Web Service Selection for Resolving Conflicting Service Requests

Guosheng Kang, Jianxun Liu, Mingdong Tang
Department of Computer Science and Engineering,
Hunan University of Science and Technology
Xiangtan, 411201, China
{guoshengkang, ljx529}@gmail.com
tangmingdong@ict.ac.cn

Xiaoqing (Frank) Liu, Kenneth K Fletcher
Department of Computer Science
Missouri University of Science & Technology
Rolla, MO 65401, USA
fliu@mst.edu
kkft3c@mail.mst.edu

***Abstract*—Web service selection based on quality of service (QoS) has been a research focus in an environment where many similar web services exist. Current methods of service selection usually focus on a single service request at a time and the selection of a service with the best QoS at the user's own discretion. The selection does not consider multiple requests for the same functional web services. Usually, there are multiple service requests for the same functional web service in practice. In such situations, conflicts occur when too many requesters select the same best web service. This paper aims at solving these conflicts and developing a global optimal service selection method for multiple related service requesters, thereby optimizing service resources and improving performance of the system. It uses Euclidean distance with weights to measure degree of matching of services based on QoS. A 0-1 integral programming model for maximizing the sum of matching degree is created and consequently, a global optimal service selection algorithm is developed. The model, together with a universal and feasible optimal service selection algorithm, is implemented for global optimal service selection for multiple requesters (GOSSMR). Furthermore, to enhance its efficiency, Skyline GOSSMR is proposed. Time complexity of the algorithms is analyzed. We evaluate performance of the algorithms and the system through simulations. The simulation results demonstrate that they are more effective than existing ones.**

Keywords- web service selection; matching degree; Euclidean distance; 0-1 integral programming; Skyline

I. INTRODUCTION

As a new web paradigm, web services (WS) are being rapidly developed in recent years and as a result, an increasing number of web services are emerging on the Internet. How to select a feasible single or composite web service (CWS) to meet the demands of different users both functionally and non-functionally has become a popular research area. Existing work focuses on a single request for a web service (single or composite) [2-14], or multiple requests for the same functional service waiting in a queue at that moment [15, 16]. Depending on the current request, the web service with the best quality of service (QoS) is selected. Usually, not only are there multiple requests, but also multiple optional web services. Owing to this, conflicts among these requests arise when a number of users request for the same web service. In addition, the QoS of the web service in question drops because of overload, since its service ability is limited. Actually, it is unnecessary for too many requests to select the same best web service, since they have different QoS demands. Besides, the service with best QoS may also not be the most matched service to the requests. However, users never know which service is busy at that moment and which is not. There is therefore the need for a system to assist users as to the web service selection in order to make maximum requests met and ensure that the services do not overload at the same time. This paper focuses on the atomic web service selection, since composite web services are just considered as complex ones, which is transparent to users. We consider web service selection under an environment of multiple requests for the same functional web services and multiple similar functional optional web services. To solve the conflict, a web service selection framework based on QoS is proposed and the corresponding global optimal service selection model and algorithm for multiple requests are presented.

The contributions of this research are as follows: (1) According to the practical problem of web service selection, the paper analyses the effectiveness of web service selection and presents an inclusion of criteria that both requesters and providers should meet; (2) A web service selection framework based on QoS is proposed and an aggregation model for web service QoS is given. A method of quantizing reputation grades is given as well; and (3) A global optimal web service selection model for multiple conflicting service requests based on QoS is proposed. This avoids the problem of improper load balancing and enhances the performance of the whole system. To solve the conflict issue effectively, a Global Optimal Service Selection for Multiple Requesters (GOSSMR) algorithm together with the modified Skyline GOSSMR algorithm is introduced.

The rest of the paper is organized as follows. Section II presents some related work. Section III provides a motivated example of web service selection for multiple conflicting service requests. In Section IV, we present the web service selection framework based on QoS. Section V provides a QoS aggregation model for web services, and quantizes the reputation grades by a method of membership degree in fuzzy mathematics. Section VI standardizes the values of QoS, calculates preferences and gives the definition of matching degree and its computation method. Furthermore, a global optimal web service selection model based on matching degree and 0-1 integral programming, and the corresponding GOSSMR algorithm are proposed. In this section, time complexity of the algorithm is discussed and a modified Skyline GOSSMR algorithm is proposed.

Simulations of the model are done in Section VII and the paper is finally concluded in Section VIII.

