QoS-based Service Ranking and Selection for Servic-based Systems

Stephen S. Yau and Yin Yin
Information Assurance Center, and
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, AZ, USA
{yau, yin.yin}@asu.edu

***Abstract*—To facilitate rapid development of service-based systems (SBS), many service discovering and matching techniques have been developed to find services according to users' functionality requirements. However, users usually also have requirements on non-functional qualities of services (QoS), such as throughput, delay, reliability and security, which are also critical for the success of SBS. In this paper, a QoS-based service ranking and selection approach is presented to help users to select the service that best satisfies users' QoS requirements from a set of services having already satisfied users' functionality requirements. To determine how well a service satisfies users' concerned QoS requirements, a set of functions is presented to normalize services' QoS on various QoS aspects with different metrics and scales, compute services' satisfaction scores on each QoS aspect, and combine each services' satisfaction scores on all QoS aspects together as an overall satisfaction scores. Compared with existing service ranking and selection techniques, our approach has the following advantages: 1) selects the service that best satisfies users QoS requirements instead of the service with the best QoS which may be much overqualified for the users' QoS requirements, 2) improves the flexibility in users' QoS requirement specification, and 3) uses the prospect theory to more accurately model the relation between services' QoS and their satisfaction scores.**

Keywords- Service ranking and selection; QoS requirements, QoS-based;satisfaction score; service-based systems; prospect theory.

I. INTRODUCTION

Web services and service-oriented architecture (SOA) have emerged as effective tools in improving distributed application systems' flexibility, interoperability, scalability and robustness, which have been widely used in many domains, such as e-commerce, e-government, health care, and homeland security [1, 2]. In SOA, service providers specify their services' functionalities as contracts through standardized service description language (WSDL) [3] and register their services in service directories. Users, such as developers of service-based systems (SBS) who are looking for services providing specific functionalities, can search for services in service directories, and integrate the found services into their SBSs instead of developing systems from scratch. This process certainly accelerates and facilitates the development of large-scale of service-based systems.

Besides the functionalities provided by services, the services' QoS, such as throughput, delay, reliability and security, are also important [4]. However, existing service discovering and matching approaches, such as UDDI, Service Location Protocol (SLP), Jini, and Bluetooth, only search for services with functionalities, and do not provide any guarantee on services' QoS. Hence, the users have no idea about whether the discovered services' QoS can satisfy their QoS requirements. Furthermore, if there are several services providing equivalent functionalities, users do not know which service should be selected.

To incorporate service QoS information in service discovery and matching, there are two major challenges. The first is to specify, measure, and monitor services' QoS. Several QoS quality models and ontologies have been presented to formally organize and specify various QoS aspects with their metrics and scales [5-9]. These QoS specifications are then integrated with extended web service standards [10-16]. Furthermore, services' QoS can be either monitored and updated through a runtime QoS monitor [17], users' feedbacks and reputations [18, 19], or data mining with historic data [20, 21].

The second challenge is to match services' QoS with users' QoS requirements. While users have requirements on 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. Furthermore, services' QoS on various QoS aspects cannot be simply combined because different users' preferences on concerned QoS aspects may be different, and various QoS aspects have different scales and value ranges.

While the first challenge has been well addressed by existing techniques [5-21], the second challenge has not been well studied. In this paper, we will present a service ranking and selection approach to meet the second challenge. This approach will include 1) normalizing services' QoS on various aspects to a unique range, 2) measuring how well services' QoS satisfies users' QoS requirements on each concerned QoS aspect with a satisfaction score, and 3) combining services' satisfaction scores on all concerned aspects together as an overall satisfaction score. Our approach has the following advantages: 1) selecting the service that best satisfies users' concerned QoS requirements instead of the service with the best QoS which may be overqualified to satisfy users' concerned QoS requirements, which is helpful in improving services' utilizations, and save resources to satisfy other users' QoS requirements; 2) improving the flexibility in users' QoS requirement specifications, and 3) using the prospect theory [22, 23] to more accurately model the relation between services' QoS and their satisfaction scores.

II. RELATED WORK

Due to the possible large variety of QoS aspects' metrics and scales, services' QoS on different aspects in a given set need to be normalized before to be combined together for service ranking. Let $\bar{S}$ be a set of m services $\{S_1, S_2, \ldots, S_m\}$, each of which satisfies user's functional requirements, and their QoS on a specific QoS aspect are $q_1, q_2, \ldots, q_m$, respectively. In [24], service S_i's q_i is normalized to $q_i/avg(q_1, \cdots q_m)$, where $avg(q_1, \cdots q_m)$ is the average QoS of all services in $\bar{S}$. In [25], q_i is normalized to $(q_{max} - q_i)/(q_{max} - q_{\min})$ or $(q_i - q_{min})/(q_{max} - q_{\min})$, where q_{max} and q_{min} are the best and worst QoS among all services in $\bar{S}$, respectively. These normalization approaches only consider the QoS of the services in $\bar{S}$ without users' QoS requirements and preferences on QoS aspects. For example, if the QoS of the all services in $\bar{S}$ cannot satisfy users' QoS requirements or perfectly satisfy users' QoS requirements, the differences between the QoS of the services in $\bar{S}$ should not have any effect to users, and hence should not be considered in the service ranking and selection.

