A Mechanism for Web Service Selection and Recommendation Based on Multi-QoS Constraints

Shao-chong Li
School of Software
Shanghai Jiao Tong University
Shanghai, P.R.China
lee.shaochong@gmail.com

Hao-peng Chen
School of Software
Shanghai JiaoTong University
Shanghai,P.R.China
chen-hp@sjtu.edu.cn

Xi Chen
School of Software
Shanghai JiaoTong University
Shanghai,P.R.China
april.chenxi@gmail.com

Abstract— **Service-Oriented Architecture (SOA) provides a flexible framework for service composition. In a service market scenario, given a functional description of service, different providers may offer diverse service implementations that match such a functional description, but differ for some QoS attributes. It is increasingly vital to provide a service selection and recommendation mechanisms that best meet the QoS requirements of the service user. Different from most of the existing approaches to service selection, we consider a Web service selection and ranking mechanism with multi-QoS attributes, focusing on simulating degree of consumer satisfaction and hypothesizing consumer preference historical information. Efficient service selection mechanism and heuristic algorithm for consumer preference of multi-QoS are presented in this article and their performances are studied by simulations.**

Keywords-web service; multi-QoS; consumer satisfaction

I.INTRODUCTION

The Serviced-oriented Architecture (SOA) strongly relies on mechanisms for advertising and discovering available services. In this vision, providers offer similar competing services corresponding to a functional description of a service, these offerings can differ significantly in some Quality of Service (QoS) attributes. On the other side, prospective users of service need to dynamically choose the best offerings for their purpose.

Web Service description Language (WSDL) describes Web service mainly addressing communication issues and syntactic description of service interfaces. Besides, current service registries such as Universal Description Discovery and Integration protocol (UDDI [1]) do not enable discovery and selection based on service capabilities and non-functional properties, since they primarily rely on the approach of keyword matching of WSDL information. Using the SOA paradigm to build applications, services can be dynamically selected, enabling system properties like flexibility, adaptability, and reusability. In this context, the key point is to build a mechanism to assure that the advertised service is effectively provided: i) the overall mechanism of service selection and recommendation, ii) the fulfillment of comprehensive QoS constraints, such as application response time and cost, (a comprehensive QoS constraints regards as many aspects, including performance, reliability, scalability, capacity, robustness, exception handling etc), iii) dynamic recommendation approach to adapt user's preference.

Due to current SOA approaches only partially address this comprehensive vision. Therefore, QoS-aware service selection and recommendation become a very active area of research and standardization. In this paper, we propose a reasonable web service mechanism which allows the service discovery based on the multi-QoS constraints. Suppose we have found a set of candidates of target service which are same in the functionality but differ in QoS. With our mechanism, we can get them ordered by the given multi-QoS constraints. We focus on the measureable QoS attributes, such as performance, reliability, availability, and so on. For the other QoS attributes, reputation is a method to solve service selection problem. Reference [2], an earlier work of our research team, has proposed an effective reputation model to rank the services.

The rest of this paper is organized as follows. In Section Ⅱ, we reviews related works in this area. Section Ⅲ describes the framework of our mechanism, and presents how to presume the consumer's preference with to ANN (Artificial Neutral Network). Section Ⅳ provides some simulations to demonstrate our method. In the final section, we conclude this paper and discuss some future works.

II.RELATED WORK

Service selecting and recommending are crucial in SOA. The SOA model enables different service providers (SP) to provide loosely coupled and interoperable services at different Quality of Service (QoS) and cost levels in a number of service domains. It provides a unique opportunity for businesses to dynamically select services that better meet their business and QoS requirements powerfully in a cost-effective manner.

QoS is an important factor for Web service selection when there is more than one service offering the same or similar functionality. Currently, it is known that QoS research mainly includes QoS model design, QoS management [3], QoS discovery [4, 5] and optimal service selection based on QoS [6, 7, 8].

Many solutions which support QoS-based WS selection have existed in this area. For instance, UDDIe[9] has been used with QoS information which is advertised by the provider. In reference [10], it focuses on augmenting web service clients as a means for determining optimal service providers. Reference [11] has implemented a reputation-enhanced service discovering algorithm, which is based on the consumer's performance feedbacks to compute

QoS. In order to meet the needs of both consumers and providers, some researchers addressed one issue of selecting web services by maximizing user's satisfaction expressed as utility functions over QoS attributes [12]. In reference [13], the researchers treated the selection of QoS-driven web service with dynamic composition as a fuzzy constraint satisfaction problem and applied an optimal search approach with adjustments to service composition. Reference [14] advocates using fuzzy set to express the user's QoS preference on a specific QoS criteria and using fuzzy expression to represent the user's trade-off among QoS criteria. Considering a set of services implement the same functionality but differ for the quality parameters, in the work presented in [11], two main approaches have been proposed: local and global optimization. For the global optimization, Reference [15] proposes a global approach for Web services selection and optimization. During the process of composing web services, it is crucial to consider the services' QoS, Reference [16] proposes an approach to combine the same or similar functional services into classes of service which is convenient for selection. Reference [17] also presents a global QoS optimizing and multi-objective web services selection algorithm based on multi-objective ant colony optimization.

