order of distance value;
14         else
15             for ∀t_i ∈ Q.W ∪ Q.B do
16                 if IL_{c_{t_i}} is empty then
17                     Break WHILE; // Lemma 2;
18                 if Q.B.P_{t_i} ∩ c.t_i.[v_i^-, v_i^+] = ∅ or Q.B.P_{t_i} ∩
                   c.t_i.pvset = ∅ then
19                     Break WHILE; // Lemma 3;
20             FOR: for each edge e in cell c_i orderly do
21                 if d_WN^min(e.start, Q) > d_Wk & d_WN^min(e.end, Q) > d_Wk
                   then
22                     break; // Lemma 4;
23                 Ẽ = ⋃̄;
24                 for ∀t_i ∈ Q.W ∪ Q.B do
25                     if IL_{e_{t_i}} is empty then
26                         Break FOR; // Lemma 5;
27                     if Q.B.P_{t_i} ∩ e.t_i.[v_i^-, v_i^+] = ∅ or Q.B.P_{t_i} ∩
                       e.t_i.pvset = ∅ then
28                         Break FOR; // Lemma 6;
29                     else
30                         Select IL_{e_{t_i}} (t_i ∈ Q.W) or the sublist of IL_{e_{t_i}}
                           (t_i ∈ Q.B) where V_{t_i}.v conforms to Q.B.P_{t_i};
31                 Sort these IL_{e_{t_i}} lists (or sublists) according to the
                   number of entities on them;
32                 for each IL_{e_{t_i}} do
33                     Ẽ = Ẽ ∩ IL_{e_{t_i}} ;
34                     if Ẽ = ∅ then
35                         break;
36                 if Ẽ ≠ ∅ then
37                     for ∀E ∈ Ẽ do
38                         Insert (E, d_WN^max(E, q)) into Ê_cand in ascending
                           order of distance value;
39                     d_Wk= the distance value of the kth entity in Ê_cand;
```

the inverted list for k_3 (Lemma 5). As for edge e_2, according to the requirement of $Q.B$, the corresponding sub-inverted lists of attribute A_1, A_2, and A_3 are retrieved, which are marked by dotted ellipses. Moreover, the inverted lists of k_2 and k_3 are also retrieved and marked. Then, these marked inverted lists are sorted by their length, and the order is $IL_{e_{2_{k_3}}}$, $IL_{e_{2_{A_1}}}$, $IL_{e_{2_{A_2}}}$, $IL_{e_{2_{A_3}}}$, and $IL_{e_{2_{k_2}}}$. After conducting an intersecting operation between the two shortest lists, $IL_{e_{2_{k_3}}}$ and $IL_{e_{2_{A_1}}}$, an empty list is obtained. Thus e_2 is pruned



	k1	k2	k3	A1	A2	A3
E1	√		√	2	5	y
E2	√	√		4	6	n
E3		√	√	6		y
E4	√		√		4	n
E5		√	√	7	3	y
E6	√	√		6		y
E7		√		4	5	n
E8	√	√		3	9	y
E9	√	√	√	9	10	n
E10	√	√	√	2	4	y
E11		√	√			n

Fig. 3. An example of Top$k$-ASKQIV query processing.

by Lemma 5. As for edge e_3, five (sub) inverted lists are first retrieved (marked by dotted eclipses). After the orderly intersecting operations of these (sub) inverted lists, a candidate set $\{E_5\}$ can be obtained.

5.3. Verification phase

This subsection verifies the candidate entities kept in candidate set $\widehat{E}_{cand}$. Algorithm 2 presents the verification steps of Top-k ASKQIV query processing. S_{Topk} is used to preserve the result entities, and is ordered by the weighted network distance to Q of the entities in it. It first locates the edge e where query Q locates. Then, based on the adjacency and travel factor component, Algorithm 2 incrementally expands the traffic network of IoV from the location of Q ($Q.L$) with a method similar to Dijkstra's algorithm to search each candidate entity $E \in \widehat{E}_{cand}$. Our method distinguishes from Dijkstra's algorithm in that we use weighted network distance as the metric while network distance is used in Dijkstra's algorithm. During the processing, for each $E \in \widehat{E}_{cand}$ found, $d_{WN}(E, Q)$ is calculated and kept, and E together with its $d_{WN}(E, Q)$ is inserted into S_{Topk} in ascending order of $d_{WN}(E, Q)$. Note that each candidate entity $E \in \widehat{E}_{cand}$ includes all the query keywords and meets each predicate requirement of query Q, which means there are keyword match and boolean expression match between Q and E. Since

Algorithm 2: Verification phase

```
Input: Ê_cand;
OutPut: S_Topk;
begin
    S_Topk = ∅;
    Locate the edge where Q locates;
    Expand the traffic network from query Q by using Dijkstra-like
    method to search candidate entities E ∈ Ê_cand, and calculate
    d_WN(E, Q);
    for ∀E ∈ Ê_cand do
        Insert (E, d_WN(E, Q)) into S_Topk in ascending order of
        d_WN(E, Q);
        if there are k entities in S_Topk then
            break;
    Return S_Topk;
```

the network expansion from Q is in ascending order of weighted distance value, thus the first k candidate entities met form the query result set of Q.

5.4. Time complexity analysis

Now we discuss the time complexity of our Top-k ASKQIV processing algorithm.

Let $|C|$ be the number of leaf cells in the system, $|C_{remains}|$ be the number of cells not pruned by Lemmas 1, 2, and 3, $|Q.W|$ ($|Q.B|$) be the average number of query keywords (query attributes), L_{IL_e} be the average length of the inverted lists for an edge, e_c be the average number of edges within a cell, and e_{uskip} be the number of edges within a cell not pruned by Lemmas 4, 5, and 6.

