Journal of Theoretical and Applied Information Technology
31st January 2014. Vol. 59 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

QOS-BASED RANKING MODEL FOR WEB SERVICE SELECTION CONSIDERING USER REQUIREMENTS

[1]G. VADIVELOU, [2] E. ILAVARASAN

[1] Research Scholar, Dept. of CSE, Bharathiar University, Coimbatore, Tamilnadu, India

[2] Professor, Dept. of CSE, Pondicherry Engineering College, Puducherry, India

E-mail: [1]vadi_ganesh@hotmail.com, [2]eilavarasan@pec.edu

ABSTRACT

Web services are widely used in the business application development to achieve interoperability among standalone systems. Efficient and effective techniques are required to find and select required services among similar services which is an important task in the service-oriented computing. Ranking process which is part of Web service discovery system helps the users to find the desired services effectively. The existing research contribution for ranking process does not consider the user's requirements which are an important factor to rank web services. In this work, vector-based ranking method is enhanced to consider user's requirements. The vector-based model is selected because of its simplicity and high efficiency. The web services are evaluated on the basis of their similarity degrees to the optimal or the best values of various quality attributes. Experiments are conducted with real dataset and the improved algorithm is compared with the other approaches and it is found that the enhanced vector-based ranking method is efficient in terms of execution time to return the result set.

Keywords: *SOA, Web Services Selection, Web Services Ranking, Vector-Based Ranking, QoS*

1. INTRODUCTION

The growing number of business applications in distributed systems has resulted in the increasing demand of communication between business modules. In context of the business community, Service Oriented Architecture (SOA) was proposed based on the idea that to provide a solution for a large problem in a more effective way, the required process can be decomposed into a collection of smaller, but related parts[1]. The most common way to implement SOA is through Web services. According to W3C (World Wide Web Consortium), a Web service is defined as a software module which is implemented through standard XML-based technologies such as WSDL and SOAP. With the increasing number of Web services, discovering and selecting best services to fulfill a required task is becoming more important.

In order to search and invoke Web services based on user's requirements, first all functional services need to be advertised by their providers in a public UDDI (Universal Description, Discovery, and Integration) registry[2]. Service providers publish descriptions and properties of their Web services in a standard file, i.e. WSDL (Web Service Description Language). A WSDL file contains the information about data types, operations and the network location of the Web services. Then consumers create their queries and use a discovery facility or an agent to search UDDI and locate the set of Web services relevant to their desired requirements. Finally, consumers need to select and invoke one of the Web services among all retrieved results[3]. More and more web services with the similar functionality are made available on the Web. In order to locate and select the appropriate Web services, additional features, i.e. non-functional attributes or quality of Web services (QoS) such as response time, scalability, etc. are taken into consideration in the discovery and selection process.

With the increasing size of the UDDI registry, it is becoming more difficult to locate and retrieve all matched web services and present them to consumers. Furthermore, it is evident that the retrieved result contains more than one matched Web services that meet the functional and non-functional criteria. Therefore, it is essential to devise an efficient technique to measure the ranking relation order between the retrieved services based on user's requirements on different QoS attributes. The process of ranking Web services is a dominant

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

part of a Web service selection system, as it helps users select their desired service easily.

2. PROBLEM STATEMENT

In the ranking process, the fundamental step is to find similarity degree between the user's request and a service. Various methods are proposed to address the problem of ranking web services. These methods compare all the quality parameters of similar Web services with the optimal values for each QoS attributes and the service with maximum similarity degree to the optimal values are returned as the result. In the some of the previous works, complicated data indexing methods are used in their query structure of the ranking process or all Web services are compared in a pair-wise method which involves more computation time. As the similar Web services are growing, the number of pair-wise comparison will increase which makes the algorithm much slower. Also most of the authors do not consider the roles of consumers in their works. They considered only services with the optimal values and not the real constraints which are as part of the query. Finally, the users are recommended with the set of Web services with minimum distance with the optimal values and different users will get the same set of Web services as recommendations.

Also various existing frameworks considered only a small number of QoS attributes and experimented only with small-sized Web service repositories. But there are various types of variables for QoS attributed, which should be considered to fulfill a desired task. Consumers prefer efficient methods which may deal with different types of constraints and large Web service repositories.

