2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC) | 978-1-6654-2463-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/COMPSAC51774.2021.00161

An Optimal Composite Service Selection Model based on Edge-Cloud Collaboration

Yan Wang, Na Zhou, Haixia Lang, Yunying Li
Inner Mongolia Engineering Lab of Cloud Computing and Service Software,
College of Computer Science, Inner Mongolia University
Hohhot, China
cswy@imu.edu.cn, 865091420,2497565532,1783930353@qq.com

***Abstract*—In the age of the Internet of everything, the Edge-Cloud collaborative service support has become a very promising development direction in the application field of the Internet of things. However, when various service components are deployed both on the cloud and the edge, the subsidence and decentralization of computing resources also have a great impact on the performance of composite services. This paper proposes an optimal composite service selection model based on Petri nets. In order to compare the composite service paths, this paper adopts a dynamic evaluation model of composite service quality based on Petri nets. Firstly, all kinds of Petri net models for the implementation structure of composite services are constructed. And then the QoS computing rules corresponding to these structures are given based on the QoS of each service component on the edge of the cloud. Finally, the dynamic execution process of each composite service with a feasible path is simulated through Petri net, and the overall performance of the service is evaluated based on the dynamic simulation of its Petri net model, finally, the optimal service path is selected among the combined services that meet the user's requirements.. Through case analysis and comparison, the feasibility and effectiveness of the method are verified by an example analysis.**

Keywords:Edge-Cloud collaboration; Composite service; Petri net; Quality evaluation model; Optimal service path

I. INTRODUCTION

With the popularity and development of the Internet of Things technology, the number of mobile devices at the edge of the network is rapidly increasing, and these devices may make various service requests. For high computing power, these services can be deployed on cloud platforms to meet the needs of mobile users. But at present, more and more services involve delay-sensitive tasks which are used to process and analyze large amounts of local data. The centralized service model with the cloud computing model as the core has been unable to meet the needs of these services for efficient edge-side data processing. In this case, an edge computing model for large amounts of data processing at the edge of the network has emerged.

Edge computing sinks computing resources and efficient services to the edge of the network, resulting in lower latency, lower bandwidth consumption, higher energy efficiency, and higher privacy protection [1]. It is essentially a new solution of cloud computing extending to the user end [2]. Therefore, edge computing and cloud computing are complementary. In recent years, the service support of Edge-Cloud collaboration has become a very promising development direction in the field of Internet of Things applications.

However, when we deploy the components of service both on the cloud and the edge, the subsidence and decentralization of computing resources also have a greater impact on the performance of the composite service [3]. First, data exchange is necessary for the execution of service composition between edge servers or between the edge server and the cloud. The performance of a composite service is different when its service components are split with different proportions on the side and the cloud. In addition, the load of the edge server also has a certain impact on the performance of the composite service. In summary, the deployment locations of service components of a composite service have a decisive impact on its performance.

From the user's perspective, different users have different needs for services. Users' different service needs can be quantified using the concept of QoS. This determines that different user QoS requirements should choose different Edge-Cloud collaboration solutions for the composite service. Especially on the edge side, different edge servers are highly heterogeneous and may be provided by different service providers. These servers have large differences in reliability, security, and cost when executing a service. Therefore, according to different deployment locations of service components, it is an urgent problem to design a dynamic evaluation method of composite service quality to choose the optimal service path for users in the service environment of Edge-Cloud collaboration. To this end, this paper proposes a dynamic quality evaluation model of composition service based on Petri nets. Based on the different QoS of each service component of a composite service executed on a single edge server or the cloud, the composite service is modeled in Petri net system with its various feasible execution paths. And then, we use a Petri net's dynamic simulation method to evaluate the overall performance of composite services, recommending to the user a composite service with the optimal collaborative execution plan in the Edge-Cloud service environment that best meets their requirements.