Top-k ASKQIV processing algorithm follows the pruning-verification framework. In the pruning phase, it takes $(|C| \times (1 + |Q.W| + |Q.B|) + |C_{remains}| \times (e_c + e_{uskip} \times ((|Q.W| + |Q.B|) \times L_{IL_e})/\sqrt{w} + (|Q.W| + |Q.B|) \times L_{IL_e}))$ for obtaining the candidate set $\widehat{E}_{cand}$. The first $O(|C|)$ is for getting the distance bounds and additional traffic factors, and calculating $d^{min}_{WN}(c_i, Q)$ of each cell c_i, $O(|C| \times (|Q.W| + |Q.B|))$ is for using Lemmas 1, 2, and 3 to prune the unpromising cells, $O(|C_{remains}| \times e_c)$ is for skipping unpromising edges of each cell remained, and $O(|C_{remains}| \times e_{uskip} \times (|Q.W| + |Q.B|) \times L_{IL_e})/\sqrt{w} + (|Q.W| + |Q.B|) \times L_{IL_e}))$ is for calculating the candidate entities within the un-skipped edges of the cells remained. Note that we use the state-of-the-art method [54] to calculate the intersection result of the $(|Q.W| + |Q.B|)$ (sub) inverted lists of each un-skipped edge to get the promising entities which have keyword match and attribute match with query Q. In [54], with preprocessing and indexing on the list items, given k lists, with totally n elements, the time cost of intersection is $O(n/\sqrt{w} + kr)$, where r is the intersection size and w is the number of bits in a machine-word. In our method, k equals $|Q.W| + |Q.B|$, and n equals the sum of lengths of $|Q.W| + |Q.B|$ related (sub) inverted lists of an edge for query Q, and we enlarge n to $(|Q.W| + |Q.B|) \times L_{IL_e}$ since the lengths of the qualified (sub) invert lists of query attributes depend on specific query Q. As for r, it equals the shortest length of $|Q.W| + |Q.B|$ (sub) inverted lists, and we enlarge it to L_{IL_e} since this shortest length is also query dependent.

Let e_{cand} be the number of edges covering the region from query Q to its candidate entities, and $|\widehat{E}_{cand}|$ be the number of candidate entities.

In the verification phase, it takes $O(e_{cand} + |\widehat{E}_{cand}|)$ to eliminate all the false hits. In particular, $O(e_{cand})$ is for calculating the weighted network distance from query Q to each candidate entity, and $O(|\widehat{E}_{cand}|)$ is for verifying each candidate entity to get the result set S_{Topk}. Since the verification iteration will terminate when top-k result entities found, the actual cost of this phase is no larger than $O(e_{cand} + |\widehat{E}_{cand}|)$.

6. Discussion

This section discusses how the ASKTI index can be extended to handle more complex boolean expressions and operators, and the generality of our ASKTI index to support variants of ASKQIV queries.

6.1. Complex boolean expression supporting

Up to now, only a simple situation where the boolean expression of query Q is a conjunction of predicates has been discussed. However, in the practical applications, these simple boolean-expressions may not completely meet the requirement of query users. This subsection discusses how to extend our ASKTI index to support ASKQIV queries with boolean-expressions in CNF and DNF.

CNF: $(P_{11} \cup P_{12} \cup \cdots \cup P_{1n_1}) \bigcap \ldots \bigcap (P_{m1} \cup P_{m2} \cup \cdots \cup P_{mn_m})$
DNF: $(P_{11} \cap P_{12} \cap \ldots \cap P_{1n_1}) \bigcup \ldots \bigcup (P_{m1} \cap P_{m2} \cap \ldots \cap P_{mn_m})$

For CNF format, each disjunctive clause in a boolean expression is considered as a simple disjunctive boolean expression: i.e., $BE = BE_1 \bigcap BE_2 \bigcap \ldots \bigcap BE_m$, with each $BE_i = P_{i1} \cup \cdots \cup P_{in_i}$. Given an ASKQIV query Q, after retrieving the relevant (sub) inverted lists according to $Q.W$ and $Q.B$, for each set of (sub) inverted lists belonging to BE_i, a merging operation is conducted to form a longer inverted list, i.e., IL_{BE_i}. Then these m merged lists are used to do intersection with the inverted lists of $Q.W$.

ASKTI index can also be used to handle boolean expressions in DNF. After retrieving the relevant (sub) inverted lists according to $Q.W$ and $Q.B$, the (sub) inverted lists of $Q.B$ first do intersecting operation and then merging operation according to boolean expression logic. Similarly, the result list is used to do interaction with the inverted lists of $Q.W$.

ASKTI index can also be expanded to handle complex predicate operators.

Set operator $\in_s$ and $\in_i$ are usually used in boolean expressions. In our method, $\in_s$ and $\in_i$ can be rewritten by using the standard operators. For example, A $\in_s$ {2,4,6} can be rewritten into A = 2 $\bigcup$ A = 4 $\bigcup$ A = 6. Another example, A $\in_i$ [2,5] can be rewritten into A $\geq$ 2 $\bigcap$ A $\leq$ 5.

6.2. Processing boolean range ASKQIV queries

A boolean range ASKQIV query Q finds all the matched entities within a specified region. In a traffic network of IoV, a range usually refer to the roads (parts of roads) within a network distance limitation d_{limit} of the position of Q.

