International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A QoS Aware Cloud Service Composition Algorithm for Geo-Distributed Multi Cloud Domain

S. Bharath Bhushan[1*] Pradeep Reddy CH[2]

[1] VIT University, Vellore, India
[2] VIT University, Vellore, India
* Corresponding author's Email: bharath.bhushan4@gmail.com

Abstract: Publishing the cloud services on the internet has proven to be effective for cloud providers over the internet. To meet user requirements, services should be combined through service composition from multiple cloud domains. In this paper, we proposed an algorithm which composes best QoS aware services with minimal number of cloud combinations which is an NP-hard optimization problem. The proposed algorithm will select a cloud with more number of service files and then we applied PROMETHEE a multi criteria decision making method that selects the best service based on QoS criteria (i.e. Response time, Throughput, Availability, Successability, Price) from optimal cloud combinations. Our experimental results show that the algorithm provides better QoS services with minimal number of clouds by assessing previous benchmarks.

Keywords: Service Composition; Cloud Computing; Multi-cloud Domain; Multi Criteria Decision Making; Quality of Service.

1. Introduction

Cloud computing is a leading platform for providing elastic web services, which helps for the seamless composition of enterprise applications to create new value added services [1]. Today, users and enterprises are increasingly using the cloud to access software resources in the form of web services [2]. As web services are self-contained, loosely coupled processes deployed over standard middleware platforms that can be described, published, discovered and invoked over a network.

The different service providers will publish a large number of similar services with different quality of service in various clouds [3]. QoS parameters can be used to select the best service among similar services and will make a QoS aware service composition [4], which is an MCDM (multi criteria decision making) problem. The traditional service composition methods compose services from single cloud, which won't always meet user requirements. There is a need to compose the services from multiple clouds [5,6] by taking advantage of different QoS levels.

The proposed QoS aware service composition is based on PROMETHEE [7] (Preference Ranking Organization METHod for Enrichment Evaluation) which is MCDM solution. All the web services in a composition sequence are now QoS aware services. The main objective is to find the best service composition [8] by taking QoS parameters into consideration with minimal number of clouds. As the services that are communicating from multiple clouds are expensive and time consuming.

1.1 PROMETHEE

PROMETHEE is a special type of MCDM tool, developed by Brans in 1982 [9]. The method compares pairs of alternatives for each criterion, and it should specify the importance of criteria through weights. The preference difference between a pair of alternatives for each criterion is specified by preference function, which specifies numerical difference ranges from 0 to1. When the preference function value is zero means there is no difference between a pair of alternatives. If the value is one, then the alternative is strictly outranking the other alternative. The difference of the preference is directly proportional to preference function's value.

There are various MCDM approaches which have been used for QoS aware service selection. In this paper, we propose a heuristic optimization algorithm based on PROMETHEE method. PROMETHEE is chosen over other MCDM methods, as there is no need to define utility functions for each criterion. The PROMETHEE supports six basic types of preference functions which will be defined on QoS criteria. Another advantage is that it doesn't have fixed weighting scheme, and thus allows the inclusion of any good method. Finally, PROMETHEE is a user friendly and has been successfully applied to solve real time problems.

1.2 Related Work

Service selection from a minimal number of clouds has been an important issue for service composition. This is because any improper selection of the service can affect the overall QoS of a composite service and leads to user dissatisfaction and number of clouds leads to expensive and time consuming. Researchers have adopted different approaches to compose the best services from a minimal number of clouds.

Qi Lianyang et al. proposed a cross platform QoS aware service composition, in which the absent services which are outside the cloud platform will be invoked. The proposed local optimization and enumeration [10] method got an optimal candidate service set with more computation time and financial charges. An agent based multi cloud service composition was proposed by Gutierrez-Gareia Jo et al. [11]. It deals with different types of cloud services like one time virtualized services, infrastructure as a service, homogeneous services and heterogeneous services that are configured in the multi cloud environment. The proposed method fails to achieve optimal cloud combinations. Microsoft has put forward to the challenges associated with multiple cloud services [12].

Klein Adrian et al. proposed a network aware service composition for cloud [13]. The proposed method is successful in differentiating the QoS of a service and QoS of a network. The method handles both QoS criteria independently and achieves near optimal solution with low latency. The number of candidate services is more by which it attains more time to obtain solution. The proposed method is based on linear discriminant analysis, by which the candidate services are reduced drastically but still it suffers from number of cloud combinations and computation time [14]. Kevin kolfer et al. [16] proposed a customer driven service composition in a cloud environment. The proposed heuristic approach is based on historical information of service composition and which define happiness measure to meet user preferences. This method consumes more execution time to generate a service composition sequence. Qiang yu et al. [1] proposed an ant colony based web service composition algorithm in a cloud environment. The algorithm finds the optimal clouds and it consumes less computation time than COM2.

Zou G et al. [6] proposed three different cloud combination algorithms, which are all clouds, base cloud and smart cloud. All clouds algorithm finds a composition sequence with a minimal execution time, but it fails to minimize the number of clouds. The base cloud algorithm obtains an optimal cloud combination with extensive execution time. The smart cloud finds a sub optimal cloud combination at a reduced cost with a significant execution time. So, the above three algorithms suffer from either execution time or minimal number of clouds. Heba kurdi et al [5,17], proposed a novel combinatorial optimization algorithm for cloud service composition with minimal number of clouds. But they failed to compose best services from the optimal cloud combination. OWLS-Xplan is a web service dataset and a service composition planner [18,19] applied for emergency medical assistance. The OWLS-Xplan converts the services into an equivalent domain description which are specified in the PDDL 2.1 (planning domain description language) and invoke an efficient AI planner Xplan to generate a service composition plan which meets user preferences. The proposed QSC_MCD and all benchmark algorithms are based on web service dataset provided in the OWL-S Xplan package.

