Entropy-based Service Selection with Uncertain QoS for Mobile Cloud Computing

Yue Wang
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
wangyue2714@gmail.com

Zibin Zheng
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
zibinzheng2@yeah.net

Michael R. Lyu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Hong Kong
lyu@cse.cuhk.edu.hk

***Abstract*—With the prevalence of mobile computing and its convergence with cloud computing, there is an increasing trend of composing existing cloud services for rapid development of cloud-based mobile applications. It is vital for developers to find services not only satisfying their functionality requirements, but also meeting the requirements on non-functional quality of services (QoS). These QoS requirements, such as throughput, delay, reliability and security, are critical for the success of cloud-based mobile applications. In this paper, a QoS-based service ranking and selection approach is proposed to help developers select the service that best satisfies developers' QoS requirements from a set of services having already satisfied developers' functionality requirements in mobile cloud computing. Compared with state-of-the-art service ranking and selection techniques, our approach has the following advantages: 1) it uses intervals instead of fixed values to represent QoS of services, which are more flexible and practical in mobile cloud computing; 2) it enables developers to specify their QoS requirements in a more simple way; and 3) it employs the hybrid weights that incorporate the Entropy-based weighting technique to overcome the weakness caused by subjective weights, which ignore the knowledge of different services' performance in different QoS aspects. Experiments validate the effectiveness of the proposed method.**

Keywords-mobile cloud computing; service ranking; service selection; QoS requirements; entropy weighting

I. INTRODUCTION

Cloud computing has recently emerged as a new paradigm for hosting and delivering services over the Internet. The core idea of cloud computing is to provide computing resources, services, and applications as an integrated utility, which can be employed by users on demand. The provisioning of cloud services occurs at the Infrastructural level (IaaS), the Platform level (PaaS), or the Software level (SaaS). The vast number of services provided in the cloud has become a commodity in people's day-to-day life. For instance, Google Docs are extensively used by millions of people around the world for document sharing, while services such as Google Maps and Bingo Maps enable the provisioning of location-based services. However, a cloud service usually does not provide a rich functionality by its own. It becomes increasingly popular to combine multiple services for developing more complex and richer applications (usually referred to as mashup applications). For example, by integrating Google Maps and SoundCloud (a media file sharing service), it is possible to visualize the songs and tracks uploaded by a user in the geographical location where the media files were recorded or uploaded [1].

In the meanwhile, the mobile computing domain also has advanced rapidly, which enabled the new generation of cloud-based and context-aware mobile applications. Consequently, clouds are looking forward to the mobile domain, having their expectations focused on the idea of fostering the access and consumption of cloud services at the different levels of mobile devices. Nowadays, mobile devices are equipped with embedded sensors and input devices such as cameras, GPS signal receivers, accelerometers, and magnetic sensors, among others. Moreover, these new capabilities can be combined with other services and mashup applications, giving place to mobile mashup applications. A mobile mashup application not only blends into a single application several services, but also uses the data gathered by embedded sensors and devices in order to enrich the mobile application. For example, foursquare uses the GPS sensors embedded in the device to determine the user's location and to provide information about the nearby services. The combination of mobile computing and cloud computing has led to the Mobile Cloud Computing (MCC) domain [2].

Applications from such Mobile Cloud Computing domain usually combine cloud-based services with basic functionalities. However, with the massive amount of services present in the cloud, and with the increasing number

978-1-5090-0089-0/15 $31.00 © 2015 IEEE
DOI 10.1109/CIC.2015.28

of functionally equivalent services, to select high performance services for building dependable cloud-based mobile applications becomes an urgently-required research problem. The performance of cloud services is usually described by Quality-of-Service (QoS), which refers to a set of non-functional properties including throughput, delay, reliability, security, price, and so on. QoS is an important research topic in cloud computing. When making optimal cloud service selection from a set of functionally equivalent services, QoS values of cloud services provide valuable information to assist decision making. There are some existing investigations on QoS-based service selection for cloud computing [3][4]. However, how to accurately match cloud services' QoS with service users' QoS requirements in mobile cloud computing is still not well addressed. While users have requirements in multiple concerned QoS aspects, it is difficult to find the best service by comparing multiple concerned QoS aspects simultaneously. For example, it is not clear how to compare a service with high throughput and high price to another service with low throughput and low price. Besides, a service's QoS values in various QoS aspects cannot be simply combined because different users' preferences in the concerned QoS aspects may be different, and various QoS aspects have different scales and value ranges. Furthermore, it is hard for an inexperienced user to assign accurate subjective weights to various QoS aspects, and thus some problems will occur if we apply these weights strictly without any adjustment.