Our ASKTI index can be directly used to support boolean range queries in traffic networks of IoV. Specifically, in pruning phase, d_{Wk} is simply replaced by the distance limitation d_{limit} which remains unchanged during the whole phase. In verification phase, the candidate entities whose network distance to query Q is no larger than d_{limit} form the final result set.

6.3. Processing ranking ASKQIV queries

This subsection discusses how to extend our ASKTI index to handle ranking ASKQIV queries in traffic networks of IoV. In real life applications, the ranking function may take into account multiple factors, such as network distance and each spatial keyword, in the manner of weighted combination.

To efficiently support ranking ASKQIV queries, we slightly argument the ASKTI index by maintaining the keyword weight information in the two-level spatial–textual component. For each

Table 2
Characteristics of the data sets.

Attribute	T_a	T_b
Number of entities	10 M	1.2 M
Avg. number of keywords in entities	7	7
Avg. number of attributes in entities	5	5
Number of queries	1k	1k

cell (edge) in the component, the maximal keyword weight among the entities within the cell (edge) are also kept. Thus, the upper bound ranking score of each cell (edge) can be calculated by using the maximal keyword weights and the minimal network distance. Consequently, the cell (edge) with the highest upper bound can be accessed firstly, and the process terminates when no more better results can be found in the remaining cells.

7. Performance evaluation

This section evaluates the performance of our method for supporting intelligent augmented keyword search on spatial entities in real-life traffic networks of IoV environments. It first describes the experimental setup in Section 7.1, and then discusses the experimental results in Section 7.2.

7.1. Experimental settings

One real traffic network and two synthetic spatial entity data sets are employed in our experiment evaluation. To simulate the real-world traffic network of IoV, the real data of the traffic network for City of San Joaquin County in USA [55] is adopted. The San Joaquin traffic network consists of 16130 vertices and 20178 edges. The five additional traffic factors of each edge (road) are randomly selected from their value ranges, to simulate the impact of road traffic conditions in the real-life traffic networks. As for the weights of the traffic factors, we give high priority to weather condition (f_{wc}) and crowd congestion (f_{cc}) over other three factors according to people's driving experience.[2] The two synthetic spatial entity sets, T_a and T_b, both include a set of spatial entities, and the locations of entities are randomly distributed on the edges of the San Joaquin traffic network. The keywords and attributes of entities are obtained from Twitter (http://twitter.com), the number of keywords (attributes) in an entity varies from 5 to 9 (3 to 7), and the average number is 7 (5). The values of attributes are randomly selected from 1 to 1000. In T_a data set, all the keywords and attributes in an entity are randomly selected. The entity number of T_a is varied from 2 to 10 million to test the scalability of the methods. As for T_b data set, both the keywords and attributes follow the Zipf distribution ($\alpha = 1$), and the entity number of T_b varies from 0.4 to 1.2 million. A set of ASKQIV queries which include query locations, keywords, and boolean expressions, are also generated. The keywords and attributes of the ASKQIV queries are also obtained from Twitter. Besides, three main operators ($=, \leq, \geq$) are supported and the values of attributes are randomly selected from 1 to 1000. The query keyword number (query attribute number) varies from 1 to 5, and the default value is 2. Table 2 presents some characteristics of each data set.

To our best knowledge, there is no other work on dealing with ASKQIV problem up to now. Our ASKTI index based method will be compared with IR^2-tree [1] based method, ILQ (Inverted Linear Quadtree) [28] based method, and RP-tree [49] based method, which are efficient and often used in performance comparison tests for spatial keyword query processing. In general, memory consumption and processing time are the two main indicators of query processing, which we also use in the performance evaluation. IR^2-tree is an R-tree augmented with superimposed text signatures. The signature file, in the form of bitmap, is added to each node of the IR^2-tree, to indicate the presence of a given term (keyword) in the sub-tree of the node. IR^2-tree stores both spatial and textual information for a set of objects, and can be used to jointly prune the search space, which results in high efficiency. Since IR^2-tree method uses the Euclidean distance which is usually different from the true network distance in real traffic network, it may cause false query results, which may in turn reduce the precise of the method. To extend the IR^2-tree based method to handle ASKQIV queries precisely, we incorporate the PMR quad-tree and the adjacency and travel factor component of our ASKTI index into the IR^2-tree index. Thus, the IR^2-tree is first used to get the k relevant entities for a query, and then the PMR quad-tree and the adjacency file are used to calculate the weighted network distances between the k relevant entities and the query Q, and the largest weighted network distance of the k entities from Q are denoted as d_{Wk}. The k entities are inserted into a result list in ascending order of weighted network distance value. Next, resume the terminated expansion, and search for textual relevant entities, until it reaches d_{Wk} in each expanding direction. During the expansion, the relevant entities are also verified by the boolean expression of Q, and inserted into the result list in ascending order of weighted network distance. Finally, the first k entities in the result list form the final query result of Q.

ILQ naturally combines the advantages of inverted R-tree and IR^2-tree techniques. Specifically, for each keyword a linear quadtree is built for the related objects, so that the objects which do not contain any query keyword can be immediately excluded from computation. ILQ is carefully designed to exploit both spatial and keyword based pruning techniques to effectively reduce the search space. RP-tree integrates an R-tree index and a boolean expression index together, and can efficiently and simultaneously prune irrelevant objects based on boolean expressions and spatial dimension. In the RP-tree based method, an RP-tree is built based on the spatial and boolean expression information of spatial entities. Since the Euclidean distance is also used in ILQ and RP-tree index, the PMR quad-tree and the adjacency and travel factor component are also incorporated into ILQ and RP-tree index, and the following traffic network expansion and verification processing are similar to that of the IR^2-tree based method.