The service selection based on the QoS problem can be solved by methods such as Multi-Criteria Decision Making (MCDM), fuzzy logic and linear programming. In literature [20,21] they adopted certain hybrid methods to solve a QoS aware service selection. Most of the researchers adopt MCDM methods to solve QoS based service selection. For instance, AHP [22,23], ANP [24], and PROMETHEE [15,25] were applied for service selection. The two types of service selection models which are widely adopted are evaluation-based and predication-based service selection models. Almost all the MCDM methods are evaluation-based service selection models which are accurate and provide in-depth estimates of service quality, but the evaluation process is complex. In [26,27] they proposed integrated PROMETHEE and GAIA to solve machine selection problem. An analytical hierarchy process is used to find priority weights for the selected criteria and results are visualized using GAIA.

In this work, we propose a heuristic optimization algorithm which is QSC_MCD (QoS aware service composition in multiple cloud domain) that can compose best QoS aware services with minimal number of cloud combinations. The PROME-THEE method is used to select the best service from similar services. To fulfil the user service request with minimal number of clouds, cloud with maximum number of services will be selected first and then it will repeat until it meet the user's requirement.

The paper is organized as follows: In Section.2 illustrates a multi criteria QoS based cloud service composition. The system design and heuristic optimization algorithm is presented in Section.3. Evaluation is presented in Section.4. The effectiveness and efficiency of a heuristic optimization algorithm are demonstrated by an experiment in section.5. The final conclusion and future work are discussed in Section. 6.

2. Multi Criteria QoS Based Cloud Service Composition Method

In order to solve our MCDM problem, we applied PROMETHEE method. As similar services are deployed across several clouds with different QoS values, we have to select a QoS aware service.

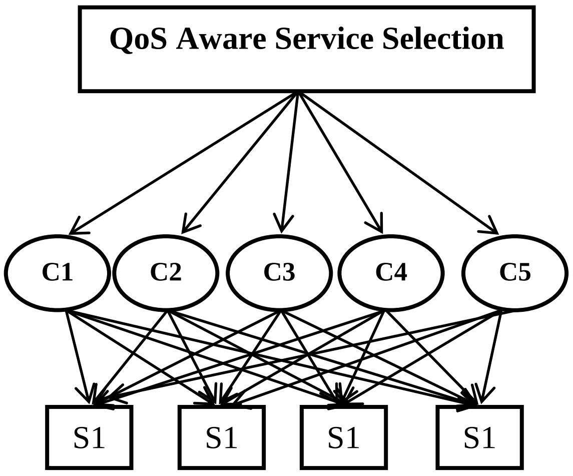

Figure.1 Basic parameters in PROMETHEE method (alternative Si and criteria Ci).

Here, the alternatives are services which are deployed in different clouds; these are compared pairwise based on QoS criteria. The best services will obtain by evaluating service alternatives (S_i), Cj=5 criteria have been used in Figure.1.The criteria are represented with Cj and they include: response time (C1), throughput (C2), availability (C3), sucessability (C4) and price (C5).

In this section, we presented PROMETHEE based QoS aware cloud service composition in detail as follows.

Step:1 Representation of cloud services

A multiple cloud domain (MCD) is a set of clouds, i.e. MCD= {C1, C2,…,Cn}. A set of service files, i.e. F= {F1, F2,….,Fn} are associated with each cloud, where each service file contains a set of services i.e. s= {S1, S2,….,Sn}. But similar services which are in different clouds have different QoS value.

Step:2 Selection of quality parameters

In order to select the best service from similar services in different clouds, the proposed model uses QoS criteria. So, according to significance and measurement of the service quality parameters, five attributes are chosen to assess the QoS metrics of a service as defined in section. 2.

Step:3 Determination of quality weights

In our experiments, we assume that all five quality parameters have equal importance, i.e. equal weightage for all five quality parameters.

Step:4 Define the constraints of quality factors and decide preference function

A constraint evaluates a service quality parameter to a preference function or by the extreme points (Minimum & Maximum).

To satisfy a user, service quality factor can either be minimized or maximized and to determine deviations based on pairwise comparisons by evaluating alternative services on each criterion by following equation. Where $g_j(a)$ and $g_j(b)$ are criteria weights of alternatives a and b.

$$d_j(a,b) = g_j(a) - g_j(b) \qquad (1)$$

Then apply a preference function on all alternatives for each criterion by doing

$$p_j(a,b) = F_j[d_j(a,b)] \quad j=1,2,\ldots.k \qquad (2)$$

Where, the preference for alternative 'a' with regard to alternative 'b' on each criterion is denoted by $p_j\ (a,b)$, as a function of $d_j\ (a,b)$ and F_j is a preference function.

To quantify various service quality factors, PROMETHEE provides six basic types of preference functions which are usual criterion, quasi criterion, criterion with linear preference, level criterion, criterion with linear preference and indifference and gauss criterion which quantifies the degree of preference with a value ranging from 0 to 1. Table 1 shows the service quality parameters along with their preference functions.

Table1. QoS parameters and its preference functions

S.NO	Parameters	Unit	Preference Function
1	Response Time	ms	Quasi criterion
2	Throughput	ms	Quasi criterion
3	Availability	%	Gaussian criterion
4	Successability	%	Gaussian criterion
5	Price	$	Level criterion

For each criterion, the value of an indifference threshold ‘q’, the value of a strict preference threshold ‘p’, and the value ‘s’ is an intermediate value between p and q has to be fixed. These values are the upper and lower bounds of a criteria value.

