QoS Aware Service Clustering to Bootstrap the Web Service Selection

Banage T. G. S. Kumara
Faculty of Applied Sciences
Sabaragamuwa University of Sri Lanka
Sri Lanka
btgsk2000@gmail.com

Incheon Paik, T. H. A. S. Siriweera
School of Computer Science and Engineering
University of Aizu, Aizu-Wakamatsu,
Fukushima, Japan
paikic@u-aizu.ac.jp, siriweera@gmail.com

Koswatte R. C. Koswatte
Sri Lanka Institute of
Information Technology,
Sri Lanka
ckoswatte@gmail.com

***Abstract*— Web services have been extended to give value-added customized services to users through service composition. Service Composition consists of four stages: planning, discovery, selection, and execution. Web service discovery and selection are becoming challenging and time-consuming tasks due to large number of Web services available on the internet. Organizing Web services into clusters of similar services to aid the pruning of the query space is one of the solutions for the issues. Web services can be clustered into similar groups by considering functional attributes to increase the performance of the service discovery. In selection stage, algorithm has to select a single service among the large number of functional equal services by considering the Quality of Service (QoS) values. In this paper, we propose QoS aware service clustering approach to increase the performance of the service selection. Here, service clusters are created using functionality as the first factor, then QoS properties being considered as secondary factors. Our approach considers QoS profit values to compute the QoS similarity. In this paper, we apply a spatial clustering technique called the Spherical Associated Keyword Space which is projected clustering result from a three-dimensional sphere to a two dimensional (2D) spherical surface for 2D visualization. Empirical study of our approach has proved the effectiveness of clustering results.**

Keywords: Web service selection, Service clustering, QoS similarity, Service Visualization

I. INTRODUCTION

Web services are loosely coupled software components that are popular implementation of the service-oriented architecture. Existing technologies for the Web services have been extended to give value-added customized services to users through service composition [1]. Service Composition consists of four stages: planning, discovery, selection, and execution [1]. The core idea is to orchestrate existing services to achieve a larger task, resulting in a new composite and value-added Web service. Developers and users can then solve complex problems such as travel planners by combining the available basic services. In the planning stage, an abstract workflow is constructed. For each workflow task, the discovery stage tries to identify available related services as candidates. Set of functionally equivalent services with varying Quality of Service (QoS) values can be identified. In the selection stage, a QoS-aware service selection algorithm selects optimal services for each task. In the execution stage, the selected services are operated in the workflow.

However, Web service discovery and selection are becoming challenging and time-consuming tasks due to large number of Web services available on the internet. Organizing Web services into clusters of similar services to aid the pruning of the query space is one of the solutions for the issues. Many researchers adopted the functional based clustering algorithms to bootstrap the service discovery [2]. They considered the similarity of functional properties such as *service name*, *input* and *output* in clustering process. In our previous work, we proposed Hybrid Term Similarity (HTS) based clustering approach [3] to cluster the services by considering the functional properties of services. Our core idea was to increase the performance of the discovery stage.

In contrast to that work, in this paper we focus on QoS properties of the services. QoS is a set of attributes describing the non-functional properties of Web services, such as *cost*, *availability*, *reliability*, etc. QoS attributes can be divide into two categories namely: user independent and user dependent. Values of the user independent QoS attributes such as *cost* and *popularity* are identical for different users. User dependent QoS properties such as *response time* vary from user to user, influenced by the Internet connections and heterogeneous environments. As we mentioned, service selection algorithm has to select a single service among the large number of functional equal services by considering the QoS values. Further, QoS plays an important role in other related issues such as service recommendation [4].

In this paper, we propose a QoS aware service clustering approach to increase the performance of service selection and recommendation by reducing the search space. Here, service clusters are created using functionality as the first factor, then QoS properties being considered as secondary factors. In QoS aware clustering first, we calculate the QoS profit values of each attribute of the services. Then, affinity values between the services are calculated using the Euclidean distance based method. Finally, Spherical Associated Keyword Space (SASKS) algorithms [5] is applied to cluster the services. The algorithm aids to search Web services by visualization of the service data on a spherical surface and it is effective for noisy data. User can use QoS based clusters to identify good,

2474-2473/17 $31.00 © 2017 IEEE
DOI 10.1109/SCC.2017.37

moderate, or poor instances from a collection of functionally equal services.

The rest of this paper is organized as follows. In section II, we present the architecture of the proposed approach. Section III discusses functional based clustering process. Section IV describes our proposed QoS aware clustering approach. Section V discusses our experiment and its evaluation. In Section VI, we discuss related works. Finally, Section VII concludes the paper.

II. Motivations and Proposed Approach

A. Motivating Example

Assume we want to select or recommend a Web service from collection of functionally equivalent services with varying QoS values. Then, selection algorithm has to select a service with optimal QoS values from all the services in the search space. However, if we cluster the services based on QoS values, then we can reduce the search space as in Fig 1. First, we can identify the cluster with better QoS value and then, selection algorithm can limit the searching process within the cluster.

