IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 11, 2018, accepted February 7, 2018, date of publication February 15, 2018, date of current version March 13, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2806391

Efficient Dynamic Service Maintenance for Edge Services

YIWEN ZHANG[1], JIN LI[1], ZHANGBING ZHOU[2], AND XIAO LIU[3], (Member, IEEE)

[1]Department of Computer Science and Technology, Anhui University, Hefei 230601, China
[2]School of Information Engineering, China University of Geosciences, Beijing 100083, China
[3]School of Information Technology, Deakin University, Melbourne, VIC 3122, Australia

Corresponding author: Xiao Liu (xiao.liu@deakin.edu.au)

This work was supported in part by the National Key Technology Research and Development Program under Grant 2015BAK24B01, in part by the General Research for Humanities and Social Sciences Project of Chinese Ministry of Education under Grant 15YJAZH112, in part by the Key Project of Nature Science Research for Universities of Anhui Province of China under Grant KJ2016A038, and in part by the National Natural Science Foundation of China under Grant 61300042.

ABSTRACT The emergence of many new computing applications, such as Internet of Vehicles (IoV) and smart homes, has been made possible by the large pool of cloud resources and services. However, the cloud computing paradigm is unable to meet the requirements of delay-sensitive business applications, such as low latency, mobility support, and location awareness. In this context, Mobile Edge Computing (MEC) is introduced to improve the quality of experience (QoE) by bringing cloud resources and services closer to the user by leveraging available resources in the edge networks. However, the performance of MEC is dynamic in nature due to its location awareness, mobility and proximity. As a result, an effective mechanism is needed for providing efficient dynamic service maintenance for edge services. In this paper, we propose applying the Skyline Graph Model and employing the Directed Acyclic Graph theory to store and update mobile edge services. Specifically, the Skyline Graph (SG) algorithm is designed to solve the insertion, deletion, updating and searching of mobile edge services to achieve efficient maintenance for edge services. Comprehensive experiments are conducted on both real-world web services and simulated datasets to evaluate the effectiveness and efficiency of our approaches. The results show that our algorithms can achieve significantly better performance and robustness than the baseline algorithm.

INDEX TERMS Mobile edge computing, edge service, service maintenance, skyline, skyline graph.

I. INTRODUCTION

With the rapid growth of the mobile Internet and the Internet of Things (IOT), numerous intelligent terminals (such as sensors and cameras) are being widely used in the areas of manufacturing, transportation, smart homes and environmental protection. As a result, many new applications (such as Internet of Vehicles (IoV), smart home and Augmented Reality (AR)) are becoming increasingly popular [1]. However, the limited resources of mobile devices are insufficient for meeting the application demand. Therefore, we often need to offload computation-intensive applications to the cloud and leverage the rich computing resources in cloud data centers [2]. One typical problem with cloud-based applications is the long-distance communication between the users' devices and the cloud, which may cause intermittent connectivity and long latency, which cannot satisfy the requirements of emerging interactive applications such as real-time face recognition and online gaming [3]. To tackle this issue, various computing paradigms, such as fog computing [4], cloudlets and edge computing [5], were introduced. Fog computing can be considered an expanded cloud computing model from the core network to the edge network. Fog is highly virtualized and factors such as latency, energy-consumption, network flow, capital and operation cost are taken into consideration to promote mobility support, real-time interaction and scalability. Mobile Edge Computing (MEC) is introduced to bring the cloud services and resources closer to the user by leveraging available resources in the edge networks [6]. MEC can not only improve the performance of user applications but also reduce the volume of network traffic. Edge service providers need to deploy edge services in the local network for mobile users to improve their productivity. As a result, an increasing number of new edge services are being deployed by edge service providers to increase the capabilities and context-awareness of edge services.

However, given the fast growth of the mobile edge infrastructure and edge devices, the number of edge services has surged. In addition, mobile edge services can be relocated

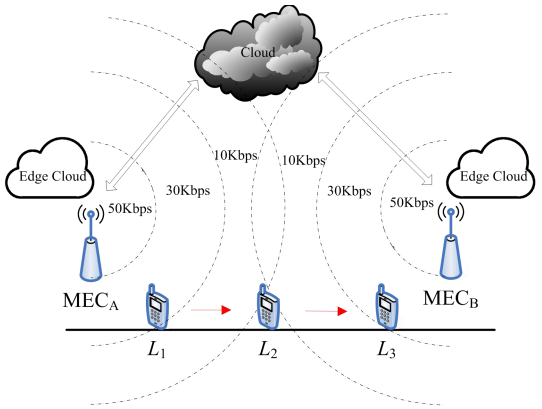


FIGURE 1. Example of edge service maintenance in mobile networks.

or changed due to changes in user locations. As a result, the quality of customer experience may fluctuate. Clearly, MEC requires an effective mechanism for providing efficient maintenance of dynamic edge services. Such a mechanism needs to support the insertion, deletion, update and search of the mobile edge services. An example of edge service maintenance in mobile networks is shown in Fig. 1. Here, if a local edge system contains two edge nodes, which are denoted as MEC_A and MEC_B, the coverage of its broadband will resemble the arcs that are shown in the example. If a user moves from location L_1 to L_3 and passes by L_2, then during the movement, the following will occur: (1) the quality of the edge service will change (50 Kbps→30 Kbps→10 Kbps); (2) the edge service will be relocated or changed ($MEC_A \rightarrow MEC_B$). In such a case, maintaining the quality of experience (QoE) of users is a big challenge. To tackle this challenge, this paper investigates a mechanism for efficient maintenance of dynamic edge services in the mobile edge computing environment. Our contributions in this paper can be summarized as follows:

1) We propose the use of the Skyline Graph Model, which can provide the storage and update of mobile edge services based on the theory of directed acyclic graphs. Given the dominance relationship and transitivity of the skyline, it can achieve the efficient storage and dynamic update of mobile edge services, to solve the fast selection problem of MEC.
2) Based on the Skyline Graph Model, this paper proposes the Skyline Graph Algorithm, which can help accelerate the building of dominance and anti-dominance regions and further optimize the insertion, deletion, update and search of the edge services to realize efficient maintenance of dynamic edge services.
3) Comprehensive experiments that are based on both real-world and simulated datasets are conducted and the results successfully demonstrate the effectiveness and efficiency of our algorithm.

The remainder of the paper is organized as follows: Section II introduces the related work. Section III presents a formal description of the research questions. Section IV introduces the Skyline Graph Model and Section V proposes the Skyline Graph Algorithm in detail. Section VI presents the evaluation results. Finally, Section VII presents the conclusions of the paper.