II. RELATED BACKGROUND RESEARCH

Service, as a convenient cloud online application, has become a widely popular way of providing software. In many cases, a single service instance has been unable to satisfy the complicated and complex requirements requested by users [4]. Composite services can meet the needs of users by combining existing simple services with different logical

978-1-6654-2463-9/21/$31.00 ©2021 IEEE
DOI 10.1109/COMPSAC51774.2021.00161

structures, which is an effective means to respond to users' personalized service needs. In the process of service composition, two points need to be considered. First, whether the composite service can achieve the functions required by the user or not is need to concerned. Second, whether the performance of the combined service can meet the user's QoS requirements or not is also need to be considered. The execution performance of a composite service is an important criterion for a user to choose the service. By quantifying the quality of different composite services and comparing them with the user's QoS requirements, it is possible to recommend the optimal service to the user and improve the user's satisfaction with the service.

In order to recommend the most suitable composite service to users, common methods include QoS ontology-based model, QoS-based ranking, QoS-based prediction, and heuristic-based QoS optimization. In [5] method, the QoS characteristics are comprehensively considered and a reliable QoS ontology model is constructed. In [6] method, the QoS ontology model to support the semantic description and measurement of heterogeneous QoS parameters is constructed, and then the semantics and values of QoS in this model are used to select services for users to meet their personalized needs. In [7] method, a framework for overall service selection and ranking is proposed. The candidate services are classified into different levels using the associative classification algorithm based on users' QoS requirements and preferences. In [8] method, a personalized QoS prediction method is presented, which considers the influence of network, server environment, and user input on QoS. Through pattern mining, the invocation pattern of the composite service is extracted from its historical QoS data, and its QoS is further predicted based on the invocation pattern and user input. In [9] method, a three-level QoS evaluation and matching algorithm is proposed by combining QoS ontology with QoS ranking. However, considering the massive scale of services in the cloud, in order to effectively reduce the search space for composite service, a local heuristic optimization method is proposed to filter candidate services by assigning quality constraints to each task, and then all possible service composition plans from the filtered candidate services are evaluated in [10]. The above algorithm has achieved good results in a centralized management service platform. However, the new service requirements and new service environment in the era of the Internet of Everything have brought new challenges to optimal composite service selection based on QoS evaluation.

Unlike centralized cloud data centers, edge servers that provide services for the Internet of Things have some characteristics such as geographic dispersion, limited computing and storage capabilities and unstable network communications [11]. The dynamics, heterogeneity and instability of edge computing resources directly affect the quality of composite services that are executed on them, which leads to the poor user service experience. For this, in [12] the authors analyzed the nature of edge computing and the nature of things and then gave some detection standards for how to maintain service quality in the edge computing. Based on these standards, the execution of composite service can be more accurate and efficient. In [13], on the premise of collaboration of edge computing resources and optimization of scheduling QoE, an improved beast particle swarm algorithm is proposed for two-phase edge service composition and scheduling. Inspired by these methods, our target is to explore an optimal composite service selection method based on the Petri net model. It can perform dynamic performance simulations of different service deployment schemes in a cloud-based collaboration service environment to find a composite service closest to the user's QoS requirements.

III. MODEL AND METHOD

A. Basic definitions

To facilitate problem modeling and description, the basic concepts are presented. QoS is the key index for evaluating a composite service. For different types of applications, QoS description methods are also different. Our paper focuses on four common QoS indicators, including response time, cost, reliability and availability.

Definition 1 (**QoS**): QoS is represented as a four-tuple,

$$QoS=\{t, c, r, a\}, \quad (1)$$

where t is for the response time of a service to process the user request, its unit is ms; c is for the fee required to invoke a service, it's in dollars; r is for the reliability of a service, which represents the ability or possibility of performing the function of the service without failure, $r \in [0,1]$; a is for the availability of a service, which represents the probability that the service will function normally, $a \in [0,1]$.

Definition 2 (**Service**): A service is represented as a five-tuple,

$$S=\{Id, SFlag, AcceptedMessage, OutputMessage, QoS\}, \quad (2)$$

where *Id* is for the unique identifier of the service; *SFlag* is for the place where the service is executed. There are two types: cloud server and edge server. If *SFlag*=0, it means that the service is executed on the cloud platform; if *SFlag*>0, it means that the service is executed on an edge server and the value of *SFlag* corresponds to the number of the edge server. *AcceptedMessage* and *OutputMessage* are the inputs and outputs of the service. *QoS* are non-functional parameters of the service, see definition 1 for details.