To consider users' QoS requirements in the QoS normalization, the range of services' QoS on each concerned aspect is divided to several discrete levels [27, 28]. Different levels represent users' different degrees of satisfaction, and QoS within the same level is normalized to the same number. These approaches, however, are too coarse to distinguish services which have the same QoS level. Furthermore, these approaches require a lot of interactions with users and do not consider users' preferences on various concerned QoS aspects.

Let service S_i's QoS on all n concerned QoS aspects be $q_{i1}, q_{i2}, \ldots, q_{in}$, which are normalized as $n_{i1}, n_{i2}, \ldots, n_{in}$ respectively. To consider all QoS aspects in service selection, $n_{i1}, n_{i2}, \ldots, n_{in}$ need to be combined together. Typically, the combination is done by adding the normalized QoS $n_{i1}, n_{i2}, \ldots, n_{in}$ together with weights [26]. That is, the service S_i will be ranked based on the value of $\sum w_j * n_{ij}$, where the weight w_j represents the user's preference on the j-th concerned QoS aspect. The user assigns larger weights for QoS aspects considered more important, and smaller weights for QoS aspects considered less important. This approach ranks and selects services

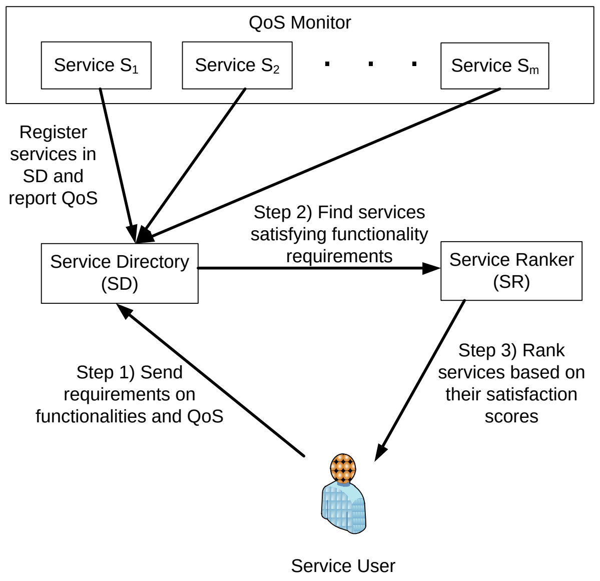
QoS Monitor
Service S1
Service S2
Service Sm
Register services in SD and report QoS
Step 2) Find services satisfying functionality requirements
Service Directory (SD)
Service Ranker (SR)
Step 1) Send requirements on functionalities and QoS
Step 3) Rank services based on their satisfaction scores
Service User

Figure 1. The overall process of our QoS-based service ranking and selection approach.

based on only services' QoS. To consider users' QoS requirements, the Euclidian distance between services' QoS and users' QoS requirements is used to rank services [25]. In [27], a price model is used to combine services' QoS on multiple concerned QoS aspects. The price model converts a service's QoS on each concerned QoS aspect to a price, and adds all prices together as the service's total price, and ranks services based on their total prices.

Moderated Fuzzy Discovery Method (MFDM) [29] and Intuitionist Fuzzy Sets (IFS) [30, 31] have been used to help users specify QoS requirements and preferences on various concerned QoS aspects using linguistic terms. For example, users can specify their requirements and preferences as "the service is very cheap" and "the price of the service is very important" instead of "the price of the service should be less than one dollar per day" and "the importance weighting factor of the service's price is 0.9.

III. OUR OVERALL APPROACH

To monitor services' QoS and facilitate the service ranking and selection, we assume that all users' concerned services' QoS aspects are monitored and reported to the service directory SD, which saves services' QoS information along with their registrations. This information will be used by the service ranker SR to rank services and help users select services when there are more than one registered services satisfying users' functionality requirements.

Our service ranking and selection approach is depicted in Figure 1 and can be described in the following three steps:

Step 1). The user specifies a set of concerned QoS aspects, such as throughput, delay, reliability, security, and price. For each concerned QoS aspect, the user specifies his/her QoS requirements of this QoS aspect, including the expected lower bound and upper bound of each acceptable QoS aspect, the importance of the requirement of each QoS aspect, and the confidence in the specified QoS requirements as discussed in Section IV. These QoS requirements are sent to the service directory SD with the user's functionality requirements on services.

