International Journal of Software Engineering and Its Applications
Vol.8, No.4 (2014), pp.169-180
http://dx.doi.org/10.14257/ijseia.2014.8.4.18

Selection of Web Services by Using Diversified Service Rank

Ayaz Nazir Ahmed[1] and Farooque Azam[2]

[1]*Virtual University of Pakistan, M.A. Jinnah Campus, Defence Road, Lahore, Pakistan*
[2]*Department of Computer Engineering, College of EME, National University of Sciences & Technology, H-12, Islamabad, Pakistan*
[1]*ayaznazir@gmail.com*, [2]*farooq@ceme.nust.edu.pk*

Abstract

Traditional techniques for web service discovery mostly rely on string matching and/or semantic logic-based matching. They are generally developed on service description and functional attributes which are exposed in service advertisement. The selection is made from the result output which is typically presented in order of their ranks based on degree of match of functional or operational parameters. The QoS attributes such as service performance, reliability, reputation, price and user preferences are often ignored in the query. Moreover, it is very difficult for the requester or any software agent to select the best service out of many potential candidates from the search result. This paper proposes a framework that takes into account QoS attributes along with user preferences and devise an algorithm namely Diversified Service Rank (DSR) and re-ranks top k web services in the search result, this helps service requesters in prompt selection and composition of web services.

Keywords: *Discovery and Selection of Web Services, Re-ranking Web Services on QoS Attributes and User Preferences, Diversified Service Rank, Distributed Databases*

1. Introduction

SOA is a major breakthrough in the field of distributed applications which has enabled the software development community to exploit the reusability of web services. Web sites can be made more dynamic by automatic selection and composition of web services. The trend to build core business structural web sites and accomplish rest of the functionality through web services is gaining popularity day by day. However, selection of a best web service from a huge repository of multiple heterogeneous registries is a challenging task. Traditional techniques of string matching are good for searching web sites, where the search result is primarily ranked by PageRank algorithm [4]. Web services on the other hand, are well defined entities and their search also takes into account functional parameters and the output result is thus ranked on parameter degree of match. However, discovery techniques in the case web services, lack the account of QoS attributes (also referred to as non-functional attributes) and user preferences. A candidate service in search result output although may be fully qualifying the functional requirements but may prove to be unreliable, slow and expensive or may even be a malicious at times. Resultantly, web service selection and composition at the endpoint involves user intervention. This makes the selection process tedious, unreliable and contrary to the objective of any automatic web service composition [2]. Moreover in scenarios where different users want to have their own search preferences over QoS attributes, there is a need to re-rank a set of top *k* web services in a given search result, so as to minimize user intervention in the selection process.

ISSN: 1738-9984 IJSEIA

2. Research Contributions

This paper aims at proposing a framework for selection of a best web service based on a DSR function. Following objectives are set to achieve this aim:-

- Target a set of top *k* web services retrieved through any of the traditional web service discovery techniques.
- Propose a QoS model that helps in computation of different non-functional attributes of web services belonging to a specific domain.
- Propose a model for obtaining users' preferences over QoS attributes.
- Define a logical relationship between individual quality scores and user preferences and work out an overall quality score, which acts a ranking measure (or DSR score) for web services.
- Re-order the targeted set of top *k* web services on DSR score and weighted user preferences.

3. Related Work

Skoutas *et al*., in their paper [8] proposed a framework for retrieval of top *k* web services based on different input and output parameters and other criteria that may match to web service description in a given domain. Their paper set three challenges for their research: firstly how to attain the best degree of match between different parameters; secondly how to group the matching results after attaining a best degree of match and thirdly how to carry out ranking of the result. They used the idea of dominance relationships in devising their algorithms. The idea of dominance implies that given two sets of parameters X and Y; X is considered to be dominating Y if and only if X is superior or equal to Y with respect to all parameters, and exactly superior in at least one parameter. Basing on this idea of dominance, they proposed three ranking criteria namely *dominated score (dds), dominating score (dgs)* and *dominance score (ds).* The first two scores implied that (1) a match was to be considered as good if few other matches dominated it, and (2) it was also considered as good if it dominated many of the others, respectively; while the third type of score was based on combination of two. Accordingly, they proposed three ranking algorithms based on these intuitions namely TKDD, TKDG, and TKM to work out a dominance score. They showed with their experimental evaluation that TKM was more appropriate for applications demanding accurate results whereas TKDD was recommended for time sensitive applications.

