Quality of Service (QoS)-Aware Microservices Selection Based on Local Constraints

Chellammal Surianarayanan[1,*], Pethuru Raj Chelliah[2], Manikandan Sethunarayanan Ramasamy[1], and Baby Nirmala M[3]

[1]Centre for Distance and Online Education of Bharathidasan University, Tiruchirappalli, Tamilnadu, India
[2]Edge AI division of Reliance Jio Platforms Ltd., Bangalore, India
[3]Department of Computer Applications, Holy Cross College, Tiruchirappalli, Tamilnadu, India
Email: chellammals@bdu.ac.in (C.S.); peterindia@gmail.com (P.R.C.); manirs2004@yahoo.co.in (M.S.R.); babynirmala7@yahoo.co.in (B.N.M.)
[*]Corresponding author
Manuscript received February 22, 2023; revised April 11, 2023; accepted August 14, 2023; published April 1, 2024

***Abstract*—In the world of microservices, to deliver a particular task, there may exist several microservices. Though they are functionally similar, their Quality of Service (QoS), which refers to their non-functional attributes would be different. Quality of Service helps to select services having adequate QoS as per the demands of service consumers so that the composition of microservices can happen within the given QoS constraints. Global QoS based selection which selects microservices at composite service level is associated with inherent exponential time complexity. In this work, a method has been proposed for microservices selection based on local QoS constraints. The given QoS constraints are divided into local constraints corresponding to the number of tasking involved in the process. Then for each task, the services which satisfy the local constraints of the task, are identified as candidate services. Afterwards the service having the highest utility is returned as the best service for that task.**

The proposed method has been evaluated with a case study and the results are discussed. The case taken for the study is microservices based travel_plan process, consisting of three sequential tasks namely, flight ticket booking, hotel booking and cab booking tasks. The proposed method has found to yield two feasible solutions which guarantee the given global constraints of QoS attributes. Also, in case candidate service is not found for a task, then the method of assigning local constraints is relaxed as follows. For a task, whenever maximum value of a QoS attribute is less than that computed by the proposed method of equal distribution, then the maximum value of the attribute itself will be assigned as local constraint and the remaining QoS attribute is used in the allocation of local constraint of that attribute to other tasks involved in a process.

***Keywords*—Quality of Service (QoS) attributes, QoS-aware microservices composition, local constraints based microservices selection**

I. Introduction

In MicroServices Architecture (MSA), each microservice is designed with limited functionality and thus realization of any business process requires the composition of many such microservices which are required to implement the process at hand. In the services based architectures, service consumers tend to avail the business processes according to their Quality of Service (QoS) demands and preferences. The demands of QoS vary considerably among consumers. Consider a user looking for a travel_plan service to plan for a travel trip to Singapore for a medical treatment he should undergo in a short while. In this case, the user's primary QoS requirements would be availability of ticket. Also, his preference would be availability rather than cost. Consider another user who plans for a casual travel trip to Singapore with his family. In this case, the user does not have any urgent requirement like the former one and here, the user considers both availability as well as cost into account. In addition, may demand for a service having low cost. Hence in practice, service providers implement microservices, though of same function, but of different QoS, corresponding to varying QoS needs of consumers. Thus, during composition, QoS becomes the discriminator factor while selecting appropriate microservices for composition. Basically, the microservices composition is a well systematic process, carried out in 4 different stages namely planning, discovery, selection and invocation [1] as shown in Fig. 1.

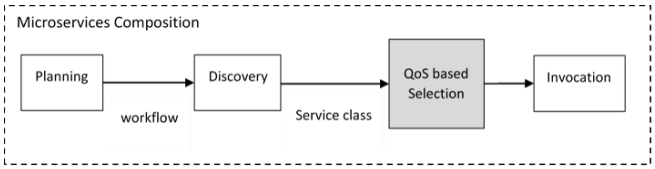

Fig. 1. Microservices composition.

In planning, the required business process is captured and represented as workflow which consists of several abstract tasks that are to be executed according to a specific pattern so that the desired process is realized. In discovery, microservices that are functionally capable to implement the tasks are discovered from the pool of microservices. Typically, this results in numerous functionally similar microservices and this due to the fact that service providers intentionally publish functionally similarly microservices but having different QoS to meet the disparate QoS needs of consumers. Also, the functionally similar microservices that corresponds to a particular task is termed as service class. In selection, the most appropriate service combination that corresponds to the workflow of process at hand has to be found out based on the QoS demands and preferences of consumers. In invocation, the selected services would be invoked according to the execution pattern so that the process is implemented with respect to the QoS needs of consumers.