These researches advanced the research in QoS-aware service discovery. However, there is no practical and comprehensive research on service recommendation yet. Many problems still need to be addressed. For example:

- In order to satisfy the consumer's preference, the QoS attributes should be consumer-aware which means that they need to be dynamically updated.
- The contribution weight of each QoS attributes to the selecting decision should be determined objectively but not subjectively.

In this paper, to provide an effective multi-QoS-aware service discovery, a new web service selecting and recommending mechanism is proposed. This mechanism can:

- Model the QoS constraints accurately. In this paper, we propose a mathematic approach to model QoS constraints according to the preferences given by the service consumers in the process of selecting web services.
- Dynamically adjust the contribution weights of QoS attributes by self learning. We keep and handle the historical service selecting information for each service consumers to update the contribution weights of QoS attributes. This approach results in the improvement of the effectiveness of service discovery.

Detailed introduction about this model is described in the next sections.

III.FRAMEWORK OF THE SERVICE RANKING AND SELECTING

In this section, we present the model of service selection and recommendation framework, and then introduce the front-end and back-end used. Figure 1 illustrates a high-level architecture view of the framework we consider.

On behalf of the further discussion, we begin by making the assumption as follows:

- There is an existing efficient and effective solution for functional selection. This paper just focuses on the approach to manage the same functional services with different QoS.
- As a dynamic feature of services, the QoS attributes are changed from time to time. This implies that we have to use them as variables to calculate the indexes of services, such as customer's satisfaction, to be convincing.

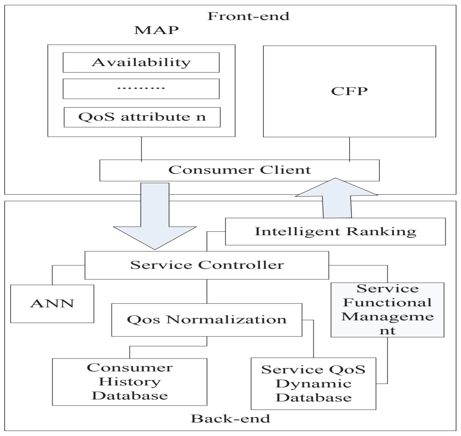

Figure 1. Service selecting mechanism

- The QoS data is feedback by service providers and kept in Service registry.
- We consider measurable QoS attributes such as performance, response time as individual attributes and treat all the immeasurable QoS attributes as a single attribute which is reputation [2].

As is shown in Figure 1, the *Font-end* is a consumer client which is divided into two parts: *MAP* (Multi-QoS Attributes Process Part) and *CFP* (Consumer Functional Process Part). In *MAP*, consumers give their QoS constraints by specifying the degrees of satisfaction of a set of reference points of each QoS attribute. In *CFP*, consumer input their functional requirement. Both of the two parts data will be packed by Front-end and sent to Back-end.

Back-end is composed by multi-components. *Service Controller* can receive consumer data which come from front-end and save data to *Consumer History Database*. It also transfers the result of *Qos normalization* part to *Intelligent Ranking* part to recommend service. *QoS normalization* part unifies the QoS attributes unit to consumer's satisfaction. *Service functional management* can generate a service set which contain the same functional requirements. All the services' QoS values are maintained by *Service QoS dynamic Database*. *ANN* is commanded by *Service Controller* which can accommodate consumer's preference.

Next Sections would illustrate each component in detail.

A. Framework introduction

Our framework provides a multi-selection client for each QoS attributes. Due to various QoS attributes differing in measurement span and meaning, it will be convenient for consumer to compose multi-constraint in a composite constraint. For example, response time and reliability is measured by seconds and errors/month separately. If all of these measures can be measured by a unified unit, it will be helpful for statistics. In our framework, consumer's multi-constraint is formatted as a vector where response time and reliability are normalized by consumer's satisfaction.

The relation between consumer's satisfaction and each QoS attribute value is depicted in a rectangular coordinate. According to such discipline, for each QoS attributes, consumer's selection constraint would form a set which contains disperse coordinate points. All the selected sets are undoubtedly taken as consumers' preference. As a result, our framework sends each selection sets and functional requirements to *back-end*.