II. RELATED WORK

A. EDGE AS A SERVICE

To meet the needs of high bandwidth and low latency due to the fast development of the mobile Internet and IoT, the concept of mobile edge computing has been introduced and has attracted wide attention from both academy and industry. Currently, the research on mobile edge computing mainly covers edge-cloud placement, computation offloading, edge-cloud service migration, group intelligence synergy and the application of mobile edge computing. Xu *et al.* [7] studied the placement of the capacity-limited edge cloud in large-scale WMAN. They investigated the allocation of the requirements of mobile users to the edge cloud to minimize the average access delay between the mobile users and the edge-cloud services. Xiang *et al.* [8] proposed self-adaptive edge-cloud placement based on the locations of mobile applications. The core idea of the method is to maximize the number of mobile devices that are covered by the active area of the edge cloud. Computation offloading can use the capability of the cloud resource to expand the capability of a mobile device and decrease its energy consumption to improve the quality of experience of the mobile edge services. Eduardo *et al.* [9] presented the MAUI computation offloading model in 2010. Wang *et al.* [10] incorporated dynamic voltage scaling (DVS) into computation offloading. You *et al.* [11] aimed at minimizing the mobile energy consumption and studied the resource allocation for a multi-user mobile-edge computation offloading (MECO) system based on time-division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA). Zhang *et al.* [12] studied energy-efficient computation offloading (EECO) mechanisms for MEC in 5G heterogeneous networks. Sardellitti *et al.* [13] formulated the offloading problem as the joint optimization of the radio resources and the computational resources. They aimed at minimizing the weighted sum of energy consumption in a multiple-input multiple-output system and proposed a successive-convex-approximation-based iterative algorithm. Varghese *et al.* [14] presented an Edge-as-a-Service (EaaS) platform. They aimed at realizing distributed cloud architectures and integrating the edge of the network into the computing ecosystem. Xu *et al.* [15] presented EAaaS, which is a scalable analytic service for enabling real-time edge analytics in IoT scenarios. Kaur *et al.* [16] adopted lightweight containers instead of the conventional virtual machines to reduce the overhead, response time, and overall energy consumption of fog devices. Therefore, the deployment of edge services will not only improve the performance of mobile applications but also reduce the volume of signaling traffic to the cloud, to improve the QoE of the mobile users. However, due to the

location relevance and mobility of mobile edge computing, the quality of edge services changes dynamically. In addition, other situations such as state unavailability and relocation of edge services may also occur frequently in MEC. Therefore, an effective mechanism for edge service maintenance is needed to meet the location-awareness, portability and proximity requirements of mobile edge services. In this paper, we will apply the theory of directed acyclic graphs to facilitate the storage and update of mobile edge services and design a dynamic maintenance algorithm for achieving the efficient selection and maintenance of dynamic mobile edge services.

B. SKYLINE QUERY FOR SERVICE SELECTION

The Skyline algorithm has been widely used in the areas of database query and multiple-objective decision making. It was first proposed by Borzsonyi *et al.* [17]. They designed two algorithms for calculating the Skyline: Block Nested Loop (BNL) and Divide and Conquer (D&C). BNL is a simple algorithm that directly compares every point to every other point (and adds non-dominated points to the Skyline set). D&C divides the data space. After every part has calculated its own Skyline set, it combines the results. Chomicki *et al.* [18] introduced a deformation algorithm of BNL: Sort Filter Skyline (SFS). Its basic strategy is to sort data according to a specific type of monotonic function. It scans the sorted data while determining whether a point is dominated by other points. The Skyline algorithm has attracted the attention of many researchers and many algorithms, such as Bitmap [19], Nearest Neighborhood [20] and Branch and Bound [21], were proposed over the years. Over the last several decades, the development and increasing popularity of e-business and e-commerce, especially the pay-as-you-go business model with cloud computing, have driven the rapid growth of services [22]. One of the fundamental issues is the NP-complete quality-aware service selection problem. In recent years, many efforts have been dedicated to reducing the complexity of this problem [23-24]. It was first proposed by Alrifai *et al.* [23] to select skyline services that were not dominated by any other candidate services. Since then, many researchers have attempted to improve the skyline-based service recommendation approach to accommodate more sophisticated application scenarios [25-28]. Benouaret *et al.* [25] proposed a concept named alpha-dominant service skyline for addressing the problem of quality of service (QoS)-based Web service selection. Benouaret *et al.* [26] also proposed an improved Skyline-based approach for service recommendation that handles services' probabilistic quality values. However, current skyline service selection methods mainly select services according to QoS. This approach will select the best candidate service set by reducing the number of candidate services. However, the time complexity of the skyline computing process is very high, which means it cannot be directly applied to dynamic service selection. Especially under the scenario of mobile edge services, the quality of the edge services will change as the user location changes, and as the mobility of services increases, the computation time will increase. As a result, efficiently providing users with good mobile edge services becomes an urgent issue. To overcome this issue, this paper proposes a dynamic mobile edge service maintenance algorithm, which analyzes the skyline feature of mobile edge services.

III. PROBLEM FORMULATION

In this section, we define the key concepts of service maintenance in mobile edge computing.

Definition 1 (Edge Service (ES)): An edge service is modeled as a quadruple es=$\{Id, Fun, Des, QoES\}$, where

1) *Id* is the unique identifier of the edge service;
2) *Fun* is the functional information of the edge service;
3) *Des* is the basic descriptive information of the edge service, including the name, location, and provider;
4) *QoES* is a series of non-functional properties of the edge service.

Definition 2 (Quality of Edge Service (QoES)): QoES is a set of attributes $QoES = \{q_1, q_2, \ldots, q_n\}$ and each attribute q_i has a default value.

The *QoES* attributes of an edge service may include bandwidth, energy consumption, transmission delay, computational delay, and cost. Normally, bandwidth is the amount of data that can be transmitted in a fixed amount of time and energy consumption is the required energy for subscriber terminals to request services from the edge devices. As the user location changes, mobile devices may be connected with or disconnected from the edge cloud, which may cause significant power consumption of mobile devices for service connection. The time needed for data transmission between the mobile devices and the edge cloud is the transmission delay. The transmission delay can be affected by the network status, the distance between the mobile devices and the edge devices, and the data size; the computation delay is the time that is needed by the edge devices to complete a job; and cost is the price that users must pay for using the edge services.

With the fast development of the mobile Internet and IoT, mobile edge service providers will produce and deploy increasing amount of edge services, which may have similar functions but different non-functional properties. For example, suppose in an urban area there are 9 edge services and every edge service has an image feature processing function. However, due to the mobility of intelligent terminals, the *QoES* of the edge services will change dynamically. At a certain moment, the quality of edge services resemble the data in Table 1 (here, we will take Computation Delay and Energy Consumption as example quality attributes). The selection of better edge services for users under such a dynamic environment is a problem that needs to be solved.

Skyline query is the selection of a subset from the given object set *ES* of the *n*-dimensional space. No point of this subset can be dominated by other points in the set *ES*. The points that satisfy these requirements are called Skyline points. If es_i

TABLE 1. Attribute values of the edge services.

sid	Computation Delay	Energy Consumption
es_1	3	21
es_2	12	24
es_3	18	23
es_4	10	20
es_5	15	18
es_6	9	15
es_7	17	7
es_8	11	5
es_9	19	16

TABLE 2. Symbols and descriptions.