Step 2). The SD first matches the user's functionality requirements on services with all registered services in SD [3]. Let the set of services in the SD satisfying all the user's service functionality requirements be the set $\overline{S} = \{S_1, S_2, ..., S_m\}$. $\overline{S}$ is sent to the service ranker SR.

Step 3). With the user's concerned QoS requirements, SR computes a satisfaction score for each service in $\overline{S}$, and ranks the services in $\overline{S}$ according to their satisfactory scores in descending order. If there are several services have the same satisfaction score, the order among the services with the same satisfaction score is random. SR returns the list of ranked services in $\overline{S}$ to the user. The top ranked service is the service that best satisfies the user's concerned QoS requirements.

In the next section, we will discuss how to specify the user's QoS requirements in *Step 1)*. In Section V, we will discuss how to compute the satisfaction scores of a service on various QoS aspects and rank the services in $\overline{S}$ in *Step 3)*. An example will be given to illustrate our approach in Section VI.

IV. QoS Requirement Specification

Let $\{c_1, c_2, ..., c_n\}$ be the set of concerned QoS aspects specified by the user. For each QoS aspect c_j, the user specifies his/her QoS requirement of c_j as a tuple

$$req_j = \{l_j, u_j, w_j, r_j\},$$

where l_j and u_j are the lower bound and upper bound of user's acceptable QoS on c_j, w_j is the weighting factor of requirement req_j representing the user's preference on c_j in the service selection, and r_j is is within (0, 1], representing the user's confidence in the values of l_j, u_j and w_j specified in req_j

A. Expected QoS l_j and u_j

The values of l_j and u_j of a different QoS aspect c_j may have different meanings and be represented with different units. For example, the QoS aspects delay and availability are represented as seconds and percentage respectively. Services are perfect if their delays are smaller than l_j, and unacceptable if their delays are larger than u_j. However, 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 them, 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 c_j has lower bound and upper bound, then l_j and u_j can be set as the metric's lower bound and upper bound for c_j, respectively. For example, the metric of 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 availability, we can set $l_j = 0\%$ and $u_j = 100\%$ if they are blank in Req_j.

Second, if the c_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, the values of l_j and u_j are defined as the smallest and largest QoS of all available services in $\overline{S}$, respectively.

B. Importance Weight w_j

As mentioned before, the weighting factor w_j of requirement req_j represents the user's preference on c_j in the service selection, and can be specified as either a value within (0, 1]. A larger w_j means that c_j is more important for users, and services' QoS on c_j will have more impact on the service ranking and selection.

When the user does not know how to specify his/her preferences on 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. In this paper, we map linguistic terms to weight values shown in Table I. If the user does not specify their importance weights in req_j, the midpoint 0.5 as the default value will be applied.

TABLE I. The Mapping Between Linguistic Terms and Importance Weight Values

Linguistic terms	**Weight values**
Very unimportant	0.1
Unimportant	0.25
Medium	0.5
Important	0.75
Very important	1

C. Confidence Value r_j

The confidence value r_j allows inexperienced users to specify req_j with uncertainty, which is a number within (0, 1]. 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.

With confidence value 1, services' satisfaction scores will be computed according to the specified l_j, u_j and w_j. For confidence values within (0, 1), the specified values of l_j, u_j and w_j will be adjusted with the value of r_j.

Because our ranking approach does not consider the difference between services' actual QoS when their QoS does not belong to the range [l_j, u_j], the range [l_j, u_j] will be extended to preserve more QoS information in the normalization process when the confidence r_j is not 1. The adjustments for l_j and u_j are shown in the functions *Norm₁* and *Norm₂* described in Section V.A.

The importance weight w_j, is used in computing and combining services' satisfaction scores. When r_j is not 1, w_j will be adjusted with r_j as follows:

$$w'_j \leftarrow w_j r_j + 0.5(1 - r_j) \tag{1}$$

When $r_j = 1, w'_j = w_j$. When r_j is close to 0 (i.e., the user has little confidence in the specifications), w'_j approaches to the default value 0.5. This adjustment tends to decrease the difference between the user's specified value of w_j and the default value of w_j, when the user is not confident in his/her specifications. Hence, when the user is not clear about which QoS aspects are more important and unimportant, all QoS aspects will be treated as equally important.

If the user does not specify r_j in req_j, the default value 1 will be applied.

V. Satisfactory Score

The ranking of services in our approach is based on how well services in $\overline{\text{S}}$ satisfy the user's QoS requirements. In this section, we will define the satisfaction score and discuss how to compute the overall satisfaction score for each service.