After *back-end* has received consumer's data, at first, *Service Functional Manage* part obtains a service set with customer's functional requirements. Each element in service set contains its QoS attributes values which are dynamically managed by *Service QoS Dynamic Database*. And then, according to consumer's preference in *Consumer History Database*, *QoS Normalization* part will normalize each service's QoS attributes values as a vector. At last, *Intelligent Ranking* part will recommend the best service to *front-end* according to each service's vector. In Section C, a detailed description will be given on this mechanism.

If the service recommended by Intelligent Ranking part is not in accord with the consumer's selection, our framework would accommodate the customer's preference automatically through ANN part. ANN part will coordinate relating data with consumer's choice so as to shorten the disparities between the recommended and the actual selection. Consumer's preference will be modified and saved in Consumer History Database. This paper exploits the inverted diffuse method in Neural Network to handle the case of service accommodation. Section D will introduce detailed solution.

B. Front-end

According to Figure 1, suppose we have registered a group of services with the functional requirement. As different service QoS attributes may lead to different results in service selection, we assume that each service consumer's satisfaction has a relation which is defined as follow:

$$fs=\{(fd\,,fc)\} \tag{1}$$

Where fd is denoted as the value of single service's QoS attribute. fc denotes the weight of the consumer's satisfaction of service. For each service selection, for example, when fd (eg.. response time)reach 1s, consumer *a* considers its satisfaction as 99%, while consumer *b* may consider it as 90%. The relation can be described as follows:

$$fs_a = (1s,\ 99\%) \tag{2}$$

$$fs_b = (1s,\ 90\%) \tag{3}$$

We build a data table for each customer as follow:

TABLE I. REFERENCE OF RESPONSE TIME

FD(second)	1	10	20	30	40	50
FS (%)	99	99	81	72	20	10

In the case that we have:

$$fs= (900s,\ 99\%) \tag{4}$$

In this mechanism, we consider the fs as a noise datum for the reason that it should be a bad service when a service's response time reaches 900s. But in this function fs, the consumer's satisfaction is 99%. Single datum cannot reflect the trend of customer's satisfaction as regards the specific *QoS* attributes. According to different requirements, each with different service consumer satisfaction and different *QoS* attributes, all together we have studied 300 cases. After simulating each of these cases, the exponential function is selected as the model to simulate the relation between service customer satisfaction and *QoS* attributes.

In exponential function, when x comes to zero, the value of exponent function is on the verge of maximum infinity. This function form can simulate the response time of QoS attributes exactly. For example, when response time targets zero infinitely, consumer's satisfaction reaches the maximum value. On the other side, when x is approximately infinite, consumer's satisfaction reaches the minimum value correspondingly. So choosing the exponent curve as the above relation formalize style can provide the consumer accurate satisfaction value. Given consumers' satisfaction discipline our exponent curve needs to be a decreasing function. Although the consumers' selected points are dispersive and random, utilizing exponent curve model, we can describe their satisfaction accurately.

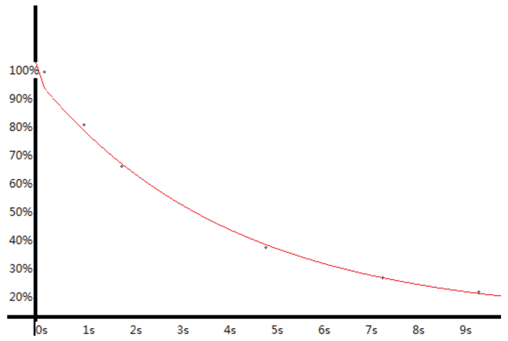

Figure 2. exponent curve of response time

We tackle the simulating problem through transforming it into the exponent curve with the conic fitting method in Figure 2, and the data are from the table Ⅰ.

With the above definition, we define fs as the relation of a single QoS attributes x_i and consumer's satisfaction y_i.

$$fs_i = (x_i,\quad y_i)\ (i=0,1,2,3\ldots\ldots) \tag{5}$$

According to each QoS attribute, the relation Sc is defined as follow:

$$Sc = \{ fs_i \}(i=0,1,2,3\ldots\ldots\ldots) \qquad (6)$$

For each set Sc, we utilize conic fitting method to simulate the model. According to above description, our model is a decreasing function. The function is described as:

$$y = e^{-bx} \qquad (7)$$

For each consumer, constant b would be given different values according to consumer's selected data. In the *front-end* part, we expect to get a formula set for each consumer's selected QoS attributes. So after selecting appropriate values with QoS attributes, we iterate each set Sc to formulize the formula. If correlative QoS attribute data Sc are vacant, we consider that the consumer ignore this QoS attribute.