Notation	Description
ES	A set of edge services
es_i	The ith edge service in the set ES
$es_i.q_l$	The lth attribute of the ith edge service in the set ES
$es_i \prec es_j$	es_i dominates es_j
SES	The set of edge services that are not dominated by other edge services in the set ES
$DR(es_i)$	The set of edge services that es_i dominates
$AR(es_i)$	The set of the edge services that dominate es_i
$mae(DR(es_i))$	The maximum element of $DR(es_i)$
$mie(AR(es_i))$	The minimum element of $AR(es_i)$
$es.D$	$es.D$= $mae(DR(es))$
$es.AD$	$es.AD$= $mie(AR(es))$
$edge<es_i, es_j>$	A directed edge from es_i to es_j

dominates es_j, then the value of es_i is better than or equal to that of es_j in any arbitrary dimension and strictly better in at least one dimension. For example, as shown in Table 1, for set ES=$\{es_1, es_2, es_3\}$, suppose a smaller value in every dimension of the data is better. Then, es_1 dominates es_2 and es_3, but es_2 and es_3 do not dominate each other. The only point that is not dominated by other points in set ES is es_1, so es_1 is the only skyline point. Thus, the query result of Skyline is $\{es_1\}$. Table 2 lists some notations that are used in this paper.

Definition 3 (Edge Service Dominance): For edge services es_i and es_j, if for any non-functional attribute l of the edge services, $es_i.q_l \leq es_j.q_l$ and at least one non-functional attribute m exists such that $es_i.q_m < es_j.q_m$ (suppose that smaller values of a non-functional attribute of an edge service are better), then the edge service es_i dominates the edge service es_j.

Definition 4 (Skyline Edge Service (SES)): If edge service set $ES = \{es_1, es_2, \ldots, es_m\}$ is a set of candidate edge services that have similar functions and different non-functional attributes, then SES is the set of edge services that are not dominated by other edge services in the ES.

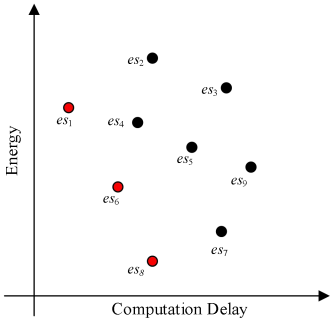


FIGURE 2. Example of skyline edge services.

According to Fig. 2, in Table 1, $SES = \{es_1, es_6, es_8\}$. Specifically, es_4 and es_5 do not dominate each other because the computation delay of es_4 is smaller than that of es_5 but its energy consumption is larger than that of es_5. However, neither es_4 nor es_5 belongs to SES because es_4 and es_5 are both dominated by es_6.

IV. SKYLINE GRAPH MODEL

In this section, we will introduce the Skyline Graph Model (SGM). SGM applies the theory of Directed Acyclic Graphs (DAGs) to the storage of the dominance relationships between the edge services. Using this model, we can compute SES easily and reduce the computation time to improve the efficiency.

Definition 5 (Dominance Region): In the n-dimensional edge service set $ES = \{es_1, es_2, \ldots, es_m\}$, the set that is composed of all the edge services that are dominated by es_i is the Dominance Region of es_i, which is denoted as $DR(es_i)$:

$$DR(es_i) = \{es_j | es_i \prec es_j\}$$

The Dominance Region of es_i includes all the edge services that are dominated by es_i in the candidate edge service set. Accordingly, the section that dominates es_i is called the Anti-Dominance Region of es_i, which include all the edge services that dominate es_i.

Definition 6 (Anti-Dominance Region): In the n-dimensional candidate edge service set $ES = \{es_1, es_2, \ldots, es_m\}$, all the edge services that dominate es_i compose the Anti-Dominance Region of es_i, which is denoted as $AR(es_i)$:

$$AR(es_i) = \{es_j | es_j \prec es_i\}$$

Taking the edge services in Table 1 as an example, in Fig. 3, only es_3 is dominated by es_5, so $DR(es_5) = \{es_3\}$. The same is true when es_5 is only dominated by es_6 and es_8. Thus, $AR(es_5) = \{es_6, es_8\}$. In the two-dimensional Cartesian coordinate system, if we draw one vertical line and one horizontal line that start from the same point, then the top-right corner and lower-left corner that are produced by the two lines and the coordinate axis are the point's Dominance Region and Anti-Dominance Region, respectively.

Property 1 (Transitivity of Dominance): If $es_i \prec es_j$ and $es_j \prec es_k$, then $es_i \prec es_k$.

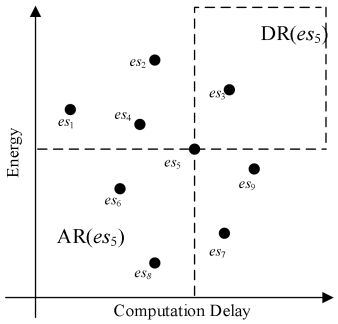


FIGURE 3. Example of *DR* (es_5) and *AR* (es_5).

Proof: For the n-dimensional candidate edge service set $ES = \{es_1, es_2, \ldots, es_m\}$, every *QoES* of the edge service can be written as $< q_1, q_2, \ldots q_n >$. If $es_i \prec es_j$ and $es_j \prec es_k$, then for all $l \in (1, n)$, $es_i.q_l \leq es_j.q_l$ and $es_j.q_l \leq es_k.q_l$ always hold. Therefore,$es_i.q_l \leq es_k.q_l$, that is to say, on every dimension, es_i is not less than es_k. In addition, $\exists m \in (1, n)$, $es_i.q_m \prec es_j.q_m \leq es_k.q_m$, so $es_i.q_m \prec es_k.q_m$. Moreover, in at least one dimension, the value of es_i is better than that of es_k, so $es_i \prec es_k$.

Deduction 1: If $es_i \prec es_j$, then $DR(es_j) \subseteq DR(es_i)$.

Proof: $\forall es_k \in DR(es_j)$, $es_j \prec es_k$. If $es_i \prec es_j$, then, because of the transitivity of dominance,$es_i \prec es_k$. Therefore, $es_k \in DR(es_i)$, which implies $DR(es_j) \subseteq DR(es_i)$.

Deduction 2: If $es_i \prec es_j$, then $AR(es_i) \subseteq AR(es_j)$.

Proof: Like Deduction 1, $\forall es_k \in AR(es_i)$, $es_k \prec es_i$. If $es_i \prec es_j$, then, because of the transitivity of dominance, $es_k \prec es_j$. Therefore, $es_k \in AR(es_j)$, which implies $AR(es_i) \subseteq AR(es_j)$.

Normally, an edge service can be dominated by many edge services and the dominating edge services may also dominate each other. Similar to the scenario in Fig. 3, suppose both es_4 and es_6 dominate es_3, and es_4 is dominated by es_6. Then, given $es_6 \prec es_4$ and $es_4 \prec es_3$, we know that $es_6 \prec es_3$. Thus, es_6 dominates es_4 directly and dominates es_3 indirectly. To further explain the direct dominance relationship and indirect dominance relationship, we define the maximum element and the minimum element as follows.

Definition 7 (Maximum Element): For the n-dimensional candidate edge service set $ES = \{es_1, es_2, \ldots, es_m\}$, $DR(es_i)$ is a subset of ES. Then, the maximum element of $DR(es_i)$ is defined as the element that satisfies the following conditions:

1) $mae(DR(es_i)) \in DR(es_i)$;
2) $\neg\exists es \in DR(es_i), es \prec mae(DR(es_i))$.

According to the definition of the maximum element, in Fig. 3, es_6 dominates es_4 directly. Therefore, es_4 can be considered the maximum element of $DR(es_6)$ because there is no other edge service that can dominate es_4 in $DR(es_6)$. In Fig. 3, the maximum elements of $DR(es_6)$ also include es_5 and es_9, which are both directly dominated by es_6.