Let C={c_1, c_2, ..., c_n} be the set of concerned QoS aspects of the user. The QoS of all S_i in $\overline{\text{S}}$ from SD is represented by the following matrix

$$\text{QoS}\begin{pmatrix} S_1 \\ \vdots \\ S_m \end{pmatrix} = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ \vdots & \vdots & \ldots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} \end{pmatrix}, \tag{2}$$

where {q_{i1}, q_{i2}, ..., q_{in}} represents the QoS of S_i on {c_1, c_2, ..., c_n}, $1 \le i \le m$, and {q_{1j}, q_{2j}, ..., q_{mj}} represents the QoS of all services {S_1, S_2, ..., S_m} on c_j, $1 \le j \le n$. The user specifies his/her QoS requirements on each c_j as req_i as follows:

$$\text{Req}(\ c_1, \cdots, c_n\ ) = (\ req_1, \ \ \cdots, \ \ req_n\ ) \tag{3}$$

For each service S_i, a satisfactory score sc_{ij} is computed for each c_j, which is a number within [0, 1] to indicate how well S_i's QoS on C satisfies $Req(c_1, c_2, ..., c_n)$. When $sc_{ij} = 0$, q_{ij} does not satisfy req_j; when $sc_{ij} = 1$, q_{ij} satisfies req_j perfectly; when $0 < sc < 1$, q_{ij} partially satisfy req_j. A larger sc_{ij} represents better satisfaction.

To compute the satisfaction scores of {S_1, S_2, ..., S_m} on the user's QoS requirements (3), all elements in (2) will first be normalized with normalization functions *Norm₁* and *Norm₂* to be defined in Subsection V.A to [0, 1], and then compared them to (3) with the satisfaction score function $SSF^{w'}$ to be defined in Section V.B, which will be used for comparing the satisfaction scores for each S_i and each c_j as follows:

$$\text{SC}\begin{pmatrix} S_1 \\ \vdots \\ S_m \end{pmatrix} = \begin{pmatrix} sc_{11} & sc_{12} & \cdots & sc_{1n} \\ \vdots & \vdots & \ldots & \vdots \\ sc_{m1} & sc_{m2} & \cdots & sc_{mn} \end{pmatrix} \tag{4}$$

Finally, each S_i's overall satisfactory score sc_i is computed with the combination function *CF* to be defined in Subsection V.C by combining {sc_{i1}, ..., sc_{in}} together.

A. QoS Normalization Functions

QoS aspects can be classified in two types: The first type of QoS aspects are *utility* type, such as throughput and security, which users want to maximize. The second type of QoS aspects are *cost* type, such as delay and services' prices, which users want to minimize. In this subsection, we will present two normalization functions *Norm₁* and *Norm₂* for these two types of QoS aspects respectively.

For a service S_i and a QoS aspect c_j, a larger q_{ij} means better quality if c_j is a utility type QoS or worse quality if c_j is a cost type QoS Hence, the QoS normalization function *Norm₁* for utility type QoS should increase with q_{ij}, while the *Norm₂* for cost type QoS should decrease with q_{ij}. *Norm₁* and *Norm₂* are defined as follows:

$$Norm_1(q_{ij}) = \begin{cases} 0, & \text{if } q_{ij} < l_j r_j \\ \frac{q_{ij} - l_j r_j}{u_j/r_j - l_j r_j}, & \text{if } l_j r_j \le q_{ij} \le u_j/r_j \\ 1, & \text{if } q_{ij} > u_j/r_j \end{cases} \tag{5}$$

$$Norm_2(q_{ij}) = \begin{cases} 0, & \text{if } q_{ij} > u_j/r_j \\ \frac{u_j/r_j - q_{ij}}{u_j/r_j - l_j r_j}, & \text{if } l_j r_j \le q_{ij} \le u_j/r_j \\ 1, & \text{if } q_{ij} < l_j r_j \end{cases} \tag{6}$$

where req_j = {l_j, u_j, w_j, r_j} is the QoS requirement specified by the user for the concerned QoS aspect c_j.

The *Norm₁* and *Norm₂* adjust the range [l_j, u_j] to the range [$l_j r_j$, u_j/r_j] according to the user's confidence in u_j and l_j. While existing normalization functions used in [24, 25] always generate larger numbers for services with better QoS, the *Norm₁* and *Norm₂* normalize services' qualities to 0 if $q_{ij} < l_j r_j$ for utility type QoS c_j, or $q_{ij} > u_j/r_j$ for cost type QoS c_j, regardless how bad these services' QoS. Similarly, the *Norm₁* and *Norm₂* will normalize services' qualities to 1 if $q_{ij} > u_j / r_j$ for utility type QoS, or $q_{ij} < l_j r_j$ for cost type QoS, regardless how good these services' QoS.

B. Satisfactory Score Function

In this subsection, we will present the satisfaction score function $SSF^{w'}$ to computes the satisfactory scores sc_{ij} in (4) for each service S_i and each QoS aspect c_j.