According to conic fitting method and taking the exponent curve as compute model, we get a normal equation with the Table Ⅰ data. The equation form can be depicted as the following:

$$\begin{cases} \mathrm{cA} + \mathrm{db} = \mathrm{f} \\ \mathrm{bA} + \mathrm{eb} = \mathrm{g} \end{cases} \qquad (8)$$

The values of A, c, d, e, f, g, is calculated by the satisfaction data. Then with the normal equation (8) being solved, the constant b could be calculated. In the end, we get customer's satisfaction with function $y = e^{-bx}$.

Before we describe our algorithm, we define S(sc) as an Sc set that comes from the customer's choice. Our algorithm is shown as follow:

```
1   getFormulaSet( S(sc) )
2   $Set_s \leftarrow S(sc)$
3   $Set_{formula} \leftarrow$ {new set of <s>}
4   for each item in $Set_s$
5   do $Set_{formula} \leftarrow$getformula(item);
6   return $Set_{formula}$
```

Figure 3. Algorithm 1

As is described by above algorithm, input parameter is S(sc) that indicates the consumer's choice. Line 3 states that we create a new empty set to contain the formula. Line 4 to Line 5 iterates every Sc set in set S(sc) to generate the formula and insert into the formula set $\mathrm{Set}_{\mathrm{formula}}$. Line 6 returns the formula set. After these steps, we have a utility to put and generate formula to multiple QoS constraints.

C. Back-end

Due to formula set we have proposed, for each QoS attribute, we get a formula Fb. Taking fc as the satisfaction of consumer, we can get the function as follows:

$$fc_i = Fb(x_i)(i = 0,1,2\ldots.\text{and where each i delegates the single QoS attributes}) \qquad (9)$$

As different service QoS attributes may lead to different results in service selection, we assume that each service has an associated vector:

$$fc(a_1, a_2, a_3, a_4, \ldots\ldots a_i \ldots.)(i=1,2\ldots\ldots.n) \qquad (10)$$

Where n is the number of different QoS classes. The element of this vector delegates the consumer's satisfaction of each QoS attributes, such as response time. Suppose we have an optimal service which contains the best value of QoS attributes. Since we take percentage as the customer satisfaction, the max value should be number 1. Here we take fp as the optimal service:

$$fp(1,1,1,1,\ldots\ldots 1\ldots\ldots) \qquad (11)$$

Suppose we get a group services with the same function requirement. The services have a vector set as follow:

$$\mathrm{Fc} = \{\mathrm{fc_i}\}(\mathrm{i}=1,2,3\ldots\ldots\ldots\ldots) \qquad (12)$$

In order to select the optimal service, we compare the distance between each service's vector and optimal vector.

$$\mathrm{D_i} = \overline{\mathrm{fc_i fp}} \ \ (\mathrm{i}=1,2,3\ldots\ldots.) \qquad (13)$$

According to the formula of space vector, we can calculate the $\mathrm{distance_i}$:

$$\mathrm{D_i} = \sqrt{\sum_{i=1}^{n} |1 - a_i|^2} \ \ (\mathrm{i}=1,2,3\ldots\ldots\ldots\ldots) \qquad (14)$$

In our study, we consider that the further $\mathrm{distance_i}$ is, the worse the service evaluated. From the vector set we consider the optimal service i need to satisfy the following condition:

$$\mathrm{D_{min}} = \mathrm{Min}\{\mathrm{D_i} \ \ (\mathrm{i}=1,2,3\ldots\ldots..)\} \qquad (15)$$

Although we can use such method to rank service, the distance cannot represent the actual situation because in above situation the QoS attributes they selected are merely taken into equal consideration. In this paper, we set weight for each QoS attribute in order to coordinate consumer's requirements. We define $\mathrm{w_i}$ to denote each QoS attribute weight. In the following, the vector distance formula is transformed as:

$$\mathrm{D} = \sqrt{\sum_{i=1}^{n} \frac{(1-a_i)^2 w_i n}{\sum_{i=1}^{n} w_i}} \qquad (16)$$

Section D would discuss in detail on how to adjust the weight value $\mathrm{w_i}$ for consumer's preference.

Based on the computing model introduced in the above two sections, we can define the recommendation algorithm as Algorithm 2.

In line 1, the input element $\mathrm{Set_{service}}$ indicates a group service with the same functional requirement and S (sc) represents customer's choice for each QoS attribute from the front-end. Line 2 and Line 3 initialize operation set and recommendation set. In line 4, we can use the former algorithm to get a formula set for compute customer's satisfaction. And then, we iterate every service in Set_{sc} from line 5 to line 10. Line 7 uses a function which is defined in Algorithm 3 to create a new vector. Line 8 computes the distance between input service's satisfaction and optimal with a function which employing formula (16) for calculation. After each service's distance has been calculated, we sort all the input service by their distance in line 10. The last line returns the result.

```
1   recommend($Set_{service}$ , $S(sc)$)
2   $Set_{sc} \leftarrow Set_{service}$
3   $Set_{recommend}$ ← {new set of <s>}
4   $Set_{formula} \leftarrow getFormulaSet(S(sc))$
5   for each item in $Set_{sc}$
6   do
7     $vector_s = createVector(item.sc, Set_{formula})$
8     item.distance=getDistance($vector_s$);
9         $Set_{recommend} \leftarrow (item)$;
10    Sort $Set_{recommend}$ according to distance
11    return $Set_{recommend}$
```