II. Related work

In 2003 Ran [1] proposed a new web service discovery model in which the functional and non-functional requirements are taken into account for service discovery. This resulted in the study of web service based on QoS to become a wide research area. Zeng et al [2] proposed that multiple criteria (e.g., price, duration, and reliability) should be considered in service composition. It should take into account global constraints and preferences set by the user (e.g., budget constraints). Their work proposed the QoS aggregation of web service based on weighted summation of QoS factors and the aggregation functions for computing QoS of execution plans. Service selection in composite service can be formulated as an optimization problem which can be solved using linear programming methods. Furthermore, Zeng et al [3] present a middleware platform for composition in a way that maximizes user satisfaction expressed as utility functions over QoS attributes. A local (task level) selection of services and a global allocation of tasks to services using integer programming were given. However, these approaches need improvement due to their low efficiencies. Another drawback to their approach is the assumption that QoS data have been obtained. Liu, Anne, Ngu, and Zeng [4] proposed a QoS computation and policing in dynamic web services selection. By ranking the web services QoS, users can obtain their best needed services. Maximilien and Singh [5] proposed an agent-based architecture for autonomic web service selection. Their work in [6] included service trust in the selection. To obtain service quality and trustworthiness, Maximilien and Singh [7] gave a framework model and ontology for dynamic web service selection. Yu and Lin [8, 9] proposed a service selection model for web services with end-to-end QoS constraints by maximizing an application specific utility function under the end-to-end QoS constraints and to solve the model, algorithms based on Multiple Choice Knapsack Problem (MCKP) and Multi-dimension Multi-choice 0-1 Knapsack Problem (MMKP) were presented. However, these algorithms may be still not efficient enough in some real time environments. Danilo and Barbara [10] modeled the service composition problem as a mixed integer linear problem where the local and global constraints can both be specified, which proposes the formulation of the optimization problem as a global optimization, not optimizing each possible execution path as in previous approaches. Further, a local and global QoS in web service composition were discussed by Alrifai and Risse [11], and Tang et al [12] to improve efficiency in service composition. Alrifai, and Risse [11] use mixed integer programming to find the optimal decomposition of global QoS constraints into local constraints. Then the distributed local selection is employed to find the best web services that satisfy the local constraints. Tang et al [12] proposed a heuristic service composition method, named local optimization and enumeration method. It aims at filtering the candidates of each task to a small number of promising ones by local selection, and then enumerates all the composite solutions to pursue a near-to-optimal one. Zheng, Ma, Lyu, and King [13] and Cheng et al [14] aim at recommending web service for users. However, these existing research works on service selection only focus on a single service request (or a service node in CWS) requesting a type of functional web service. Some aim at finding an efficient algorithm to solve their proposed CWS model. In sum, the existing work does not consider multiple service requests for the same functional service at a time. Similarly, Shahand et al [15] and Dyachuk and Deters [16] considered multiple requests waiting in a queue. Actually, multiple services can be distributed to multiple requests, since different requests have different QoS demands, thereby avoiding the overload of web services and improving the performance of the whole system, which will be mentioned later in our work and that is also the highlight in this paper.

III. Motivated example

As is mostly the case, there are multiple requests for the same functional web service at a particular point of time. This may cause conflicts among the requests. For example, during festivals or holidays, many people reserve tickets online using web services and there are multiple ticket reservation systems for users to select from. Though users' response of time demands for web services may vary, they are unable to select the best ticket reservation system at their discretion. Therefore, if too many users select the same best system, there will be conflict since the service ability of the system is limited. Actually, it is unnecessary for all of them to select the best web service since they have different QoS demands. This scenario of multiple requests for the same functional service and multiple optional services is common especially now that web services are being popularized. In general the best ticket reservation system at users' discretion would be selected. This arouses competition for this web service and also increases its request load thereby reducing its reputation. Conversely, those web services that actually meet users' needs may not be requested at all as users always select the best web service even though they do not need such good service. For instance, web services with high QoS may be requested and selected by users that do not need a high QoS requirement. In this case, requests from users who actually need the high QoS cannot be met because the Web service with high QoS is unavailable. This results in long wait time for such users and causes wasted resources, decreasing the QoS of the entire system or network.

Therefore in order to ensure resources are not wasted and that users receive only the needed web service, global optimal selection policy is applied. It aims to maximize matching degree between requests and web services selected under an environment of multiple requests and web services. That is to say, a web service should accept the suitable request according to its service ability; a requester should not just select the best web service, but to make most requests to be met; and effective load balancing for optimization should be achieved.

IV. WEB SERVICE SELECTION FRAMEWORK

QoS is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow. In web environment, the demand of requesters always changes, which reflects their diversified QoS demands. At the same time, QoS of web services changes dynamically, and its reputation also becomes unpredictable. Therefore, user preferences should be acquired according to the requests' demands. Feedback from users on the effectiveness of a web service must be gathered frequently and with this information, the QoS data can be updated. By doing so, the reputation of a web service can be calculated. Based on such information, web service optimal selection for multiple conflicting requests can be achieved.