Step:5 Determination of preference index

The global preference index *π(a,b)* indicates the preference of service ‘a’ equate with service ‘b’ by taking service quality parameters into consideration.

The preference index is defined by

$$\forall a,b \in A \ \pi(a,b) = \sum_{j=1}^{k} p_j(a,b) w_j \tag{3}$$

Where *π(a,b)* is weighted sum *p(a,b)* for each criterion and w_j is weight associated with jth criterion

Step:6 Determination of outranking flows

In order to rank cloud service candidates from best to worst one, the outgoing and incoming flow for each alternative is defined as follows

$$\varphi^+(a) = \frac{1}{n-1} \sum_{x \in A} \pi(a,x) \tag{4}$$

$$\varphi^-(a) = \frac{1}{n-1} \sum_{x \in A} \pi(x,a) \tag{5}$$

The outranking flow $\varphi^+(a)$ indicates the degree at which service ‘a’ is preferred than other alternative services, x is an alternative service. The incoming flow $\varphi^-(a)$ indicates the degree at which other alternative services are preferred over service ‘a’. Based on the outgoing and incoming flow, the net flow *φ(a)* is defined by equation 6, which represents the overall preference of service ‘a’ comparing with other similar cloud services.

$$\varphi(a) = \varphi^+(a) - \varphi^-(a) \tag{6}$$

So, services are ranked from best to the worst one by calculating net flow. If $\varphi(a_i) = \varphi(a_j)$, than the alternative a_i is indifferent to a_j. If $\varphi(a_i) > \varphi(a_j)$, than the alternative a_i is preferential to a_j.

3. System Design

To solve the cloud service composition problem with minimal number of clouds, architecture is developed and is elucidated in Figure. 2. This figure shows that the five main components are user interface, multi cloud domain, cloud combiner, service composer and PROMETHEE.

1. A user sends a service composition request through a user interface and which displays the resultant service composition sequence.
2. A multi cloud domain (MCD) is a set of clouds, i.e., MCD= {C1, C2,…,Cn}. A set of service files from different service providers, i.e., F= {F1, F2,…,Fn} is associated with each cloud, where each service file contains a set of services i.e., S= {S1, S2,…,Sn}.
3. Cloud combiner process service composition request, based on that it will generate suitable cloud combinations with minimal number of clouds. The cloud combiner list is given as input to the composer list to select best composition sequence.
4. Service composer will find better services from cloud combinations received from cloud combiner. If similar services are presented more than once in a received cloud combinations from cloud combiner, then it will send to PROMETHEE in order to select best among them.
5. PROMETHEE is MCDM method, which receives a request from the service composer to decide best service from alternative services. Based on QoS metrics with corresponding preference function, quality weights the PROMETHEE method will select the best service within a minimal number of cloud combinations.

The service composer will check for a required service in received cloud from a cloud combiner. If it finds a similar service that is already in service composer, then PROMETHEE will select a service whose aggregate QoS value is maximum. Suppose if the service is alone from that cloud, then service composer will remove the cloud from the cloud combiner. The proposed method will compose the best services from a selected cloud combination set, it will neglect some better services from other clouds which are not part of the selected cloud combination set.

The complete proposed algorithm for QoS Aware Service Composition in Multi Cloud Domain (QSC_MCD) is illustrated in below Algorithm.

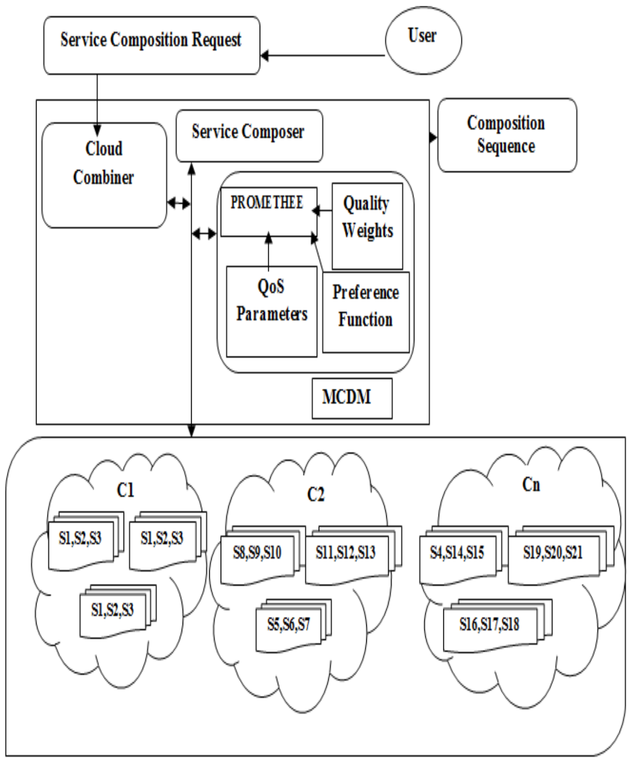

Figure 2. Architecture for proposed QSC_MCD method.