Figure 4. Algorithm 2

The algorithm of created vector is described as Algorithm 3.

```
1   createVector(sc,$Set_{formula}$ )
2   $Set_s \leftarrow sc$
3   $vector_s \leftarrow new\ vector(n)$
4   for each item in $Set_s$
5   do formula = getFormula( $item_{QoS\ Type}$ , $Set_{formula}$ )
6     if (formula==null)
7       then $vector_s(item_{QoS\ Type}) = 0$
8       else
9       $vector_s(item_{QoS\ Type}) =$
            $getSatisfaction(item_{QoS\ value}, formula)$
10    return $vector_s$
```

Figure 5. Algorithm 3

Input parameters conclude a service's QoS attributes and a formula set. Line 2 and Line 3 initialize the QoS attributes set and a new empty vector. Line 4 to Line 9 iterates each QoS attribute in Set_s to form a vector. For each QoS attribute, $\mathrm{Set}_{formula}$ contains corresponding formula to compute customer's satisfaction. So in line 5, we use function to get the formula with QoS type. If the formula is null, it represents that customer have no constraint on this QoS attribute. As the line 7 represents, the element of this vector is defined as 0. If the formula is not null, we should record the customer's satisfaction with the formula. Line 9 indicates such situation. At last, vector_s would return as the result.

D. Optimizing attribute weight

The above sections describe the web service static ranking method. However, it is possible for the consumer to choose a service in the service rank sequence other than the first one in practice. (In our service registry, we put the service with best QoS in the first place of the rank sequence). For example, a consumer would choose a service other than the first one in the service rank sequence due to its much lower price though its performance and reliability are worse. This fact shows that compared with the congruent ones, unequal weights, particularly the customer-specific weights, are more meaningful for customers.

Furthermore, if the historical records of service selection of customers are available, our framework can recommend the most suitable service to consumers according to the weights self-learned from the historical data in the case that the customers don't provide any clues about the weights when they send the service discovery requirements. Therefore, as the running of the services at runtime, this approach is proposed to handle service ranking and selecting in a dynamic situation.

In our framework, from the above description, the optimal service is selected by the formula (16) which with a weight w_i that indicates customer's preference to each QoS attribute. When consumer's selection cannot follow the framework rank sequence, we need to change the value w_i for QoS attribute to coordinate the preference. In order to adjust the rank sequence, we expect to build a dynamic self-accommodate network to revise the correlate weight. *ANN* (Artificial Neural Network) automatically learns from the former experiences through adjusting the connection weights and can use the knowledge learned before. In this paper, we use the BP algorithm (Back Propagation Algorithm) in neutral network to learn the weight information in our framework, which is widely used as an effective autonomy method.

In the beginning, we suppose each QoS attributes weight has a same value. With the accumulation of the application record, we can adjust the weight value to satisfy customer's preference. In order to obtaining an appropriate weight value, a neural network needs a fixed expecting value and training weight. During the learning process, neutral network shorten the distance between expect value and output value with the adjusting weight.

In this paper, we consider neurons number of Input layer can be confirmed by the QoS attribute number. The output layer has two neurons note which contain the values of customer contentment and discontent. Similarity, we get a vector (x, y) to represent the output. Our framework take the vector (1, 0) as the marking of consumer's acceptation and (0, 1) represents the consumer's rejection. On behalf of making our neutral network to reach stable state, we compute the error between consumer's choice and (1, 0) or (0, 1) and then send it to neutral network. It is a contradictive issue to ensure the number of hidden neurons because there is no terminal verdict at present. In our framework, the hidden layer has three units because such design can solve the core functions problem, and if we add more layers or more units in the hidden layer, we'll find that the precision couldn't be improved remarkably, on the contrary, the training time would be prolonged greatly. According to the principle above, our framework builds the ANN as Figure 6:

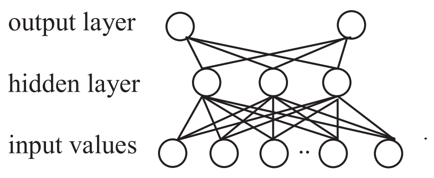

Figure 6. The form of ANN

In *ANN*, the learning process includes forward propagation and back propagation. During forward propagation, input information is managed by hidden layer to arrive at output layer. According to formula (16), we take a service's each *QoS* attribute as an input unit a_i and initialize the weight w_i to a small random number. Input information is transmitted to hidden layer by active formula (18). Here we take w_{ij} to represent the weight of each input unit of hidden layer. The formula is transformed as follow:

$$f_1(a_i)= \sqrt{\sum_{i=1}^{n} \frac{(1-a_i)^2 w_{ij}\, n}{\sum_{i=1}^{n} w_{ij}}} \quad (i= 1,2,3\ldots\ldots n) \tag{17}$$