In this paper, we will present a service ranking and selection approach to meet the above challenge. This approach firstly normalizes a service's QoS in various aspects to a unique range, then measures how well the service's QoS satisfies users' QoS requirements in each concerned QoS aspect with a satisfaction score, and finally combines the service's satisfaction scores in all concerned aspects together as an overall satisfaction score. Our approach has the following advantages: 1) it uses intervals instead of fixed values to represent QoS of services, which are more flexible and practical in mobile cloud computing; 2) it enables developers to specify their QoS requirements in a more simple way; and 3) it employs the hybrid weights that incorporate the Entropy-based weighting technique to overcome the weakness caused by subjective weights, which ignore the knowledge of different services' performance in different QoS aspects. Experiments validate the effectiveness of the proposed method.

II. RELATED WORK

Service discovery and selection with QoS have recently attracted extensive interests from the researchers in the field of services computing and cloud computing. Service selection is heavily based on ranking of services according to their QoS values. Since there are multiple QoS factors and users' requirements on QoS are quite different, it is not easy to find the right services with an optimal QoS value for the users.

Many previous tasks employ weighted aggregate of QoS factors to rank services. One of such work was proposed by Masri and Mahmoud [5], which first normalized the values of different QoS factors into a range, and then computed the overall quality of services by adding the normalized QoS together with their weights. It should be noted that each weight represented the user's preference on one concerned QoS factor. Based on the overall quality, services were ranked and the top-ranked services were recommended for selection. Similarly, Comuzziand Pernici [6] used a price model to combine multiple QoS factors. The price model converted each QoS factor of a service to a price, and added all prices together. The services were then ranked based on their total prices. To allow the specification of elastic QoS requirements using linguistic terms or fuzzy propositions, several service selection methods based on fuzzy sets were proposed [7][8][9]. Different from previous approaches, Liu, Fletcher, and Tang [9] allowed users to specify personalized tradeoffs among QoS factors in QoS requirements and used several fuzzy operators to aggregate users' satisfaction degrees in individual QoS aspects. Yau and Yin [10] proposed a service ranking and selection method which could support more flexible QoS requirement specifications. The method selected the service that best satisfied users QoS requirements instead of the service with the best QoS which may be overqualified for the users' QoS requirements.

However, most existing service ranking and selection methods assume that the QoS value of a service in every QoS aspect is unique and fixed in the time of selection. In practice, depending on the network conditions and other influencing factors, QoS values of a service, such as response time, throughput, and reliability, are likely to have different values at different times and for different users [11]. Therefore, it is not appropriate to use a fixed and unique value to represent the QoS of a service in every aspect. Only a few investigations have taken the uncertainty of QoS into consideration in service selection [12][13]. However, they mostly focused on evaluating the uncertainty of service QoS and aimed at selecting the most reliable service. How to measure users' satisfaction degrees in various QoS aspects of a service with uncertain QoS, unfortunately, is not considered.

Recently, interests in service provision, discovery and selection in mobile environments have been increased [14][15][16]. In [17], the authors introduced MCC to integrate the cloud computing into the mobile environment, which overcame the obstacles related to performance, environment, and security. In [18][19], the QoS assurance problem, that is, how a service provider can ensure QoS of its cloud services, especially for mobile users in MCC, is addressed. Elgazzar et al. [20] proposed a novel discovery framework that addressed various aspects of mobile Web service discovery in resource-constrained environments. Shi and Gu [21] proposed a framework for mobile cloud computing service selection, and they engaged a Markov chain model to evaluate the service selection process. However, these tasks seldom addressed the mobile service selection problem with uncertain QoS, which is urgent in mobile environments.

III. OVERVIEW OF OUR METHOD

Fig. 1 depicts our service ranking and selection method. To monitor a service's QoS and to facilitate the service ranking and selection, we assume that all users' concerned services' QoS aspects are monitored and reported to a service selection engine, which saves the services' QoS information along with their functionality information. This QoS information will be used by the service selection engine to rank services and to help users select services when there are more than one registered services satisfying the users' functionality requirements. Our service ranking and selection method can be described in the following four steps:

A. Filtering

A user submits a service request in which the user's requirements on both service functionalities and QoS are specified. At first, the functional requirements will be processed by the service selection engine. The service selection engine matches the user's functional requirements with all services' functionalities and a list of acceptable services will be identified, whose QoS will be further evaluated based on the user' QoS requirements in the following steps.