Skoutas *et al.* in their paper [7], proposed a unified framework for ranking and clustering of web services taking into account multiple criteria for carrying out parameter degree of match. Ranking entailed working out a score for each advertisement based on dominance relationships among services, while clustering grouped the similar advertisements with respect to the request so that user could select desired service from the relevant cluster. For example, while clustering on QoS attributes of price and execution time, two clusters would be formed; the first cluster would contain web services that were expensive in cost but good in terms of response time and the second cluster would contain those low in price but taking long execution time.

Skoutas *et al.* in another paper [6], proposed a method for re-ranking discovered services as per varied users' unknown preferences. Their approach aimed at diversifying the search results by combining functional parameters and other nominal attributes of web services which were specified in the service description but lacking a well defined ordering, which were otherwise very difficult to incorporate in match making process. They proposed a diversification method by making use of semantic web service. Their goal was to retrieve top *k* result containing similar and dissimilar web services with a balanced approach expecting that at least one service would be present in the result that sufficiently fulfilled the requester's requirement. The metric

for computing similarity measure was worked out by using Jaccard similarity. In order to achieve this goal they provided a diversification objective giving maximum coverage to the whole result as compared to the diversification objective proposed by Gollapudi and Sharma [5], which gave coverage to the services that meet the degree of match and dissimilarity only.

Zeng *et al.* in their paper [9], addressed the issue of user satisfaction over web service composition by providing a middleware platform namely AgFlow, which took into consideration different preferences set by the user over execution plans of elementary web services. They also described a QoS model and proposed some methods for computing different QoS attributes of web services. The issue of computing QoS of composite web services was also addressed and two approaches for selection of elementary web services were discussed and compared namely service selection by *local optimization* and *global planning*.

4. Problem Statement

Given an ordered list of web services $L = (a_1, a_2, a_3,\ldots,a_n)$ which is output of a query over a set of web service advertisements, where L is derived from functional attributes of web services and some ranking algorithm like proposed by Skautas *et al*. [8]. Assuming that at least one service is present in first *k-elements* of L that satisfies user requirement, we shall re-rank top *k-elements* and define a new order list $DSR = (s_1, s_2, s_3,\ldots,s_k)$ such that its first element is closest to the user requirement/satisfaction, where *DSR* is a function defined on QoS attributes and user preferences.

5. Diversified service rank (DSR)

5.1. Defining DSR

Ranking of web services means to assign a score to each advertisement that can provide a measure of selection during a request, while the clustering involves organization of advertisements into different groups so that services within each group provide similar matches to a user [7]. Diversification is a means of re-producing a search result after re-ranking as per user intent [1]. In our case, DSR function entail computation of quality score for every candidate web service in the domain, aggregating all quality scores to get an overall quality score (which we call a scalar value of our DSR function) and finally ordering result based on overall quality score and user preferences.

5.2. Computing individual quality scores for QoS attributes

While computing individual quality scores for QoS attributes, we shall take into account a domain specific criteria *e.g.*, a set of QoS attributes (which we also refer to as our quality vector) in a business domain. This demands that the complete set of targeted top *k* services should respect our quality vector and information about QoS attributes should be collected in a fair manner. For example, while getting feedback from users the service provider must verify the validity of user through some mechanism such as user ID and password. Similarly, QoS attributes published by the service provider *e.g.*, price, execution time etc should also be true and correct. Here in succeeding sub-sections, we suggested some ways to compute quality score of some of the core QoS attributes in a business domain.

5.2.1. Price

The price is a fee that a service consumer has to pay for execution of an operation of a web service. Suppose there is service *s* having an operation *op* then execution price can be expressed

as $q_{price}(s, op)$ [9]. The provider of web service may either advertise the price of operations, or allow some method to inquire about it. The price of all candidate web services should be expressed in a common currency.

5.2.2. Execution Duration

For the service *s* providing an operation *op*, the execution duration $q_{duration}(s, op)$ is the time taken in seconds from the moment of initiating the request by the user till receiving of result. It can be computed as follows [9]:

$$q_{duration}(s, op) = T_{transmission}(s, op) + T_{process}(s, op) \quad \ldots\ldots (1)$$

where, $T_{process}(s, op)$ is the process time which is advertised by the service provider or some method is provided to inquire about it. $T_{transmission}(s, op)$ is transmission time which is an average of past transmission times and is expressed as under:

$$T_{transmission}(s, op) = \frac{\sum_{i=1}^{n} T_i(s, op)}{n} \quad \ldots\ldots (2)$$

where $T_i(s, op)$ is transmission time observed previously, and *n* is total number of observations made in the past.