The web service selection framework includes three main roles: requesters, web services and the selector which takes charge of making the global optimal service selection policy. Our system has searched approximately 6,000 web services from the Internet and many of them have similar functions. We named the entire framework web service supermarket. Synonymous to a typical supermarket, web services or composite web services are the products and users are the customers. The difference is that the system helps customers to pick products according to their demands instead of picking products at their own discretion. Our work in this paper is based on the web service supermarket. Fig. 1 shows the web service selection framework in our web service supermarket based on QoS. The selector provides a bridge for requests and web services. In that way, the system can offer as many satisfied web services as possible for service requesters and keep their load balanced.

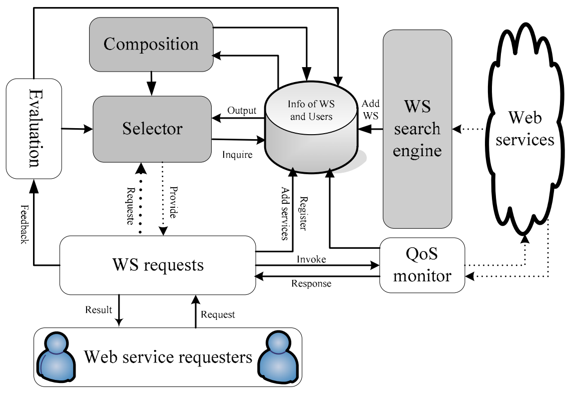

Figure 1. Web service selection framework based on QoS

The database in the framework stores the information pertaining to web services and users. The web service search engine searches web services from the Internet and adds them into the database. Also, users can add web services into the database manually. The composition component is to transform atomic web services into a composite web service. By doing so, various web services can be offered to users. QoS monitor component monitors QoS data of web services during execution time. After invoking finished, users can log in the system anytime to evaluate the Web services that they invoked before. The evaluation component will collect feedbacks from users and send them to the database or the selector for further calculation. The selector develops the global optimal service selection policy. More specifically, the selector includes QoS computing, QoS normalization, QoS quantization, and user preference computing. Based on the information about QoS of web services and users' preferences gathered by these four components, global optimal service selection for multiple conflicting service requests can be achieved. The web service selector is illustrated in Fig. 2.

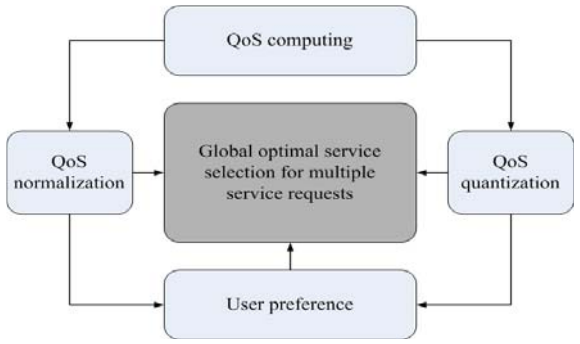
QoS computing
QoS normalization
Global optimal service selection for multiple service requests
QoS quantization
User preference

Figure 2. Web service selector

V. QOS AGGREGATION MODEL OF WEB SERVICE

QoS of a web service is collected and calculated by the system mentioned in Section IV in the process of practical application. Different QoS attributes may have different computation methods and importance. When giving different weights to every quality component, the freshness of QoS should be ensured. A Web service QoS is calculated as shown in (1).

$$QoS(\mathrm{ws}) = \sum_{i=1}^{n} \omega_i q_i \ , \ \sum_{i=1}^{n} \omega_i = 1 \qquad (1)$$

Where q_i is one of QoS attributes of a web service, *ws*, including cost (cost for using the web service), response time (the time interval between when a user requests the service and when the user receives the response), service availability (the probability that a service is available), service reliability (the probability that a requester is handled correctly within the expected time), and service reputation (the mean of evaluations which are given by all the requesters). ω_i is the weight of QoS attribute q_i, which is calculated from QoS demands. Note that since the evaluation grades are fuzzy, the set of fuzzy evaluations may as well be considered as {very good, good, general, and bad}. Their corresponding numerical values are {5, 4, 3, and 2}. It is easily known that requesters are more sensitive to dissatisfaction than satisfaction, i.e., requesters may complain much for a decreased reputation grade but are not satisfied for a same margin increased reputation grade. Hence, based on this fact, the partial large Cauchy distribution membership function to quantize the grades is used (see (2)).

$$f(x)=\begin{cases}[1+\alpha(x-\beta)^{-2}]^{-1}, & 1\le x\le 3\\ a\ln x+b, & 3\le x\le 5\end{cases} \quad (2)$$

Where α, β, *a*, and *b* are undetermined constants. Let *f*(5) = 1, *f* (3) = 0.8, and *f* (1) = 0.01, then the quantization value of fuzzy evaluation set is {1, 0.9126, 0.8, 0.5245}. After quantization, the reputation of the web service is the mean of all the quantization values. Because the reputation is based on the whole evaluation of a web service, Equation (1) is revised to incorporate this complete evaluation (see (3)).

$$QoS(\mathrm{ws})=(1\text{-}\alpha)\sum_{i=1}^{4}\omega_i q_i+\alpha\cdot\mathrm{rep}\,,\ \sum_{i=1}^{4}\omega_i=1, 0\le\alpha\le 1 \quad (3)$$

Where *rep* is the reputation, *α* indicates how much users value the web service reputation compared to the synthesized QoS of other attributes. This is determined by what degree users accept the reputation. For example, when α=1/2, that means the reputation has the equal importance with the synthesized QoS of other attributes.