Let n_{ij} be the normalized value of q_{ij}. $SSF^{w'}$ should satisfy the following conditions:

- If the user does not care about services' QoS qualities on c_j, the user will not specify any QoS requirement for c_j. Then, services' qualities on c_j will not be considered in the ranking of services.
- If the user only specifies a threshold requirement on services' QoS qualities on c_j (e.g., the user does not care about the price as long as it is within the budget), the user will use the same lower bound and upper bound in the QoS specification. This means that all services' qualities on c_j are normalized to 1 if their QoS satisfies the threshold or to 0 if their QoS qualities do not satisfy the threshold. Hence, the $SSF^{w'}$ does not need to compute the satisfaction score for the normalized QoS within (0, 1).
- If the user defines a range of acceptable QoS, the $SSF^{w'}$ should output 0 for all services' QoS below the lower bound of the range, and 1 for all services' QoS above the upper bound of the range. That is, $SSF^{w'}(0) = 0$ and $SSF^{w'}(1) = 1$.
- If sc_{ij} is between 0 and 1, a larger sc_{ij} means that the user will be better satisfied with S_i's QoS, and S_i will be ranked higher in service selection.

A simple satisfaction function can use the value of n_{ij} as sc_{ij}, which is linear and satisfies all of the above three conditions. However, this simple function assumes that the user's satisfaction of a service is proportionate to the service's QoS which is not always true as shown in [28] and the prospect theory in economics [22, 23].

Because the expected lower and upper bounds represent the range of minimum acceptable QoS and best QoS, all QoS qualities within this range is acceptable, but users will be more satisfied with better QoS. Using the normalized QoS as the satisfaction score implies that the satisfaction of services is only determined by the normalized QoS, and services' satisfaction will be doubled if their normalized QoS is doubled. However, it follows the prospect theory [22, 23] that the satisfaction should be based on the gains and losses relative to the reference point instead of absolutely determined by services' normalized QoS.

If we define the reference point as $n_{ij} = 0.5$ for each c_j, the user will have gains if the user chooses a service with normalized QoS larger than 0.5, and will suffer losses if the user chooses a service with normalized QoS smaller than 0.5. According to the prospect theory, the satisfaction function should be concave for gains, and convex for losses. For example, the delay of VoIP services is usually expected to be within the range (*50 ms*, *400 ms*) recommended by

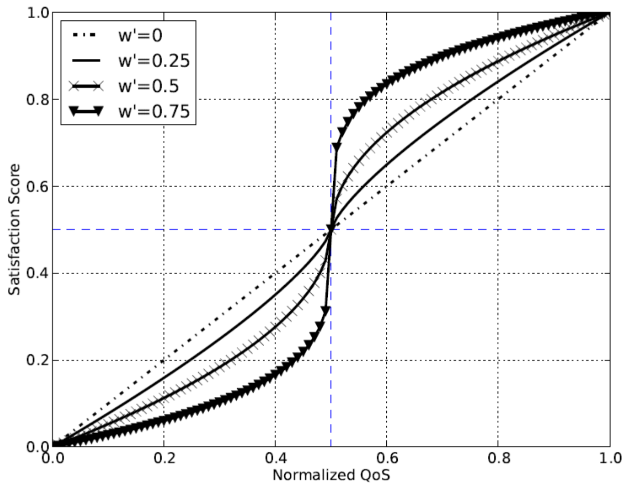

Figure 2. The effect of parameter w' on the function $SSF^{w'}$

ITU G.114. Hence, the reference point of delay in VoIP is *225 ms* (*i.e.*, $Norm_2(225) = 0.5$). It is easier for the user to discriminate a longer delay from *150 ms* to *200 ms* than to discriminate a longer delay from *50 ms* to *100 ms* because the pauses in speech are generally longer than *100 ms*. Furthermore, It is also easier for the user to discriminate a longer delay from *250 ms* to *300 ms*, than to discriminate a longer delay from *350 ms* to *400 ms* because *350 ms* is already longer than most pauses in speech.

Satisfaction is subjective and may be different for different users even for the exact same service. Consider the delay of VoIP services again. First, a user who usually talks fast may specify the range of expected delay as (*50 ms*, *300 ms*), while a user who usually talks slowly may specify the range as (*150 ms*, *400 ms*). Because the normalization functions presented in Subsection V.A normalize services' QoS with users' QoS requirements, the actual delay of a normalized delay 0.5 will be different for different users. Second, even with the same expected range, the relation between the satisfaction and the normalized QoS also depends on users' subjective feelings. For example, if a user is more sensitive about the speed of speaking, the VoIP service's satisfaction score will change faster with the change in delay. That is, the satisfaction function becomes steeper around the reference point.