B. Normalizing

After a list of services meeting the user's functional requirements are identified, our method will measure the user's satisfaction degrees on these services based on their QoS values and the user's QoS requirements. This is done by firstly normalizing the services' QoS values in various QoS aspects into a unique range in order to facilitate aggregating different QoS aspects. We normalize the services' QoS values in a QoS aspect according to the user's QoS requirements in the concerned QoS aspect, instead of the best-case and worst-case QoS values of the QoS aspect.

C. Computing

The user's satisfaction degrees in every individual QoS aspect for each service will then be computed. Since QoS values of the services are likely to be intervals, computing the user's satisfaction degree in an individual QoS aspect is more complex than using fixed QoS values. Since wider QoS intervals have high uncertainty and vice versa, the user will likely prefer service with narrow QoS intervals. Base on this consideration, we incorporate the width of QoS intervals of the services into a satisfaction degree function designed for computing the user's satisfaction degrees in individual QoS aspects for the services.

D. Aggregating

The satisfaction degrees of individual QoS requirements for each service will be aggregated to produce the overall satisfaction degree. To do this, a hybrid weighting technique which combines subjective weights and objective weights is employed. The subjective weights in every concerned QoS aspect are specified by the user. If the user does not specify a subjective weight, the medium value (0.5) will be applied as the default subjective weight. However, simply employing the subjective weight ignores the knowledge of all the QoS aspects of different services. Therefore, we introduce an entropy-based objective weighting method, combined with the subjective weights, to achieve a more effective and reasonable ranking outcome. The overall satisfaction degree function is implemented by using an additive weighting operator on the hybrid weights.

Finally, the candidate services are ranked in order of decreasing overall satisfaction degrees, which are returned to the user for selection. In the following sections, we will describe our method in detail. We focus on how to match the services' QoS with the user's QoS requirements, assuming that a list of service with acceptable functionalities have already been identified by using existing service functionality matching techniques.

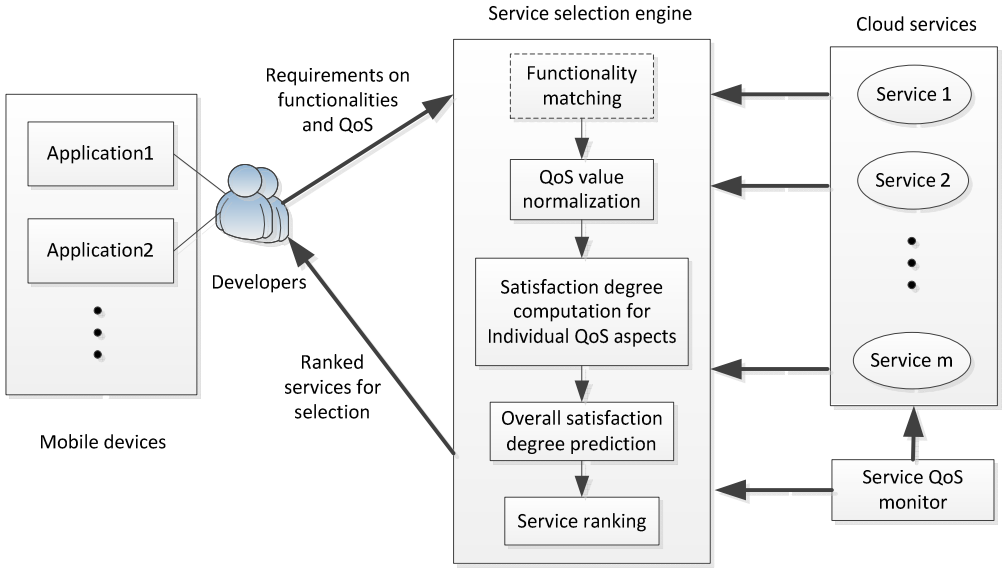

Figure 1. Overview of our QoS-based service ranking and selection method

IV. MODELING QOS AND QOS REQUIREMENTS

Suppose there are n QoS aspects of services, which can be represented by a vector $A = (A_1, A_2, \ldots, A_n)$. The set of QoS aspects can be divided into two subsets: positive and negative QoS aspects. The values of positive aspects need to be maximized (e.g. throughput and security), whereas the values of negative aspects need to be minimized (e.g. price and delay).