5.2.3. Reputation

A web service reputation may be computed by taking graded average of users' feedbacks. One of the ways to get users' feedback is through a survey. Let there be six grades (A through F) for a service *s*. Users may be asked to grade the service by voting in a survey. Each grade is assigned with a weighting as shown in Table 1. The Grade Count column in Table 1 shows the number of users who voted for their corresponding grades:

Table 1. Grading Obtained through Survey

Grade	Description	Weighting (W)	Grade Count (G)	Weighted Grade (W×G)
A	One of my favorites	1	5	5
B	Very good	.8	15	12
C	Good	.6	20	12
D	Fair	.4	5	2
E	No idea	.2	3	0.6
F	Poor	0	2	0
	Total	**3**	**50**	**31.6**

The overall reputation $q_{reputation}(s)$ of the web service *s* can now be obtained by dividing the sum of weighted grades by total number of users who took part in the survey as shown in following generalization [9]:

$$q_{reputation}(s) = \frac{\sum_{i=1}^{m}(W_i \times G_i)}{\sum_{i=1}^{m} G_i} \quad \ldots\ldots (3)$$

such that $W_i \in [0, 1]$

where *m* represents the total number of grades offered to the users, W_i is the weighting assigned to the *i th* grade which has range [0, 1] and G_i is the grade score or the number of users who

voted for the *i th* grade. Also note that $q_{reputation}(s)$ is a probability measure and has range [0, 1]. For the data given in Table 1, reputation score can be computed as:

$$q_{reputation}(s) = \frac{31.6}{50}$$
$$= 0.632$$

5.2.4. Link Popularity

Link popularity of a web service can be defined as number of inbound links pointing to the web service. This can be measured by applying PageRank algorithm proposed by Brin and Page [4]:

$$q_{popularity}(s) = (1\text{-}d) + d\,\{PR(T_1) / C(T_1) + \ldots + PR(T_n) / C(T_n)\} \quad \ldots\ldots (4)$$

where,

- $q_{popularity}(s)$ is Link Popularity (or Page Rank) of a web service *s*
- $PR(T_i)$ is Page Rank of T_i pages that link to *s*
- $C(T_i)$ counts for outbound links on T_i pages (outbound links are other links that do not point to the page under consideration).
- *d* is the Damping Factor, its value ranges from 0 to 1.
- Notice that $q_{popularity}(s)$ is a probability measure and has range [0, 1]. It is developed on random surfer model and therefore it is reduced by *d*. The justification given for *d* by Brin and Page is that a surfer may not choose to click on unlimited number of links but may get fed up sometimes and is expected to surf to some other pages in a random fashion.

5.2.5. Throughput (frequency of usage)

It refers to the frequency of a web service by its usage during a specific period. If web service *s* was used *n* times during a period from date d_1 to date d_2, then frequency of usage $q_{frequency}$ for *s* can be expressed as:

$$q_{frequency}(s, d_1, d_2) = \frac{n}{d_2 - d_1} \quad \ldots\ldots (5)$$

where, d_2 is later than d_1 *i.e.*, $(d_2 > d_1)$

5.2.6. Success rate

The success rate of a web service refers to the number of times the users' requests were successfully served within the execution time limit published in the web service description. The success rate is also dependent on software, hardware and network configurations involved in execution process. The success rate of a service is estimated by number of its previous successful invocations. If a web service *s* remained successful for *N* times during past *K* invocations then its successful execution rate may be computed as under [9]:

$$q_{success}(s) = N/K \quad \ldots\ldots (6)$$

5.2.7. Availability

The availability of a service *s* is a probability measure that may be computed by following expression [9]:

$$q_{availability}(s) = \frac{T_{availability}(s)}{\theta} \quad \ldots\ldots (7)$$

where, $T_{availability}(s)$ is the total time measured in seconds for which the service s remained available in last θ seconds (θ is treated as a constant and its value may be set depending upon the context (domain) of web service by an administrator of that community). A smaller value of θ is more appropriate for kind of web services that are accessed more frequently and vice versa. Here the definition of availability is based on the assumption that web services keep sending notifications of their availability state (*i.e.*, available or unavailable) to the system or some means is available to inquire about it.