Based on the prospect theory and the above discussion, $SSF^{w'}$ is defined as follows:

$$SSF^{w'}(n_{ij}) = \begin{cases} 0.5(2n_{ij} - 1)^{1-w'} + 0.5, & if\ n_{ij} > 0.5 \\ -0.5(-2n_{ij} + 1)^{1-w'} + 0.5 & if\ n_{ij} \leq 0.5 \end{cases} \quad (7)$$

where n_{ij} is the normalized QoS, and w' is the adjusted importance weight of the QoS aspect c_j given in (1). For any value of w', $SSF^{w'}(0) = 0$, $SSF^{w'}(1) = 1$, and $SSF^{w'}(n_{ij})$ increases with the increase of $n_{ij} \in [0,1]$. Because larger n_{ij} indicates better QoS for both utility type and cost type QoS,

$SSF^{w'}(n_{ij})$ always outputs a larger satisfactory score for better QoS.

The $SSF^{w'}(n_{ij})$ (7) has the following properties:

- When $w' \rightarrow 0$, $SSF^{w'}$ is becoming a linear function with n_{ij} as SSF^0 shown in Figure 2.
- $SSF^{w'}$ uses 0.5 as the reference point. In Figure 2, $SSF^{w'}(0.5)=0.5$ for any w'.
- With the increasing of n_{ij}, the increasing speed of $SSF^{w'}$ accelerates at the beginning until $n_{ij} = 0.5$, where $SSF^{w'}$ reaches the maximum increasing speed. After $n_{ij} = 0.5$, the increasing speed of $SSF^{w'}$ starts to decrease. Finally, $SSF^{w'}$ reaches 1 at $n_{ij} = 1$. This trend is consistent with the prospect theory [22, 23].

C. *Combination Function*

After the satisfaction scores for each concerned QoS aspects have been computed by $SSF^{w'}$, we need the combination function *CF*, which will be presented in this subsection, to compute the overall satisfaction scores for all the services in $\bar{S}$ by combining their satisfaction scores of various QoS aspects with weights.

The combination function *CF* is defined as the average of the weighted sum of satisfaction scores on all QoS aspects as follows:

$$CF(S_i) = \frac{\sum_{1\le j\le n} sc_{ij} w'_j}{\sum_{1\le j\le n} w'_j} \quad (8)$$

where sc_{ij} is S_i's satisfaction score on c_j computed with (7), and w'_j is the adjusted importance weight w_j of c_j with (1).

VI. An Example

In this section, we will give an example to illustrate our approach. In this example, we assume that a developer of a voice communication system wants to upgrade the system by providing secure peer-to-peer voice communication. The developer would like to select an existing encryption service to be used in the system. The encryption service should accept a data stream and output an encrypted data stream. Furthermore, the developer has QoS requirements on 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 [32], the developer requires the encryption service to prevent attackers from cracking messages with probability at least 60%. But, because 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.

Following our approach, we have

Step 1) The developer specifies the QoS requirements shown in Table II, and sends the information to SD. Some parameters in the QoS requirements are blank because the developer does not know how to specify them. Throughput and security are utility QoS aspects. Delay and price are cost QoS aspects.

Step 2) SD finds three encryption services S_1, S_2 and S_3. Their QoS are shown in Table III.

Step 3) SR computes a satisfaction score for each service S_1, S_2 and S_3 as follows:

i. SR sets blank parameters in Table II as follows: Because there is no natural upper bound for the throughput, the parameter *u* will be set to the largest throughput of all encryption services. The importance weight for price will be set to the default value 0.5. Furthermore, all linguistic terms will be mapped to values according to Table I, and all *w* will be adjusted with *r* according to (1). The adjusted QoS requirements are shown in Table IV.

ii. SR normalizes throughput and security according to (5), and delay and price according to (6). The normalized QoS are shown in Table V.

iii. SR computes the satisfaction scores for each QoS aspect according to (7) and combines them together according to (8). All satisfaction scores are shown in Table VI.

At the end of this step, SR returns the ranked services $S_2 > S_3 > S_1$ to the user.