VI. GLOBAL OPTIMAL WEB SERVICE SELECTION

Web service selection for multiple requests is essentially an optimal problem about selecting matched web services for all the requests to improve the performance of system and the QoS of web services. The global optimal web service selection is recommended to deal with the issue of optimal selection under an environment with multiple requests as well as web services.

A. QoS Standardization and Preference Computation

With reference to the previously mentioned QoS computing model for web services in Section IV, they are considered as five-dimension vectors. Let a web service be ws_i, then QoS of ws_i is $P_i= (P_{i1}, P_{i2}, P_{i3}, P_{i4}, rep_i)$ and let a request be r_i, then QoS demand of r_i is $R_i= (q_{i1}, q_{i2}, q_{i3}, q_{i4}, rep_i)$. To eliminate the difference among measurement methods of different QoS attributes, QoS data should be unitarily processed to [0, 1]. This implies that the higher the value, the higher the quality. Employing the maximum difference normalization method [10] for standardizing all QoS attributes, non-dimensional attributes are yielded. To be more specific, positive criteria, i.e., quality attributes such that the higher the value the higher the quality, and negative criteria, i.e., quality attributes such that the higher the value the lower the quality, are scaled in different ways, as defined in (4) and (5) respectively:

$$v_i=\begin{cases}\dfrac{q_{\max}-q_i}{q_{\max}-q_{\min}}, if q_{\max}-q_{\min}\ne 0\\ 1, \qquad if q_{\max}-q_{\min}=0\end{cases} \quad (4)$$

$$v_i=\begin{cases}\dfrac{q_i-q_{\max}}{q_{\max}-q_{\min}}, if q_{\max}-q_{\min}\ne 0\\ 1, \qquad if q_{\max}-q_{\min}=0\end{cases} \quad (5)$$

Where $q_{\min}=\min\{q_{k,j}\}$ and $q_{\max}=\max\{q_{k,j}\}$ are the minimum and maximum values respectively for the attribute dimension *j*. Therefore, after normalization, the preference vector $W_i= (w_{i1}, w_{i2}, w_{i3}, w_{i4})$ of r_i can be easily calculated by (6).

$$w_{jk}=q_{jk}\Big/\sum_{\mathrm{k}=1}^{4}q_{jk} \quad (6)$$

B. Computational Model of Matching Degree of Web Services

With reference to the thought of vector classification methods, a set of five-dimension vectors can be actually viewed as a multiple space where each vector is a point. Therefore, the problem of web service matching is basically distance measurement among objects. This can be solved by extending the traditional Euclidean distance measurement method to add weights in order to measure the similarity between two vectors. The lower the distance is calculated, the higher the degree of matching between a request and a web service is. Equation (7) calculates the distance $dis(ws_i,r_j)$ between ws_i and r_j. This kind of matching degree meets not only the quantitative demand but also the preference demand.

$$dis(ws_i,r_j)=\sqrt{(1-\alpha)\sum_{k=1}^{4}w_{jk}(p_{ik}-q_{jk})^2+\alpha(rep_i-rep_j)^2}, \sum_{k=1}^{4}w_{jk}=1, 0\le\alpha\le 1 \quad (7)$$

To use (7), a decision must be made in advance to distinguish between surplus and deficiency in numerical value of a matching degree. Let

$$A=(1-\alpha)\sum_{k=1}^{4}w_{jk}(p_{ik}-q_{jk})+\alpha(rep_i-rep_j)$$

If $A<0$, then the web service does not completely meet the demand. Hence $dis(ws_i,r_j)$ =Max, where Max is a relatively large number. Otherwise, $dis(ws_i,r_j)$ is calculated according to (7).