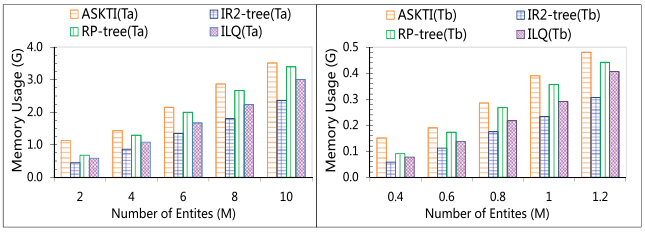
The following experiments verify the performance of the four methods by varying the number of entities, number of results, number of query keywords, and number of query attributes. The memory consumption for these methods is also evaluated. Moreover, the performance of handling boolean range ASKQIV queries and ranking ASKQIV queries is also evaluated.

Table 3 shows the main parameters and values used, and the default values are presented in bold face.

Table 3
Parameters evaluated in the experiments.

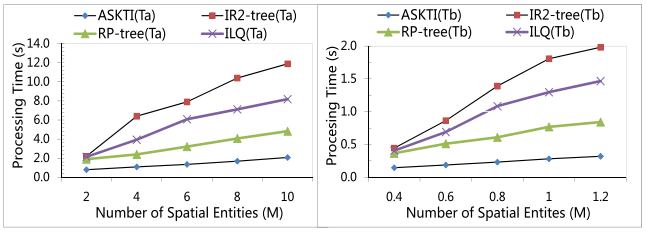
Parameter	Values
Number of entities	T_a: 2, 4, **6**, 8, 10 (M) T_b: 0.4, 0.6, **0.8**, 1.0, 1.2 (M)
Number of query results (K)	5, 10, **15**, 20, 25
Number of query keywords	1, **2**, 3, 4, 5
Number of query attributes	1, **2**, 3, 4, 5
The diameter of query range (%)	0.6, 0.8, **1.0**, 1.2, 1.4 (percent of traffic network width)
Preference parameter α	0.1, 0.3, **0.5**, 0.7, 0.9

[2] Due to space limitation, we leave how to get the true additional traffic factors and the more reasonable weights of traffic factors to the future work. Moreover, we will also discuss how to dynamic adjust these factors and weights in real-time.



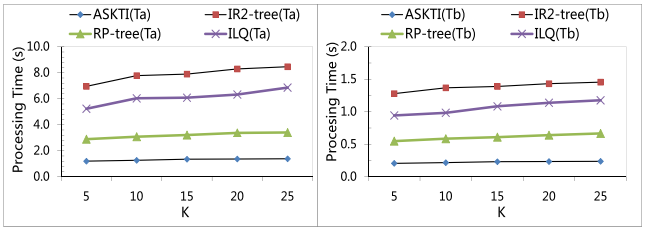
(a) T_a data set (b) T_b data set

Fig. 4. Effect of number of spatial entities on memory consumption.



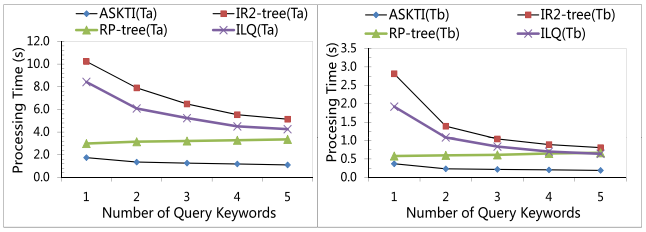
(a) T_a data set (b) T_b data set

Fig. 5. Effect of number of spatial entities on processing time.



(a) T_a data set (b) T_b data set

Fig. 6. Effect of number of results on processing time.



(a) T_a data set (b) T_b data set

Fig. 7. Effect of number of query keywords on processing time.

7.2. Experimental results

7.2.1. Memory consumption

The memory consumption of the four methods is first evaluated. As shown in Fig. 4, the memory space needed for the methods all increases when there are more spatial entities in the system. The more entities there are, the more space the entities take up. In general, the RP-tree based method and the ASKTI based method consume more memory resources than the IR^2-tree based method and the ILQ based method for both t_a and t_b data sets, since the former two methods need to build an inverted list for each attribute and the entity ID is stored with multiple copies in the inverted lists.

7.2.2. Varying the number of spatial entities

As shown in Fig. 5, when the number of entities in the system increases, the running time of IR^2-tree, RP-tree, ILQ, and ASKTI all increases. As entity cardinality becomes larger, more entities need to be considered whether could be one of results of the query. Obviously, the increase tread of IR^2-tree is rapid than other three methods, since IR^2-tree is centralized index that all entities are inserted into one tree. On the contrary, RP-tree (ASKTI, resp.) is inverted list-based index, and partitions the entities based on the attributes (keywords and attributes, resp.). Thus, the entity which does not contain any query attribute (query keyword or query attribute, resp.) can be pruned safely, which makes these two methods scalable than IR^2-tree method. As for ILQ method, the entities are indexed by the quadtrees of the keywords they contain, thus large amounts of irrelevant objects can be pruned by query keywords. In general, the performance of ASKTI is better than its competitors due to the cell pruning and edge pruning techniques based on weighted network distance, keywords and boolean expression dimensions simultaneously. On the contrary, IR^2-tree based method and ILQ based method prune the search space based on Euclidean distance and query keywords simultaneously, and RP-tree based method prunes the search space based on Euclidean distance and boolean expression dimensions simultaneously. These three methods are less effective than ASKTI in space pruning, and therefore less efficient than ASKTI. For example, when the entity number is 6M, ASKTI requires only 42.02% (17.09% and 22.19%, resp.) processing time of the compared RP-tree (IR^2-tree and ILQ, resp.) method for the T_a data set.