Definition 8 (Minimum Element): For the n-dimensional candidate edge service set $ES = \{es_1, es_2, \ldots, es_m\}$, $AR(es_i)$ is a subset of ES. The minimum elements of $AR(es_i)$ are defined as members that satisfy the following conditions:

1) $mie(AR(es_i)) \in AR(es_i)$;
2) $\neg\exists es \in AR(es_i), mie(AR(es_i)) \prec es$.

According to the definitions that are presented above, in Fig. 3, $DR(es_6) = \{es_2, es_3, es_4, es_5, es_9\}$ and the set of maximum elements of $DR(es_6)$ is $mae(DR(es_6)) = \{es_4, es_5, es_9\}$; $AR(es_3) = \{es_1, es_4, es_5, es_6, es_7, es_8\}$ and the set of minimum elements of$AR(es_3)$ is $mie(AR(es_3)) = \{es_1, es_4, es_5, es_7\}$.

Now, we will define the Skyline Graph.

Definition 9 (Skyline Graph): The Skyline Graph is a triple $SG=<V(G), E(G), \prec>$, where $V(G)$ is a non-empty set of nodes. The node G is also a triad $G =< ID, AD, D >$, where ID is the identity of the node, AD is the set of the minimum elements of the anti-dominance region of the node, and D is the set of the maximum elements of the dominance region of the node. Suppose that in the Skyline Graph, there is a node es. Then, $es.AD$=$mie(AR(es))$ and $es.D = mae(DR(es)).E(G)$ is the set of edges. We denote the directed edge from es_i to es_j as edge<es_i,es_j>. $\prec$ is the dominance relationship between the edge set E and the nodes.

Based on the definitions and properties that are presented above, we will introduce the concrete construction of the skyline graph algorithm for the dynamic maintenance of edge services.

V. SKYLINE GRAPH ALGORITHM

In this section, we will introduce the dynamic selection and maintenance algorithm of the edge services, which is based on the Skyline Graph. Specifically, Subsection A introduces the construction algorithm of the Skyline Graph. Subsection B presents the addition algorithm of the edge services and Subsection C presents the deletion algorithm for when the original edge services become invalid. Subsection D explains the dynamic update algorithm of the skyline graph for when *QoES* changes. Subsection E introduces the *SES* search algorithm. Finally, Subsection F presents the complexity analysis of the algorithms.

A. BUILD SKYLINE GRAPH

For a given edge service set $ES = \{es_1, es_2, \ldots, es_m\}$, every edge service has an n-dimensional *QoES* value. For every edge service es_i, all the edge services that are dominated by es_i form set $DR(es_i)$. The maximum element of $DR(es_i)$ is $mae(DR(es_i))$. This element is added into $es_i.D$. Similarly, it is possible to obtain $AR(es_i)$ and $mie(AR(es_i))$. Finally, for all $es_i \in ES$, if $es_j \in es_i.D$, we build the edge<es_i, es_j>. Then, the skyline graph is complete.

Take Fig. 3 as an example. The constructed skyline graph is shown in Fig. 4. The detailed process is as follows: (1) Find all the edge services that are dominated by es_1 and compute $DR(es_1) = \{es_2, es_3\}$; (2) Compute the maximum element of $DR(es_1)$, which is $mae(DR(es_1)) = \{es_2, es_3\}$; (3) From es_1, draw two directed edges that point to es_2 and es_3; (4) Repeat these steps until all the edge services have been processed.

Algorithm 1 SG-Construction

Input: the set of edge services *ES*
Output: skyline graph (SG)

Begin
1: **for each** $es_i \in ES$ **do**
2: //compute $es_i.D$
3: find es where $es_i \prec es$
4: $DR(es_i) \leftarrow es$
5: $es_i.D \leftarrow mae(DR(es_i))$
6: //compute $es_i.AD$
7: find es where $es \prec es_i$
8: $AR(es_i) \leftarrow es$
9: $es_i.AD \leftarrow mie(AR(es_i))$
10: **for each** $es_j \in es_i.D$
11: new edge$<es_i, es_j>$
12: **end for**
13: **end for**
End

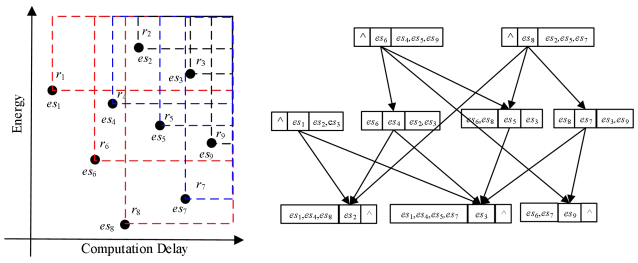


FIGURE 4. Construction of skyline graph.

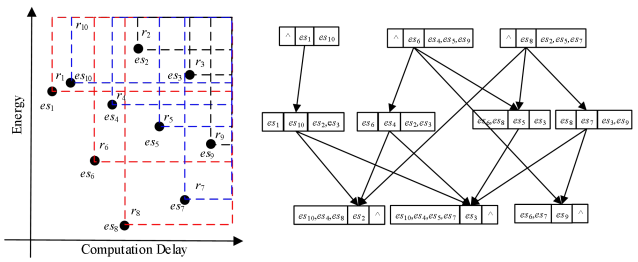


FIGURE 5. Example of inserting a node into the SG.

In Fig. 4, there is no directed edge from es_2 to any other node, as no service is dominated by es_2.

In the following subsections, we will introduce the dynamic insertion, deletion, update and search algorithms for the dynamic maintenance of edge services.

B. INSERTION

Suppose we are given *SG* of edge service set *ES* and a new edge service *es*. The insertion process is as follows: (1) Compute $DR(es)$ and $AR(es)$; (2) Obtain $es.D$ and $es.AD$; (3) For all $es_i \in es.D$, add edge$<es,es_i>$; (4) For all $es_j \in es.AD$, add edge$<es_j,es>$. The detailed process is described as Algorithm 2.

Algorithm 2 SG-Insertion

Input: *SG*, a new edge service *es*
Output: *SG*

Begin
1: //compute $es.D$
2: find sp where $es \prec sp$
3: $DR(es) \leftarrow sp$
4: $es.D \leftarrow mae(DR(es))$
5: //compute $es.AD$
6: find sd where $sd \prec es$
7: $AR(es) \leftarrow sd$
8: $es.AD \leftarrow mie(AR(es))$
9: **for each** $es_j \in es.AD$
10: new edge$<esj, es>$
11: **end for**
12: **for each** $es_i \in es.D$
13: new edge$<es, es_i>$
14: **end for**
End

Algorithm 3 SG-Deletion

Input: *SG*, an invalid edge service *es*
Output: *SG*

Begin
1://delete edges that are from *es* to the nodes in $es.D$
2: **for each** $es_i \in es.D$
3: delete edge$<es,es_i>$
4: **end for**
5://delete edges that are from the nodes in $es.AD$ to *es*
6: **for each** $es_j \in es.AD$
7: delete edge$<esj, es>$
8: **end for**
End

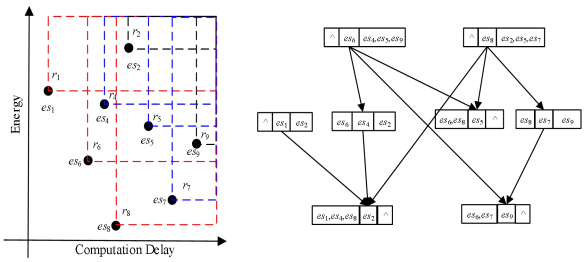


FIGURE 6. Example of deleting a node from the SG.