Definition 3 (**QoSFilter**): QoSFilter is the minimum QoS requirement for a service to the user,

$$QoSFilter=\{Ft, Fc, Fr, Fa\}. \quad (3)$$

The QoSFilter is used for preliminary filtering conditions for the services, and its value is user-defined. Among them, *Ft* and *Fc* represent the maximum response time and cost that the user can accept, respectively; *Fr* and *Fa* represent the minimum value of reliability and availability that the user can accept, respectively. Only when all QoS indicators of the basic services meet the value of *QoSFileter*, the service can be selected as a candidate service. In this way, the complexity of subsequent composite service evaluation and recommendation

will be greatly reduced. Assume that QoS(s)=$\{t_s, c_s, r_s, a_s\}$, then when $t_s<Ft$, $c_s<Fc$, $r_s>Fr$, $a_s>Fa$, s is a candidate service.

Definition 4 (**QW**): QW represents the subjective preferences of the user for each sub-attribute in QoS,

$$QW=\{Wt, Wc, Wa\}, \quad (4)$$

where Wt is for time wight, Wc is for cost weight, Ws is for availability weight, and Wt+Wc+Wa=1. The reliability preference is not be set because if the reliability of a composite service is lower than the user's requirements, the service will not be recommended to the user, no matter how high the other values are.

Definition 5 (**Request**): QoS is represented as a five-tuple, describing the functional and non-functional requirements of user requests, as well as service constraints and expectations.

$$R=\{ID, InputMessage, ExpectedMessage, QoSFilter, QW\} \quad (5)$$

where *ID* is for the unique identifier of the request, *InputMessage* and *ExpectedMessage* are responsible for the user's definition of the overall function of the request. *InputMessage* represents the parameters that the user needs to input, which can be empty, and *ExpectedMessage* represents the output form expected by the user. *QoSFilter* and *QW* are defined as above.

B. Model description

1) Petri net model of composite services

For convenience, the names, types, and meanings of each element in the Petri net model of composite services are first given, as shown in Table I. In this paper, we abstract the services, user requests, and intermediate results into different places, and take the processes of service matching and composition as transitions.

TABLE I. ELEMENTS IN THE MODEL AND THEIR MEANINGS

Name	Element type	Meaning
Service	Place	a service
Request	Place	a user request
ValueMerge	Transition	parameters aggregation
ValueTrans	Place	parameters passing
Condition	Transition	defense conditions
midRequest	Place	intermediate results
STEP	Place	service selection
Combine	Place	path aggregation
Result	Place	final output

First, the model of a single service in a Petri net is given, as shown in Fig.1. Request place R_1 and service place S_1 can jointly trigger transition *valueMerge*, and the transition aggregates the parameters of R_1 and S_1 into the intermediate place *valueTrans*, which can further fire the transition *Condition*. The defense expression in *Condition* compares the I/O parameters in R_1 and S_1. If S_1 meets the requirements of R_1, the service is executed and its result *OutputMessage* is passed to place *Result*.

For composite services that contain multiple service components, the execution flow of service components can be categorized and simplified into three categories: sequential structure, branch structure, and federated structure. The sequential structure represents the sequential execution of the service components. The branch structure means that the output of a service component is the input of multiple service components. The federated structure means that a service needs to receive the output of two or more services simultaneously as its input. The Petri net models for these three structures are given as follows.

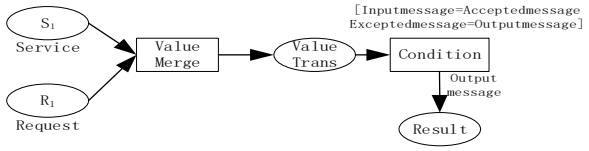

Fig. 1 Petri net model of a single service

The Petri net model of the sequential structure for the feasible service path $S_{33}+S_{34}$ is shown in Fig.2. S_{33} is the first service that needs to be called by this sequential model. After S_{33} is completed, its output *OutputMessage* replaces the parameter *InputMessage* in *Request*. The parameter is combined with other parameters in *Request* to generate an intermediate result into place *midRequest*, whose type is Request. Next, the second service place S_{34} in the sequential model is introduced, and the following steps are similar to the above steps until the service whose output is *ExpectedMessage* is completed.