After initialization, in the line 1 the algorithm accepts user's service composition request. As per our assumption the clouds are sorted based on the number of service files in each cloud. As shown by lines 3-5, it will select first cloud that contains a large number of service files, and then process each and every file in that particular cloud. Then the services that match with the user request in line-7 will be added to composer list 'P' in line-17, and add that cloud to combiner list 'B' in line-21. Then algorithm in line-22 determines whether the user request is fulfilled or not. If it is fulfilled means, it generates a composition sequence; otherwise the next cloud is checked until user request is fulfilled. The algorithm will terminate, if it fails to meet user requests after processing all clouds. In line-8 it will execute PROMETHEE method, it checks whether the selected services are already present in composer list or not. If similar services are their means, line 9-16 will determine the best services from alternative services based on QoS parameters, then it will update the composer list with better QoS service, as demonstrated by lines 17-18.

Algorithm: A QoS Aware Service Composition Algorithm in Multi Cloud Domain

```
Input: User service request and multi cloud domain information.
Output: QoS aware service composition if available, otherwise the algorithm terminates.
Assumption: Clouds are sorted in decreasing order, based on a number of services.
// Initialize:
$B \longleftarrow \varphi$ // B is the combiner list of clouds
$p \longleftarrow \varphi$ // P is the composer list of services
$N \longleftarrow$ N is number of clouds in MCD
$K \longleftarrow$ K is the number of service files in Cn
$R \longleftarrow$ R is user request
$y, t, p_1, x \longleftarrow$ are temp variables
1.  Get the user request R
2.  Select the cloud Cn from sorted clouds
3.  for ( i=1; i≤n; i++)
4.  {
5.  for ( j=1; j≤k; j++)
6.  {
7.  if  $p_1$= (($C_{ij}$ ∩ R)! == ∅) then
8.  Exec PROMETHEE
9.  if  $y_s$= (($p_1$ ∩ t)! == ∅)
10. for ( l=1; l≤s; l++)
11. {
12. if ($c_i.y_l$ > X. $y_l$) then
13. p= p- X. $y_l$
14. else
15. $p_1$= $p_1$- $c_i.y_l$
16. }
17. P= P ∪ $P_1$
18. t= p;
19. }
20. x= $c_i$
21. B=B+$C_i$
22. if (P==R) then
23. Generate composition sequence
24. else
25. goto step.3
26. }
27. else
28. All clouds in the MCD have been checked
29. Exit
```

The working of the algorithm is illustrated by an example, suppose that an MCD contains four clouds: MCD1= {C1, C2, C3, C4}. Table 2 and Table 3 illustrate the cloud service files and the services in each file. If the user service request is R= {S1, S5, S9, S11, S14, S15, S18}, then the algorithm selects the first cloud in sorted list i.e., C4. Then it checks all service files in the cloud C4 to fulfil the user request R. The cloud C4 service files are {F1, F2, F3, F5} which checks (F1∩R), (F2∩R), (F3∩R), (F5∩R), then the resultant service list is {S1, S5, S9, S11, S14, S15} which will be added to composer list and C4 to combiner list. Still the composer list is not equal to R, and then it will select the next cloud from the sorted list i.e., C1. The cloud C1 service files will be processed and the resulting service list is {S1, S5, S9, S11, S14, S15}. As similar services are already there in the composer list, in order to find better services, invoke PROMETHEE to rank services from best to worst. From the results, the services S5, S9, S11 of cloud C1 have best QoS value when compared with cloud C4 services. So, update the composer list with best QoS services and add C4 to combiner list, i.e. B= {C1, C4}. Now select C2 or C3 randomly which contains the same number of service files. Suppose that we select C2, the resulting list after processing C2 service files are {S18}. Therefore, S18 will be added to the composer list, i.e., {S1, S5, S9, S11, S14, S15, S18} and C2 will be added to the combiner list i.e. B= {C4, C2, C1}. The composer list equals to R, so the algorithm terminates by generating the composition sequence with best QoS through a minimal number of clouds.

4. Evaluation

A novel QSC_MCD algorithm is evaluated in contrast to four benchmark algorithms; they are All cloud combination algorithm, the base cloud combination algorithm, the smart cloud combination algorithm and combinatorial optimization algorithm for cloud service composition (COM2). The proposed and all benchmark algorithms are based on web service dataset provided in the OWLS Xplan package.

OWLS-Xplan first converts the domain ontology and service description in OWL and OWLS to equivalent planning domain description language (PDDL) problem and domain description, which are used by the AI planner Xplan and PROMETHEE method to generate QoS aware service composition sequence. In our example the MCD consist of four clouds {C1, C2, C3, C4}, each cloud consists a set of service files {F1, F2,…,Fn}, which in turn contains a set of services {S1, S2,….,Sn}. For example, one of the service dataset in OWLS-Xplan is Health-Scallops [17,18] which is emergency medical assistance planning tasks, here we assume {F1, F2, F3, F4, F5} corresponds to {“EMA Services”, “Medical Flight Company Service”, “Non-Medical Flight Company Service”, “Medical Transport Company Services”, “Non-Medical Transport Company Services”} and each service file contains number of services.

Table 2. MCD settings for assessment.

MCD	C1	C2	C3	C4
MCD1	F1,F2,F3	F4,F5	F3,F4	F1,F2,F3,F5
MCD2	F1,F2	F3	F2,F5	F1,F4,F5
MCD3	F1,F3,F5	F5	F1,F2	F3,F4
MCD4	F2,F3,F5	F3,F4	F1,F2,F3	F4,F5
MCD5	F1,F2	F2,F3	F3	F1,F4,F5

Table 3. The cloud services in each file.

Service File	Services
F1	S1,S6
F2	S9,S10,S11
F3	S3,S5,S8,S12,S14,S15,S16,S17
F4	S2,S4,S18
F5	S19,S7,S13

The MCE settings and list of services in each file are illustrated in Table 2 and Table 3. The service composition request R= {S1,S5,S9,S11, S14,S15,S18} was assumed for all experiments.