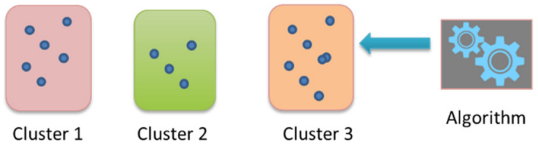


Figure 1. Motivating example

B. Proposed Approach

Figure 2 shows a block diagram of the architecture of the proposed approach. First, we cluster the services by considering the services functionality. Here, HTS [3] method is used to compute the semantic similarity between the services. SASKS algorithm [5] is used as the clustering algorithm. Then as the next step, we extract the clusters and compute the QoS similarity values between the services in each cluster. To calculate the QoS similarity, first we generate QoS vector for each service and employ Euclidean distance based similarity calculation approach. Next, affinity matrix is generated using the QoS similarity values. Finally, services are clustered and plotted onto the sphere using the SASKS algorithm according to the QoS similarity values.

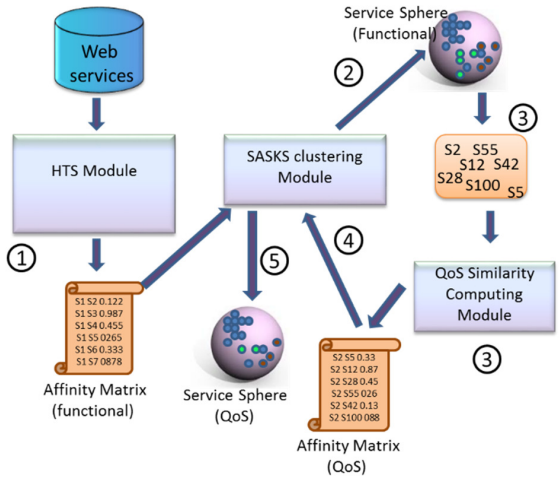


Figure 2. Architecture of the proposed approach

III. Functional Based Service Clustering

As we explained in the previous section, we apply functional based clustering approach as the first step. In this chapter, first, we explain the procedure of functional based similarity calculation method and then, we explain about the clustering algorithm.

A. Functional Based Similarity Calculation

We proposed HTS method to calculate the functional similarity between services in our previous work [3]. In this work also, HTS method is used to calculate the functional similarities of services,

Summary of HTS method: HTS approach first adopts ontology-learning method to generate ontologies via the hidden semantic patterns existing within the complex terms. If calculating similarity using the generated ontology fails, it then applies an Information Retrieval (IR) based method. HTS approach is able to capture the hidden semantics via an ontology-learning phase that generates ontologies for service domains. In ontology generation step, we defined rules namely *Head–Modifier Relation Rule*, *Compound Noun Rule, Concept and Modifier Rule* and *Modifier Only Rule* to generate concepts and relationships between concepts. Then, we used different logic filters namely: *Exact*, *Property-&-Property*, *Property-&-Concept*, *Plug-in*, *Sibling*, *Subsumes*, *Logic Fail* and *Fail* to calculate the similarity. However, if the concepts belong to different ontologies, then ontology learning method fails to compute the semantic similarities between services. Thus, we used IR based method as an alternative to compute the semantics similarity. WordNet and search engine based techniques were used as the IR based methods. Empirical study of our HTS based clustering approach has proved the effectiveness of clustering results. Thus, HTS method is used to calculate functional similarity in this work.

In calculating the semantic similarity, first we extract *service name*, *operation name*, *input* and *output* as the functional attributes. Then, individual feature similarity values of the services are calculated using HTS method. Finally, individual feature similarity values are integrated using following (1) to calculate the final functional similarity values between the services.

$$FuncSim(S_i, S_j) = W_N \times Sim(Sname_i, Sname_j) + W_{ON} \times Sim(ON_i, ON_j) + W_I \times Sim(In_i, In_j) + W_{OP} \times Sim(Out_i, Out_j) \quad (1)$$

Here, $Sname_i$, ON_i, In_i, and Out_i are *service name*, *operation name*, *input* and *output* of service *i* respectively. Further, *sim*(,) represents a particular feature similarity value between service *i* and *j*. Weights for each feature elements, W_N, W_{ON}, W_I and W_{OP} are real values between 0 and 1, with $W_N + W_{ON} + W_I + W_{OP} = 1$.

B. Clustering Algorithm

In this paper, we apply a spatial clustering Algorithm called SASKS [5] to cluster the services. The SASKS algorithm is an extended version of ASKS algorithm. ASKS algorithm is able to represent services in 3D space and we can get a visual output. However, this 3D form is difficult to visualize on a 2D screen. We therefore apply the SASKS technique proposed in [5] as our clustering approach. The SASKS technique plots services onto a 2D spherical surface for easy visualization on a 2D screen. We need to give an affinity matrix as the input for the SASKS algorithm. The affinity matrix is created using the calculated semantic similarity values of the services.