The Petri net model of the branch structure for the feasible service path is shown in Fig.3. S_{20} is the first service invoked by this model. When the execution result of S_{20} enters the first intermediate place *midRequest*, S_{32} and S_{41} can be triggered simultaneously. And then two independent execution processes are continued separately according to their respective service paths. As is shown in the figure, S_{32} fires S_{30}, and S_{41} fires S_{35} until the last transition of each path is completed. Finally, the final output of each path is sent to place *Result.*

The Petri net model of the federated structure for the feasible service path is shown in Fig.4. Since S_{37} can fire its execution condition only when the two branch service paths are completed, Request is divided into two branch places $Request_1$ and $Request_2$ by two-way arc and transition. $Request_1$ is the starting place for the service path $S_{20}+S_{32}$, and $Request_2$ is the starting place for the service path S_{33}. The final output places of the two paths, $midRequest_2$ and $midRequest_3$, will trigger transition $ValueMerge_3$.

2) Qos quantization of composite services

A composite service is composed of several basic services, so its value of QoS can be obtained through QoS aggregation functions. For some QoS attributes, the aggregation functions of the same QoS attributes are also different in the face of different execution flows [14]. The aggregation function of cost is accumulation, the aggregate function of reliability is multiplication, and the aggregate function of availability is the minimum value. The aggregation function of time is related to the executive structure of composite service.

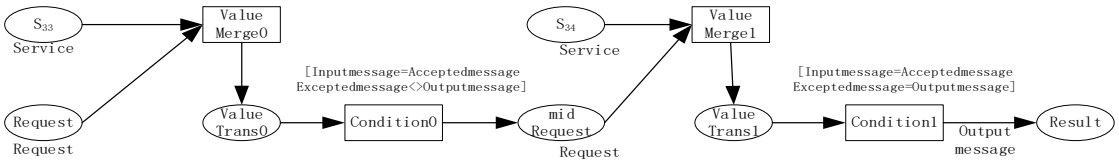

Fig.2 Petri net model of sequential structure

For parallel service execution paths, the aggregation function of time is the maximum value; while for sequence service execution paths, the aggregation time is accumulation [15]. Here are the aggregate functions used for each of the attributes in this paper, as shown in Table II. Composite services have multiple QoS attributes, and their ranges and units are different. From the perspective of QoS global optimal analysis, it is not conducive to the overall evaluation of QoS attributes of composite services. In order to facilitate the comparison of composite services and to select the best composite service, it is necessary to standardize the QoS attributes and then convert it to a composite global QoS. That is, the QoS vector of each feasible composite service is mapped to a real number.

Firstly, QoS attributes can be divided into positive attributes and negative attributes. The higher the value of positive attributes, the better the quality of service, such as availability; the smaller the negative attribute value, the better the quality of service, such as time. For positive attributes, standardize with formula (6); for negative attributes, standardize with formula (7).

$$D_{h,i} = (h_i - a_i)/b_i \quad (6)$$

$$D_{h,i} = 1 - (h_i - a_i)/b_i \quad (7)$$

TABLE II. ELEMENTS IN THE MODEL AND THEIR MEANINGS

QoS Attribute	Execution Structure	Aggregation Function
Time	Sequence	$\sum_{i=1}^{n} q_t(s_i)$
	Parallel	$MAX_{i=1}^{n} q_t(p_i)$ $(p_i = s_{i1} + s_{i2} + \ldots + s_{im})$
Cost	Any	$\sum_{i=1}^{n} q_c(s_i)$
Reliability	Any	$\prod_{i=1}^{n} q_r(s_i)$
Availability	Any	$MIN_{i=1}^{n} q_s(s_i)$

Because of the large number of servers in the edge service environment, there is a significant cost involved in traversing and evaluating all feasible composite service deployments on these servers [16]. In fact, it is completely unnecessary. Some servers are overloaded or are too far away from the mobile user, so it is impossible to meet the user's performance needs when the service components are executed on them. In order to reduce the space for service evaluation, we will first filter out some candidate servers by basic QoS constraints, that is