For example, suppose we need to add a new edge service es_{10} in Fig. 4 and the *QoES* of es_{10} is (5,22), which means that the computing delay is 5 and the energy consumption is 22. According to the original *SG*, es_{10} dominates es_2 and es_3, while es_{10} is dominated by es_1. Based on Algorithm 2, $DR(es_{10}) = \{es_2, es_3\}$ and $es_{10}.D = \{es_2, es_3\}$. Therefore, es_{10} needs to be connected to es_2 and es_3, that is, new directed edges that point to es_2 and es_3 from es_{10} are needed.

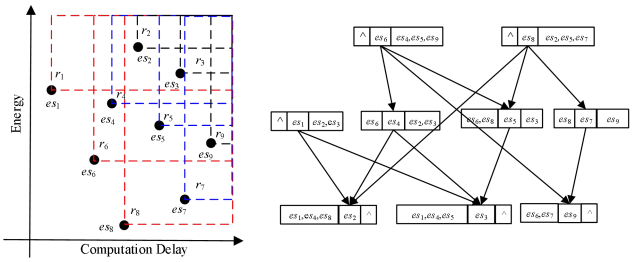


FIGURE 7. Example of updating a node in the SG.

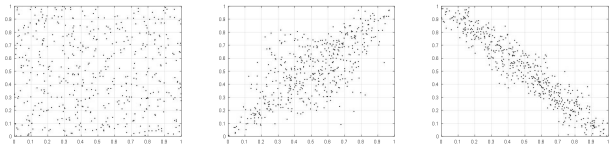


FIGURE 8. Two-dimensional example of three datasets. (a) Independent dataset. (b) Correlated dataset. (c) Anti-correlated dataset.

In addition, $AR(es_{10}) = \{es_1\}$ and $es_{10}.AD = \{es_1\}$. Thus, we need a new directed edge that points to es_{10} from es_1. After es_{10} is added, the result of SG is as shown in Fig. 5.

C. DELETION

Given *SG* of edge service set *ES* and an invalid edge service *es*, the deletion process is as follows: (1) For all $es_i \in es.D$, delete edge$<es,es_i>$; (2) For all $es_j \in es.AD$, delete edge$<esj, es>$. The detailed process is described in Algorithm 3.

Still taking Fig. 4 as an example, suppose edge service es_3 becomes invalid. The process of deleting es_3 is as follows: (1) $es_3.D = \emptyset$; (2) $es_3.AD = \{es_1, es_4, es_5, es_7\}$; (3) Delete edge$<es_1,es_3>$, edge$<es_4,es_3>$, edge$<es_5,es_3>$ and edge$<es_7,es_3>$. After deleting es_3, the new Skyline Graph is as shown in Fig. 6.

Algorithm 4 SG-Search

Input: *SG*
Output: *SES*

Begin
1: **for each** $es_i \in ES$
2: **if** $es_i.AD = \emptyset$
3: add es_i into *SES*
4: **end if**
5:**end for**
End

D. UPDATE

When the *QoES* of edge services changes, we need to change the *SG* as well. The update operation simply deletes one edge service and adds another edge service. Therefore, the update algorithm is omitted here.

TABLE 3. Experimental parameter settings.

Parameter	Experimental process	
	#A	#B
The number of candidate edge services (n)	2400	400 to 2400
The service quality dimensions (q)	1 to 9	6

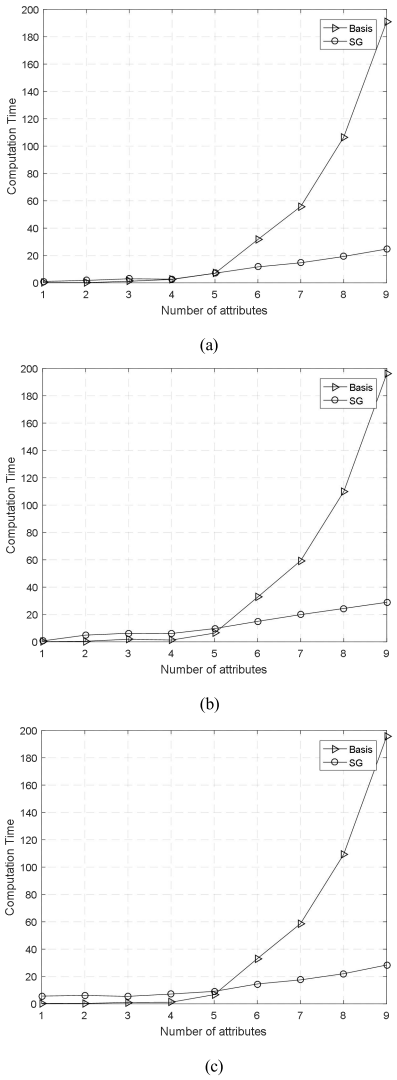


FIGURE 9. QWS dataset: performance vs. dimension (A). (a) Addition. (b) Deletion. (c) Update.

For example, suppose the *QoES* of edge service es_3 changes from (18,23) to (16,23). Then, the computation delay of es_3 decreases while the energy consumption remains the same. As a result, es_7 no longer dominates es_3. The process of updating the skyline graph is as follows: (1) Delete es_3 and all the interfacing edges. That is, delete edge$<es_1,es_3>$, edge$<es_4,es_3>$, edge$<es_5,es_3>$ and edge$<es_7,es_3>$; (2) Since now $es_3.D = \emptyset$ and $es_3.AD = \{es_1, es_4, es_5\}$, add edge$<es_1,es_3>$, edge$<es_4,es_3>$ and edge$<es_5,es_3>$. The updated skyline graph is shown in Fig. 7.

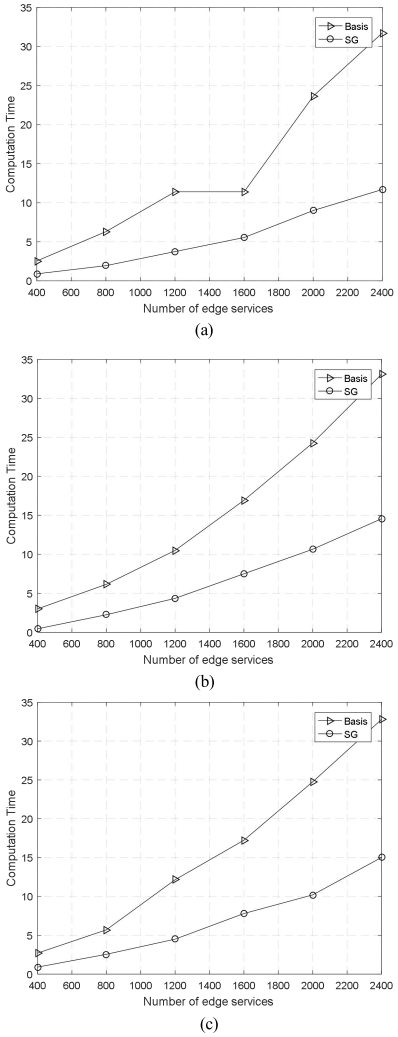


FIGURE 10. QWS dataset: performance vs. number of edge services (B). (a) Addition. (b) Deletion. (c) Update.