5.3. Computing Overall Quality Score

The overall quality score provides a scalar value for our DSR, which can be obtained by combining individual quality scores in a logical way. Following sub-sections provide this logical relationship.

5.3.1. Taking Probability Measures

Individual quality scores that we have computed have different unit of measurements. For example q_{price} is measured in some currency like \$40; $q_{duration}$ is measured in seconds; $q_{reputation}$ *and,* $q_{popularity}$ are probability measures; and $q_{frequency}$, $q_{success}$, $q_{availability}$ are measured in units. Our DSR value takes into account their probability measures i.e. each complying to a range [0, 1]. To find probability measure, we divide an individual quality score by the maximum quality score in the space. For example probability of *success rate* of a service s_1 expressed as $P(q_{success}(s_1))$ can be obtained by dividing its success rate with maximum success rate among k services as expressed in following generalization:

$$P(q(s_i)) = \frac{q(s_i)}{MAX(q(s_k))} \quad \text{..... (8)}$$

For the quality attributes like execution duration or price, whose higher score reflect low probability for selection, the probability measure may be reversed by subtracting it from 1. For example, the probability measure for price may be computed as under:

$$P(q_{price}(s_i)) = 1 - \frac{q_{price}(s_i)}{MAX(q_{price}(s_k))} \quad \text{..... (9)}$$

5.3.2. User Preferences

Different users may have different preferences over QoS attributes and they would also like to change their preferences later. Therefore, it is important to present a model for obtaining user preferences. There are number of ways to get users' preferences *e.g.*, by creating user profiles on a server or saving user preferences locally on user machine in the form of a cookie, local database or XML file. Here, we have hypothesized preferences of a user in weighted form as reflected in Table 2. We have also shown an assumed quality score (in terms of probability) for each quality attribute, which we shall use to compute an overall quality score of a service s_1 as shown in working equation (11).

5.4. DSR Function

Zeng *et al*. have proposed a formula for computing an overall quality score as sum of their weighted probabilities [9]:

$$Q(s) = \sum_{i=1}^{m}(W_i \times P(q_i)) \quad \text{..... (10)}$$

such that $W_i \in [0, 1]$ and $\sum_{i=1}^{m} W_i = 1$

where, m is total number of QoS attributes in a given domain, $P(q_i)$ is the probability of *i th* quality attribute and W_i is the weighting/preference assigned by the user to the *i th* attribute. The formula at equation (10) computes a scalar value for our DSR function. For example, to compute DSR value of a service s_1 for the data provided in Table 2, we have:

$$Q(s_1) = 0.23 \times 0.62 + 0.15 \times 0.75 + 0.27 \times 0.32 + 0.05 \times 0.45 + 0.09 \times 0.10 + 0.16 \times 0.96 + 0.05 \times 0.80$$
$$= 0.566 \quad \text{..... (11)}$$

In order to rank the search result in proper order we need to sort set of top k services in the order defined below:

$$DSR\ (Q(s), P(q_1), P(q_2), \ldots, P(q_m)) \text{..... (12)}$$

where, m represents the total number of quality attributes in the domain.

Thus, our DSR Function entails computation of DSR scalar value for every service in top k by formula defined at equation (10) followed by ranking order given at (12). In the next section we give pseudo code of our DSR algorithm and also carry out its analysis and experimental evaluation.

Table 2. Weighted User Preferences

Index	QoS Attribute	User Preference (W)	Probability (P)
1	$q_{price}(s_1)$	0.23	.62
2	$q_{dur}(s_1)$	0.15	.75
3	$q_{rep}(s_1)$	0.27	.32
4	$q_{pop}(s_1)$	0.05	.45
5	$q_{freq}(s_1)$	0.09	.10
6	$q_{success}(s_1)$	0.16	.96
7	$q_{avail}(s_1)$	0.05	.80

6. Results and Discussion

6.1. DSR Algorithm

The pseudo code of our algorithm and its running time is enlisted in Table 3. The pseudo code and running time for selection sorting i.e. call to function ISERTION-SORT() at Line#7 is already worked out by Brassard and Bratley [3].