Also the existing frameworks take more time for processing a request. With the number of published web services in getting increased, it is very important to return the results fast as the user's tolerance on slow response is usually very low.

The main goal of this work is to develop a Web service ranking model in which, the user's request and preferences is considered along with the optimal values. Equal weights are given for both mentioned factors.

As in the previous works, a simple and more straightforward method to rank retrieved Web services is used to achieve accurate results. In this work, a methodology for ranking web services is proposed by developing a vector-based framework and considering user's requests and preferences and the quality and efficiency of the result of the proposed method is compared with one of the existing ranking algorithms. The proposed methods are also compared with a simple positional algorithm to show that the proposed methods are reliable and efficient. Different number of web services and different types of QoS attributes such as interval data, Boolean etc are considered to compare the algorithms. A real QoS dataset is considered for simulation.

The main contributions in this work are:

1. An improved rank aggregation based algorithm (Borda Fuse Algorithm) is proposed to cover the user's requirement.
2. A new enhanced ranking algorithm based on a vector-based model is proposed which is capable of dealing with user's requirements and to measure the ranking relation between services.
3. Finally the enhanced algorithms are compared with one of the famous skyline ranking algorithms (Sort-Filter-Skyline) with complex structure to show their efficiency on large sized datasets with a large number of attributes and different data types.

3. RELATED WORKS

Most researchers classify Web service discovery and ranking methods into two different groups: **Syntactic** and **Semantic** approaches. In semantic methods, the ontology concepts are used in the discovery process, whereas in syntactic methods, the selection process is based on the syntactic information. As semantic-based approaches suffer from massive human effort and complicated computational process, processing time is slow. It is also assumed that there is no standard definition of ontology to use for different situations. To address these issues, another category of service discovery approaches has been developed which is based on the syntactic information. It is believed by many researchers that syntactic-based models are more efficient than semantic-based approaches.

In Rank-based aggregation technique proposed by Aslam and Montague [4], first the services are ranked in different lists based on each individual attribute, and then the algorithm combines different ranked lists to compute the final ranked list.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

To aggregate *m* ranked lists generated by *n* sources, rank aggregation problem is used. There are two types of rank-based aggregation methods:

1. Supervised rank aggregation technique which relies on the training data and un-supervised rank aggregation method with no need of the training data.
2. Unsupervised rank aggregation technique is categorized in two groups: positional methods and Majoritarian techniques.

Positional methods generate the final ranked list based on the combination of all ranking scores gained by summing all the positional values of each element in each ranked list. The most common method in regards with the rank aggregation method is the Linear Score Combination method in which the scores of items are aggregated by some operators such as weighted sum to compute the final ranked list.

Another important algorithm in this context is referred as Borda-Fuse proposed by Bartell et al.[5]. It is considered an effective algorithm to rank a set of data points. The algorithm was introduced to solve the voting problem. It is a very simple procedure, which has been proved to be effective.

Another positional algorithm that can be named in this area is Median-Rank aggregation method introduced by Fagin et al.[6], in which the candidate documents are ranked based on their median ranks.

Majoritarian rank aggregation approaches are another type of unsupervised rank aggregation methods. In this type of algorithms, every item is compared with another candidate[7]. The method consists of repetitive steps. First they considered a list of all candidates and then each item in the list is compared with the next one. The winner stays in the list, but the loser will be removed from the list. The comparison steps are repeated until there is no other item in the list to be compared. This method suffers from low speed, as the number of comparisons gets larger, when the number of items in the dataset increases.

There is another type of matchmaking and ranking algorithms based on Skyline query concept which is a dominant topic in the database field .The skyline operation was introduced by Kossman et al[8] to solve maximum vector problems. The model calculates and filters the desired points relevant to a query and returns all possible solutions among a large set of data points in a given domain. Skyline points are composed of services that are not dominated by other services. Skyline points assist consumers to select their desired service easier based on their preferences.