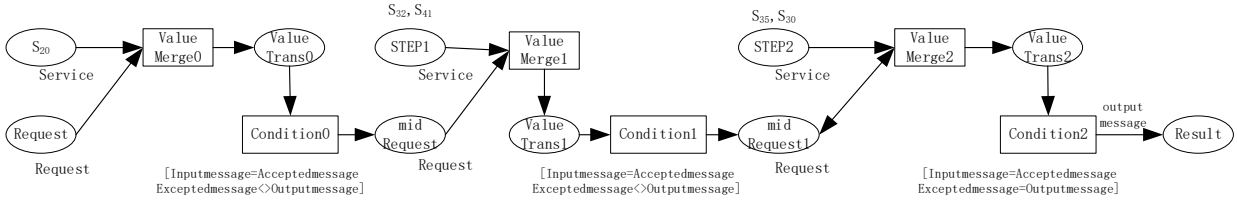

Fig.3 Petri net model of branch structure

The i-th comprehensive QoS attribute of a composite service is defined by $D_{h,i}$, where h_i, a_i and b_i represents the highest value, the mean value and the standard deviation.

Definition 6 (**Comprehensive QoS**) The comprehensive QoS of a composite service is calculated based on user preferences and its overall QoS,

$$ServiceQoS = \sum_{i=1}^{n} W_{h_i} * D_{h,i} \ . \quad (8)$$

QoSFilter, which can improve the recommendation efficiency of composite services.

In the focused four QoS properties of a service, considering the relationship between the reliability and availability of the service, the initial availability value of all services is set to 0. When the reliability of service is less than *Fr*, the availability of the service does not need to be compared. Because if the service is not reliable, even if the service is available, we will

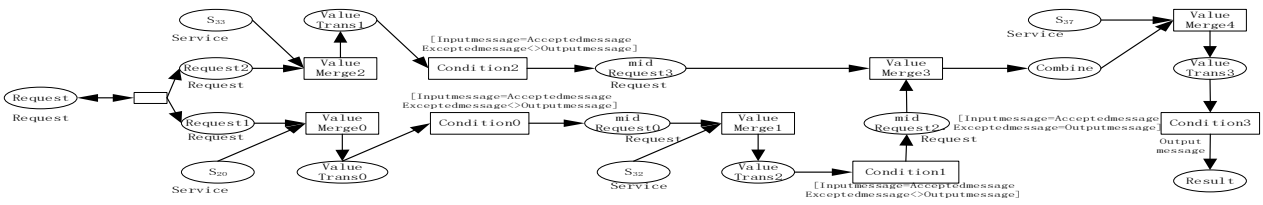

Fig.4 Petri net model of federated structure

not recommend it. The availability of service can be calculated according to the formula (9), that is,

$$s = \begin{cases} 0, & r \le F_r \\ (r - F_r)/F_r, & r > F_r \end{cases} \quad (9)$$

C. Algorithms

algorithm:

1: Set s_i of S ;
2: for(all s_i)
3: Filter(s_i, *QoSFilter*)
4: for(all s_i)
5: if F_r >=r s = 0;
6: else s =(r-F_r)/ F_r;
7: end
8: matchService(S);
9: for(j=0; j++)
10: if InputMessage=AcceptedMessage
11: if ExpectedMessage=OutputMessage
12: insert S_j and set next "MAX";
13: else
14: OutputMessage=o;
15: o=InputMessage;
16: p=p+s_i;
17: matchService(p);
18: end
19: Output(CS)
20: for (j=0;j++)
21: if s<=F_s s = 0; else s=s/F_s ;
22: Get(QoS)
23: for CS_i in CS
24: T_i=f_t (CS_i): C_i=f_c (CS_i): R_i=f_r (CS_i): A_i=f_s (CS_i)
25: Standardized(T_i, C_i, R_i,A_i)
26: $ServiceQos_i = \sum_{i=1}^{n} w_{h_i} * D_{h_i i}$;

The explanation of the algorithm: Steps 2-3 are used to filter the basic services. Steps 4-8 are used to filter services in the edge according to their reliability, its purpose is to emphasize the importance of reliability through combining reliability and availability. It can filter out invalid services deployed on edge servers to improve the recommendation efficiency of composite services, leaving valid services deployed on edge servers that can execute a component of the composite service. Steps 11-19 match all feasible paths for the composite service. Through a recursive algorithm, all-composite services that meet the user's functional requirements are found. Steps 20-26 evaluate the QoS attributes of each candidate composite service and calculate their comprehensive QoS values, and compare the comprehensive QoS value to select the optimal composite service that most meets the needs of the user.