We can represent the distance D_{ij} between two services using following (2):

$$D_{ij} = -f\left(x_j^{(k)} - x_i^{(k)}\right), \quad (2)$$

where x_i and x_j are the locations of the services *i* and *j*, respectively. k represents the dimension of the space in which the services are located.

Further, function f can be defined using (3):

$$f(\theta) = \begin{cases} |\theta|^2, & |\theta| < a \\ 2a|\theta| - a^2, & |\theta| \geq a \end{cases} \quad (3)$$

Here, parameter *a* is the density-control parameter.

Finally, following iterative computation (4) is used to calculate the new service locations on the sphere.

$$x_i^{(k)}(t+1) = \frac{\sum_{j=1}^{n} M_{ij} \left\{ D\left(x_j^{(k)}(t) - x_i^{(k)}(t)\right)\left(x_j^{(k)}(t)\right)\right\}}{\sum_{j=1}^{n} M_{ij} D\left(x_j^{(k)}(t) - x_i^{(k)}(t)\right)} \quad (4)$$

Here, x_i: $ij = 1,2\ldots\ldots,n$, $k = 1,2,3,\ldots,p$ and $t = 1,2,\ldots$ Further, M_{ij} is the affinity value between service *i* and *j*.

SASKS algorithm cluster services based on domain and services of the same domain are arranged into the same region.

IV. QoS Aware Service Clustering

As we explained in the section III, we cluster the services based on functional similarity using HTS method and SASKS algorithm. In service discovery, first we can identify the most suitable service cluster for the user request. Then, candidate services are identified from that cluster. Thus, functional based clusters can be used to avoid many unnecessary similarity calculations in the service discovery process. However, the clusters contain functionally equivalent services with varying QoS values. In service selection, when several target services have similar estimation of functional similarity for a given request *R*, the QoS properties become a critical issue to provide better quality service for the *R*. Thus, we have to consider about the QoS attributes. Here, we cluster the services within the functional cluster by considering QoS attributes. First, we extract the relevant functional based cluster and then cluster the services by generating QoS affinity matrix and using SASKS algorithms. We consider two methods to calculate the affinity values. In the first method, affinity values are calculated without considering the QoS profit values. In the second method, affinity values are calculated considering the profit values. The two methods are compared in section V. In this section, first we explain the first method and then explain the steps for profit based affinity calculation method.

We select QoS attributes such as *availability, cost, throughput* and *reliability. Availability, throughput,* and *reliability* are considered as positively affecting criteria. *Cost* is considered as the negatively affecting criteria. The *availability* of a service is the probability that a service is accessible to use. *Throughput* is the maximum amount of requests that the service provider can process in a given period. *Reliability* refers to the probability of given request successfully invoke the given service. *Cost* is the charge that a service requester has to pay for invoking the service.

A. QoS Affinity Matrix Generation (without profit) and Clustering

As we mentioned, a service has several QoS attributes. Thus, QoS vector of service S_i is defined as follows;

Definition 1 (QoS Vector). *QoS vector of a service candidate S_i can be define as $S_iQoS = \{ Q_1(S_i), \ldots Q_r(S_i),.. Q_n(Si)\}$ and $Q_r(S_i)$ represents the estimated value of the r-th QoS attribute of service S_i and determines the service quality.*

Example 1: Suppose that there is a service S_1 with *availability* value 0.5, *throughput* value 0.7, *reliability* value 0.6, and *cost* value 0.3. Then QoS vector for service S_1 is $S_1QoS = \{ 0.5, 0.7, 0.6, 0.3\}$.

Then, to generate the QoS affinity matrix first we calculate affinity values between the services. Affinity value between service S_i and S_j is defined as follows;

Definition 2 (Affinity Value). *Affinity value $AV(S_i, S_j)$ between the services S_i and S_j represents the QoS closeness*

between the services. Euclidean Distance is used to compute the similarity and denoted as follows:

$$AV(S_1, S_2) = 1 - ED(S_i - S_j) \quad (5)$$

Where,

$$ED(S_i - S_j) = \sqrt{\sum_{r=1}^{n} [\, Q_r(S_i) - Q_r(S_j)\,]^2} \quad (6)$$

Here, $ED(S_i - S_j)$ is Euclidean Distance between two services and $Q_r(S_i)$ represent the r-th QoS value of service S_i. Further *n* is the number of QoS attributes of a service.

Example 2: Suppose that there are two services S_1 and S_2 with QoS vectors S_1QoS = {0.5, 0.7, 0.6, 0.3} and S_2QoS = {0.6, 0.6, 0.7, 0.3} respectively. Then using (5) and (6), Affinity value $AV(S_1, S_2)$ between services S_1 and S_2 can be calculated as follows;

$$AV(S_1, S_2) = 1 - \sqrt{0.03} = 0.83$$

According to the results, the services are close to each other. Let's consider another service S_3 with QoS vector {0.3, 0.2, 0.4, 0.3}. Then, $AV(S_3, S_2)$ between services S_3 and S_2 is 0.41. Thus, S_3 is far away from S_2 compare to S_1 according to the QoS values.