In context of the skyline query field, Papadias et al.[9] introduced a progressive algorithm which relies on Branch and Bound Skyline (BBS) based on a nearest-neighbor search method. On a given set of points, this model computes the skyline points based on their distances to a query point in an ascending order. In this work they first indexed the data by applying an R- tree technique to reduce the computation cost by decreasing the number of pair-wise comparisons. Then they computed the dominance relationship between each two services. They argued that in their framework any pre-computation functions would not be required. BBS is widely used in multi-criteria optimization problem.

To extend the Skyline query model to relational databases[10],[11] presented a new algorithm called Sort Filter Skyline (SFS) model. They implemented their model based on a sorting technique. According to their theory all data points are sorted by using a sorting technique and considering a monotone function. In other words, SFS sorts all candidates that maximize the scoring function in an ascending order. After sorting the data, the services which dominate the other services over most attributes will appear in upper positions. Thus the number of pair-wise comparisons will decrease. Any service with the best score over the monotone function will appear in the skyline list. This method is used extensively and is a fundamental structure for methodologies invented later. This model is considered as a baseline for comparison purpose in many works.

3.1 Limitations

Most of the reviewed models are reasonable work; however they suffer from some deficiencies:

1. Data indexing and sorting techniques are used in most of the reviewed models which generally takes a longer processing time.
2. They mostly either ignore the role of user's requirements are ignored or their methods require users to compute the importance degree of each parameter. More load is given to the users.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. Only limed number and a few types of attributes (mostly numeric type) are considered, while in reality there are various types of data.

In the proposed work, the above mentioned issues are addressed by developing a simple but effective method which considers user requirement as an important factor while ranking web services.

4. PROPSOED WORKS FOR RANKING WEB SERVICES

Web service selection and discovery system is essential to provide clients with proper results according to their requirements. It is impossible to fulfill this task without considering the ranking relation between thousands of available candidates with similar functionalities. Ranking process is a fundamental step in a Web service selection system, as it integrates the results gathered from previous stages (functional and non-functional matching process) and presents them to the requestors. This work focuses on the ranking process by considering user's QoS requirements. Skyline algorithm is used as baseline for the comparison purpose because it is well-accepted algorithm in database area for the multi-criteria based selection problem, and recently is being used in Web service selection area due to the accuracy of the generated results.

However, efficiency is one of the big concerns on this algorithm. The proposed work is to test whether a much simpler algorithm with higher efficiency could achieve the similar level of accuracy. The simple algorithms chosen in this work are vector-based (Distance-based) algorithm and a rank aggregation algorithm (Borda Fuse). Since both of them (also the skyline algorithm) do not consider the actual user requirements at all, in this work it is proposed to enhance those algorithms by taking into account the user's QoS requirements. The goal of the proposed work is to provide a simple and effective method for generating a ranked list of desired Web services with consideration of user's requirements. The QoS data considered in this work includes different types of data such as interval, numeric, boolean etc.

4.1 Proposed Enhanced Borda Fuse Algorithm Considering User's Requirements

Rank Aggregation methods are used for ranking data. One of the efficient methods in this context is Borda Fuse[5]. This model was proposed to solve the voting problems in different areas. In the context of Web service discovery, a service which appears in the highest positions in the most ranking lists will receive higher ranking score. In this model all services are first ranked in different lists in terms of different QoS attributes. Each service is assigned a score based on its positional value in each individual ranked list. Then the final ranked list is generated by computing the summation of all obtained scores from all ranked lists. The Borda Fuse algorithm suffers from an important deficiency that may affect the accuracy of the results. In this model the user's actual request is ignored which is an important factor in selection systems. For different requirements from different users, the output of the algorithm is always the same.

To overcome this issue, this work proposes an improved algorithm to cover user's requirements in the ranking process.

The final ranking score in Borda Fuse method is calculated by adding the positional value of each service in each individual ranked list. The user's requirements are not considered in the ranking process which returns the same result to the user for different requirements. To involve user's requirements in the algorithm, in the enhanced method the query attributes are considered as a sample service S_q and is added it to the list of offered services. As a result, a new ranked lists including S_q is generated as indicated in the Table 1. It is noticed that the position order of service S3 and S5 has changed in the new ranked list. Service S5 does not meet the requirements for the last 2 attributes.