7.2.3. Varying the number of results

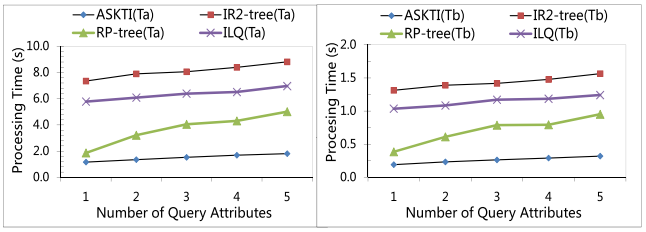
Next, the effect of varying the number of results (value k) on the performance of these four methods is evaluated. Fig. 6 shows that the processing time of the methods do not increase very much with increasing k. In IR^2-tree method, the increase tendency for T_b data set is less obvious than that for T_a data set. This is because that there are less entities in T_b, they are less entities meeting the requirement of both query keywords and query attributes, and those candidate entities may locate in a large range of the search space, thus most of the tree nodes still need to be accessed even for smaller k value. The performance of ILQ and RP-tree both degrades with larger k, since larger k means more relevant entities need to be retrieved and verified. For our ASKTI method, the effect of increasing k is not obvious due to our efficient pruning techniques.

7.2.4. Varying the number of query keywords

This set of experiments examine the scalability of the methods as the number of query keywords increases. As shown in Fig. 7, the processing time of IR^2-tree, ILQ, and ASKTI decreases with the increment of query keyword number. This is because that a candidate tree node (cell) is required to contain all the query keywords, and most of the nodes (cells) in IR^2-tree and PMR quad tree of our ASKTI can be safely pruned when there are many query keywords. Similarly, the candidate entities of ILQ method is fewer for larger query keyword number, thus cutting down the total processing cost. The running time of RP-tree method slightly increases in both data sets since it require more verification cost to determine whether an entity includes all the query keywords. As expected, our ASKTI achieves the best performance among all the four methods.

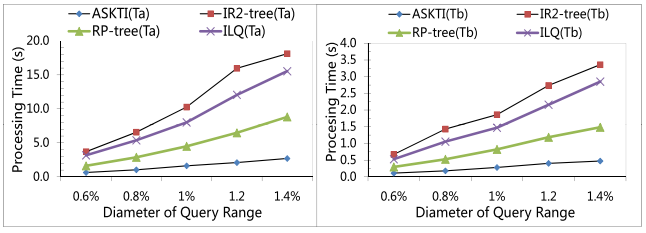
7.2.5. Varying the number of query attributes

The effect of number of query attributes on the performance of these methods is also evaluated. As we can see from Fig. 8, the processing time of the methods all increases as the number of query attributes increases. For IR^2-tree and ILQ, the running time slightly increases since more verification cost is needed to determine whether an entity matches the boolean expression of the



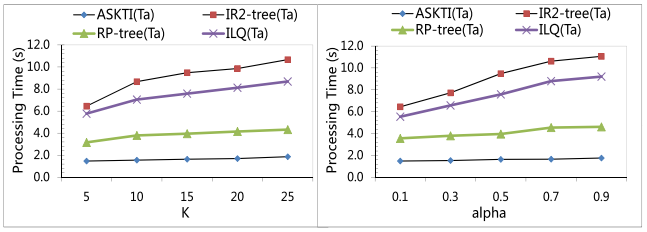
(a) T_a data set (b) T_b data set

Fig. 8. Effect of number of query attributes on processing time.

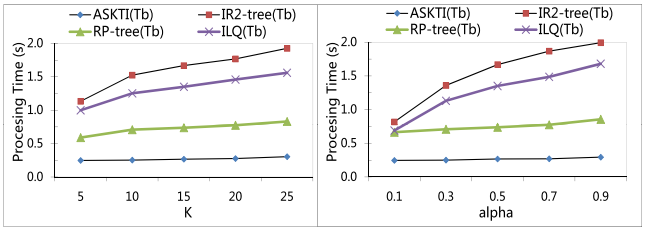


(a) T_a data set (b) T_b data set

Fig. 9. Processing boolean range ASKQIV queries.



(a) T_a data set (b) T_a data set



(c) T_b data set (d) T_b data set

Fig. 10. Processing ranking ASKQIV queries.

query. For RP-tree, more query attributes means more partitioned subscription lists need to be accessed, incurring higher processing overhead. As for our ASKTI, the reason is twofold. Firstly, a candidate cell (edge) is required to contain all the query attributes, and most of the cells (edges) in the PMR quad tree of our ASKTI can be safely pruned when there are many query attributes. On the other hand, as query attributes increase, more relevant reverse lists need to be accessed. In general, the latter factor outweighs the former, thus the total processing time of ASKTI increases slightly with the increase in query attributes.

7.2.6. Processing boolean range ASKQIV queries

We then evaluate the performance of the four methods for processing boolean range ASKQIV queries with different query ranges. Fig. 9 illustrates the processing time of the methods when the diameter of query range increases from 0.6% to 1.4% of the traffic network width. As can be observed, the cost of these four methods increases with the increase of query scope. This is because a larger query range means there are more entities in a given query region and more information needs to be retrieved and evaluated. Generally speaking, the performance of our ASKTI is much better than that of its competitors. On average, it incurs only 15.39% (34.93% and 18.90%, resp.) and 14.61% (34.33% and 18.50% resp.) of processing time compared to IR^2-tree (RP-tree and ILQ, resp.) for T_a and T_b data sets, respectively.