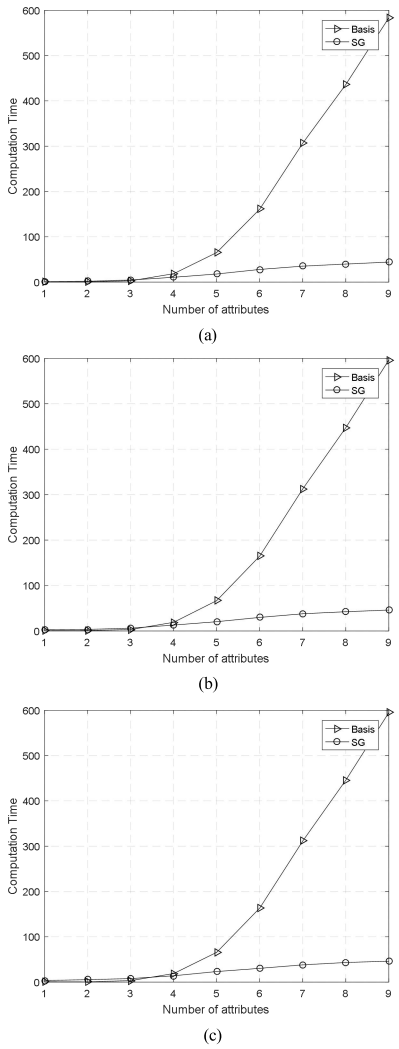


FIGURE 11. Anti-correlated dataset: performance vs. dimension (A). (a) Addition. (b) Deletion. (c) Update.

E. SEARCH

For a given *SG*, to search the *SES* is to solve the problem of finding the nodes whose in-degree is zero in *SG*. Based on the definition of the minimum element, if the *AD* set of a node is empty, then no other node dominates that node. Hence, the node is the *SES* that we need. For example, in Fig. 4, $es_1.AD = \emptyset$, $es_6.AD = \emptyset$, and $es_8.AD = \emptyset$. Therefore, $\{es_1, es_6, es_8\}$ is the *SES* that we need. The process is for *SG* of the given edge service set *ES*, and for every node *es* in *SG*, if $es.AD = \emptyset$, add *es* into *SES*. The search algorithm is given as Algorithm 4.

F. ANALYSIS OF THE COMPLEXITY OF THE ALGORITHM

Here, we analyze the time complexity of the SG algorithms for the construction, addition, deletion, update and search of SG.

(1) The SG-Construction algorithm for candidate edge service set $ES = \{es_1, es_2, \ldots, es_n\}$ is as follows: First, compute the Dominance Region of every edge service *es* in the edge service set. Then, obtain the maximum element of $DR(es)$, namely, $mae(DR(es))$. For each $mae(DR(es))$, add edge$<es, mae(DR(es))>$. Therefore, the time complexity of the SG-Construction algorithm is $O(n^2)$.

(2) For the SG-Insertion algorithm, first, we identify the minimum element of the anti-dominance region of the addition service, namely, $mie(AR(es))$. Then, for every $mie(AR(es))$, we add edge$<mie(AR(es)), es>$. Next, we find the maximum element of the dominance region of the *es*, namely, $mae(DR(es))$ and add edge$<es, mae(DR(es))>$ into *SG*. Therefore, the time complexity of the SG-Insertion algorithm is $O(n)$.

(3) The SG-Deletion algorithm is different from the SG-Insertion algorithm. We do not have to compute the maximum element of the dominance region of the invalid service *es* or the minimum element of the anti-dominance region of *es* because all the information is stored as node information of the *SG*. All we need to do is to delete. Therefore, the time complexity of the SG-Deletion algorithm is $O(1)$.

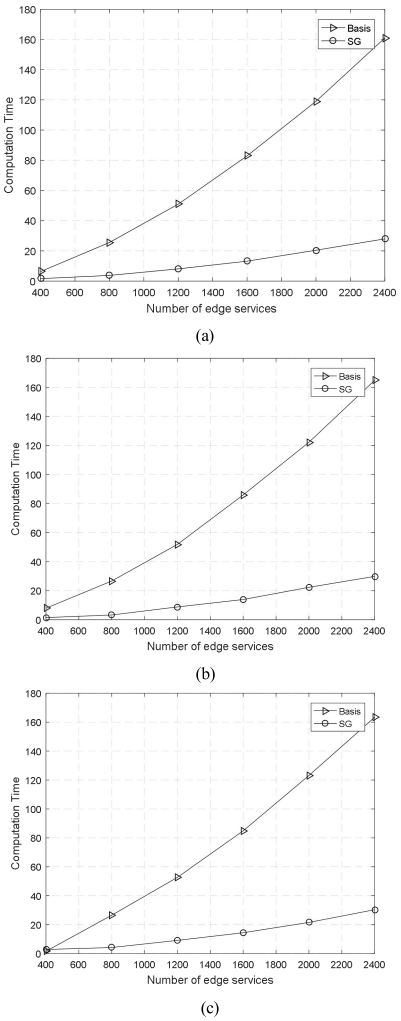


FIGURE 12. Anti-correlated dataset: performance vs. number of edge services (B). (a) Addition. (b) Deletion. (c) Update.

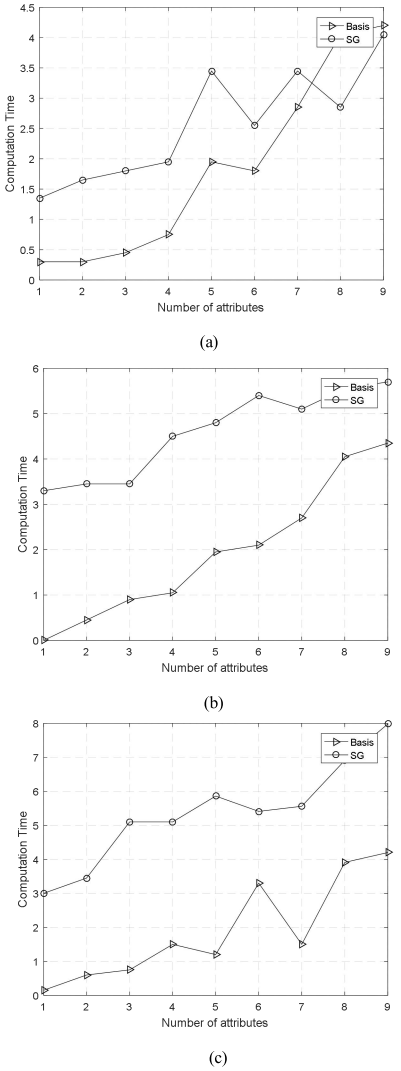


FIGURE 13. Correlated dataset: performance vs. dimension (A). (a) Addition. (b) Deletion. (c) Update.