The scores are assigned to services depend on their position in each ranked list. A negative scores are assigned to those services which appear in the each rank list after S_q. The negative score for each service is computed according to position of service in the new ranked list. Suppose *n* as the number of services, S_i .*position* as the position value of service S_i in each ranked list, S_q.*position* as the positional value of S_q in the new list with considering user's requirements.

Journal of Theoretical and Applied Information Technology
31st January 2014. Vol. 59 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Ranked Lists Based On The User's Requirements

Ranked list based on availability	Ranked list based on response time	Ranked list based on cost	Ranked list based on reliability	Final ranked list
S4	S1	S2	S2	S2
S2	S5	S1	S4	S4
S5	S4	S3	S3	S1
S1	S3	S4	S_q	S3
S3	S2	S_q	S5	S5
S_q	S_q	S5	S1	S_q

Considering the new list, if a service is in a higher position than S_q, then the score for each list is computed as: Rankscore = (n- S_i.position) + 1 In case a service has a lower position than sq in the ranked list, the score is calculated as: Rankscore = S_q.position - S_i.position

The total score (Score S_i) for each service is calculated as: Score S_i $= \sum_{i=1}^{n}$ wi Rankscore where n is the number of services, S_i.position is the position value of service S_i, S_q.position is the position value of S_q, m is the number of QoS attributes and w_i is the weight of attribute. The final ranked list could be generated by sorting the final scores in a descending order.

4.2 Proposed Enhanced Distance-Based Algorithm Considering User's Requirements

In the basic distance-based algorithm, the problem of matching and ranking Web services based on the requirements is considered as a distance measurement problem[12]. To solve the problem, a published Web service is modeled in a vector including its *n* QoS attributes. The optimal values of each QoS attributes are also modeled in another vector. The distance between the two vectors is measured according to Weighted Euclidean Distance formula. Then in the ranking process, the ranked list is generated by sorting the distance values in an ascending order.

The Distance-Based algorithm is an efficient method to generate a list of ranked services. However, the original algorithm only considers optimal value for each attribute. The algorithm can be improved by considering user's requirements.

In the proposed work, the values of n QoS attributes of a service S is modeled as a vector: $Q_s=(Q_{s1},Q_{s2},\ldots,Q_{sn})$ where s1,s2,…,sn are services and the values of QoS requirements requested by a consumer as $Q_r=(q_{r1},q_{r2},\ldots,q_{rn})$. The consumer's preferences values on each QoS attribute is modeled as $P_r=(P_{r1},P_{r2},\ldots,P_{rn})$ where $P_{r1},P_{r2},\ldots,P_{rn}$ preferences where values of P_{ri} ranges between 1 and n. If the consumer has not specified the preferences, n will be considered as the preference. Weight values of the preferences are calculated using the below given formula.

$$w_i = \frac{p'_{ri}}{\sum_{i=1}^{n} p_{ri}} \quad i = 1,2,\ldots. n \qquad \text{where}$$

$$p'_{ri} = (p_{r\max} + 1) - p_{ri} \qquad (1)$$

where w_i is the weight of an attributes, p_r is the preferences values of each attributes, n is the total no. of services and $p_{r\max}$ is the maximum value in vector p_r. The score to service S is computed based on its distance to the optimal values of the QoS attributes using the Euclidian formula. In order to consider the real query values, the distance between the QoS of service S and the real requested constraints specified in the query is also calculated. The final distance score which indicates the distance between a service and both optimized and required values is calculated as explained below.

Based on the Euclidian distance method, score for each service (S) based on the optimal QoS(Q_o) is calculated using the formula given below.

$$Score_{osi} = | Q_{si} - Q_o | = \sqrt{w_j (q'_j - q'_{oj})^2} \qquad (2)$$

where Q_o is the optimal values of QoS attributes, w_j is the weight of an attributes, q'_j is the normalized values of an attributes and q'_{oj} is the normalized value of an optimal value.

Based on the tendency of the QoS parameters, the normalized values for numeric and Boolean attributes are computed based on method discussed in[13] and the normalized values for range type is calculated based on the method discussed in [14].