TABLE II. User-specified QoS Requirements of Acceptable Encryption Services

QoS aspect	*l*	*u*	*w*	*r*
Throughput	1Mbps		0.7	0.9
Delay	0	10 ms	Important	0.8
Security	60%	80%	Very important	1.0
Price	0	1 dollar/day		1.0

TABLE III. The QoS of Encryption Services Returned By SD

	Throughput	Delay	Security	Price
S_1	10 Mbps	10 ms	0.7	1 dollar/day
S_2	1.5 Mbps	5 ms	0.8	0.5 dollar/day
S_3	5 Mbps	1 ms	1.0	2 dollar/day

TABLE IV. ADJUSTED QOS REQUIREMENTS OF ACCEPTABLE ENCRYPTION SERVICES

QoS aspect	*l*	*u*	*w'*	*r*
Throughput	1Mbps	10 Mbps	0.68	0.9
Delay	0	10 ms	0.74	0.8
Security	60%	80%	1.0	1.0
Price	0	1 dollar/day	0.5	1.0

TABLE V. NORMANIZED QOS OF ENCRYPTION SERVICES RETURNED BY THE SD

	$Norm_1$ (Throughput)	$Norm_2$ (Delay)	$Norm_1$ (Security)	$Norm_2$ (Price)
S_1	0.89	0.2	0.50	0
S_2	0.06	0.6	1	0.5
S_3	0.4	0.92	1	0

TABLE VI. THE SATISFACTION SCORES OF ENCRYPTION SERVICES RETURNED BY SD

	S_1	S_2	S_3
Satisfaction Score on Throughput	0.96	0.02	0.20
Satisfaction Score on Delay	0.06	0.83	0.98
Satisfaction Score on Security	0.50	1.00	1.00
Satisfaction Score on Price	0	0.50	0
Overall Satisfaction Score	0.410	0.643	0.637

VII. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have presented a QoS-based service ranking and selection approach to matching services with users' requirements on both functionalities and QoS, which helps users select the services that best satisfy their QoS requirements. Our approach has the following advantages:

- The top ranked service is the service with the largest satisfaction score which means that the service's QoS can best satisfy users' QoS requirements. Hence, the services whose QoS qualities are not the best among those of all available services satisfying the user's QoS requirements can still be ranked on the top as long as they can perfectly satisfy users' QoS requirements. This is helpful in improving services' utilizations, and save resources to satisfy other users' QoS requirements.
- Users are allowed to specify their QoS requirements on each concerned QoS aspect as a range of acceptable QoS with the importance weight and uncertainty rather than only a value indicating the required QoS. This is more flexible and easier for users to specify their QoS requirements, especially for inexperienced users. Furthermore, our approach allows users to specify QoS requirements with linguistic terms and leave some parameters blank if they do not know how to specify them.
- The accuracy of service ranking is improved by considering users' QoS requirements and satisfaction on services' QoS. First, the QoS normalization eliminates the differences among services' actual QoS when all of them are unacceptable or perfect because these services' QoS qualities have no difference from the users' perspective. Second, users' feeling on how well services' QoS qualities satisfy their QoS requirements is modeled with the prospect theory, which considers human behavior and psychological principles in the computation of services' satisfaction scores.

Our approach can be improved with the following research: Although our approach has facilitated the specification on QoS requirements by allowing users to specify their requirements with linguistic terms, these terms are simply mapped to predefined values as shown in Table 1. However, the values for the same linguistic terms may be different for different users. This can be improved if the mapping between values and linguistic terms are adaptable for users. Furthermore, it is more flexible if users can specify their expected range of QoS with linguistic terms as well as the importance weights.

The prospect theory well describes users' decision among several choices with uncertainty, and is used in our approach to compute services' satisfaction scores. Although this theory has been supported by many social studies and widely used in economics, the evaluation on the correctness and effectiveness of this theory in the service ranking and selection require more studies in service users' behavior.

ACKNOWLEDGMENT

This work is supported by National Science Foundation under grant number CCF-0725340.

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, "Web Services - Concepts, Architectures and Applications", M. J. Carey and S. Ceri (eds.), Springer, 2004.

[2] N. Josuttis, *Soa in Practice: The Art of Distributed System Design*, O'Reilly Media Inc., 2007.

[3] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1", W3C Note, available at http://www.w3.org/TR/wsdl

[4] J. Osullivan, "What's in Service? Towards Accurate Description of Non-Functional Service Properties," *IEEE Trans. on Distributed and Parallel Database*, vol. 12, 2002, pp. 117-133.

[5] E. Kim, Y. Lee, "Quality Model for Web Services v2.0", Committee Draft, 2005, available at http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc

[6] D. Martin, M. Burstein, J. Hobbs, *et. al.* "OWL-S: Semantic Markup for Web Services," W3C Member Submission, 2004, available at http://www.w3.org/Submission/OWL-S/

[7] J. Bruijin, C. Bussler, J. Domingue *et. al.* "Web Service Modeling Ontology (WSMO)," WSMO Final Draft, 2006, available at http://www.wsmo.org/TR/d2/v1.3/

[8] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, "A QoS-aware Selection Model for Semantic Web Services", *Proc. Int'l Conf. on Service-Oriented Computing*, 2006, pp. 309-401.