C. Model and Algorithm for Global Optimal Service Selection

Let *N* be the number of web services available and *M* the number of requests. In order to meet the maximum request demands and optimize the web service resource deployment, a selection policy should be developed. This will make the matching degree maximum (i.e., minimize the whole sum of distances). The result of web service selection policy is a 0-1 matrix *X*, where x_{ij} is the element in *X* as shown in (8).

$$x_{ij} = \begin{cases} 1 & \mathrm{ws}_i \ \text{for} \ \mathrm{r}_j \\ 0 & \mathrm{ws}_i \ \text{not for} \ \mathrm{r}_j \end{cases} \quad (8)$$

Taking the minimum value of the sum of all distances as the objective function, a policy of web service selection for each request is made. Therefore, the global optimal service selection model can be expressed by (9) and (10).

$$\min = \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} dis(\mathrm{ws}_i, \mathrm{r}_j) \quad (9)$$

$$s.t. \begin{cases} 0 \le \sum_{j=1}^{M} x_{ij} \le T(\mathrm{ws}_i) - C(\mathrm{ws}_i) & i = 1, \cdots, N \\ \sum_{i=1}^{N} x_{ij} = 1 & j = 1, \cdots, M \end{cases} \quad (10)$$

Where $T(\mathrm{ws}_i)$ indicates that $\mathrm{w}s_i$ can serve at most $T(ws_i)$ requesters at the same time. And $C(ws_i)$ indicates that $\mathrm{w}s_i$ is serving $C(ws_i)$ requesters at the present, i.e., considering that some web services can be executed in multiple threads. However, under the situation of $\sum_{i=1}^{N}(T(ws_i) - C(ws_i)) \ge M$, some of the requests have to wait to be served later.

After the values of x_{ij} are calculated, the matrix X becomes the policy for web service optimal selection. The GOSSMR algorithm based on the above model is shown in Fig. 3.

```
Algorithm GOSSMR
Input: QoS  // QoS of web service and QoS demand of request
Output: X  //Policy of global optimal web service selection
1. rep=f(Rep);  // quantization of reputation grades
2. standardization(QoS);  // standardize the values of QoS
3. W=preference (QoS_ r);  //calculate preferences with (6)
4. for i=1:N, j=1:M  // N requests and M Web services
5.   if A<0
6.     dis=Max;
7.     else  dis=distance(QoS_ws_i, QoS_ r_j);  //calculate distances with (7)
8.   end if
9. end for
10. X=LP(DIS); // solve 0-1 integral linear programming model
```

Figure 3. The GOSSMR Algorithm

The complexity of GOSSMR algorithm is $O(N!/(N-M)!)$. However, the 0-1 integral programming problem is NP-hard. For large systems, it will be very difficult to always find the optimal solution. In a real problem situation with N web services and M requests where $N>>M$, some web services with lower QoS could be abandoned before the algorithm is applied. This can highly enhance the efficiency of the algorithm by reducing the number of optional web services to form a reduced service set named the Skyline service set. This Skyline service set constitutes the services which are not dominated by any other services. In this case, only services in the Skyline service set will be chosen. The concept of dominance and Skyline service set are explained in detail by Alrifai, Skoutas, and Risse [18]. When there are still not enough web services in the Skyline service set for selection, then additional Skyline service set could be generated among the rest services until there are enough services for selection. Fig. 4 is an example where an additional Skyline service set was added in a 2-dimension space.

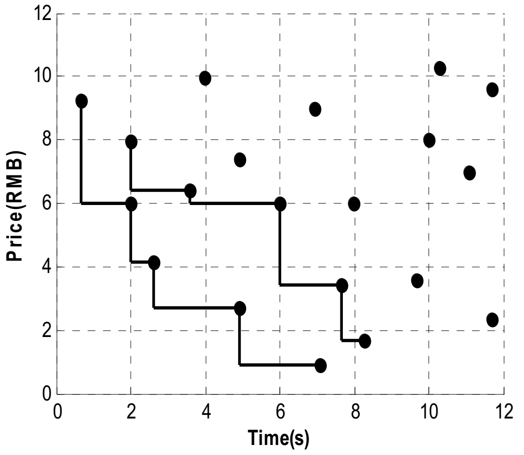

Figure 4. An example of solving Skyline service set twice

The Skyline GOSSMR algorithm, an extension of the GOSSMR algorithm to include these modifications, is to enhance the feasibility of the model. Fig. 5 gives the description of the Skyline GOSSMR algorithm.

```
Algorithm Skyline GOSSMR
Input: QoS  // QoS of web service and QoS demand of request
Output: X  // Policy of web services global optimal selection
1. rep=f(Rep);  // quantization of reputation grade
2. standardization(QoS);  // standardize the value of QoS
3. if N>=k*M   // that is N>>M
4.    skyline(QoS_ws); //solve the Skyline service sets
5.    update(ws);  // update the set of web services
6. end if
7. W=preference (QoS_ r);  //calculate preferences with (6)
8. for i=1:N, j=1:M  // N requests and M Web services
9.   if A<0
10.     dis=Max;
11.     else  dis=distance(QoS_ws_i, QoS_ r_j); //calculate distances with (7)
12.   end if
13. end for
14. X=LP(DIS); //solve 0-1 integral linear programming model
```

Figure 5. The Skyline GOSSMR Algorithm

VII. Simulations and Analysis

In this section, a computational example is given to show the effectiveness of the model. Let N=5, M=3, where N is the number of web services and M the number of requests. For the sake of simplicity, we assume that each web service can only serve one more request in this example.