To calculate the distance between each service S_i and the requested QoS(Q_r), a vector $P_i=\{p_{i1},p_{i2},\ldots p_{in}\}$ is considered where p_{ij} € [1..n] and the vector P_i includes 0 or 1 based on whether

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the service S_i meets user requirement in the query or not.The distance between each services and the real required QoS(Q_r) is calculated as:

$$Dis_{si} = | Q_{si} - Q_r | = \sqrt{p_{ij} w_j (q_j - q'_{rj})^2} \quad (3)$$

where Q_r is the distance between each service and real required QoS, p_{ij} is the preference values of an attribute, w_i is the weight of an attributes, q'_j is the normalized value of attributes and q'_{rj} is the normalized value of specified query. The distance between two vectors of QoS and offered services interval data type attributes is calculated as:

$$Dis_{si} = | Q_{si} - Q_r | = \sqrt{\sum_{j=1}^{n} p_{ij} w_j ((q'_{jl} - S'_{jl})^2 + (q'_{ju} - S'_{ju})^2)} \quad (4)$$

In the above equation, q'_{jl} and q'_{ju} are the normalized values of lower and upper bounds of the j^{th} QoS attribute respectively and S'_{jl} and S'_{ju} are the normalized values of lower and upper bounds of the j^{th} attribute respectively. Finally, we calculate the final score for each services as:

$$Score_{si} = W_1 Dis_{si} + W_2 Score_{osi} \quad (5)$$

where w_1 is the weight for the score calculated by user's requirement, w_2 is the weight for the score calculated by optimal values, Dis_{si} is the distance score for each services and $Score_{osi}$ is the score value for each services.

5. IMPLEMENTATION AND OBSERVED RESULTS

The proposed enhanced algorithms and baseline algorithms which includes SFS has been implemented as a console-based application in java using NetBeans6.1 under Windows environment. Different subsets derived from the QWS dataset provided by Al-Masri, and Mahmoud[15] was considered in this work. The original dataset includes information of over 2000 web services available on the Web. The dataset includes real data for various QoS attributes such as response time, availability, throughput, successability, reliability, compliance, best practices, latency and documentation. The service name and its WSDL address are also included in the dataset. Different scenarios are considered for the experiments to test the efficiency of proposed methods algorithm by using real datasets. Skyline algorithm is considered as the baseline for comparison with our improved algorithms in terms of efficiency.

The observed results are shown in the following tables and figures. The abbreviations used for each algorithms are: DS for the original Distance-based algorithm, DS_I for the improved Distance-based algorithm, BF for the original Borda Fuse, BF_Q for the Borda Fuse algorithm with consideration of user's query, and SFS for the Sort Filter Skyline algorithm. Table 1 and Figure 1 shows the execution time of different algorithms on various number of web services for numeric type attribute and Table 2 and Figure 2 shows the execution time of different algorithms on various number of web services for interval type attribute.

6. CONCLUSION AND FUTURE WORKS

To measure the efficiency of the improved algorithms, the average execution time of each algorithm is calculated by using various datasets with different number of QoS attributes from different data types. Then the improved methods are compared with the original methods in terms of the average execution time.

In order to validate the framework and compare the optimized algorithms with other models, extensive experiments were conducted on various datasets with different specifications. By increasing the size of datasets and the number of QoS attributes, all algorithms are compared in terms of the processing time. According to the observations, SFS is the fastest algorithm for small sized dataset and one QoS attribute. By increasing the size of the dataset and the number of attributes, SFS runs slower. On the contrary DS and DS_I are the fastest algorithms on the large sized datasets. BF and BF_Q run faster than SFS on large sized datasets with larger number of QoS attributes. Different experiments were performed with different type of attributes such as numeric, Boolean and data interval. It is noticed that SFS has a poor performance on attributes with data interval type. DS and DS_I with a slight different in execution time have the best performance for all data types when the size of the dataset is large.

As future work, the proposed framework may be improved to support top-*k* query processing effectively so that the processing time could be much lower, and users would be able to select their desired services easily.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES:

[1] T. Earl, "SOA: Principles of Service Design", 2nd Edition. Upper Saddle River, N.J., Prentice-Hall, ISBN: 9780132344821, 2008.