All experiments were implemented using an HP Pavilion dv6 laptop with 2.10 GHz Intel core is a processor and 2 GB RAM. The performance factors are number of combined clouds, number of services examined and aggregate QoS value for a composed service.

5. Experimental Results

The proposed algorithm undergone a series of experiments to test the effectiveness and its performance, the experimental results are shown in Table 4 and Table 5. Table 4 gives the cloud combinations (CC) and the number of services processed (SP) in order to generate a composition sequence by different algorithms. Table 5 gives the composition sequence and its aggregate QoS value for COM2 and QSC_MCD algorithms.

Comparing the QSC_MCD algorithm with all four algorithms discussed above, the proposed algorithm is effective when compared to existing algorithms with respect to the number of services processed |N|. The QSC_MCD algorithm obtains better results across all multi cloud domains in terms of number of clouds processed.

Let there be a service 5 present in cloud 1, cloud 3 and cloud 5 with different QoS values, to select the best service the PROMETHEE is applied to obtain the service ranking as shown in Table 6 along with positive preference flow, negative preference flow and net flow. The proposed QSC_MCD selects services from cloud 1 based on the high PHI value of service5. The remaining models do not consider the QoS value while selecting the service. The service 5 from cloud 3 is neglected by both methods,

Table 4. Experimental results for the MCD settings.

Algorithm MCD	All Clouds		Base Cloud		Smart Cloud		COM2		QSC_MCD	
	CC	SP	CC	SP	CC	SP	CC	SP	CC	SP
MCD1	C1 C3 C4	46	C1 C2	65	C1 C3	70	C4 C2	35	C4 C2 C1	35
MCD2	C1 C2 C3 C4	27	C1 C2 C4	148	C1 C2 C4	48	C4 C2 C3	45	C4 C1 C2	27
MCD3	C1 C3 C4	32	C3 C4	128	C3 C4	48	C1 C4 C3	50	C1 C4 C3	29
MCD4	C1 C2 C3 C4	44	C2 C3	68	C2 C3	140	C1 C3 C2	49	C1 C3 C4	38
MCD5	C1 C2 C3 C4	32	C2 C4	112	C1 C2 C4	56	C2 C4	30	C2 C4	19

Table 5. Performance of the COM2 and QSC_MCD algorithms.

MCD	COM2		QSC_MCD	
	Composition Sequence	QoS	Composition Sequence	QoS
MCD1	$S1^{c4}+S9^{c4}+S11^{c4}+S5^{c4}+S14^{c4}+S15^{c4}+S18^{c2}$	0.5185	$S18^{c2}+S1^{c4}+S9^{c1}+S11^{c1}+S5^{c1}+S14^{c4}+S15^{c4}$	0.5885
MCD2	$S1^{c4}+S18^{c4}+S5^{c2}+S14^{c2}+S15^{c2}+S9^{c3}+S11^{c3}$	4.2026	$S1^{c4}+S18^{c4}+S5^{c2}+S14^{c2}+S15^{c2}+S9^{c1}+S11^{c1}$	5.0719
MCD3	$S1^{c1}+S5^{c1}+S14^{c1}+S15^{c1}+S9^{c3}+S11^{c3}+S18^{c4}$	2.7725	$S1^{c3}+S9^{c3}+S11^{c3}+S5^{c1}+S14^{c4}+S15^{c4}+S18^{c4}$	3.609
MCD4	$S9^{c1}+S11^{c1}+S5^{c1}+S14^{c1}+S15^{c1}+S1^{c3}+S18^{c2}$	0.3615	$S9^{c1}+S11^{c3}+S5^{c3}+S14^{c3}+S15^{c3}+S1^{c3}+S18^{c4}$	0.4039
MCD5	$S1^{c4}+S18^{c4}+S9^{c2}+S11^{c2}+S5^{c2}+S14^{c2}+S15^{c2}$	1.5724	$S1^{c4}+S18^{c4}+S9^{c2}+S11^{c2}+S5^{c2}+S14^{c2}+S15^{c2}$	1.5724

Table 6. Preference flow of service 5 in MCD 1.

Cloud Services	**PHI+**	**PHI-**	**PHI**	**Rank**
S5-C1	0,5554	0,0199	0,5354	**1**
S5-C3	0,1699	0,3947	-0,2248	2
S5-C4	0,1593	0,4699	-0,3107	3

because the cloud 3 is not in the optimal cloud combination set.

Figure.3 shows the results of QoS values of COM2 and QSC_MCD algorithm. We observe that QSC_MCD algorithm can find composite services with better QoS value compared with COM2. All most in all multi cloud domains it generates better QoS except in MCD5. Figure.4 presents the number of services processed by all five algorithms. The proposed algorithm achieved the best results across multi cloud domains. All Clouds are close to the QSC_MCD algorithm, but it incurs high communication cost and financial charges between clouds. At its best, it processed 19 services with two clouds in MCD5 and it maintains the same aggregate QoS value. The proposed QSC_MCD achieved a significant improvement in composition time by processing a minimum number of services than other benchmarks.

The proposed method outperforms other existing techniques in terms of number of clouds and in achieving better aggregate QoS value. The number of clouds is reduced by evaluating clouds with respect to their service files. The PROMETHEE is adopted to evaluate the service QoS by which the proposed method attains better aggregate QoS value.

The number of clouds involved in generating a composition sequence in the multi cloud domain is illustrated in Figure. 5. The QSC_MCD algorithm is successful in maintaining the same number of clouds as COM2. Both Base cloud and Smart cloud performed well with a maximum margin of one cloud, which falls behind in the number of services processed.