Then the QoS and reputation model of web services in this environment is shown in Tab. I. The QoS and reputation demand model of requests is also shown in Tab. II.

TABLE I. QoS AND REPUTATION MODEL OF WEB SERVICES

	Time	Cost	Availability	Reliability	Reputation
ws$_1$	10	500	1	0.9	very good
ws$_2$	15	100	0.8	0.7	general
ws$_3$	5	200	0.6	0.8	bad
ws$_4$	20	300	0.9	1	very good
ws$_5$	15	200	0.7	0.6	general

TABLE II. QoS AND REPUTATION DEMAND MODEL OF REQUESTS

	Time	Cost	Availability	Reliability	Reputation
r$_1$	20	250	1	0.9	general
r$_2$	20	200	0.8	0.7	very good
r$_3$	15	300	0.7	0.8	good

After QoS is normalized, preferences of requests are calculated and reputation grades are quantized, let $\alpha = 1/2$, and *Max*=10. According the model and algorithm mentioned in Section VI, the result is calculated as follows: $x_{12} = x_{23} = x_{41} = 1$, the rest of $x_{ij} = 0$, and their corresponding matching degrees are $dis(\mathrm{ws}_1, r_2) = 10$, $dis(\mathrm{ws}_2, r_3) = 0.31$ and $dis(\mathrm{ws}_4, r_1) = 0.27$ respectively. It can be seen from the above result that under the condition of making maximum service requests to be met, request r_2 fails. When this happens, the system may prompt the user to decrease its QoS demand in order to be served or wait for the next selection round. From the above example, it is clear that the result is just the best matching web services for r_1, r_3 under the objective function, which proves the effectiveness of the model.

By the above example, we can know how the global optimal web service selection for multiple conflicting requests can be achieved. Next, we give some large-scale simulations and evaluate the performances using our global optimal web service selection algorithm for multiple conflicting service requests as well as with the classical web service selection algorithm based on weighted summation of QoS factors. We also evaluate the performance based on the way of selecting the most matched web service for each request not considering whether the selected services are overloaded or not. Let *N*=50 and 100, *M*=80 where *N* is the web service number and *M* is the service request number. Also, the QoS data are generated by random functions in a certain range. However, some QoS data with short time but high price will be generated according to certain probability. Tab. III, IV and V show the results from the simulations. In the tables, *SA* is the current web service ability, which is the number of requests it can still serve at the present. If the number of requests exceeds the web service ability, the service will be overloaded. Service ability for each web service is generated by a random function in the range of [1, 5]. *SN* is the number of requests which select the service. *DN* indicates the delayed number of web service requests, that is, these service requests have to wait to be served. Here therefore *DN*=*SN*-*SA*, take *ws$_6$* in Tab. III for example, 27=34-7, *ws$_6$* can still serve 7 requests at the present, but 34 requests select *ws$_6$*, so 27 requests have to wait to be served. So when a web service delays, the corresponding user might give a "bad" reputation evaluation to the service. *TT* and *TC* are the total time and the total cost respectively for the system to finish all its current requests.

TABLE III. PERFORMANCE IN CLASSICAL WEB SERVICE SELECTION ALGORITHM (REQUEST NUMBER=80)

	Classic web service selection algorithm based on weighted summation of QoS factors										
	Web Service Number=50						**Web Service Number=100**				
WS	**SA**	**SN**	**DN**	**TT**	**TC**	**WS**	**SA**	**SN**	**DN**	**TT**	**TC**
ws$_1$	7	0	0	20	31	**ws$_1$**	1	0	0	7	304
ws$_2$	8	0	0	16	120	**ws$_2$**	4	0	0	18	73
ws$_3$	10	0	0	18	69	**ws$_3$**	2	0	0	16	106
ws$_4$	7	0	0	14	212	**ws$_4$**	8	0	0	4	396
ws$_5$	8	0	0	14	187	**ws$_5$**	8	0	0	15	142
…	…	…	…	…	…	**…**	…	…	…	…	…
ws$_6$	7	34	27	11	215	**ws$_7$**	8	7	0	16	164
…	…	…	…	…	…	**…**	…	…	…	…	…
ws$_{31}$	7	3	0	8	258	**ws$_{34}$**	10	0	0	15	137
ws$_{32}$	6	25	19	15	108	**ws$_{35}$**	5	16	11	17	67
…	…	…	…	…	…	**…**	…	…	…	…	…
ws$_{43}$	8	18	10	2	436	**ws$_{68}$**	6	57	51	1	445
…	…	…	…	…	…	**…**	…	…	…	…	…
ws$_{50}$	9	0	0	11	263	**ws$_{100}$**	1	0	0	5	362
Overall	299	80	56	75	18632	**Overall**	561	80	62	68	27585