Let $S = (S_1, S_2, \ldots, S_m)$ represent a set of m services with similar functionalities such that they all meet the active user's functional requirements. Let $q_i = (q_{i1}, q_{i2}, ..., q_{in})$ denote the vector of QoS values of a service S_i, where q_{ij} represents the service's QoS value on A_j. QoS values of services are usually different when invoked by different users or at different times. This uncertainty of service QoS is even severe in mobile cloud environments because mobile applications invoke services via wireless communications. Thereby, it is not practical to use certain values to represent QoS of services. Instead, we define q_{ij} as an interval, i.e. $q_{ij} = [q_{ij}^l, q_{ij}^r]$, where q_{ij}^l and q_{ij}^r are two endpoints of q_{ij} and satisfy $q_{ij}^l < q_{ij}^r$.

The user can specify his/her QoS requirements in a QoS aspect A_j as a tuple $req_j = (l_j, u_j, w_j)$, where l_j and u_j are the lower bound and upper bound of the user's acceptable QoS on A_j, $w_j \in [0,1]$ is the weighting factor of the requirement, representing the user's preference on A_j in the service selection.

The values of l_j and u_j of a different QoS aspect A_j may have different meanings and be represented with different units. For example, the QoS aspects of delay and availability are represented as seconds and percentage respectively. The services are perfect if their delays are smaller than l_j, and unacceptable if their delays are larger than u_j. However, the services are unacceptable if their availabilities are smaller than l_j, and perfect if their availabilities are larger than u_j. If the user has specified no lower bound or upper bound for the acceptable QoS or does not know how to specify it, the user can leave l_j and u_j blank in req_j and we can set the values of l_j and/or u_j as follows:

First, if the metric for A_j has a lower bound and an upper bound, then l_j and u_j can be set as the metric's lower bound and upper bound for A_j respectively. For example, the metric of a service's availability is measured as the percentage of time that the service is available, which has a lower bound 0% and an upper bound 100%. Hence, for the QoS aspect of availability, we can set l_j = 0% and u_j = 100% by default if they are blank in req_j. Second, if the A_j's metric has no lower bound and/or upper bound, such as the service's price which has a lower bound 0, but no upper bound, then the values of l_j and u_j are defined as the smallest and largest QoS of all available services in S, respectively.

When the user does not know how to specify his/her preferences in QoS aspects with weight values, the user can specify his/her preferences through linguistic terms, such as very unimportant, unimportant, medium, important, and very important. These linguistic terms can be easily mapped to weight values, as shown in Tabel 1. If the user does not specify their importance weights in req_j, 0.5 as default weights will be employed.

TABLE I. MAP LINGUISTIC VALUES TO WEIGHT VALUES

Linguistic Values	Table Column Head
Very importance	*1*
Importance	*0.75*
Medium	*0.50*
unimportance	*0.25*
Very unimportance	*0*

Compared with the method proposed by Yau and Yin [10], who introduced a confidence rate r_j in the tuple $req_j = (l_j, u_j, w_j, r_j)$ to specify the user's confidence on the assignments of his/her requirements. A larger r_j indicates that the user has higher confidence in the specification of all the specified values of l_j, u_j and w_j, and a smaller r_j means lower confidence. However, this strategy will cause some potential problems. First, one inexperienced user may find it hard to specify the confidence rate since he/she may be unfamiliar with the QoS aspect. Besides, Yau and Yin used r_j to extend the interval $[l_j, u_j]$ to $[l_j, r_j, u_j/r_j]$, and this technique is likely to violate the user's subjective opinion subtly, which leads to meaningless results in some cases. For example, if a user is pretty sure about the lower bound but has no confidence in the upper bound, an r_j will change the lower bound that he/she initially has full confidence to set.

V. INDIVIDUAL QOS EVALUATION

This section discusses how to evaluate a service's individual QoS based on the user's requirements in a concerned QoS aspect. This evaluation process returns a satisfaction degree which represents how much the user is satisfied with the service's QoS in a concerned QoS aspect. The first step is to normalize QoS values into a unique range. The range is set as [0,1] in this paper.