After calculating the Affinity values between the services, Affinity matrix is generated as shown in Fig. 3. Here, S_i represents the i-th service. Value of S_iS_j position represents the Affinity value between service S_i and S_j.

	S_1	S_2	S_3	S_4	S_5
S_1	-	0.33	0.87	0.21	0.11
S_2	0.33	-	0.42	0.55	0.84
S_3	0.87	0.42	-	0.34	0.74
S_4	0.21	0.55	0.34	-	0.88
S_5	0.11	0.84	0.74	0.88	-

Figure 3. Sample affinity matrix

Then, we use the Affinity matrix to cluster the services based on QoS values. In this step also, SASKS algorithm is used to cluster the services as in functional clustering steps. We have explained about the algorithm in section III. We use value $AV(S_i, S_j)$ as the M_{ij} in (4).

B. Profit Based QoS Affinity Matrix Generation and Clustering

Above method does not consider about the category (positive or negative) of the QoS attributes in calculating the Affinity value. However, we have to handle the positive and negative attributes in different ways to cluster the services as good, moderate, or poor clusters according to the QoS value. Normally, in service selection positive QoS values needs to be maximized and negative QoS values needs to be minimized. Thus, next we proposed QoS profit based method to cluster the services to overcome the above issue.

Instead of using QoS vector, we use QoS profit vector to generate the Affinity values in this method. QoS profit vector is based on individual QoS attribute values and it is defined as follows;

Definition 3 (QoS Profit Vector). *QoS profit vector of a service candidate S_i can be define as S_iQoSP = { $Q_1P(S_i)$, … $Q_rP(S_i)$,.. $Q_nP(S_i)$} and $Q_rP(S_i)$ represents the QoS profit value of the r-th QoS attribute of service S_i and determines the service quality.*

To calculate the normalized profit values of positive QoS attributes, we use the following (7). Further, normalized profit values of negative QoS attributes are calculated by (8) [6].

$$Q_rP(S_i)^+ = \frac{Q_r(S_i) - Q_r^{min}}{Q_r^{max} - Q_r^{min}} \quad (7)$$

$$Q_rP(S_i)^- = \frac{Q_r^{max} - Q_r(S_i)}{Q_r^{max} - Q_r^{min}} \quad (8)$$

Here, $Q_rP(S_i)^+$ and $Q_rP(S_i)^-$ are positive and negative QoS profit values of r-th QoS attribute in service S_i. Q_r^{max} and Q_r^{min} represent the maximum and minimum values of r-th QoS attribute among all the services. However, if $Q_r^{max} - Q_r^{min} = 0$, then we consider profit value as 1 in both cases.

Example 3: Suppose that we want to generate QoS profit vector for service S_1 in the example 1. The service has values 0.5, 0.7, 0.6 and 0.3 for *availability, throughput*, *reliability,* and *cost* respectively. Assume that the maximum and the minimum values of each QoS attribute values are 0.9 and 0.1 from all the services. Here, *availability, throughput* and *reliability* are positive QoS attributes, whereas cost is a negative attribute. Thus, profit value of *availability* is calculated using (7) as follows;

$$Q_rP(S_{availability}) = \frac{0.5 - 0.1}{0.9 - 0.1} = 0.5$$

Further, $Q_rP(S_{Throughput})$ and $Q_rP(S_{Reliability})$ can be obtained as 0.75 and 0.63 from (7) and profit value of *cost* is calculated as follows using (8).

$$Q_rP(S_{cost}) = \frac{0.9 - 0.3}{0.9 - 0.1} = 0.75$$

Then, QoS profit vector for service S_1 can be represented as S_iQoSP = {0.5, 0.75, 0.63, 0.75}.

We can observe the difference between QoS vector and QoS profit vector from the example 1 and the example 3. Value in the relative position of the *cost* attribute of QoS profit vector is higher compare to the value of QoS vector. Here, *cost* of the S_1 is a low value. Thus, profit value of the *cost* must be high. We can use the advantage of this profit calculation during the clustering process.

After creating the QoS profit vectors for each service, we use the vectors to calculate the affinity values between the services. Here also we use (5) and (6) to calculate the affinity values as in above sub section. However, instead of using QoS values of service $S_i (Q_r(S_i))$, QoS profit values ($Q_rP(S_i)$) are used to calculate the Affinity in (6). Then, Affinity matrix is generated using calculated Affinity values as shown in fig. (3). Finally, services are plotted into a 3D sphere based on QoS profit values using the generated Affinity matrix and the SASKS algorithm.