TABLE IV. PERFORMANCE BY SELECTING THE MOST MATCHED WEB SERVICE FOR EACH REQUEST (REQUEST NUMBER=80)

	Select the most matched Web service for each request										
	Web Service Number=50						**Web Service Number=100**				
WS	**SA**	**SN**	**DN**	**TT**	**TC**	**WS**	**SA**	**SN**	**DN**	**TT**	**TC**
WS_1	7	0	0	20	31	**WS_1**	1	0	0	7	304
WS_2	8	3	0	16	120	**WS_2**	4	0	0	18	73
WS_3	10	0	0	18	69	**WS_3**	2	1	0	16	106
WS_4	7	0	0	14	212	**WS_4**	8	0	0	4	396
WS_5	8	0	0	14	187	**WS_5**	8	0	0	15	142
…	…	…	…	…	…	**…**	…	…	…	…	…
WS_{50}	9	5	0	11	263	**WS_{100}**	1	0	0	5	362
Overall	299	80	22	45	17904	**Overall**	561	80	28	64	18745

TABLE V. PERFORMANCE IN GLOBAL OPTIMAL WEB SERVICE SELECTION (REQUEST NUMBER=80)

	Global optimal Web service selection algorithm for multiple service requests										
	Web Service Number=50						**Web Service Number=100**				
WS	**SA**	**SN**	**DN**	**TT**	**TC**	**WS**	**SA**	**SN**	**DN**	**TT**	**TC**
ws_1	7	4	0	20	31	**ws_1**	1	0	0	7	304
ws_2	8	6	0	16	120	**ws_2**	4	1	0	18	73
ws_3	10	2	0	18	69	**ws_3**	2	2	0	16	106
ws_4	7	2	0	14	212	**ws_4**	8	0	0	4	396
ws_5	8	4	0	14	187	**ws_5**	8	1	0	15	142
…	…	…	…	…	…	**…**	…	…	…	…	…
ws_{50}	9	5	0	11	263	**ws_{100}**	1	0	0	5	362
Overall	299	80	0	20	18336	**Overall**	561	80	0	20	18340

From the tables above, in GOSSMR algorithm, requests do not need to wait and therefore recorded the least total time. Also, no service was overloaded during the whole process, which means the reputation of services is improved. In contrast, only 3 or 4 good web services are selected in the classical service selection algorithm as shown in the Tab. III and nearly all selected web services were overloaded. In the Tab. IV, more web services were selected than in the Tab. III, but less than in Tab. V, which implies some of the selected web services were overloaded. The reputation of services which were overloaded may drop because some requests need to wait and as a result users might evaluate them as "bad". However, the total cost in Tab. V may be a little more than that of Tab. IV when the number of requests exceeds the total service ability of all most matched web services. But this is not always the case since cost is only one of the factors to calculate the matching degree. As shown in Tab. III, when the number of services got to 100, the total time increased but the total cost decreased. This is because requests are more likely to select their best needed services according to their QoS demands when optional services become more but the cost of web services are usually high due to the best QoS. Conversely, in Tab. IV and V, requests are more likely to be offered matched services according to their QoS demands when optional services become more. Nevertheless, some better services may be left to be offered to later requests which have higher QoS demands. In this way, service resources will not be wasted, though total time and cost may increase when optional services increase sometimes. In sum, the overall performance of the system can be improved by GOSSMR algorithm. This is obvious especially when every service can only serve one request at a time. Also, from the perspective of users, the total response time is much less than in the previous.

We perform a simulation to compare the execution efficiency of GOSSMR with that of Skyline GOSSMR. Again, assuming an environment of N web services and M requests let M=10, with 6 independent values of N being 10, 50, 100, 150, 200, 250, and 300 respectively. These values serve as inputs for the algorithms and the results are shown by CPU time-cost in Fig. 6. It can be seen from the graph that when M=10, CPU time costs is the same for both algorithms. This is because when $N<k*M$, there is no need to calculate Skyline service set. For this experiment, in Skyline GOSSMR we let k=5. While the number of optional web services increase, CPU time-costs of both algorithms increase. But the increase with the GOSSMR algorithm is much faster than that of Skyline GOSSMR. Figure 6 also illustrates that the CPU time-cost of GOSSMR algorithm is constantly higher than that of Skyline GOSSMR algorithm. At the mean time, the disparity between the two computation times widen as M increases. The CPU time cost of Skyline GOSSMR algorithm also increases with the increase of M especially when generating Skyline repeatedly.

From the above observation and analysis, it is obvious that compared with GOSSMR, Skyline GOSSMR is much more efficient and feasible.