Depending on whether the concerned QoS aspect A_j is a positive one or not, the normalization function would differ. For instance, when A_j is a positive QoS aspect that the user wants to maximize, such as reliability, then the QoS value on A_j (e.g. x) will be normalized using (1):

$$Norm_1(x) = \begin{cases} 0, & \text{if } x < l_j \\ (x - l_j)/(u_j - l_j), & \text{if } l_j \le x \le u_j \\ 1, & \text{if } x > u_j \end{cases} \quad (1)$$

Otherwise, when A_j is a negative QoS aspect that the user wants to minimize, such as price, x will be normalized using (2). It is obvious that the normalized QoS produced by both (1) and (2) will be in the range [0,1], where larger values indicate better QoS. In particular, 0 represents the lowest QoS, and 1 represents the best QoS. This normalization

function is distinguished as it normalizes a QoS value based on how well it satisfies the user's QoS requirement, instead of how good it is. In other words, the normalized QoS actually can reflect the satisfaction degree of the user on q_{ij}.

$$Norm_2(x) = \begin{cases} 0, & \text{if } x > u_j \\ (u_j - x)/(u_j - l_j), & \text{if } l_j \le x \le u_j \\ 1, & \text{if } x < l_j \end{cases} \quad (2)$$

In this paper, we use a real interval instead of a fixed value, to represent the QoS of a concerned QoS aspect. The normalization and satisfaction degree computation of the QoS will be a little more complex. Suppose that the concerned QoS aspect A_j of service i has QoS $q_{ij} = [q_{ij}^l, q_{ij}^r]$, by applying (1) and (2) to its endpoints, we get its normalized form $q_{ij}^{'} = [n_{ij}^l, n_{ij}^r]$. However, it is unsuitable for using $q_{ij}^{'} = [n_{ij}^l, n_{ij}^r]$ to represent the satisfaction degree of the user on q_{ij}. A single value is more preferred in doing this. There are two straightforward methods in addressing this issue:

- Take the average value of $q_{ij} = [q_{ij}^l, q_{ij}^r]$ and normalize it using (1) and (2), then use the result to represent the user's satisfaction degree.
- Take the average value of $q_{ij}^{'} = [n_{ij}^l, n_{ij}^r]$ to represent the user's satisfaction degree.

However, the above methods will both introduce another confusing issue. For example, suppose the user's QoS requirements on delay is $req = (50ms, 100ms, 1)$, and there are two services, one of which has delay $q_1 = [50ms, 100ms]$ and the other has delay $q_2 = [10ms, 140ms]$. Both of the above methods will produce the same result, i.e. 0.5, for the user on the two services' delay performance. However, based on our intuition, the first service obviously have better delay performance than the second service, since its QoS on delay is perfectly matched with the user's QoS requirements on delay. This example indicates that the QoS interval width of a service, i.e. the certainty of QoS, will impose significant influence on the user's satisfaction degree. Therefore, we incorporate the service's QoS interval width in predicting the user's satisfaction degree. This is done based on adjusting the average value of $q_{ij}^{'} = [n_{ij}^l, n_{ij}^r]$ via introducing an adjustment factor φ_{ij}, which depends on how much proportion of the service's QoS interval satisfies the user's QoS requirements. The adjustment factor φ_{ij} is designed as follows.

When $q_{ij} = [q_{ij}^l, q_{ij}^r]$ is completely worse than the user's QoS requirements, φ_{ij} will be set as 0. Otherwise, when $q_{ij} = [q_{ij}^l, q_{ij}^r]$ is completely better than the user's QoS requirements, φ_{ij} will be set as 1. In the other cases, φ_{ij} will be the proportion of the service's QoS interval on A_j that satisfies the user's QoS requirements on A_j. Depending on whether A_j is a positive QoS aspect or not, the formula for calculating φ_{ij} will be different. When A_j is a positive QoS aspect, φ_{ij} is calculated with (3); otherwise, φ_{ij} is calculated with (4).

$$\varphi_{ij} = \begin{cases} 0 & q_{ij}^r < l_j \\ 1 - (l_j - q_{ij}^l)/(q_{ij}^r - q_{ij}^l) & q_{ij}^l \le l_j \le q_{ij}^r \\ 1 & q_{ij}^l > l_j \end{cases} \quad (3)$$

$$\varphi_{ij} = \begin{cases} 0 & q_{ij}^l > u_j \\ 1 - (q_{ij}^r - u_j)/(q_{ij}^r - q_{ij}^l) & q_{ij}^l \le u_j \le q_{ij}^r \\ 1 & q_{ij}^r < u_j \end{cases} \quad (4)$$

Finally, the user's satisfaction degree in the individual QoS aspect A_j of service i is computed with (5). We can see, a smaller φ_{ij} will lead to a smaller satisfaction degree, and vice versa.

$$SD(q_{ij}) = \frac{1}{2}(n_{ij}^r + n_{ij}^l) \cdot \varphi_{ij} \quad (5)$$

VI. OVERALL QOS EVALUATION

To evaluate the overall QoS of a service, we need to aggregate its normalized QoS values in every individual QoS aspect. Like many existing works such as [10], we use a weighted aggregate function to combining the user's satisfaction degrees of various QoS aspects with weights. The critical issue is how to determine the weight of every QoS aspect. Let $req(A_1, A_2, ..., A_n) = (req_1, req_2 ..., req_n)$ be the user's QoS requirements in all the concerned QoS aspects.