[2] W. Rong, K. Liu, and L.Liang, "Towards Personalized Ranking in Web Service Selection", in *Proceedings of the IEEE International Conference on e-Business Engineering*, Xi'an, China, pp.165-172, 2008.

[3] I. Toma1, D. Roman, D. Fensel, B. Sapkota, and J.M. Gomez, "A Multi-criteria Service Ranking Approach Based on Non-Functional Properties Rules Evaluation", in *Proceedings of the Fifth International Conference on Service- Oriented Computing*, Vienna, pp. 435-441, 2007.

[4] J. A. Aslam, and M. Montague, "Models for Metasearch", in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, pp. 276-284, 2001.

[5] B. T. Bartell, G. W. Cottrell , and R.K. Belew, "Automatic combination of multiple ranked retrieval system", in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY , pp. 173-181, 1994.

[6] R. Fagin, R. Kumar, and D. Sivakumar, "Efficient Similarity Search and Classification via Rank Aggregation",in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, New York, NY,pp. 950-961, 2003.

[7] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank Aggregation Methods for the Web", in *Proceedings of the 10th World Wide Web Conference*, New York, NY, pp. 613-622, 2001.

[8] D. Kossmann , S. Borzsony, and K. Stocker, "The Skyline Operator", in *Proceedings of the 17th InternationalConference on Data Engineering*, Heidelberg, pp. 421-430 , 2001.

[9] D. Papadias, Y. Tau, G. Fu, and B. Seeger, "Progressive Skyline Computation in Database Systems", *ACM Transactions on Database Systems (TODS)* , vol. 30, no. 1, pp. 41-82, 2005.

[10] J. Chomicki, B. Godfrey, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting", in *Proceedings of the 19th International Conference on Data Engineering*, Bangalore, pp. 71-719, 2003.

[11] H. Han, H. Jung, S. Kim, and H.Y. Yeom, "A Skyline Approach to the Matchmaking Web Service", in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, Shanghai,China, pp. 436-443, 2009.

[12] J. Yan, and J. Piao, "Towards QoS-based Web Service Discovery", in *Proceedings of the International Conference on Service Oriented Computing*, Sydney, pp. 200-210, 2009.

[13] J. Hu, C. Guo, H. Wang, and P. Zou, "Quality Driven Web Services Selection", in *Proceedings of the IEEE International Conference on e-Business Engineering* , 681-685, 2005.

[14] Y.M. Wang, and T.M.S. Elhag, "On the Normalization of Interval and Fuzzy Weights", *Journal of Fuzzy Sets and Systems*, vol. 157, no. 18, pp. 2456-2471, 2006.

[15] E. Al-Masri, Q.H. Mahmoud, "QoS-based Discovery and Ranking of Web Services", in *Proceedings of the 16th International Conference on Computer Communications and Networks*, Honolulu, HI, pp. 529 – 534, 2007.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Execution Time Of Different Algorithms On Various Number Of Web Services For Numeric Type Attribute

Type of attribute	No. of services	No. of attributes	DS	DS_I	BF	BF_Q	SFS
Numeric	50	1	203	208	212	218	189
		3	223	238	239	251	203
		5	223	238	292	294	218
		7	250	281	328	332	219
	100	1	216	218	223	226	193
		3	230	239	242	253	218
		5	266	281	297	298	219
		7	281	283	344	344	234
	200	1	232	236	242	246	203
		3	242	248	256	258	232
		5	268	283	298	303	263
		7	368	375	383	385	385
	300	1	243	247	249	249	248
		3	248	253	259	263	250
		5	313	329	391	394	263
		7	344	346	384	386	392
	500	1	250	266	268	272	248
		3	252	257	279	279	281
		5	266	281	283	284	293
		7	403	418	422	454	458

Journal of Theoretical and Applied Information Technology
31st January 2014. Vol. 59 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