V. Experiments and Evaluation

The experiments were conducted on a computer running Microsoft Windows 10, with an Intel core i7-4702MQ, a 2.20 GHz CPU and 8 GB RAM. Java was used as the programming language to implement the program for affinity calculations, and the SASKS algorithm was implemented using MATLAB. First, we applied the functional based clustering method and analyzed the visual output of clusters for different domains. Here, WSDL documents related to the *Book*, *Medical*, *Food*, *Film*, and *Vehicle* domains were gathered from real-world Web service repositories, and the OWL-S [7] test collection as the services dataset. We manually assigned QoS values for the services. Then, we extracted clusters by clusters and applied the QoS aware clustering method. First, we analysed the visual output of first method and then, profit based method was evaluated with visual outputs. However, in this paper we select only *vehicle* cluster to analyse the results due to the space limitation.

A. Visualization of Functional Based Clusters

We calculated the similarity of Web services by HTS method to calculate the functional similarity. Then, the Affinity matrix was generated. The matrix was used as input for the SASKS clustering algorithm. Web services were placed into five main separated regions in the sphere. Most of the same domain services were placed into same region. We briefly analysed and discussed about the clusters in our previous work [8]. Thus, here we do not discuss about the results in details. Figure 4 shows the visualization of the functional based clusters. It shows the service sphere and part of the *Vehicle* cluster. We can observe *Vehicle* domain services such as *JaguarCarPrice* and *ThreeWheeledCarPrice, ExpensiveCarPrice, CarPrice* and *ToyotaPrice* in that region. In the bottom region of the sphere, we observed *Medical* domain services such as *InformHospital* and *HospitalPhysian*.

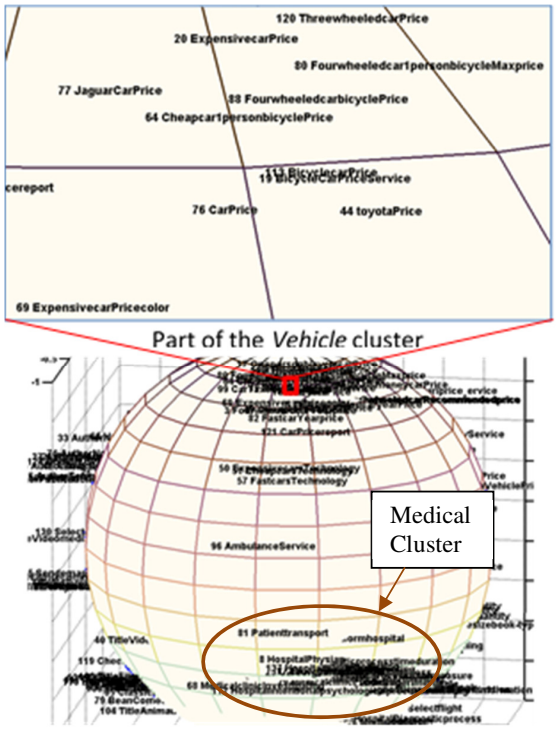


Figure 4. Visualization results of the functional based clustering

B. Visualization of QoS Aware (without profit) Clusters

After the functional based clustering, we extracted the clusters for QoS aware clustering. First, we clustered the services based on QoS attribute values without calculating the profit. Figure 5 shows service sphere for the QoS aware clustering of vehicle domain services. In this case, we observed four main regions.

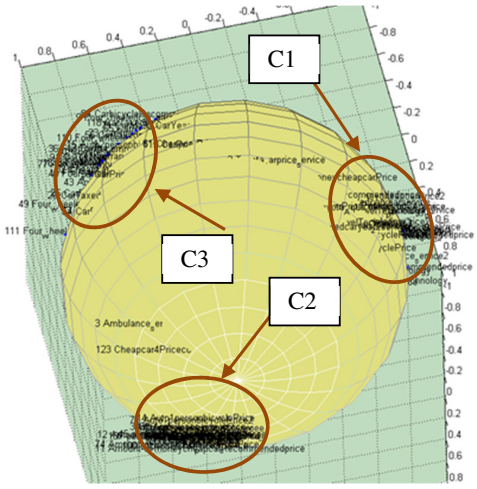


Figure 5. Service sphere of QoS aware (without profit) clustering

Figure 6 shows a part of cluster *C1* in above fig. 5. Further, Fig 7. shows a part of cluster *C2* in above fig 5. *C1* contains services such as *CarReport* and *CarMaxPrice*. Cluster *C2* contains services such as *CarsTechnology* and *AmbulanceService2*. Moreover, *CarTaxedPrice3Report* (highlighted service in fig. 6) service is in C1. But, *CarTaxedPriceReport* service (highlighted service in fig. 7) is in *C2*. These two services are identical services with functional similarity value 1. But, the services have different QoS values. QoS vector for *CarTaxedPrice3Report* is {0.3, 0.2, 0.3, 0.6} and QoS vector for *CarTaxedPriceReport* is {0.7, 0.9, 0.8, 0.8}. Thus, the services are in two separate clusters. In the vector 0-th position represents the *availability* value. Further, 1-st, 2-nd and 3-rd positions represent the *cost*, *throughput* and *reliability* values respectively. We used the same format to represent the QoS vector in the rest of this paper.