7.2.7. Processing ranking ASKQIV queries

In the final set of experiments, we evaluate the performance of these four methods for processing ranking ASKQIV queries. Similar to many previous work [56] for ranking spatial keyword queries, our ranking function is a linear combination of keyword rating score and normalized network distance, and a query preference parameter α is set to strike a proper balance between text relevance and network proximity. In particular, a larger value α increases the importance of textual relevance over network proximity. Fig. 10 illustrates the results of methods with varying k and α. As expected, the processing time of all the methods increases with the increment of k, since more related entities need to be considered when more result entities are wanted. The performance of the methods degrades as α increases, because the result entities may exist in a larger range of the traffic network of IoV for larger α. An entities can be one of the query result as long as its text relevance with query q is sufficiently large, even it is far away from query Q. Our ASKTI is more scalable when the k and α vary due to its strong pruning ability, which takes into account query keywords, boolean expressions, and network distances, intelligently and synchronously.

8. Conclusion

This paper addresses the issue of processing intelligent augmented spatial keyword queries in real-life traffic networks of IoV environments (ASKQIV). ASKQIV is a new type of queries which considers keywords, expressions, spatial locations, traffic condition, and traffic network structure, intelligently and simultaneously. To address ASKQIV problem, an efficient hybrid index, called ASKTI, is designed. In particular, ASKTI consists of three components, two-level spatial–textual component, adjacency and travel factor component, and distance bound and traffic factor bound component. Moreover, several lemmas are proposed to prune huge amounts of irrelevant spatial entities for queries. Based on ASKTI, an efficient and intelligent algorithm for Top-k ASKQIV query processing is proposed. Our method can also be extended to handle boolean range ASKQIV Queries and ranking ASKQIV Queries. Finally, experimental study on a real road network of IoV and two spatial entity sets to demonstrate the efficiency of the proposed index and query processing algorithm. The results show that the proposed method is more efficient and scalable than its competitors. In future work, the proposed method will be extended to handle other types of queries, such as skyline queries, in IoV. We will discuss how to get the true additional traffic factors and the more reasonable weights of traffic factors. Moreover, we will also discuss how to dynamic adjust these factors and weights in real-time.

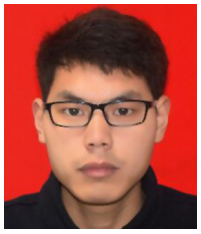
Acknowledgments

This work is supported by National Science Foundationof China (No. 61309002, No. 61772562, No. 61272497), the Hubei Provincial Natural Science Foundation of China for Distinguished Young Scholars (No. 2017CFA043), the Hubei Provincial Natural Science Foundation of China (No. 2017CFB135), Fundamental Research Funds for the Central Universities (CZZ17003, CZP17043), the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No. VRLAB2018B03), and the Youth Elite Project of State Ethnic Affairs Commission of China.

References

[1] I.D. Felipe, V. Hristidis, N. Rishe, Keyword search on spatial databases, in: IEEE International Conference on Data Engineering, 2008, pp. 656–665.
[2] D. Zhang, K.L. Tan, A.K.H. Tung, Scalable top-k spatial keyword search, in: International Conference on Extending Database Technology, 2013, pp. 359–370.
[3] K. Zheng, H. Su, B. Zheng, S. Shang, J. Xu, J. Liu, X. Zhou, Interactive top-k spatial keyword queries, in: IEEE International Conference on Data Engineering, 2015, pp. 423–434.
[4] y.E. Carlsson, Keyword search on spatial network databases : Road network indexing for efficient query processing, Inst. Datateknikk Og Informasjonsvitenskap (2011).
[5] Y. Gao, X. Qin, B. Zheng, G. Chen, Efficient reverse top-k boolean spatial keyword queries on road networks, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1205–1218.
[6] G. Li, Q. Zhou, J. Li, A novel scheduling algorithm for supporting periodic queries in broadcast environments, IEEE Trans. Mob. Comput. 14 (12) (2015) 2419–2432.
[7] Q. Zhou, G. Li, J. Li, L.C. Shu, C. Zhang, F. Yang, Dynamic priority scheduling of periodic queries in on-demand data dissemination systems, Inf. Syst. 67 (2017) 58–70.
[8] D. Zhang, Y. Li, X. Cao, J. Shao, H.T. Shen, Augmented keyword search on spatial entity databases, Vldb J. 27 (2) (2018) 225–244.
[9] Y. Zhang, R. Yu, S. Xie, W. Yao, Y. Xiao, M. Guizani, Home m2m networks: Architectures, standards, and qos improvement, Commun. Mag. IEEE 49 (4) (2011) 44–52.
[10] R. Yu, Y. Zhang, S. Gjessing, W. Xia, K. Yang, Toward cloud-based vehicular networks with efficient resource management, IEEE Netw. 27 (5) (2013) 48–55.
[11] T. Wang, L. Song, Z. Han, Coalitional graph games for popular content distribution in cognitive radio vanets, IEEE Trans. Veh. Technol. 62 (8) (2013) 4010–4019.
[12] M. Jin, X. Zhou, E. Luo, X. Qing, Industrial-qos-oriented remote wireless communication protocol for the internet of construction vehicles, IEEE Trans. Ind. Electron. 62 (11) (2015) 7103–7113.
[13] M. Zhang, T. Wo, T. Xie, X. Lin, Y. Liu, Carstream: an industrial system of big data processing for internet-of-vehicles, Proc. VLDB Endow. 10 (12) (2017) 1766–1777.
[14] H. Song, B.R. Danda, J. Sabina, B. Christian, Cyber-Physical Systems: Foundations, Principles and Applications, Morgan Kaufmann, 2016.