(4) The SG-Update algorithm combines the SG-Deletion and SG-Insertion algorithms. Therefore, the time complexity of the SG-Update algorithm is $O(1) + O(n) = O(n)$.

(5) The SG-Search algorithm needs to traverse all the nodes in *SG*. If the in-degree of a node is 0, it is added into the *SES* set. Therefore, the time complexity is $O(n)$.

VI. EVALUATION

A. EXPERIMENTAL SETTINGS

In this section, we use both public web service quality dataset QWS [29] and a simulated data set to evaluate the performance of the SG algorithms. The QWS dataset includes 2507 instances of real *QoS* information of web services. The service quality of QWS includes 9-dimensional attributes. To further evaluate the effectiveness of the algorithms, we simulate three different types of datasets: (1) an independent data set, where all the data are independently distributed; (2) a correlated dataset, where the data obey a positive-correlation distribution, such that one attribute value grows as another attribute value grows; and (3) an anti-correlated dataset, where the data obey a related distribution, according to which an attribute value will grow as another attribute value declines. Fig. 8(a), Fig. 8(b) and Fig. 8(C) show the independent distribution, correlated distribution and anti-correlated distribution of two-dimensional data, respectively.

The correlated dataset is considered the easiest case for skyline computation since a single point that is close to the origin can quickly be used to prune a small portion of the data from consideration. The anti-correlated dataset is considered the most challenging of the three for skyline computation. This is because points in the skyline dominate only a small portion of the entire dataset. More skyline points exist for anti-correlated data of a given cardinality compared with the independent and correlated cases.

B. RESULTS AND ANALYSIS

Here, we analyze how the number of edge services and the dimensions of *QoES* affect the performance of

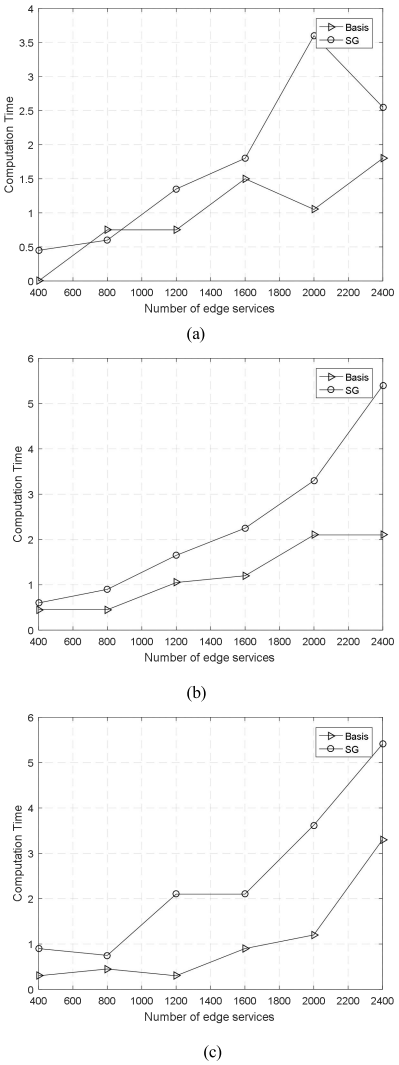


FIGURE 14. Correlated dataset: performance vs. number of edge services (B). (a) Addition. (b) Deletion. (c) Update.

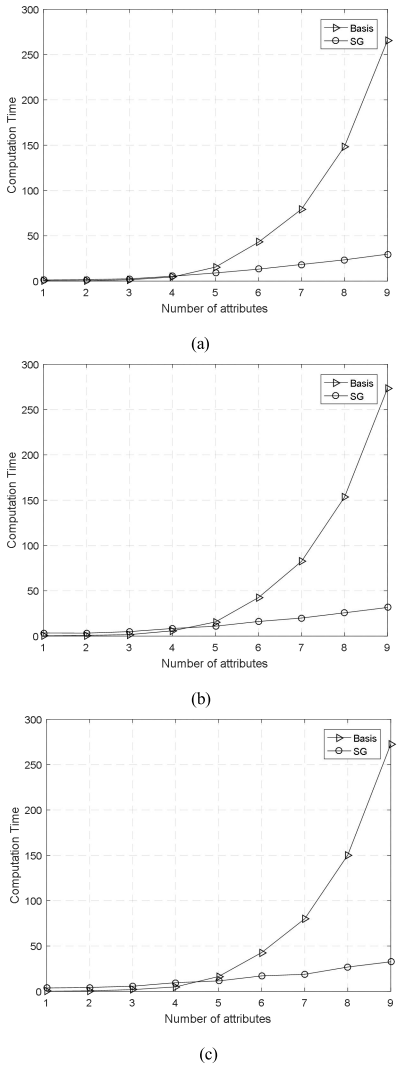


FIGURE 15. Independent dataset: performance vs. dimension (A). (a) Addition. (b) Deletion. (c) Update.

SG algorithms. In the experiments, under different parameter settings, we have two experiment processes: A and B. Table 3 shows the parameter settings for the two experiments.

We compare the SG algorithms that were proposed in this paper to the classic Skyline algorithm: Block Nested Loop (BNL). The BNL algorithm is often used as the baseline algorithm in skyline service selection.

Experiment 1 (QWS Data Set): The process of comparing the algorithm is as follows: (1) Add (delete or change) one edge service at a time; (2) Use SG algorithms and BNL to update *SES*; (3) Repeat (1) and (2) 100 times; (4) Compute the average time. The results of addition, deletion and update for experiment A are shown in Fig. 9(a), (b) and (c), respectively. According to Fig. 9, the computation times of the BNL algorithm and the SG algorithms grow as the dimension of the *QoES* grows. However, the growth for the BNL algorithm is exponential, while the growth for the SG algorithms shows a linear trend. When the number of service quality dimensions is smaller than 5, BNL slightly outperforms the SG algorithms. However, when it is larger than 5, the computation time of the SG algorithms is much shorter than that of the BNL algorithm. The results of experiment B are shown in Fig. 10. According to Fig. 10, the computation time of SG algorithm is better than that of the BNL algorithm in all cases.

Experiment 2 (Simulated Dataset): Fig. 11 and Fig. 12 present the comparison results between the SG algorithms and the BNL algorithm on the anti-correlated dataset. Compared to the result on the QWS dataset, the advantage of the SG algorithm is much more obvious under anti-correlated dataset. In the process of experiment A, in the 4th dimension, the performance gap between the SG algorithm and the BNL algorithm grows. In addition, when number of dimensions of service quality is 9, the computation time of the BNL algorithm is nearly 600 ms, while SG algorithm only needs approximately 46 ms. In the process of experiment B, the advantage in terms of computation time of the SG