And then, when the hidden layer enters into output layer, the output unit is activated by the following formula:

$$o_k = \begin{cases} 1-\dfrac{3*\sum_{j=1}^{3} b_j w_{jk}}{\sum_{j=1}^{3} w_{jk}} & \text{if } k=1 \\ \dfrac{3*\sum_{j=1}^{3} b_j w_{jk}}{\sum_{j=1}^{3} w_{jk}} & \text{if } k=2 \end{cases} \tag{18}$$

The value of o_k is the customer's anticipant output. The value o_1 indicates the output satisfy degree of customer's requirement and o_2 indicates the output indicates that the output does not satisfy degree of customer's requirement. If o_1 is three times larger than o_2, we consider this service has satisfied the customer's requirements. On the contrary, if o_2 is three times larger than o_1, we deem the service does not satisfy customer's requirements. In actual situation, if there is contradiction between customer's choice and expected output, the information will be transferred to back propagation process.

At the back propagation phase, feedback path shortens the distance between actual output and expected output by regulating weight of each neuron. Then we repeat iterations in this way for reducing the errors to the permissible error range.

Before we introduce the method of adjusting weight, we define the following variable for neutral network:

- Variableη indicate the learning rate. It is a constant with very small value.
- Variable n_{in} is the dimension of the network input vector.
- Variable n_{out} is the dimension of the network output vector. Here we evaluate it as 2.
- Variable n_{hid} is the number of units in the hidden layer, we evaluate it as 3.
- The input from unit i to unit j is denoted x_{ij}.
- The weight from unit i to unit j is denoted w_{ij}.
- α is a momentum constant which is very small.
- o_u indicates the output of each unit u in the network.
- t_k represents the expect output of each unit

For each network output unit k, we calculate its error term:

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k) \tag{19}$$

For each hidden unit h, we calculate its error δ_h term:

$$\delta_h \leftarrow o_h(1-o_h)\sum_{k\in output} w_{kh}\,\delta_k \tag{20}$$

When output can't satisfy the expect value, we should adjust the weight w_{ij}. Increment is computed as follow:

$$\Delta w_{ij} = \eta\delta_i x_{ij} + \alpha\Delta w_{ij}(n-1) \tag{21}$$

And then the weight w_{ij} is adjusted to:

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \tag{22}$$

Variable Δw_{ij} would be changed to coordinate w_{ij} until output meets the customer's preference. Through this operation, neutral network will continue to make the whole system reach a stable state spontaneously.

IV.SIMULATION

We built a prototype to simulate our computing mechanism. In the prototype, we measure 5 QoS attributes: response time (unit: milliseconds), availability (unit: TTR which means the cost time of modifying an inefficacy service), interoperability analysis (unit: % which means ratio of the errors and the warnings reported), cost of service (unit: cent per service request), reliability (unit: errors/month). According paper [18], we build a registry center to save and manage register service's QoS attributes feedback by the service consumers. We developed the front-end and background-end with Eclipse 3.4.

In our simulation environment, there are three consumers and five sets of services, each of which has eight services which are same in functionality but differ in QoS. We depict the process of service discovery and how the contribution weights of *QoS* attributes are adjusted as follows.

Suppose the consumer A input the multi-QoS constraints from the front-end. The Next Tables show the font-end data.

TABLE II. THE REFERENCE POINTS OF PERFORMANCE

FD(*milliseconds*)	1	10	20	25	30	31	35
FS (%)	98	80	50	48	10	8	5

TABLE III. THE REFERENCE POINTS OF PERFORMANCE

FD(*second*)	1	5	7	10	15	18	20
FS (%)	99	95	89	70	50	30	5

TABLE IV. THE REFERENCE POINTS OF INTEROPERABILITY

FD (%)	1	15	25	32	48	50	80
FS (%)	99	96	88	75	61	30	10

TABLE V. THE REFERENCE POINTS OF COST

FD ($)	9	11	17	19	21	23	27
FS (%)	97	91	80	76	61	22	8

TABLE VI. THE REFERENCE POINTS OF COST

FD (*errors/math*)	1	10	15	20	21	25	30
FS (%)	98	90	79	72	63	25	3

Applying the curve fitting described in algorithm 1 on the data in above tables, we can get the following formula to express the multi-QoS constraints.