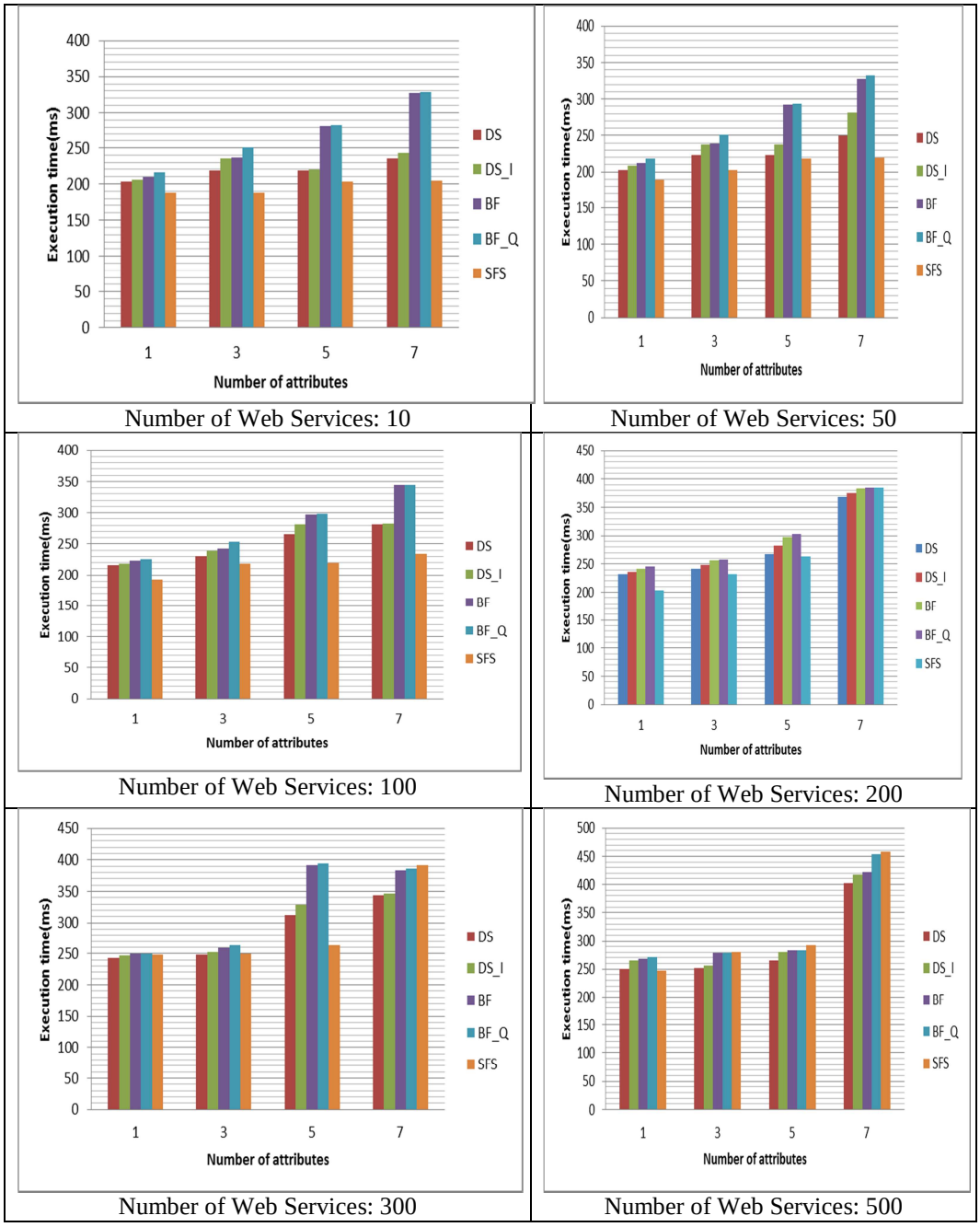

Figure 1: Execution Time Of Different Algorithms On Various Number Of Web Services For Numeric Type Attribute

Journal of Theoretical and Applied Information Technology
31st January 2014. Vol. 59 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Execution Time Of Different Algorithms On Various Number Of Web Services For Interval Type Attribute

Type of attribute	No. of services	No. of attributes	DS	DS_I	BF	BF_Q	SFS
Data Interval	50	1	141	143	156	158	144
		5	143	145	156	162	132
	100	1	160	163	168	171	163
		5	158	163	168	172	153
	200	1	188	189	203	206	190
		5	163	165	173	174	168
	500	1	193	208	212	218	215
		5	203	242	253	256	258