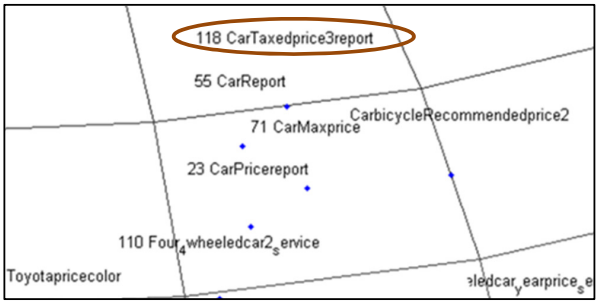


Figure 6. Part of cluster *C1* in fig. 5

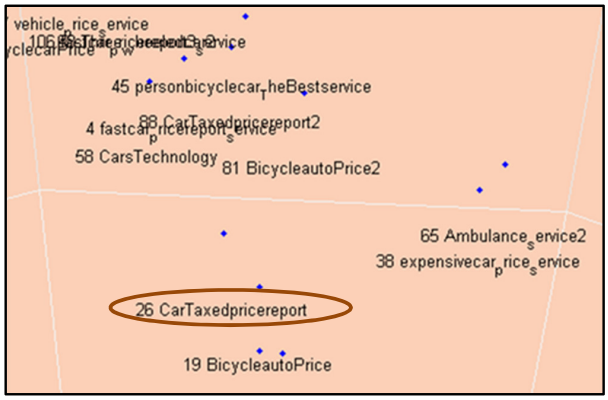
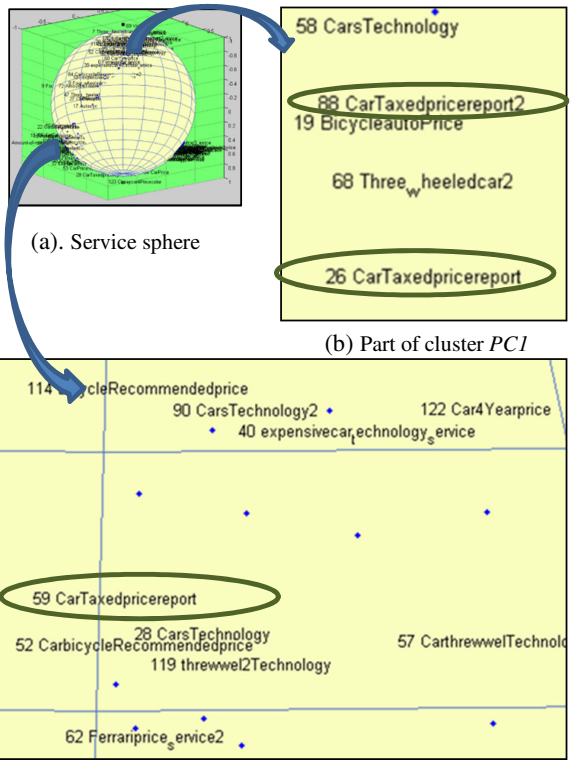


Figure 7. Part of cluster *C2* in fig. 5

C. Visualization of Profit Based QoS Aware Clusters

As we mentioned, the above method does not consider about QoS profit values in clustering process. But, our profit based QoS aware clustering method used individual QoS profit values. This sub section discusses the visualization results of the method. In this method, we identified four main regions as in previous method. Figure 8 (a) shows the services sphere and fig. 8 (b) shows part of the profit based cluster *PC1*.



(a). Service sphere

(b) Part of cluster *PC1*

(c) Part of cluster *PC2*

Figure 8. Visualization results of the profit based QoS aware clustering

Further, fig. 8 (c) shows part of the profit based cluster *PC2*. Here also we observed that some services with high functional similarity are in different clusters. For example, there are services with service ID 88, 26 and 59 (see highlighted areas in fig.8). Name (*CarTaxedPriceReport*) and functionality of the services are functionally identical and functional similarity is 1. But, service with ID 59 is in cluster *PC2* and others are in *PC1* due to the QoS variation. Furthermore, *CarsTechnology* service with ID 58 is in *PC1* and *CarsTechnology2* service with ID 90 is in *PC*2. When we analyzed the QoS profit vectors of the services, we observed that the services have totally different QoS profit values. QoS profit vector of service 58 is {0.88, 0.13, 1.0, 0.88} and vector {0.25, 0.88, 0.25, 0.25} is the QoS profit vector of service 90. Here, service 58 has high QoS profit values for all QoS attributes except *cos*t. But, service 90 has low QoS profit values for all QoS attributes except *cos*t. As a result of this, the services are placed into two separate clusters.