If $T(n)$ is the total running time of our algorithm, then it can be computed as below:

$$T(n) = c1 \times n + c2 \times (n-1) + c3 \times \sum_{j=1}^{n} t_j + c4 \times \sum_{j=1}^{n} (t_j - 1) + O(n^2) + m \times O(n^2) + c8$$
$$= c1 \times n + c2 \times n - c2 + c3 \times \frac{n(n+1)}{2} + c4 \times \frac{n(n-1)}{2} + O(n^2) + mO(n^2) + c8$$
$$= c1 \times n + c2 \times n - c2 + \frac{c3}{2} \times n^2 + \frac{c3}{2} \times n + \frac{c4}{2} \times n^2 - \frac{c4}{2} \times n + (m+1)O(n^2) + c8$$

$$T(n) = \left(\frac{c3}{2} + \frac{c4}{2}\right)n^2 + \left(c1 + c2 + \frac{c3}{2} - \frac{c4}{2}\right)n + (c8 - c2) + (m+1)O(n^2)$$

It can be seen that *T(n)* is a quadratic function of the form $an^2 + bn + c$ for constant *a*, *b* and *c*. Thus worst case running time expressed in big *O* notation is:

$$Running\ Cost = O(n^2) + O(n) + c + O(n^2)$$

$$= O(n^2)$$

Also note that computational cost for function call *getQScore()*at Line#4 is taken as one primitive step in our algorithm. This is because of the reason that algorithm for computing every individual quality score is not uniform rather it depends upon the domain it belongs to and the logical relevance of various contributing factors that build up the score.

Table 3. DSR Algorithm

```
Algorithm DSR(S, Q, W)
Description: Computes DSR value of each element in array of web services S and sort
       elements on DSR ranking.
Input:  An unsorted array S of top k services.
       An array V of m number of QoS attributes in descending order of preference.
       An array W of m number of weights assigned to corresponding elements in V.
Output: A sorted array DSR based on DSR ranking.
```

Line#		Running Time Cost	Running Time Time
1	for i ← 1 to k	c1	n
2	q ← 0; //for individual quality score	c2	n-1
3	for j ← 1 to m	c3	$\sum_{j=1}^{n} t_j$
4	DSR[i] ← q + W[j] * getQScore(V[j]);	c4	$\sum_{j=1}^{n}(t_j - 1)$
	//first sort on dsr value		
5	INSERTION-SORT(DSR, dsr);		$O(n^2)$
	//resolving ties by sorting as per DSR ordering		
6	for i ← 1 to m		
7	INSERTION-SORT(DSR, V[i]);		$m \times O(n^2)$
8	return DSR;	c8	1

6.2. Analysis of DSR Algorithm

We will analyze the performance of our algorithm by comparing with those proposed by Zeng *et al.*, [9] and Skoutas *et al.*, [6, 7].

Zeng *et al.* [9] have primarily focused on composition of various web services taking into account an intermediate step of automatic selection of elementary web services based on their individual QoS score. We have gone a step ahead in the intermediate step wherein we have proposed a method to resolve ties among common scores based on user preferences set on QoS attributes, before their final selection. So our algorithm has additional computational cost of sorting. We can also optimize the overall performance of our algorithm by selecting a suitable sorting algorithm depending upon the value of *k*. For example insertion and selection sorts are good when *k* is small while merge sort is good when *k* is large. Also running time of insertion sort is further reduced when unsorted numbers are more close to each other. Insertion sort has a worst-case running time of $O(n^2)$ while merge sort runs in $O(n \lg n)$ worst-case time. If *m* is the total number of QoS attributes involved then irrespective of the sorting algorithm we choose, sorting cost will be $(m+1)$ times the worst-cost running time. Since *m* is constant coefficient

and does not participate in the growth rate of running time so we can ignore it in our asymptotic notation, which remains unchanged as $O(n^2)$. Resultantly, we get a better ordering of web services without effecting the worst-case running time.