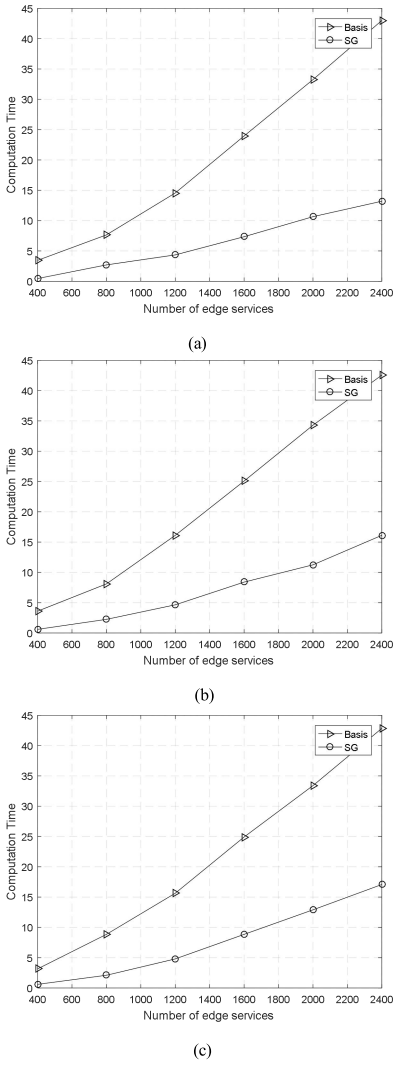


FIGURE 16. Independent dataset: performance vs. number of edge services (B). (a) Addition. (b) Deletion. (c) Update.

algorithm is still obvious. When the number of candidate edge services is 2400, the computation time of the SG algorithms is approximately 30 ms, while that of the BNL algorithm is nearly 160 ms. Compared to the QWS dataset, under the anti-correlated dataset, the computation time of the BNL algorithm fluctuates dramatically. When the number of candidate edge services is 2400 and the number of service quality dimensions is 6, the computation time grows from 30 ms to 160 ms. Therefore, the performance of the BNL algorithm is affected by the dataset significantly, while that of the SG algorithms is relatively stable. Clearly, our SG algorithm has better robustness.

The experimental results under the correlated dataset are shown in Fig. 13 and Fig. 14. According to Fig. 8 (b), the size of *SES* is smaller compared to the anti-correlated dataset, so the computation times of both algorithms are short. Although the BNL algorithm outperforms the SG algorithm under this dataset, the gap is within a few milliseconds, which is very small, and the worst computation time of the SG algorithms is under 8 ms.

We further confirm the advantage of the SG algorithms under the independent dataset. The experimental results are shown in Fig. 15 and Fig. 16. The results under the independent data set are similar to the results under the anti-correlated dataset and the QWS dataset. The curves also show a similar growing trend.

In summary, all the experiments that are presented above show that on different datasets, the SG algorithms that are proposed in this paper achieve better overall performance and robustness.

VII. CONCLUSIONS

With the development of the mobile Internet and the Internet of Things, the ecosystem of mobile edge computing services is becoming larger and more complicated. The mobile edge computing service providers will keep releasing new types of edge services and the number of edge services will grow dramatically in the near future. Therefore, recommending the best services for users according to their needs will become a more important yet more challenging issue. In the meantime, due to the location awareness, mobility and proximity of the mobile edge services, achieving efficient maintenance of dynamic mobile edge services is a difficult problem. To address this issue, this paper applied the skyline graph model, which is based on the theory of directed acyclic graphs, and designed the dynamic skyline graph algorithms for addition, deletion, update and search of dynamic edge services. The results of the experiments demonstrated that, in general, our proposed algorithms achieve better performance and robustness than the baseline algorithm (Block Nested Loop).

In the future, we plan to investigate how to ensure and improve the efficiency of skyline graph algorithms in large-scale application scenarios with the support of high-performance computing platforms such as Spark.

REFERENCES

[1] B. G. Chun, S. Ihm, P. Maniatis, P. Maniatis, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–314.

[2] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. INFOCOM*, Mar. 2012, pp. 945–953.

[3] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[4] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

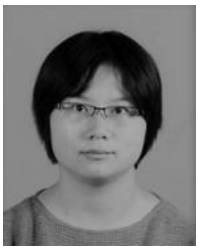
[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[6] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Generat. Comput. Syst.*, vol. 70, pp. 59–63, May 2017.

[7] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.

[8] H. Xiang *et al.*, "An adaptive cloudlet placement method for mobile applications over GPS big data," in *Proc. Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[9] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 49–62.
[10] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.
[11] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
[12] K. Zhang *et al.*, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5890–5896, 2016.
[13] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.
[14] B. Varghese, N. Wang, J. Li, Dimitrios, and S. Nikolopoulos. (Oct. 2017). "Edge-as-a-service: Towards distributed cloud architectures." [Online]. Available: https://arxiv.org/abs/1710.10090
[15] X. Xu, S. Huang, L. Feagan, Y. Chen, Y. Qiu, and Y. Wang, "EAaaS: Edge analytics as a service," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 349–356.
[16] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, "Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 48–56, Jun. 2017.
[17] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, 2001, pp. 421–430.
[18] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting: Theory and optimizations," in *Intelligent Information Processing and Web Mining*. Berlin, Germany: Springer, 2005, pp. 595–604.
[19] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *Proc. VLDB*, vol. 1. 2001, pp. 301–310.
[20] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *Proc. 28th Int. Conf. Very Large Data Bases. VLDB Endowment*, 2002, pp. 275–286.
[21] D. Papadias, Yufei Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2003, pp. 467–478.
[22] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2008.
[23] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for QoS-based Web service composition," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, Raleigh, NC, USA, 2010, pp. 11–20
[24] Q. Yu and A. Bouguettaya, "Efficient service skyline computation for composite service selection," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 776–789, Apr. 2013.
[25] K. Benouaret, D. Benslimane, and A. Hadjali, "On the use of fuzzy dominance for computing service skyline based on QoS," in *Proc. 9th IEEE Int. Conf. Web Services*, Jul. 2011, pp. 540–547.
[26] K. Benouaret, D. Benslimane, and A. Hadjali, "Selecting skyline Web services from uncertain QoS," in *Proc. 9th IEEE Int. Conf. Services Comput.*, Jun. 2012, pp. 523–530.
[27] F. Zhang, K. Hwang, S. U. Khan, and Q. M. Malluhi, "Skyline discovery and composition of multi-cloud Mashup services," *IEEE Trans. Services Comput.*, vol. 9, no. 1, pp. 72–83, Jan. 2016.
[28] S. Zhang, W. Dou, and J. Chen, "Selecting top-k composite Web services using preference-aware dominance relationship," in *Proc. 20th IEEE Int. Conf. Web Services*, Jun./Jul. 2013, pp. 75–82.
[29] E. Al-Masri and Q. H. Mahmoud, "Investigating Web services on the World Wide Web," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 795–804.

YIWEN ZHANG received the Ph.D. degree in management science and engineering from the Hefei University of Technology in 2013. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University. His research interests include service computing, cloud computing, and e-commerce.



JIN LI received the bachelor's degree in network engineering from the School of Computer Science and Technology, Anhui University, in 2017, where she is currently pursuing the master's degree. Her current research interests include service computing, cloud computing, and mobile edge computing.

ZHANGBING ZHOU is currently a Full Professor with the School of Information Engineering, China University of Geosciences, Beijing, and also with the Computer Science Department, TELECOM SudParis, France. His research interests include wireless sensor networks, service computing, business process management, cloud computing, and SOA.

XIAO LIU received the Ph.D. degree in computer science and software engineering from the Faculty of Information and Communication Technologies, Swinburne University of Technology, in 2011. He is currently a Senior Lecturer with the School of Information Technology, Deakin University, Melbourne, VIC, Australia. His research interests include workflow systems, cloud computing, and social network.