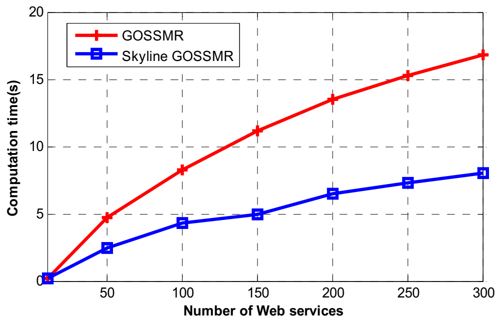

Figure 6. Computational time of GOSSMR VS Skyline GOSSMR

VIII. Conclusion

The problem of global optimal web service selection in web environment with conflicting service requests is addressed. By using Euclidean distance with weights to calculate the matching degree between a request and a web service, a global optimal web service selection model for multiple requests has been developed based on 0-1 integral programming. An effective algorithm based on the model is proposed. Simulations showed that the proposed algorithm has better performance. It allows for maximum service requests to be met and ensures that resources are used effectively and the performance of system is good. As a result, the total time for service execution time is improved a lot and overload is avoided or mitigated.

Acknowledgment

The work was supported by National Natural Science Foundation of China under grant No.90818004, Program for New Century Excellent Talents in University under grant No.NCET-10-0140, and Foundation of Hunan Educational Committee under grant No.09K085.

References

[1] S. Ran, "A model for web services discovery with QoS," ACM SIGecom Exchanges, vol. 4, pp. 1-10, 2003.

[2] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q. Z. Sheng, “Quality driven web services composition,” In Proceedings of the International World Wide Web Conference, May 2003, pp. 411-421.

[3] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, "QoS-aware middleware for web services composition," Software Engineering, IEEE Transactions, vol. 30, pp. 311-327, 2004.

[4] Y. Liu, H.Anne, H. Ngu, L. Zeng, “ QoS computation and policing in dynamic web service selection,” In Proceedings of the International World Wide Web Conference, May 2004, pp. 66-73,.

[5] E. M. Maximilien and M. P. Singh, "Agent-based architecture for autonomic web service selection," Journal on Web Semantics, pp. 261-279, 2003.

[6] E. M. Maximilien and M. P. Singh, "Toward autonomic web services trust and selection," ACM, Nov. 2004, pp. 212-221.

[7] E. M. Maximilien and M. P. Singh, "A framework and ontology for dynamic web services selection," Internet Computing, IEEE, vol. 8, pp. 84-93, 2004.

[8] T. Yu and K. J. Lin, "Service selection algorithms for Web services with end-to-end QoS constraints," Information Systems and E-Business Management, vol. 3, pp. 103-126, 2005.

[9] T. Yu, , Y. Zhang and K.J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," ACM Transactions on the Web (TWEB), vol. 1, pp.1- 6, 2007.

[10] D. Ardagna and B. Pernici, "Global and local QoS guarantee in web service selection," Third Internation Conference on Business Process Management, Sep. 2006, pp. 32-46.

[11] M. Alrifai, T. Risse, “Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition,” In Proceedings of the International World Wide Web Conference, Apr. 2009, pp. 881-890.

[12] L. Qi, Y. Tang, W. Dou, J. Chen, editors, “Combining Local Optimization and Enumeration for QoS-aware Web Service Composition,” IEEE International Conference on Web Service, 2010, pp. 31-44.

[13] Z. Zheng, H. Ma, M. R. Lyu, and I. King, “WSRec: A Collaborative Filtering Based Web Service Recommender System,” IEEE International Conference on Web Services, 2009, pp. 437-444.

[14] X. Chen, X. Liu, Z. Huang, and H. Sun, “RegionKNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation,” IEEE International Conference on Web Services, 2009, pp. 9-16.

[15] S. Shahand, S. J. Turner, W. Cai, H. Khademi, "DynaSched: a dynamic Web service scheduling and deployment framework for data-intensive Grid workflows," Procedia Computer Science, vol. 1, pp. 593-602, 2010.

[16] D. Dyachuk and R. Deters, "Scheduling of composite web services," On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, 2006, pp. 19-20.

[17] S. Galizia, A. Gugliotta, J. Domingue, “A Trust Based Methodology for Web Service Selection,” IEEE International Conference on Web Service, 2009, pp. 881-890.

[18] M. Alrifai, D. Skoutas, T. Risse, “Selecting Skyline Services for QoS-based Web Service Composition,” In Proceedings of the International World Wide Web Conference, Apr. 2010, pp. 11-20.

[19] H. U. Heiss and R. Wagner, "Adaptive load control in transaction processing systems," In 17th International Conference on Very Large Databases, Barcelona, Spain, Sep. 1991, pp. 48-54.

[20] J. Carlstrom and R. Rom, "Application-aware admission control and selection in web servers," The 21st IEEE International Conference on Computer Communications, Jun. 2002, pp. 506-515.