Since simply using the subjective weight neglect the important knowledge of different services' performance in different QoS aspects, we need to seek for objective weights to utilize this knowledge to achieve a more effective and reasonable ranking outcome.

The objective weighting technique used is based on Entropy. Entropy is a very popular objective weighting technique. However, traditional Entropy-based weighting techniques assume that each QoS apsect has a fixed value, whereas we allow each QoS aspect to use uncertain QoS values (QoS intervals). We compute the entropy for each QoS aspect and its relative weight as follows.

The entropy for the *jth* QoS aspect A_j is defined as:

$$e_j = -(1/\ln m) \cdot \sum_{i=1}^{m} p_{ij} \ln(p_{ij}) \quad (6)$$

where

$$p_{ij} = \sqrt{(q_{ij}^l)^2 + (q_{ij}^r)^2} \Big/ \sum_{i=1}^{m} \sqrt{(q_{ij}^l)^2 + (q_{ij}^r)^2}$$

Basically, the smaller the entropy of the *jth* QoS aspect A_j, the larger the weight should be assigned to A_j. In this

regard, the entropy weight for the *jth* QoS aspect is defined as

$$w_j^{'} = (1 - e_j) \Big/ \sum_{j=1}^{n} (1 - e_j) \tag{7}$$

Then we should normalize the subjective weights as follows:

$$w_j = w_j \Big/ \sum_{j=1}^{n} w_j \tag{8}$$

After we obtain the objective weights and subjective weights of each QoS aspect according to (7) and (8) respectively, the hybrid weights, combining the subjective weights and objective weights, is computed as follows:

$$w_j^{*} = \left(w_j * w_j^{'}\right) \Big/ \sum_{j=1}^{n} \left(w_j * w_j^{'}\right) \tag{9}$$

Finally, the overall satisfaction degree of the user on every service in S is calculated as follows:

$$OSD(S_i) = \sum_{j=1}^{n} SD(q_{ij}) \cdot w_j^{*} \tag{10}$$

Where $SD(q_{ij})$ is the user's satisfaction degree in the QoS aspect A_j of the service S_i, which is computed with (5).

VII. EVALUATION

In this section, we use an example to evaluate our approach. In this example, a developer plan to develop a mobile voice communication system for voice communication between vehicles. The developer would like to select an existing encryption service to be used in the system. The encryption service accepts a data stream and output an encrypted data stream. Furthermore, the developer has QoS requirements in the following four QoS aspects:

- Throughput: The encryption service can support at least 1,000 packages per second. Each package has 1,000 bits. That is, the throughput of the service should be at least 1Mbps.
- Delay: The additional delay caused by encryption of each package should be less than 10 ms.
- Security strength: The encryption service should provide appropriate security protection for voice communication of sensitive, but not classified, information. According to the security metric developed in [22], the developer requires the encryption service to prevent attackers from cracking messages with probability at least 60%. Since the information in the voice communication is not classified, the developer does not require an encryption service providing perfect security. Any encryption service with the probability 80% is already sufficient for this application.
- Price: The price of the encryption service should be less than 1 dollar per day.

TABLE II. USER-SPECIFIED QOS REQUIREMENTS OF ACCEPTABLE ENCRYPTION SERVICES

QoS aspect	l	u	w
Throughput	*1 Mbps*		*0.7*
Delay	*0*	*10 ms*	*Important*
Security	*60%*	*80%*	*Very important*
Price	*0*	*1 dollar/day*	

TABLE III. THE QOS OF CANDIDATE ENCRYPTION SERVICES

	Throughput	Delay	Security	Price
S1	*9-12 Mbps*	*5-9 ms*	*0.5-0.7*	*0.8 dollar/day*
S2	*3-4 Mbps*	*3-5 ms*	*0.9-1.0*	*1 dollar/day*
S3	*4-6 Mbps*	*1-2 ms*	*0.6-0.7*	*0.5 dollar/day*
S4	*1-8 Mbps*	*1-12ms*	*0.2-0.6*	*0.4dollar/day*