Figure.6 is a PROMETHEE diamond, where the vertical dimension corresponds to the phi net flow and each cone represents a service. In MCD2, the proposed QSC_MCD selects S9, S11 services from cloud 1 which are better services than services in cloud 3. In MCD3, the QSC_MCD selects S1, S14, S15 services from C3, C4 which are clearly preferred to all other similar services as shown in Figure.7. Suppose, if services are intersecting each other indicating that those services are incomparable.

The GAIA plane is a descriptive tool which helps to display the graphical position of alternatives corresponding to various criteria. The GAIA web is a spider web display for service S5 in C1 and C4 of MCD1as reflected in Figure.8 and Figure.9. The service S5 from C1 is a green dotted circle which implies the positive multi criteria net flow score. The service S5 in C4 is a red dotted circle indicating negative multi criteria net flow score. So, the proposed QSC_MCD prefers S5 from C1 than C4.

The conflicts between QoS criteria and the service alternatives are shown in Figure.10. In this GAIA plane the QoS criteria are denoted by axis and the alternate services by points. The criteria throughput, response time and successability are oriented in the same direction because of similar preferences between those criteria, while the other QoS criteria are conflicted by pointing in the opposite direction. We can even notice the dominated QoS criteria of each service alternative by observing the quarter in which they lie. Suppose S5-C1 is good with defined criteria, so it is ranked as one. Whereas S5-C3 and S5-C4 is far away from all axis, leading to the last rank.

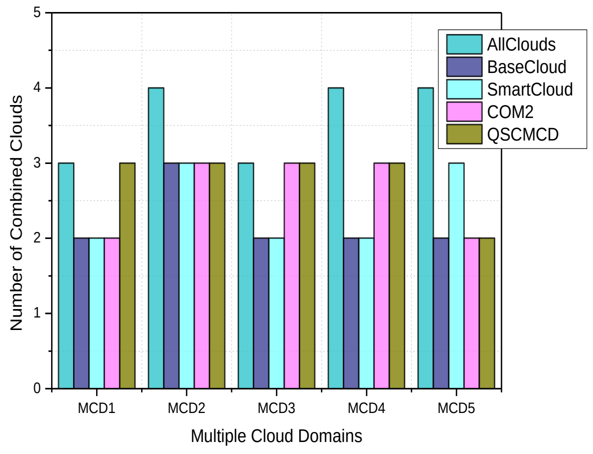

Figure 5. The number of combined clouds in all the algorithms.

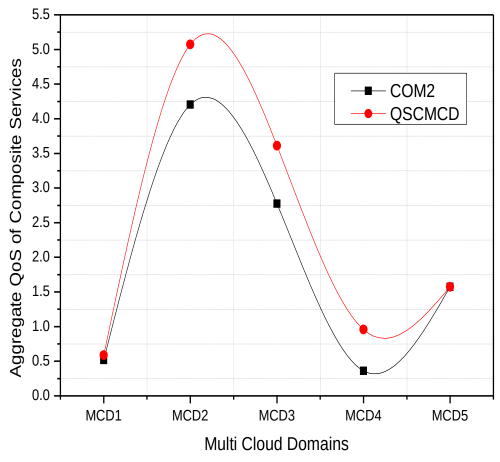

Figure 3. The comparison of aggregate QoS of COM2 and QSC_MCD algorithms.

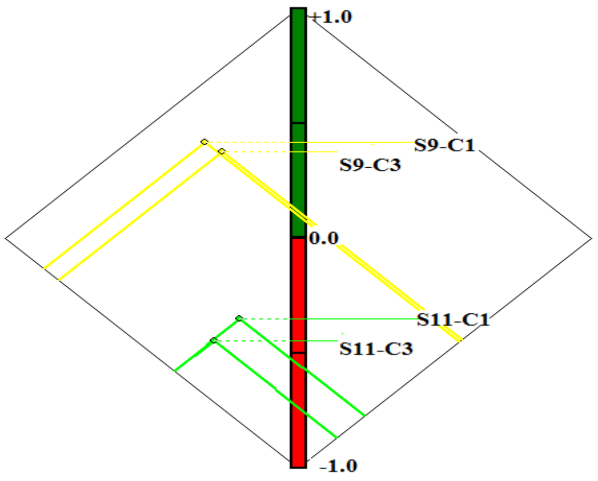

Figure 6. PROMETHEE Diamond of MCD2.

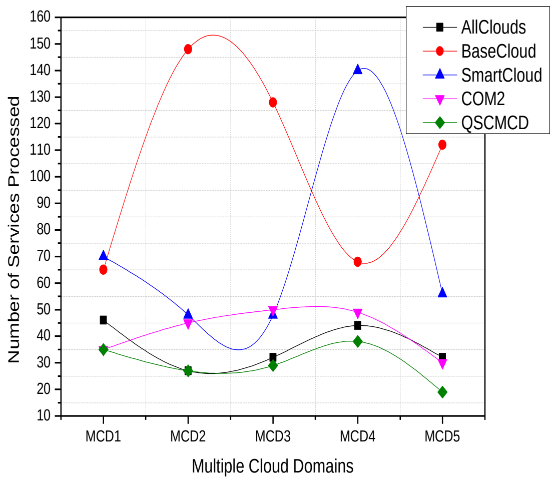

Figure 4. The number of processed services in all the algorithms.