TABLE VII. FORMULAE OF QOS ATTRIBUTE

QoS *attribute*	*Formula*
Performance	$y = e^{-0.0293x}$
Availability	$y = e^{-0.0491x}$
Interoperability	$y = e^{-1.2482x}$
Cost	$y = e^{-0.0490x}$
Reliability	$y = e^{-0.0324x}$

All the services we developed are registered in the service registry. Here we take the services A-G which are same in functionality as an example. The QoS of these eight services is shown in Table Ⅷ.

TABLE VIII. QOS ATTRIBUTES OF SERVICES

Service	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
Performance (milliseconds)	3.22	8.32	1.05	16.50	1.78	33.18	15.42	38.76
Availability (second)	8.10	6.19	1.03	1.33	12.99	8.72	2.65	25.34
Interoperability (%)	1.03	9.00	1.00	1.99	33.32	1.00	11.77	83.21
Cost ($)	16.44	14.00	20.00	20.78	12.64	7.01	6.00	29.89
Reliability (errors/math)	11.27	12.12	1.03	8.16	8.53	1.04	5.32	30.71

Applying the QoS data of the eight services into the formula set shown in Table Ⅶ, we can get the QoS vectors of each service'. In terms of formula (18), we get the distances of the services as well as their ranks. The result is shown in Table Ⅸ:

TABLE IX. SIMILARITIES OF RANK

Service	*Vector*	*Distance*	*Rank*
A	(0.91,0.67,0.99,0.45,0.69)	0.7180	4
B	(0.78,0.74,0.89,0.50,0.68)	0.6919	3
C	(0.97,0.95,0.98,0.41,0.97)	0.6283	2
D	(0.62,0.94,0.98,0.36,0.76)	0.7833	7
E	(0.95,0.53,0.66,0.54,0.76)	0.7824	6
F	(0.38,0.65,0.99,0.70,0.96)	0.7705	5
G	(0.64,0.88,0.86,0.75,0.84)	0.5056	1
H	(0.32,0.29,0.35,0.23,0.37)	1.5405	8

Suppose even though the service G is the best one in Table Ⅸ, consumer A selects service C but not service G due to the perfect performance of service C. Then, the ANN will update weights of each QoS attribute to accommodate the customer's preference. In our ANN, we take η as 0.05 and α as 0.1. The results of self-learning on contribution weights of QoS attributes are listed in Table Ⅹ in which each row indicates the result after iteration.

TABLE X. ITERATIVE RECORDS OF CONSUMER A

SetId	*Consumer A*					*Iteration*
	w1	*w2*	*w3*	*w4*	*w5*	
N1	1.00	1.00	1.00	1.00	1.00	1
	1.12	1.00	0.96	0.91	1.00	2
	1.18	0.99	0.92	0.89	1.02	3
	…….	…….	…….	…….	…….	….
	1.28	0.92	0.96	0.74	1.10	100

The last row of Table Ⅹ is the weights at the time of ANN reaching its stable status. With the weights in last row of Table Ⅹ, the rank is updated as Table Ⅺ:

TABLE XI. THE UPDATE SIMILARITIES OF RANK

Service	*Distance*	*Rank*
A	0.7082	3
B	0.7364	4
C	0.6169	1
D	0.9061	5
E	0.9132	6
F	1.1132	7
G	0.7057	2
H	1.9549	8

We simulate the behavior of consumer B and C, and respectively calculate the contribution weights for the both consumers in the same way as above. The results are shown as follows:

TABLE XII. ITERATIVE RECORDS OF CONSUMER B

SetId	*Consumer B*					*Iteration*
	w1	*w2*	*w3*	*w4*	*w5*	
N1	1.00	1.00	1.00	1.00	1.00	1
	0.99	1.00	1.01	0.99	0.85	2
	0.91	0.96	1.01	0.91	0.81	3
	…….	…….	…….	…….	…….	……
	0.01	0.87	2.89	0.56	0.67	100

TABLE XIII. ITERATIVE RECORDS OF CONSUMER C

SetId	*Consumer C*					*Iteration*
	w1	*w2*	*w3*	*w4*	*w5*	
N1	1.00	1.00	1.00	1.00	1.00	1
	1.05	1.23	1.32	1.00	1.01	2
	1.12	1.96	1.21	1.01	1.11	3
	…….	…….	…….	…….	…….	……
	0.23	0.12	3.68	0.55	0.42	100

After the weights have been modified, according to our statistic, it is reported that the error term between consumer's choice and our calculation is only 1.23%. The simulation proved that our mechanism can improve service selection precision.