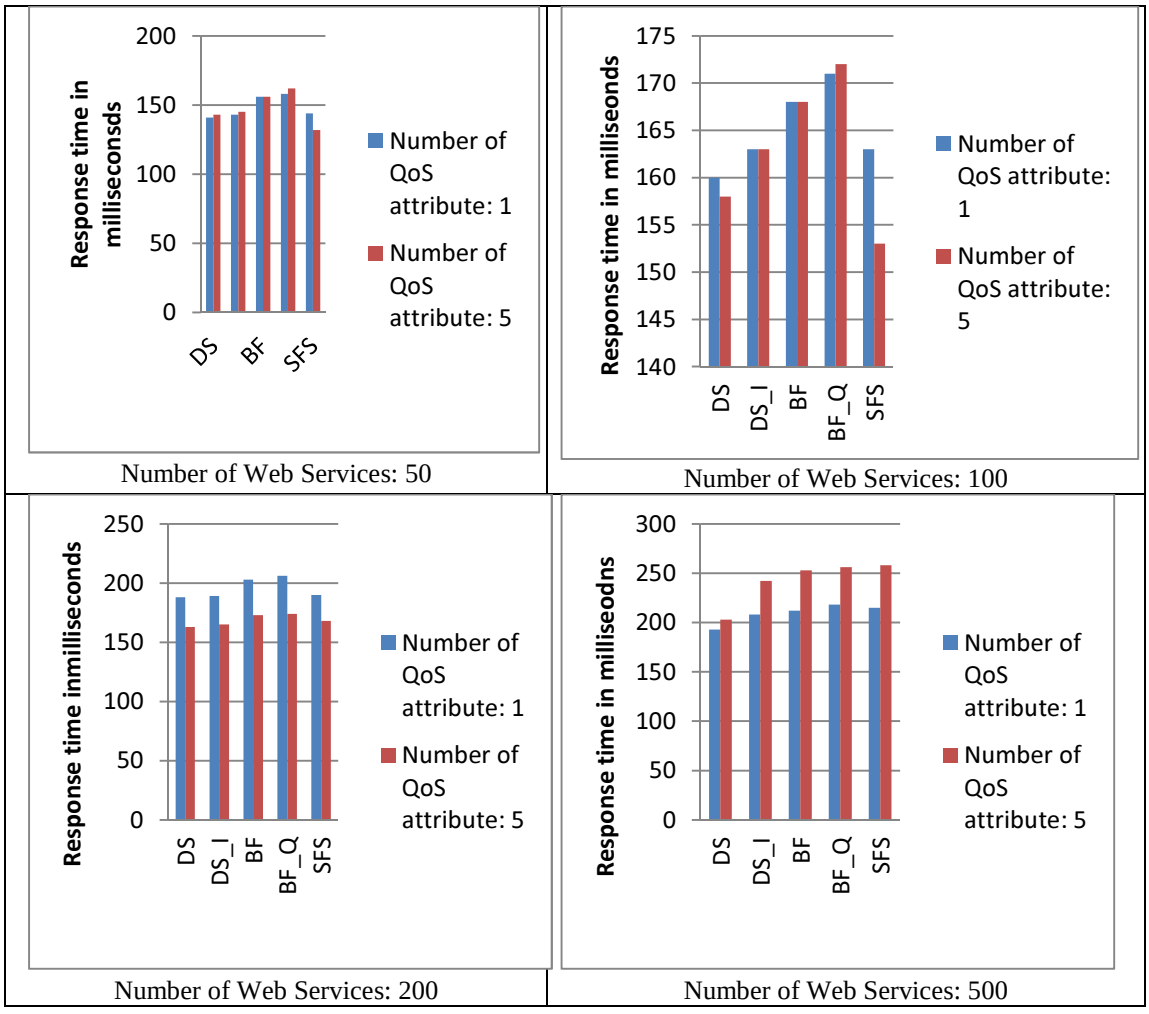

Figure 2: Execution Time Of Different Algorithms On Various Number Of Web Services For Interval Type attribute