[9] A. Andrieux, K. Czajkowski, A. Dan *et.al.* "Web Service Agreement Specification (WS-Agreement)", Open Grid Forum Proposed Recommendation, 2007, available at http://www.ogf.org/documents/GFD.107.pdf

[10] A. Ambrogio, "A Model-driven WSDL Extension for Describing the QoS of Web Services", *Proc. IEEE Int'l Conf. on Web Service*, 2006, pp. 789-796.

[11] J. Kopecky, T. Vitvar, C. Bournez and J. Farrell, "Sawsdl: Semantic Annotations for WSDL and Xml Schema," *IEEE Journal of Internet Computing*, vol 11(6), 2007, pp. 60- 67.

[12] J. Farrell and H. Lausen, "Semantic Annotations for WSDL and XML Schema", W3C Recommendation, 2007, available at http://www.w3.org/TR/sawsdl/

[13] V. Agarwal and P. Jalote, "From Specification to Adaptation: An Integrated QoS-driven Approach for Dynamic Adaptation of Web Service Compositions," *Proc. IEEE Int'l Conf. on Web Services*, 2010, pp. 275-282.

[14] A. ShaikhAli, O. Rana, R. Al-Ali and D. Walker, "UDDIe: An Extended Registry for Web Services," *Proc. Symp. on Applications and the Internet Workshops*, 2003, pp. 85- 89.

[15] Z. Xu, P. Martin, W. Powley and F. Zulkernine, "Reputation Enhanced Qos-based Web Services Discovery", *Proc. IEEE Int'l Conf. on Web Services*, 2007, pp. 249- 256.

[16] C. Lo, D. Cheng, P. Lin, and K. Chao, "A Study on Representation of QoS in UDDI for Web Services Composition", *Proc. Int'l Conf. on Complex, Intelligent and Software Intensive Systems (CISIS)*, 2008, pp. 423-428.

[17] S. S. Yau, N. Ye, H. Sarjoughian, D. Huang, A. Roontiva, M. Baydogan, and M. Muqsith, "Toward Development of Adaptive Service-Based Software Systems", *IEEE Trans. on Services Computing*, vol. 2(3), 2009, pp. 247-260.

[18] H. T. Nguyen, W. Zhao, and J. Yang, "A Trust and Reputation Model Based on Bayesian Network for Web Service," *Proc. IEEE Int'l Conf. on Web Services*, 2010, pp. 251-258.

[19] S. S. Yau, J. Huang and Y. Yin, "Improving the Trustworthiness of Service QoS Information in Service-based Systems", *Proc. 7th Autonomic and Trusted Computing (ATC)*, 2010, pp. 208-218.

[20] M. Godse, U. Bellur, R. Sonar, "Automating QoS Based Service Selection," *Proc. IEEE Int'l Conf. on Web Services*, 2010, pp. 534-541.

[21] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "Monitoring, Prediction and Prevention of SLA Violations in Composite Services", *Proc. IEEE Int'l Conf. on Web Service*, 2010, pp. 369-376

[22] D. Kahneman and A. Tversky, "Prospect Theory: An Analysis of Decision under Risk," Econometrica, vol. 47(2), 1979, pp. 263–292.

[23] Amos Tversky and Daniel Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty," Journal of Risk and Uncertainty, vol. 5, 1992, pp. 297–323.

[24] Y. Liu, A. H. Ngu, L. Zeng, "QoS Computation and Policing in Dynamic Web Service Selection", *Proc. Int'l World Wide Web Conf. (WWW 04)*, 2004, pp. 66-73.

[25] L. Taher, H. E. Khatib and R. Basha, "A Framework and QoS Matchmaking Algorithm for Dynamic Web Services Selection," *Proc. 2nd Int'l Conf. on Innovations in Information Technology (IIT)*, 2005.

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

[27] M. Comuzzi and B. Pernici, "A Framework for QoS-Based Web Service Contracting", *ACM Tran. on The Web*, Vol. 3(3), 2009, Article 10.

[28] A. Srivastava, and P. G. Sorenson, "Service Selection Based on Customer Rating of Quality of Service Attributes," *Proc. IEEE Int'l Conf. on Web Services*, 2010, pp.1-8

[29] L. W. Li, L. C. Chun, C. K. Ming, and Y. Muhammad, "Fuzzy Consensus on QoS in Web Services Discovery", *Proc. Int'l Conf. on Advanced Information Networking and Applications (AINA)*, 2006, pp. 791-798.

[30] P. Wang, "QoS-aware Web Services Selection with Intuitionistic Fuzzy Set Under Consumer's Vague Perception", *J. Expert Systems with Applications*, vol. 369(3), Part 1, April 2009, pp. 4460-4466

[31] S. Nepal, W. Sherchan, J. Hunklinger, and A. Bouguettaya, "A Fuzzy Trust Management Framework for Service Web", *Proc. 2010 IEEE Int'l Conf. on Web Services*, 2010, pp. 321-328.

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