V. Conclusion and future work

The quality-based selection of web services is an active topic recently. Many researchers have taken a great deal of investigation. The main drawback of current work in dynamic web service selection is the inability to ensure that service recommending algorithm is open, fair and trustworthy. We achieved the dynamic and fair computation of QoS values of web services through active users' feedback and active monitoring. Our selection mechanism concentrates on constraint which contains consumer's preference and multitude attributes will be taken into account. Moreover, QoS constraint can be dynamic changed by ANN to accommodate consumer's preference.

However, in the process of our simulating exponent curve, there still exist errors between our expected value and true value. Our future work will concentrate on how to eliminate the noise datum from the consumer's input data effectively for simulating the exponent curve accurately. Further, it is still a challenge on how to condense reference data of consumer's preference QoS attributes.

Acknowledge

This paper is supported by the National High-Tech Research Development Program of China (863 program) under Grant No. 2007AA01Z139.

References

[1] Tom Bellwood, Luc Clement "Uddi version 3.0.1," *http://uddi.org/pubs/uddi v3.htm* .

[2] Guang Yang, Hao-peng Chen, "An Extensible Computing Model for Reputation Evaluation Based on Objective and Automatic Feedbacks，" *Proceedings of the 2008 International Conference on Advanced Language Processing and Web Information Technology，China ,2008.*

[3] Sharma A, Adarkar H, Sengupta S，"Managing QoS through Prioritization in Web Services," *In Proc.of the* 4^{th} *Inc.Conf.on Web Information Systems Engineering Workshops(WISEW), Rome, 2003,pp.140 -148.*

[4] Shupping Ran, "A Model for Web Services Discovery With QoS, " *ACM SIGecom Exchanges, 2003, 4(1): 1-10.*

[5] Yang Shengwen, Shi Meilin, "A model of Web service discovery with QoS constraints," *Chinese Journal of Computer, 2005, pp.589-594.*

[6] Gerardo Canfora, Massimiliano Di Penta,Raffaele Esposito,et al, "An approach for QoS-aware service composition based on genetic algorithms," *In Pro.of the 2005 conference on Genetic and evolutionary computation (GECCO). USA, Washington DC, 2005, pp. 1069-1075.*

[7] Nizamuddin channa,Li Shanping, Abdul Wasim Shaikh,et al, " Constraint satisfaction in dynamic Web service composition, " *In Pro.of the* 6^{th} *Inc.Workshop on Database and Expert Systems Applications. Copenhagen, Denmark, 200, pp.658 -664.*

[8] Wang Hongbing,Xu Xun,Wang Yifei, "A Multi-Dimensional Framework for Services Selection," *In Proc.for Int.Conf.on Next Generation Web Services Practices(NWeSP). Korea, Seoul, 2005, pp. 441-442.*

[9]A.Shaikhali, O.f.Rana, R.Al-Ali and D.W.Walker, "An Extended Registry for Web Services," *Proceedings of the Service Oriented Computing: Models, Architectures and Applications, 2003.*

[10] Julian Day, Ralph Deters, "Selecting the best web service, " *In Proceeding of the 2004 conference of the Centre for Advanced Studies on Collaborative research. Canada, Markham, Ontario, 2004, pp.293 - 307*

[11]Xu Ziqiang, Martin Patrick, Powley Wendy, Zulkernine Farhana, "Reputation-Enhanced QoSbased Web Services Discovery," *The IEEE International Conference on Web Services(ICWS 2007), IEEE Computer Society, 9-13 July 2007, pp.249-256.*

[12] Liangzhao Zeng, Boualem Benatallah. Etc, "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering 30(5):311-327. 2004(5)*

[13] Yutu Liu, Anne H. Ngu, Liang Z. Zeng, "QoS computation and policing in dynamic web service selection," *In Proceedings of 13th international conference on world wide web, 2004.*

[14] Lin, M., Xie, J., Guo, H, Wang, H, "Solving QoS-driven Web Service Dynamic Composition as Fuzzy Constraint Satisfaction," *IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2005, 9-14.

[15] Ardagna, D., Pernici, B, "Global and Local QoS Constraints Guarantee in Web Service Selection," *IEEE International Conference on Web Services*, 2005, 805–806.

[16] Hongbing Wang, Ping Tong, Phil Thompsoon, PYinsheng Li, "QoS-Based Web Services Selection," Proceedings of the IEEE International Conference e-Business Engineering, pp.617-637.

[17] Fang Qiqing, Peng Xiaoming, Liu Qinghua, Hu Yahui, "A Global QoS Optimizing Web Services Selection Algorithm Based on MOACO for Dynamic Web Service Composition," *Proceedings of the 2009 International Forum on Information Technology and Applications - Volume 01.Chaina,ChengDu,2009.*

[18] Siming Xiong, Haopeng Chen, QMC: "A Service Registry Extension Providing QoS Support," *2009 International Conference on New Trends in Information and Service Science (NISS 2009), China, 2009.7, pp.145-151.*