TABLE IV. ADJUSTED QOS REQUIREMENTS OF ACCEPTABLE ENCRYPTION SERVICES

QoS aspect	l	u	w	w*
Throughput	*1 Mbps*	*12 Mbps*	*0.24*	*0.22*
Delay	*0*	*10 ms*	*0.25*	*0.53*
Security	*60%*	*80%*	*0.34*	*0.14*
Price	*0*	*1 dollar/day*	*0.17*	*0.12*

TABLE V. THE USER'S SATISFACTION DEGREES ON CANDIDATE ENCRYPTION SERVICES

Satisfaction Degree	S1	S2	S3	S4
Throughput	*0.86*	*0.23*	*0.36*	*0.32*
Delay	*0.30*	*0.60*	*0.85*	*0.13*
Security	*0.13*	*1.00*	*0.25*	*0.00*
Price	*0.20*	*0.00*	*0.50*	*0.60*
Overall Satisfaction Degree (using only subjective weights)	*0.36*	*0.55*	*0.47*	*0.21*
Overall Satisfaction Degree (using hybrid weights)	*0.39*	*0.50*	*0.62*	*0.21*

Suppose there are four candidate encryption services which have the same functionalities. Their QoS values are listed in Table III. According to our proposed service selection method, some of the user-specified QoS requirements in Table II need to be adjusted. The adjustment is described as follows. All linguistic terms will be mapped to values according to Table I. The parameter u will be set to the largest throughput of all encryption services since there is no natural upper bound for the throughput. The importance weight for price will be set to 0.5 at first since the user does not specify it. Furthermore, all w will be adjusted according to (8) and the hybrid weights will be computed according to (9). The adjusted QoS requirements are shown in Table IV.

Table V shows the user's satisfaction degrees in individual QoS aspects for all encryption services, as well as his/her overall satisfaction degrees on individual encryption services computed by using only subjective weights or using the hybrid weights. We can see, as for the hybrid weighting method, S1, S2, S3, and S4 should be ranked as S3> S2> S1> S4 according to their satisfactory degrees. As for the traditional subjective weighting method, we can rank them as S2> S3> S1> S4.

It can be seen that two results are similar except the ranking order of S2 and S3. According to our observation, it is unreasonable for S2 to outperform S3. Although S2 has perfect performance in the security aspect, S3 has better average QoS than S2 and is able to meet the security requirement as well. Through careful analysis we can find that S2 is wrongly ranked because the weight assigned to security aspect make too much contribution to the final ranking result when just employing the subjective weights, which ignore the important knowledge of all the QoS value of all the services. By employing the hybrid weights, the user's subjective opinion can be tuned by the objective knowledge of all the services and thus lead to more reasonable results.

VIII. CONCLUSION

This paper presents a QoS-based service ranking and selection method in developing high-quality applications for mobile cloud computing. Considering that services in mobile cloud computing environments are likely to have uncertain QoS values in various QoS aspects, the proposed method engages flexible intervals rather than crisp values to represent the services' QoS in every QoS aspect. Based on the services' QoS and the users' QoS requirements, the method employs a set of functions to compute a user's satisfaction degree in the QoS aspect of individual services. The method also incorporates a weighting technique for more accurate weight assignments in different QoS aspects. This is done by combining subjective weights given by the user and objective weights produced by an entropy-based technique. Finally, the user's satisfaction degrees in individual QoS aspects were aggregated with hybrid weights to infer the user's overall satisfaction degrees on individual candidate services. The current work did not consider the distribution properties of services' QoS values in various QoS aspects. To improve this work, we will take this into consideration in our future work. Furthermore, more experiments will be conducted for evaluating our proposed service selection method and its improved version.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Project No. 61332010, 61472338), Guangdong Natural Science Foundation (Project No. 2014A030313151), the Research Grants Council of Hong Kong (Project No. CUHK 415113), and Microsoft Research Asia Grant in Big Data Research (Project No. FY13-RES-SPONSOR-036).

REFERENCES

[1] Satish Narayana Srirama,Carlos Paniagua, Huber Flores, "Social Group Formation with Mobile Cloud Services," Service Oriented Computing and Applications, vol. 6, no. 4, pp.351–362, 2012.

[2] Han Qi, Abdullah Gani, "Research on Mobile Cloud Computing: Review, Trend and Perspectives," in Proc. 2nd International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), pp. 195-202, Bangkok, 16-18 May 2012.