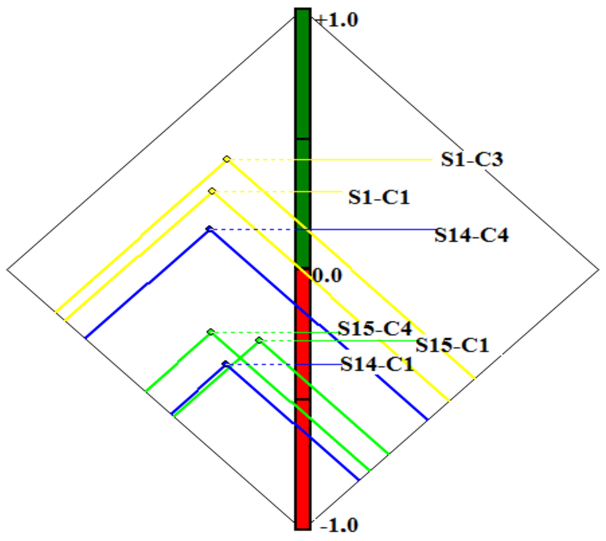

Figure 7. PROMETHEE Diamond of MCD3.

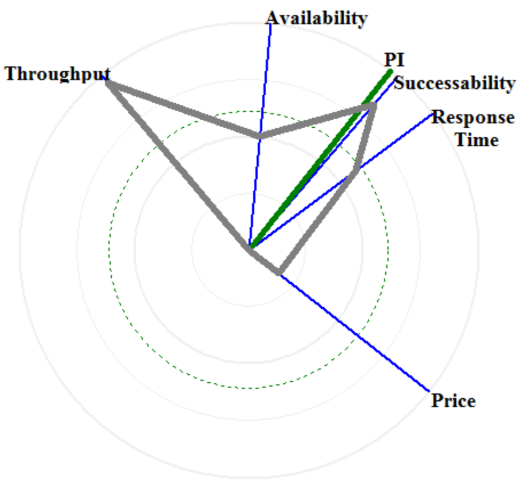

Figure 8. GAIA web of service5 in Cloud 1

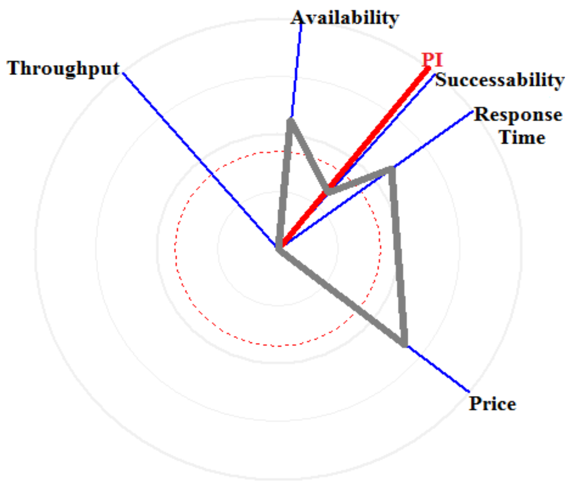

Figure 9. GAIA web of service5 in Cloud 4

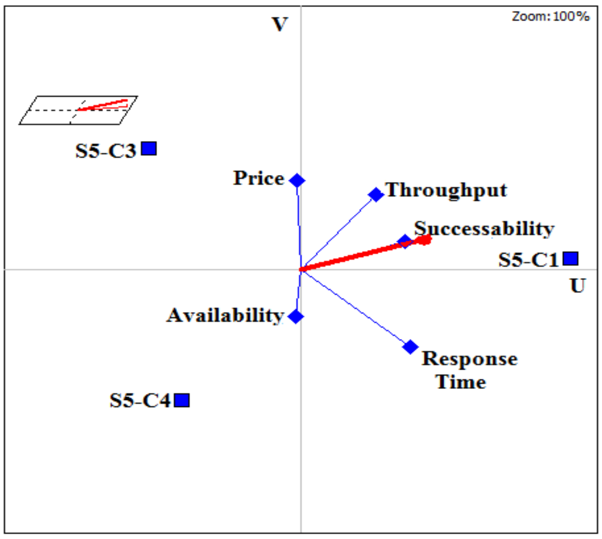

Figure 10. GAIA Plane of Service5 with Respect to QoS Criteria.

The experimental results demonstrate that the QSC_MCD algorithm found better QoS aware service composition with optimal cloud combinations. Hence the QSC_MCD algorithm is much more efficient.

6. Conclusion

The proposed QSC_MCD algorithm composes services from multiple cloud domain based on quality of service parameters with minimal number of clouds. We adopt PROMETHEE method an MCDM solution, which selects best service from similar services within the cloud combination set. Our experimental results indicate that the proposed method can efficiently and effectively find the best services with minimal number of clouds. The proposed method achieves a better aggregate QoS value in all five sceneries over COM2. Even it outperforms the existing methods in terms of number of clouds and number of services processed with a good margin leads to reduce in communication cost and finical charges between clouds.

In future work, service composition with minimal number of clouds problem will be solved by bio-inspired optimization methods.

References

[1] Y. Qiang, L. Chen, and Bin .L, "Ant colony optimization applied to web service compositions in cloud computing", *Computers & Electrical Engineering*, Vol. 41, pp.18-27, 2015.

[2] J. Amin, E. Sundararajan, and Z. Othman, "Cloud computing service composition: A systematic literature review", *Expert Systems with Applications*, Vol. 41, No. 8, pp.3809-3824, 2014.

[3] Rajeswari, M., G. Sambasivam, N. Balaji, MS Saleem Basha, T. Vengattaraman, and P. Dhavachelvan, "Appraisal and analysis on various web service composition approaches based on QoS factors", *Journal of King Saud University-Computer and Information Sciences*, Vol. 26, No. 1, pp.143-152, 2014.