[15] M. Chen, W. Li, Y. Hao, Y. Qian, I. Humar, Edge cognitive computing based smart healthcare system, Future Gener. Comput. Syst. (2018).
[16] C. Chen, H. Xiang, T. Qiu, C. Wang, Y. Zhou, V. Chang, Rear-end collision prediction scheme based on deep learning in the internet of vehicles, J. Parallel Distrib. Comput. (2017).
[17] S. Maharjan, Q. Zhu, Y. Zhang, S. Gjessing, T. Basar, Dependable demand response management in the smart grid: A stackelberg game approach, IEEE Trans. Smart Grid 4 (1) (2013) 120–132.
[18] Y. Zhang, R. Yu, M. Nekovee, Y. Liu, S. Xie, S. Gjessing, Cognitive machine-to-machine communications: visions and potentials for the smart grid, IEEE Netw. 26 (3) (2012) 6–13.
[19] H. Song, S. Ravi, S. Tamim, J. Sabina, Smart Cities: Foundations, Principles, and Applications, John Wiley and Sons, 2017.
[20] H. Song, A.F. Glenn, J. Sabina, Security and Privacy in Cyber-physical Systems: Foundations, Principles, and Applications, John Wiley and Sons, 2017.
[21] A. Paul, A. Daniel, A. Ahmad, S. Rho, Cooperative cognitive intelligence for internet of vehicles, IEEE Syst. J. 11 (3) (2017) 1249–1258.
[22] K.W. Chen, C.H. Wang, X. Wei, Q. Liang, C.S. Chen, M.H. Yang, Y.P. Hung, Vision-Based positioning for internet-of-vehicles, IEEE Trans. Intell. Transp. Syst. 18 (2) (2017) 364–376.
[23] M. Chen, Y. Tian, G. Fortino, J. Zhang, I. Humar, Cognitive internet of vehicles, Comput. Commun. 120 (2018) 58–70.
[24] M. Chen, Y. Hao, K. Lin, Z. Yuan, L. Hu, Label-less learning for traffic control in an edge network, IEEE Netw. (2018).
[25] Y. Qian, M. Chen, J. Chen, M. Hossian, A. Alamri, Secure enforcement in cognitive internet of vehicles, IEEE IoT J. 5 (3) (2018) 1242–1250.
[26] Y. Zhou, X. Xie, C. Wang, Y. Gong, W.Y. Ma, Hybrid index structures for location-based web search, in: Proc. of the 14th ACM International Conference on Information and Knowledge Management, 2005, pp. 155–162.
[27] G. Cong, C.S. Jensen, D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, Proc. VLDB Endow. 2 (1) (2009) 337–348.
[28] C. Zhang, Y. Zhang, W. Zhang, X. Lin, Inverted linear quadtree: Efficient top k spatial keyword search, in: IEEE International Conference on Data Engineering, 2013, pp. 901–912.
[29] L. Chen, G. Cong, C.S. Jensen, D. Wu, Spatial keyword query processing: an experimental evaluation, Proc. VLDB Endow. 6 (3) (2013) 217–228.
[30] W. Huang, G. Li, K.-L. Tan, J. Feng, Efficient safe-region construction for moving top-k spatial keyword queries, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, 2012, pp. 932–941.
[31] G. Li, J. Feng, J. Xu, Desks: Direction-aware spatial keyword search, in: Proceedings of the 28th International Conference on Data Engineering, 2012, pp. 474–485.
[32] X. Cao, G. Cong, C.S. Jensen, B.C. Ooi, Collective spatial keyword querying, in: ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June, 2011, pp. 373–384.
[33] B. Zheng, N.J. Yuan, K. Zheng, X. Xie, S. Sadiq, X. Zhou, Approximate keyword search in semantic trajectory database, in: IEEE International Conference on Data Engineering, 2015, pp. 975–986.
[34] J. Lu, Y. Lu, G. Cong, Reverse spatial and textual k nearest neighbor search, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, ACM, 2011, pp. 349–360.
[35] Y. Yuan, X. Lian, L. Chen, J. Yu, G. Wang, Y. Sun, Keyword search over distributed graphs, IEEE Trans. Knowl. Data Eng. PP (99) (2017) 1–1.
[36] L. Guo, J. Shao, H.H. Aung, K.L. Tan, Efficient continuous top-k spatial keyword queries on road networks, GeoInformatica 19 (1) (2015) 29–60.
[37] X. Wang, L. Guo, C. Ai, J. Li, Z. Cai, An Urban Area-Oriented Traffic Information Query Strategy in VANETs, Springer Berlin Heidelberg, 2013.
[38] T. Anakavej, A. Kawewong, K. Patanukhom, Internet-vision based vehicle model query system using eigenfaces and pyramid of histogram of oriented gradients, in: International Conference on Signal-Image Technology and Internet-Based Systems, 2013, pp. 179–186.
[39] Y. Li, S. Lei, J. Li, R. Zhu, Y. Chen, Spatial Keyword Query Processing in the Internet of Vehicles, Springer International Publishing, 2017, pp. 1–13.
[40] L. Wang, Q.Y. Niu, Research on digital intelligent query optimization of digital library based on user experience, Digital Libr. Forum (2017).
[41] K. Kolomvatsos, S. Hadjiefthymiades, Learning the engagement of query processors for intelligent analytics, Appl. Intell. 46 (1) (2017) 1–17.
[42] H. Wang, P.-C. Lim, Y. Wang, S. Chai, Sevqer: Automatic semantic visual query builder to support intelligent image search in traffic images, J. Comput. Sci. 14 (7) (2018) 1053–1063.