[3] Z. Rehman, FK Hussain, OK Hussain, "Towards Multi-Criteria Cloud Service Selection," in Proc. 5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp.44-48, Seoul, June 30 2011-July 2 2011.

[4] Zibin Zheng, Xinmiao Wu, Yilei Zhang, Michael R. Lyu, and Jianmin Wang, "QoS Ranking Prediction for Cloud Services", IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, ,pp. 1213-1222, June 2013.

[5] E. A. Masri and Q. H. Mahmoud, "QoS-based Discovery and Ranking of Web Services", in Proc. Int' l conf. on Computer Communications and Networks (ICCCN), pp. 529-534, 2007.

[6] M. Comuzzi and B. Pernici, "A Framework for QoS-based Web Service Contracting", ACM Transactions on The Web, vol. 3, no. 3, article 10, 2009.

[7] P. Wang, "QoS-aware Web Services Selection with Intuitionistic Fuzzy Set Under Consumer's Vague Perception", J. Expert Systems with Applications, vol. 369, no. 3, pp. 4460-4466, April 2009.

[8] Mohamed Almulla, Kawthar Almatori, Hamdi Yahyaoui, "A QoSbased Fuzzy Model for Ranking Real World Web Services", in Proc. International conference on Web Services (ICWS), pp. 203-210, 2011.

[9] Xiaoqing (Frank) Liu, Kenneth Kofi Fletcher, Mingdong Tang. "Service Selection based on Personalized Preference and Trade-Offs among QoS Factors and Price", in Proc. 1st IEEE International Conference on Services Economics, pp. 32-39, Hawaii, USA, June, 2012.

[10] Stephen S. Yau and Yin Yin, "QoS-based Service Ranking and Selection for Servic-based Systems," in Proc. International Conference on Services Computing (SCC), pp. 156-863, 2011.

[11] Zibin Zheng, Hao Ma, Michael R. Lyu, and Irwin King "QoS-Aware Web Service Recommendation by Collaborative Filtering", IEEE Transactions on Services Computing, vol.4, no.2, pp.140-152, 2011.

[12] Qibo Sun, Shangguang Wang, Hua Zou, Fangchun Yang, "QSSA: A QoS-aware Service Selection Approach", International Journal of Web and Grid Services, vol.7, no.2, pp.147-169, 2011.

[13] Shangguang Wang, Zibin Zheng, Qibo Sun, Hua Zou and Fangchun Yang, "Reliable Web Service Selection via QoS Uncertainty Computing", International Journal of Web and Grid Services, vol. 7, no. 4, pp.410-426, 2011.

[14] S. N. Srirama, M. Jarke, and W. Prinz and H. Zhu, "Scalable Mobile Web Service Discovery in Peer to Peer Networks", In Proc. 3nd International Conference on Internet and Web Applications and Services (ICIW), pp.668-674, 2008.

[15] Y. Kim and K. Lee, "A Lightweight Framework for Mobile Web Services", Journal on Computer Science Research and Development, vol. 24, no. 4, pp.199-209, 2009.

[16] rirama SN, Shor V, Vainikko E, Jarke M, "Supporting Mobile Web Service Provisioning with Cloud Computing", International Journal on Advances in Internet Technology, vol.3, no.34, pp.261–273, 2010.

[17] Hoan K T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches," Wireless Communication and Mobile Computing, vol. 13, pp. 1587-1611, 2013.

[18] Peng Zhang, Zheng Yan, "A OoS Aware System for Mobile Cloud Computing", in Proc. IEEE Int' l Conf. on Cloud Computing and Intelligence Systems (CCIS), pp.518 – 522, Sept. 2011.

[19] Alazawi, Z, Altowaijri S.; Mehmood, R.;AbdlJabar, M.B., “Intelligent Disaster Management System based on Cloud-enabled Vehicular Networks”, in Proc. 11th International Conference on ITS Telecommunications (lTST), pp.361- 568, Aug. 20l1.

[20] Khalid Elgazzar, Hossam S. Hassanein, Patrick Martin, “DaaS: Cloud-based Mobile Web Service Discovery”, Pervasive and Mobile Computing, vol. 13, pp. 67-84, 2014

[21] Zhefu Shi and Ruirui Gu, “A framework for mobile cloud computing selective service system”, in Proc. Wireless Telecommunications Symposium (WTS), pp.1-5, 2013

[22] S. S. Yau, Y. Yin and H. G. An, “An Adaptive Model for Tradeoff between Service Performance and Security in Service-based Environments”, in Proc. Int’l Conf. on Web Services, pp. 287-294, 2009.