According to the fig. 7, *CarsTechnology* (ID 58) service is placed near to *FastCarPriceReport* (ID 4) and *BicycleautoPrice2* (ID 81) services. But, according to the fig. 8 (a) the services are not placed near to *CarsTechnology* service. Because, the services have similar QoS values compare to QoS profit values. In Table I, we listed some cluster members of each cluster.

Table I. List of sample members of each cluster

Manually Assigned Cluster	ID	Service Name	QoS vector	Without QoS Profit	Profit Based Clustering	Avg Profit
Cluster A [15 – 35]	17	AutocyclePrice	[0.4, 0.8, 0.4, 0.2]	A	A	0.25
	27	CarthrewwelTechnology	[0.1, 0.8, 0.3, 0.3]	A	A	0.16
	33	CheapcarPrice	[0.8, 0.7, 0.1, 0.2]	C	A	0.31
Cluster B [36 – 50]	9	Four_wheeledcar_yearprice(2)	[0.2, 0.3, 0.2, 0.8]	B	C	0.47
	42	Toyota_carservice	[0.5, 0.6, 0.5, 0.4]	B	B	0.44
	22	CarbicycleRecommendedprice	[0.2, 0.5, 0.6, 0.4]	A	B	0.41
	30	CarYearprice	[0.3, 0.6, 0.5, 0.4]	B	B	0.38
Cluster C [51 –65]	75	CarMaxPrice3	[0.7, 0.4, 0.5, 0.3]	C	C	0.53
	97	FastcarPrice3report	[0.6, 0.2, 0.3, 0.6]	A	C	0.59
	82	BicyclecarPriceyear2	[0.9, 0.1, 0.2, 0.4]	C	C	0.63
	86	CarPrice3	[0.6, 0.2, 0.3, 0.6]	A	C	0.59
Cluster D [66 –80]	4	fastcar_pricereport	[0.8, 0.9, 0.9, 0.8]	D	D	0.69
	113	Bicyclecar3Price	[0.1, 0.8, 0.9, 0.8]	D	D	0.78
	37	vehicle_price_service	[0.9, 0.8, 0.9, 0.9]	D	D	0.78
	80	Bicycle4wheeledcarPrice2	[0.7, 0.1, 0.2, 0.7]	C	D	0.66

Here, we first performed a manual classification to categorize the Web service data set based on the average QoS profit values. Column 1 in Table I shows the cluster label and profit range of the each cluster. Cluster *A* has services with low average profit values. Thus, we can consider it as a poor cluster. We can consider cluster *D* as the best cluster compared to other clusters. Because, the cluster has services with higher average QoS profit values. Cluster *C* and *D* are moderate clusters.

According to the Table I, some services are in two different clusters in two methods. *CheapcarPrice* service is a member of cluster *A* according to the profit based method. Further, it is in same cluster according to the manual classification. But, it was incorrectly placed into cluster *C* without profit based method. *Four_wheeledcar_yearprice(2)* is belonging to the cluster B according to the manual classification. But, it was incorrectly clustered into cluster C with the profit based clustering method. However, when we analyzed the table, we can see that more services were incorrectly placed into wrong clusters in the without profit based method compared to profit based method. For an example, *CarbicycleRecommendedprice, FastcarPrice3report, CarPrice3* and *Bicycle4wheeledcarPrice2* services were placed into wrong clusters with that method.

In the next evaluation step, we used precision, recall and and F-measure to measure the cluster performance. Precision is the fraction of a cluster that comprises services of a specified class. Recall is the fraction of a cluster that comprises all services of a specified class. The F-measure measures the extent to which a cluster contains only services of a particular class and all services of that class.The experimental results in Table II show that the profit based QoS aware clustering approach obtained highest values for every evaluation criteria compare to the without profit base method. The profit based approach improved the precision of the cluster *C* by 36.1%. Further, the method improved the recall of the cluster *A* by 46.6% and F-measure of the cluster *B* was improved by 36.1%. Moreover, cluster *B* and *C* obtained very low values for every criteria in without profit based method. The clusters contained more noisy data compare to profit based method.

Table II: Performance measures of clusters

Cluster	QoS Aware based clustering (Without Profit)			Profit based QoS aware clustering		
	Precision %	Recall %	F-Measure%	Precision %	Recall %	F-Measure%
Cluster A	75.0	41.4	53.4	76.3	88.0	81.7
Cluster B	41.4	52.1	46.1	85.1	79.3	82.1
Cluster C	50.0	56.1	52.9	86.1	87.5	86.8
Cluster D	70.0	70.0	70.0	80.8	86.7	83.6

Thus, according to the result, profit based approach improved the performance of QoS aware service clustering.