Three major differences become prominent when we compare our results with that of Skoutas *et al.* [6, 7]. Firstly, Skoutas *et al.* [7] in their research have mainly focused on retrieving top *k* web services based on dominance relationship among different functional attributes in working out their rank (dominance score), whereas we have proposed a mechanism to re-rank top *k* web services based on their numeric rank (DSR scalar value) and order of weighted user preference. Secondly, Skoutas *et al.* [6] have used the idea of working out a similarity measure in the absence of well defined user preferences; their similarity measure primarily works on Jaccard Similarity. Whereas, we have worked in the scenario where user preferences are well defined and are made available in a weighted form. The services with higher user preference and higher DSR scalar value are presented higher in our ranked list. The third difference is that our approach is sub-modular to the general problem proposed by Skoutas *et al.*, [6]. The diversification objective presented by Skoutas *et al.* [6] for selection of top *k* web service is NP-hard; whereas we have mainly focused on re-ranking top *k* web services in the search result which takes quadratic running time.

6.3. Experimental Evaluation

Let there be a quality vector $V(q_1, q_2, q_3, q_4)$ with corresponding weights $W(0.4, 0.3, 0.2, 0.1)$. For $k=10$, let's assume some quality scores already computed for services s_1 through s_{10} as shown in Table 4. As we can see that services are ranked on overall quality score $Q(s)$ but there are still three unresolved ties T_1, T_2 and T_3. These ties may be resolved by applying ordering part of our *DSR* function as given in sequence (12) i.e. sorting in order of preference set by the user. The rule is to look at the most preferred attribute and sort data on quality score of the attribute and then repeat the process for next most preferred attribute until all the ties are resolved. After applying this rule we get service ranking as shown in Table 5, which is closest to the user intent as it fully takes into account user preferences and QoS scores.

Table 4. Service Ranking Based on Overall Quality Score with Ties

Service	*Ties*	$Q(s)$	$P(q_1)$	$P(q_2)$	$P(q_3)$	$P(q_4)$
s_1	T_1	0.58	0.3	0.8	0.7	0.8
s_2		0.58	0.7	0.7	0.3	0.3
s_3		0.58	0.6	0.7	0.4	0.5
s_4	T_2	0.53	0.7	0.5	0.4	0.2
s_5		0.53	0.7	0.6	0.3	0.1
s_6		0.53	0.7	0.5	0.2	0.6
s_7	T_3	0.48	0.5	0.6	0.3	0.4
s_8		0.48	0.5	0.6	0.2	0.6
s_9		0.48	0.5	0.6	0.4	0.2
s_{10}		0.46	0.9	0.1	0.2	0.3

Table 5. Service Ranking after Resolving Ties Based on DSR Function

Service	*Ties*	*Q(s)*	*P(q_1)*	*P(q_2)*	*P(q_3)*	*P(q_4)*	*Remarks*
s_2	T_1	0.58	0.7	0.7	0.3	0.3	T_1 is resolved on $P(q_1)$
s_3		0.58	0.6	0.7	0.4	0.5	
s_1		0.58	0.3	0.8	0.7	0.8	
s_5	T_2	0.53	0.7	0.6	0.3	0.1	T_2 is first resolved on $P(q_2)$ and the sub-tie is resolved on $P(q_3)$
s_4		0.53	0.7	0.5	0.4	0.2	
s_6		0.53	0.7	0.5	0.2	0.6	
s_9	T_3	0.48	0.5	0.6	0.4	0.2	T_1 is resolved on $P(q_3)$
s_7		0.48	0.5	0.6	0.3	0.4	
s_8		0.48	0.5	0.6	0.2	0.6	
s_{10}		0.46	0.9	0.1	0.2	0.3	

Our DSR function can further be validated by drawing graphs of dominance relationships for each tie as shown in Figure 1. The graph for tie T_1 shows that it is resolved as $s_2 \succ s_3 \succ s_1$, where $\succ$ is a symbol for dominance. Similarly, tie T_2 is broken by $s_5 \succ s_4 \succ s_6$ and so is the tie T_3 resolved by $s_9 \succ s_7 \succ s_8$.