[43] A. Farroukh, M. Sadoghi, H.A. Jacobsen, Towards vulnerability-based intrusion detection with event processing, in: ACM International Conference on Distributed Event-Based System, 2011, pp. 171–182.
[44] M. Sadoghi, H.A. Jacobsen, Be-tree:an index structure to efficiently match boolean expressions over high-dimensional discrete space, in: ACM SIGMOD International Conference on Management of Data, 2011, pp. 637–648.
[45] D. Zhang, C.Y. Chan, K.L. Tan, An efficient publish/subscribe index for e-commerce databases, Proc. VLDB Endow. 7 (8) (2014) 613–624.
[46] Z. Chen, G. Cong, Z. Zhang, T.Z.J. Fuz, L. Chen, Distributed publish/subscribe query processing on the spatio-textual data stream, in: IEEE International Conference on Data Engineering, 2017.
[47] J. Li, J.J. Chen, M. Xiong, G. Li, W. Wei, Temporal consistency maintenance upon partitioned multiprocessor platforms, IEEE Trans. Comput. 65 (5) (2016) 1632–1645.
[48] L. Guo, D. Zhang, G. Li, K.L. Tan, Z. Bao, Location-aware pub/sub system:When continuous moving queries meet dynamic event streams, in: ACM SIGMOD International Conference on Management of Data, 2015, pp. 843–857.
[49] P. Zhao, H. Jiang, J. Xu, V.S. Sheng, G. Liu, A. Liu, J. Wu, Z. Cui, Location-aware publish/subscribe index with complex boolean expressions, World Wide Web-Int. Web Inf. Syst. (2017) 1–22.
[50] I. Tsapakis, T. Cheng, A. Bolbol, Impact of weather conditions on macroscopic urban travel times, J. Transp. Geogr. 28 (2) (2013) 204–211.
[51] T. Kato, K. Uchida, A study on benefit estimation that considers the values of travel time and travel time reliability in road networks, vol. 14, no. 1–2, 2017, pp. 1–25.
[52] K. Uchida, T. Kato, A simplified network model for travel time reliability analysis in a road network, J. Adv. Transp. 2017 (476) (2017) 1–17.
[53] Y. Wang, C. Xu, Y. Gu, M. Chen, G. Yu, Spatial query processing in road networks for wireless data broadcast, Wirel. Netw. 19 (4) (2013) 477–494.
[54] B. Ding, Fast set intersection in memory, vol. 4, no. 4, 2011, pp. 255–266.
[55] F.S. University, http://www.cs.fsu.edu/lifeifei/spatialdataset.htm, 2013.
[56] D. Zhang, C.Y. Chan, K.L. Tan, Processing spatial keyword query as a top-k aggregation query, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2014, pp. 355–364.

Yanhong Li is currently an associate professor at the College of Computer Science of South-Central University for Nationalities, China. She received her PhD degree from Huazhong University of Science and Technology (HUST), China, in 2011. Her research interests include spatial information and communication, and multimedia network technology.



Meng Wang is currently a master student at the College of Computer Science of South-Central University for Nationalities, China. He graduated from University of South China, in 2016. His research interests include spatial information and communication, and multimedia network technology.

Rongbo Zhu is currently a Professor in College of Computer Science of South-Central University for Nationalities, China, and he is the director of Institute of Smart Cities. He received the B.S. and M.S. degrees in Electronic and Information Engineering from Wuhan University of Technology, China, in 2000 and 2003, respectively; and Ph.D. degree in communication and information systems from Shanghai Jiao Tong University, China, in 2006. From August 2011 to August 2012, he was a research scholar in CNSR group at Virginia Tech, USA. Dr. Zhu has published over 70 papers in international journals and conferences in the areas of mobile computing and cognitive wireless networks. He is an Associate Editor IEEE Access, International Journal of Radio Frequency Identification Technology and Applications, and Lead Guest Editor of several international journals.

Ashiq Anjum is currently a professor of Distributed Systems at the University of Derby, UK. His research interests include data intensive distributed systems, block chain, Internet of Things and high performance analytics platforms. Currently he is investigating high performance distributed platforms to efficiently process video and genomics data. Dr. Anjum has more than 70 international academic publications. He has been part of the EC funded projects in distributed systems and large scale analytics such as Health-e-Child (IP, FP6), neuGrid (STREP, FP7) and TRANSFORM (IP, FP7). He has been a general chair, programmer chair, organizing chair, track chair, publicity chair and workshop chair for more than 20 international conferences. Prof. Anjum has been a member of the technical program committees for more than 40 international conferences. He is the member of British Computer Society, IEEE, ACE, and SIGHPC. He is the Fellow of Higher Education Academy and the champion of European Grid Infrastructure.

Xiaokun Du is currently an lecturer at the College of Computer Science of South-Central University for Nationalities, China. He received her PhD degree from Huazhong University of Science and Technology (HUST), China, in 2010. Her research interests include data integration, text mining and sentiment analysis.

Yuhe Feng is currently the director of R&D of Guangdong Intelligent Robotics Institute, Dongguan, China. He received his bachelor degree from Central South University, China, in 1992. His research interests include intelligent manufacturing, industrial big data processing, and spatial information and communication.

Changyin Luo is currently a Lecturer at School of Computer in Central China Normal University. He received his PhD from Huazhong University of Science and Technology (HUST) in 2015. His research interests include spatial database, spatial keyword search and social media data processing.

Shasha Tian is currently a lecturer at the South-central University for Nationality, Wuhan, China. She received her master's degree in computer science from South-central University for Nationality in 2009. Her research interests are intelligent algorithms, and location query based on space–time.