IV. Experimental Verification

In this section, we will verify the proposed algorithm with an example. Assume that the user request is described by *R={R1, shy, happy, {100,50,0.6,0.57}, {0.57,0.2,0.23}}*. To describe the user's requested I/O interface, we abstract it with a state variable. Different values represent different data input and output. Some basic services and their corresponding deployment in the service environment of edge-cloud collaboration are given in Table III.

TABLE III. Some services and the corresponding parameters

ID	SFlag	I	O	t(ms)	c($)	r	a
S_1	12	shy	happy	200	40	90%	70%
S_2	6	shy	bored	70	28	73%	77%
S_3	0	shy	happy	33	35	80%	60%
S_4	0	shy	proud	90	87	71%	80%
S_5	0	shy	glad	51	72	80%	20%
S_6	7	bored	proud	89	44	80%	70%
...	...	...	...	...	...	...	...
S_{35}	8	proud	happy	64	12	87%	79%
...	...	...	...	...	...	...	...

Through filtering by the *QoSFilter* in *R,* 26 services that meet the basic requirements are retained. By matching the I/O interfaces of the basic services, all feasible execution paths of the composite service are found, as shown in Table IV.

Taking the service path $S_2 + S_6 + S_{35}$ as an example, t and c of the three services are all less than *Ft* and *Fc*, and r and a are all greater than *Fr* and *Fa* in the request R. In addition the AcceptedMessage of S_2 is equal to the InputMessage requested by the user, and the OutputMessage of S_{35} is the same as the ExcepMessage in the request R. Therefore, $S_2 + S_6 + S_{35}$ can become a service path to meet user's request.

TABLE IV. All possible composite services

Service structure	service path
Single	S_3, S_9, S_{43}
Sequential structure	S_2+S_6+S_{35} S_7+S_{12}+S_{18}+S_{29} S_{33}+S_{34}
Branch structure	$S_{20}+\begin{cases} S_{32}+S_{30} \\ S_{41}+S_{35} \end{cases}$ $S_{23}+\begin{cases} S_{31}+S_8+S_{35} \\ S_{41}+S_{35}+S_8 \end{cases}$ $S_{44}+\begin{cases} S_8+S_{35} \\ S_{27} \end{cases}$
Federated structure	$\left.\begin{matrix} S_{33} \\ S_{20}+S_{32} \end{matrix}\right\}+S_{37}$

In Table IV, ten feasible service paths that satisfy the user's request are found. These composite services can perform the function of request *R* and meet its performance requirements. Then the QoS values of these composite services are quantified by aggregate functions, calculate the composite QoS value of the composite service. The quantified results are shown in Table V.As for the above feasible composite services that can satisfy *R*, the quality of service varies greatly.

TABLE V. QOS OF FEASIBLE SERVICE PATHS

Service structure	Feasible service path	ServiceQoS	Standard
Single	S_3	0.667	
	S_9	0.742	
	S_{43}	0.632	
Sequential structure	$S_2+S_6+S_{35}$	0.793	
	$S_7+S_{12}+S_{18}+S_{29}$	0.627	
	$S_{33}+S_{34}$	0.865	
Branch structure	$S_{20}+\begin{cases}S_{32}+S_{30}\\S_{41}+S_{35}\end{cases}$	0.911	optimal
	$S_{23}+\begin{cases}S_{31}+S_8+S_{35}\\S_{41}+S_{35}+S_8\end{cases}$	0.814	
	$S_{44}+\begin{cases}S_8+S_{35}\\S_{27}\end{cases}$	0.879	
Federated structure	$\begin{matrix}S_{33}\\S_{20}+S_{32}\end{matrix}\Big\}+S_{37}$	0.768	

V. CONCLUSION

With the popularization of Internet of things technology and application, the centralized service model of cloud computing has not met new services for big data processing and delay-sensitive applications. Edge-cloud collaborative computing can effectively support Internet of things applications and become a new environment for the deployment and execution of new complex services. This paper proposes an optimal composite service selection model based on Petri net under the dynamic heterogeneous Edge-Cloud collaborative service environment. The main contribution is to establish the Petri Net models of composite services with different structures, give the aggregation functions of different QoS attributes of composite services, then calculate the important index to evaluate the composite service, that is the comprehensive QoS value and use the Petri Net simulation tool to realize the dynamic discovery and quality evaluation of feasible execution paths, so as to select the best composite service. Considering massive servers in the service environment of Edge-Cloud collaboration, we also propose a method to filter out invalid service deployment schemes through QoS constraints in user requests.