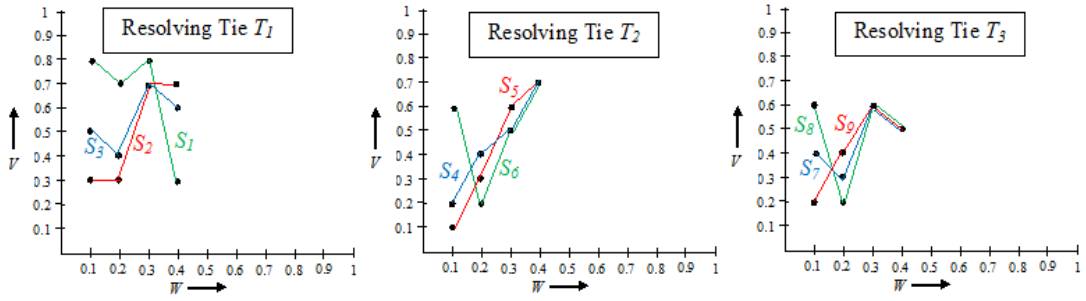

Figure 1. Resolving Ties by Dominance Relationship

7. Future Work

The future research directions may be to provide service requesters with a framework that may enable them to describe their requirements from the service in a well defined manner, manage and store their preferences along with UDDI. At the same time existing service discovery techniques which are primarily based on the functional description in WSDL only fulfill operational requirements of the requesters, the QoS specifications have been left entirely in the hands of service providers. There is a need to fill this gap by devising a standard framework that should address incorporating QoS issues in WSDL.

8. Conclusion

In this paper we have presented a framework for selection of web services based on QoS attributes (also referred to as non-function attributes) and user preferences set over them. We have assumed that there is a set of top *k* services already retrieved from UDDI based on functional attributes and other essential parameters. In order to select the best service out of top

k result, we have derived a DSR function. Our DSR function takes into account a scalar DSR value which is obtained by logically combining individual quality scores of non-functional attributes and weighted user preferences.

References

[1] R. Agrawal, S. Gollapudi, A. Halverson and S. Ieong, "Diversifying Search Results", Proceedings of 2nd ACM International Conference on Web Search and Data Mining, Barcelona, Spain, **(2009)** February 9-12.
[2] G. Baryannis and D. Plexousakis, "Automated Web Service Composition: State of the Art and Research Challenges", Technical Report ICS-FORTH/TR-409. Institute of Computer Science Information Systems Laboratory. Foundation for Research & Technology, Hellas, Greece, **(2010)** October.
[3] G. Brassard and P. Bratley, Fundamentals of Algorithmics, Edited T. McElwee, B. Zobrist, I. Zucker and P. Mailhot, Prentice Hall Publishers, Englewood, New Jersey, USA, 1st Edition, **(1996)**, pp. 62-63.
[4] S. Brin and L. Page, "The Anatomy of a Large Scale Hypertextual Web Search", Proceedings of 7th International Conference on World Wide Web, Brisbane, Australia, **(1998)** April 14-18.
[5] S. Gollapudi and A. Sharma, "An Axiomatic Approach for Result Diversification", Proceedings of 18th International Conference on World Wide Web, Madrid, Spain, **(2009)** April 20-24.
[6] D. Skoutas, M. Alrifai and W. Nejdl, "Re-ranking Web Service Search Results under Diverse User Preferences", Proceedings of 4th International Workshop on Personal Access, Profile Management and Context Awareness in Databases, Singapore, **(2010)** September 13.
[7] D. Skoutas, D. Sacharidis, A. Simitsis and T. Sellis, "Ranking and Clustering Web Services using Multicriteria Dominance Relationships", IEEE Transactions on Services Computing, vol. 3, no. 3, **(2010)**.
[8] D. Skoutas, D. Sacharidis, A. Simitsis, V. Kantere and T. Sellis, "Top-k Dominant Web Services under Multi-Criteria Matching", Proceedings of 12th International Conference on Extending Database Technology: Advances in Database Technology, Saint Petersburg, Russia, **(2009)** March 23-26.
[9] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, "QoS-Aware Middleware for Web Services Composition", IEEE Transactions on Software Engineering, IEEE Computer Society, vol. 30, no. 5, **(2004)**.

Authors

Ayaz Nazir Ahmed received his post graduate degree in computer sciences from Bahauddin Zakariya University, Multan, Pakistan in 2005. Currently, he is pursuing his M.S. degree in Software Engineering at Virtual University of Pakistan. He is a senior instructor of computer sciences at Pakistan Military Academy, Kakul, Abbottabad. His research interests lie in the areas of web services and database management.

Farooque Azam received his Bachelor of Electrical Engineering degree from N.E.D. University, Karachi, Pakistan in 1988. He did his M.S. in Software Engineering from National University of Sciences and Technology (NUST), Pakistan in 2003 and later Ph.D. in Software Engineering from BUAA, Beijing, China. Presently, he is serving at College of EME, NUST, Pakistan as a Professor.