[4] Z. Wei, J. Wen, M. Gao, and J. Liu, "A QoS preference-based algorithm for service composition in service-oriented network", *Optik-International Journal for Light and Electron Optics*, Vol.124, No. 20, pp.4439-4444, 2013.

[5] K. Heba, A. Al-Anazi, C. Campbell, and A. Al Faries, "A combinatorial optimization algorithm for multiple cloud service composition", *Computers & Electrical Engineering*, Vol.42, pp.107-113, 2015.

[6] Z. Guobing, Y. Chen, Y. Yang, R. Huang, and Y. Xu, "AI planning and combinatorial optimization for web service composition in cloud computing", *In Proc*

international conference on cloud computing and virtualization, pp. 1-8. 2010.

[7] Podvezko, k. Valentinas, and A. Podviezko, "Dependence of multi-criteria evaluation result on choice of preference functions and their parameters", *Technological and Economic Development of Economy*, Vol.16, No. 1, pp.143-158, 2010.

[8] H. Serge, L. Mokdad, and S. Youcef, "Selection of the Best composite Web Service Based on Quality of Service", *ISSS/BPSC*, Vol.10, pp.255-266, 2010.

[9] Vincke, Ph, "Note---A Preference Ranking Organisation Method", *Management Science*, Vol.31, No. 6, pp.647-656, 1985.

[10] Q. Lianyong, W. Dou, X. Zhang, and J. Chen, "A QoS-aware composition method supporting cross-platform service invocation in cloud environment", *Journal of Computer and System Sciences*, Vol.78, No. 5, pp.1316-1329, 2012.

[11] Gutierrez-Garcia, J. Octavio, and KM. Sim, "Agent-based cloud service composition", *Applied intelligence*, Vol.38, No. 3, pp.436-464, 2013.

[12] Microsoft Communications and Media Industries. Multi-Cloud Service Delivery & End-to-End Management.Reference architecture.<file:///C:/Users/Kelly/Downloads/Microsoft%20RA%20%20Multi-Cloud%20Service% 20Delivery%20and%20E2E %20 Mgmt%20v1_1.pdf>, 2013.

[13] K. Adrian, F. Ishikawa, and S. Honiden, "Towards network-aware service composition in the cloud." *In Proceedings of the 21st international conference on World Wide Web*, pp.959-968, 2012.

[14] S.Bharath Bhushan and Pradeep Reddy.CH, "A four level linear discriminant analysis based service selection in the cloud environment", *International Journal of Technology*, Vol.5, pp. 859-870, 2016.

[15] S,Bharath Bhushan and Pradeep Reddy.CH, "A Network QoS Aware Service Ranking Using Hybrid AHP-PROMETHEE Method in Multi Cloud Domain", *International Journal of Engineering Research in Africa*, Vol.24, pp. 153-164, 2016.

[16] K. Kevin, H. Irfan ul, and E. Schikuta, "User-centric, heuristic optimization of service composition in clouds", *In Euro-Par 2010-Parallel Processing, Springer Berlin Heidelberg*, pp. 405-417, 2010.

[17] C. Fabio, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M.Shan, "Adaptive and dynamic service composition in eFlow", *InAdvanced Information Systems Engineering,. Springer Berlin Heidelberg*, pp. 13-31, 2000.

[18] K. Matthias, and A. Gerber, "Fast composition planning of owl-s services and application", *In Web Services, 2006. ECOWS'06. 4th European Conference on, IEEE*, pp. 181-190, 2006.

[19] K. Matthias, A. Gerber, and M. Schmidt, "Semantic web service composition planning with owls-xplan", *In Proceedings of the AAAI Fall Symposium on Semantic Web and Agents, Arlington VA, USA, AAAI Press*. 2005.

[20] L. Lei, Y. Wang, and EP. Lim, "Trust-Oriented Composite Service Selection with QoS Constraints", *J. UCS*, Vol16, No. 13, pp.1720-1744, 2010.

[21] L. Chi-Chun, Ding-Yuan. C, Chen-Fang.T, and Kuo-Ming .C, "Service selection based on fuzzy TOPSIS method", *In Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on, IEEE*, pp. 367-372, 2010.

[22] Q. Li-Li, and Y. Chen, "QoS ontology based efficient web services selection", *In Management Science and Engineering, ICMSE 2009. International Conference on, IEEE*, pp. 45-50, 2009.

[23] Z. Meiyun, S. Wang, and B. Wu, "Research on web services selection model based on AHP", *In Service Operations and Logistics, and Informatics, IEEE/SOLI 2008. IEEE International Conference on*, Vol. 2, pp. 2763-2768, 2008.

[24] G. Manish, R. Sonar, and S. Mulik, "Web service selection based on Analytical Network Process approach", *In Asia-Pacific Services Computing Conference, APSCC'08. IEEE*, pp. 1103-1108, 2008.

[25] K. Raed, C. Ding, and Chi-Hung .C, "An enhanced PROMETHEE model for QoS-based web service selection", *In Services Computing (SCC), IEEE International Conference on, IEEE*, pp. 536-543, 2011.

[26] K. Prasad, and S. Chakraborty, "Application of PROMETHEE-GAIA method for non-traditional machining processes selecti-on",Management Science Letters, Vol. 2, No. 6, pp.2049-2060, 2012.

[27] C. Heejung, and K. Lee, "A Quality-Driven Web Service Composition Methodology for Ubiquitous Services", *J. Inf. Sci. Eng*. Vol.26, No. 6, pp.1957-1971, 2010.