ACKNOWLEDGEMENT

This work was supported in part by Natural Science Foundation of China under Grants 61662054, 61262082, Natural Science Foundation of Inner Mongolia under Grants 2019ZD15, 2019MS06029, Inner Mongolia Application Technology Research and Development Funding Project under Grant 201702168, Inner Mongolia Science and Technology Plan Project under Grant 2019GG372, Inner Mongolia Colleges and Universities Support Program for Young Scientific and Technological Talents under Grand NJYT-19-A02, Inner Mongolia Engineering Lab of Cloud Computing and Service Software, Inner Mongolia Key Laboratory of Data Processing and Social Computing, Inner Mongolia Engineering Lab of Big Data Analysis Technology, the Ecological Big Data Engineering Research Center of the Ministry of Education, National and Local Joint Engineering Research Center of Mongolian Intelligent Information Processing.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," *in IEEE Internet of Things Journal, vol. 3, no. 5, pp . 637-646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.*

[2] Peter, et al. "Mobile Edge Computing and the Internet of Things for Consumers: Extending cloud computing and services to the edge of the network." IEEE Consumer Electronics Magazine 5.4(2016):73 -74.

[3] A. M. Khan and F. Freitag, "On Edge Cloud Service Provision with Distributed Home Servers," *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2017, pp. 22 3-226, doi: 10.1109/CloudCom.2017.50.*

[4] K.C.Huang , and B. J. Shen . "Service deployment strategies for effi cient execution of composite SaaS applications on cloud platform." *J ournal of Systems & Software 107.sep.(2015):127-141.*

[5] J. Xiao, F. Cai . Research of three level match method about semantic web service based on ontology. 2011.

[6] L. Chang-Ming, L. U. Jian-Yun. "Semantic Web Service Selection Algorithm Based on QoS Ontology". *Journal of Southwest China Normal University(Natural Science Edition), 2011, 36(4):163-167.*

[7] M. Makhlughian,S. Mohsen Hashemi, Y. Rastegari, et al. "Web service selection based on ranking of qos using associative classification. " *International Journal on Web Service Computing, 2012.*

[8] L. Zhang , B. Zhang, C. Pahl, et al. "Personalized Quality Prediction for Dynamic Service Management Based on Invocation Patterns" *Eleventh International Conference on Service Oriented Computing ICSOC 2013. Springer-Verlag New York, Inc. 2013.*

[9] H. K. Zhu, X. L. Yu ,G. P. Zhuo, et al. Research on Services Matching and Ranking Based on Fuzzy QoS Ontology" *International Conference on Computational Aspects of Social Networks. IEEE Computer Society, 2010.*

[10] L. Qi, T. Ying, W. Dou, et al. "Combining Local Optimization and Enumeration for QoS-aware Web Service Composition" *IEEE International Conference on Web Services. IEEE, 2010.*

[11] A. Elmas. " Edge position detection and depth estimation from gravity data with application to mineral exploration". *Carbonates and Evaporites, 2019.*

[12] K. Sasaki, N. Suzuki, S. Makido and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), 2016, pp. 1122-1127, doi: 10.1109/SICE.2016.7749210.*

[13] C.F.Jian, K.Y.Qiu, M.Y.Zhang,"Two-stage Edge Service Composition and Scheduling Method for Edge Computing QoE". *Journal of Chinese Computer Systems, 2019.*

[14] Hayyolalam, Vahideh, Kazem, et al. "A systematic literature review on QoS-aware service composition and selection in cloud environment". *Journal of Network & Computer Applications, 2018.*

[15] Y.Y.Fan, X.Y.Mei. "QoS Evaluation Model for Web Services Composition". *Key Engineering Materials, 2012, 500:311-316.*

[16] Truong Khoa phan, Miguel Rocha, David Griffin, et.al. " Utilitarian Placement of Composite Services". *IEEE Transactions on Network & Service Management, 2018, PP(99):1-1*