VI. Related Works

A. QoS aware Web Service Clustering

Clustering the Web services enables the user to identify appropriate and interesting services according to his or her requirements in service discovery while excluding potential candidate services outside the relevant cluster. Further, we can use service clusters to increase the performance of service recommendation. Because, it enables efficient browsing for similar services within the same cluster. Current clustering approaches can be classified by considering the properties used in the clustering process: Most previous work focuses on the functionally based clustering approaches, considering the semantics of functional properties such as operations, and their input and output [2]. Non-functionally based clustering approaches cluster the services considering QoS attributes such as availability and response time. Research work [9] proposed QoS-based clustering approach to select the best Web service for the service requesters. The approach filtered irrelevant services according to the specified QoS constraints and calculated the total QoS utility score for each filtered candidate. Then it generated the cluster for relevant Web services. In addition, Xia et al. [10] presented an algorithm that used clustering technique to cluster a huge number of Web services into a number of groups according to QoS properties. Further, Zhu et al. [11] proposed clustering-based QoS prediction solution for Web services' recommendation system. However, in our approach, we used method to calculate the affinity values by considering negative and positive QoS values. Then, compute the QoS similarity to cluster the services.

B. QoS aware Service Selection and Recommendation

Gao et al [12] proposed trust-oriented genetic algorithm to find a near-optimal service composition plan with QoS constraints. The approach contained, trust evaluation method for the service composition plan based on the subjective probability theory. Further, Wang et al. [13] proposed a service recommendation approach to improve the efficiency of QoS-aware service selection for multi-tenant Software as a service. Zheng et al. [4] proposed approach to predict Web service QoS values based past web services QoS information. Their objective was to implement collaborative filtering approach by taking advantages of past usage experiences of service users.

VII. Conclusion

In this paper, we have proposed a QoS aware Web service clustering approach to increase the performance of service selection and recommendation. First, we calculated the QoS profit values of each attribute. Then, affinity matrix was generated based on QoS profit values. To calculate the affinity values between services, we used the Euclidean distance based method. Finally, the SASKS algorithm was used to cluster the Web services. Our objective was to plot the services on sphere to visualize the clusters. The algorithm clustered the service on service sphere based on QoS values. To compare the profit-based method, we implemented another clustering method without computing the QoS profit values and by taking only the QoS values. Experimental results show that profit based method outperforms without profit based QoS aware clustering approach. Average precision value of without profit-based method was 59.1% and it was increased by 23.0% in profit-based method. Further, recall value was increased by 30.5%.

In our future work, we plan to conduct a thorough evaluation of our QoS aware Web service clustering approach over existing methods. Further, we also plan to handle user dependent QoS values and user independent QoS values separately to calculate the Affinity values.

References

[1] I. Paik, W. Chen, and M. N. Huhns, "A scalable architecture for automatic service composition," IEEE Transactions on Services Computing, vol. 7, no. 1, pp. 82–95, Jan.–Mar. 2014.

[2] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering WSDL Documents to Bootstrap the Discovery of Web Services," in Proc. 8th IEEE International Conference on Web Services, 2010.

[3] B. T. G. S. Kumara, I. Paik, W. Chen and K. Ryu, "Web Service Clustering using a Hybrid Term-Similarity Measure with Ontology Learning," International Journal of Web Services Research (JWSR), Vol. 11, No. 2, pp. 24 - 45, 2014.

[4] Z. Zheng, H. Ma, M. R. Lyu, I. King, "Qos-aware Web service recommendation by collaborative filtering", IEEE Transactions on Services Computing, vol. 4, pp. 140-152, 2011.

[5] T. Sasaki, Y. Yaguchi, Y. Watanobe and R. Oka, "Extracting a Spatial Ontology from a Large Flickr Tag Dataset," In Awareness Science and Technology (iCAST), 2012 4th International Conference on IEEE, 2012.

[6] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition", IEEE Trans. Softw. Eng., vol. 30, pp. 311-327, 2004.

[7] http://projects.semwebcentral.org/projects/owls-tc.

[8] Banage T. G. S. Kumara, Yuichi Yaguchi, Incheon Paik and Wuhui Chen, "Clustering and spherical visualization of Web services," IEEE International Conference on Services Computing, 2013.

[9] R.Karthiban, "A QoS-Aware Web Service Selection Based on Clustering", International Journal of Scientific and Research Publications, Vol 4, No. 2, 2014.

[10] Y. Xia, P. Chen, L. Bao, M. Wang, and J. Yang, "A QoS-Aware Web service selection algorithm based on clustering," in Proc. 9th IEEE International Conference on Web Services, pp. 428–435, 4–9 Jul. 2011.

[11] J. Zhu, Y. Kang, Z. Zheng, and M. R. Lyu, "A clustering-based QoS prediction approach for Web service recommendation," in Proc. 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, , pp. 93-98, 2012.

[12] H. Gao, J. Yan and Y. Mu, "Trust-oriented QoS-aware composite service selection based on genetic algorithms", Concurrency and Computation: Practice and Experience, vol. 26, no. 2, pp. 500-515, 2014.

[13] Y. Wang, Q. He and Y. Yang, "QoS-Aware Service Recommendation for Multi-tenant SaaS on the Cloud," IEEE International Conference on Services Computing, New York, pp. 178-185, 2015.