The scope of the current work is bounded by QoS based selection. Selection of services according to QoS can be performed using two methods, global and local. In global method, the problem selecting appropriate service combination for a given workflow is performed by enumerating all possible service combinations that correspond to various tasks of the workflow. Here the number

of possible combinations involved in the construction of composite service increases exponentially with respect to number of services

Consider "*m*" number of tasks involved in a workflow. Also, consider that there are "*n*" number of functionally similar microservices are available for each task. To illustrate the exponential evolution of number of combinations keeping $n = 10$ and the value of m is varied from 1 to 10. The number of service combinations has been computed using the formula m^n. The exponential increase in the number of combinations with respect to number of services is also shown in Fig. 2.

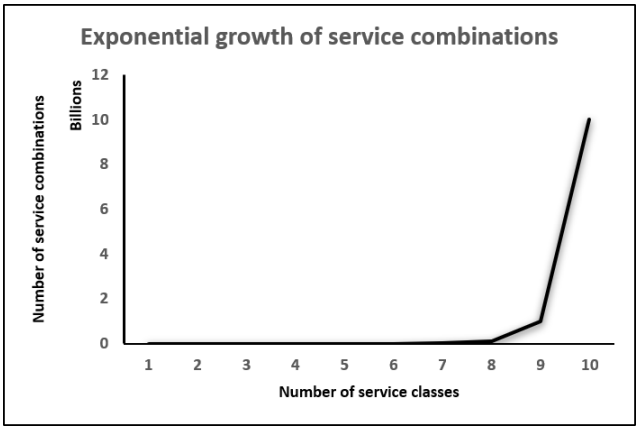

Fig. 2. Exponential increase in number of combinations with respect to increase in number of service classes.

Due to time complexity of global selection methods, local selection methods are preferred. In local selection, the selection is done based on individual tasks rather than the end-to-end workflow level. Based on the idea of divide and conquer technique, here the QoS demands for entire business process given by the service consumer is divided into task level constraints called local constraints and selection will be performed at task level. Then the services selected for individual tasks are combined to produce the appropriate service combination for the entire workflow.

In this work a method has been proposed for QoS based micoservices selection based on local constraints. Local constraints are mathematically derived from the standard aggregation expression of the QoS attributes and assigned to tasks. Then for each task the services which satisfy its local constraints are identified as candidate services. Utility function is defined to select most appropriate candidate service for each task. Here, utility function is formulated as the average of normalized QoS values of different attributes. The priority of consumers over different QoS attributes are taken into account with a simple additive weighting technique.

The rest of the paper is organized as follows. Section II highlights the research work that handle QoS related issues. Section III describes the proposed method for local selection. Section IV describes the evaluation of the proposed method with a case study. Section V describes the results and limitations of the proposed method. Section VI presents the method of relaxing local constraints and Section VII concludes the work.

II. Related Work

Microservices are typically deployed in cloud computing environment and their QoS tend to change dynamically. There are some research works which have focused on how to meet the required QoS of microservices by using effective resources allocation or suitable placement of microservices or load aware scheduling aspects, etc. Zhang *et al.* [2] developed a set of scalable and validated machine learning models to find out the performance dependencies among the microservices and allocate appropriate resources so that the required QoS is met. A runtime system called Nautilus is proposed to ensure the QoS of microservices while connecting multiple services in edge devices [3]. The core components of this system include a communication-aware microservice mapper which divides the microservice graph into partitions according to communication overhead, a contention-aware resource manager which detects the contention as well as determines the optimal resource allocation and load-aware scheduler which monitors the QoS of the entire service and migrates services according to QoS needs. Stevant *et al.* [4] proposed a method to place the participating microservices of an application on various user defined devices effectively so that the response time and thus user's satisfaction can be me. A scalable QoS-aware scheduling policy for batch placement of microservices in fog environment is proposed with an objective to satisfy the QoS needs while minimizing the fog resources [5]. In contrast to machine learning techniques which not only require huge amount of training data but also slow in adapting to dynamically changing microservices operating environment, the authors developed a light weight resource manager that performs efficient resource allocation through opportunistic resource reduction, at the same time ensuring QoS demands [6].

Some other research works have used machine learning algorithms to handle the QoS requirements of microservices. Machine learning technique is used to discover and select services according to the required context and QoS [7]. A self-adaptive framework is proposed to handle QoS requirements and managing the unexpected changes in the QoS, using reinforcement algorithm [8]. Chang *et al.* [9], proposed Bayesian algorithm based model to classify the microservices using the data related to system calls. Ding *et al.* [10] proposed a microservices selection strategy based on sub-deadline of task where sub-deadline is computed based on processing speed of service instances, transferring speed of network and degree of task concurrency

Some research works focus on monitoring aspect of QoS of microservices. For example, Stefanic *et al.* [11] proposed a novel approach which monitors the different QoS attributes of components and microservices deployed in cloud environment. In this work, the authors proposed Qualitative Metadata Markers method in which, a software component is developed right from scratch, deployed in cloud and it is monitored for various quality attributes. A specific example of file upload service is taken by the authors and in this case, the quality attributes namely, file upload time, upload speed, jitter, packet loss, IO read, IO write, total time, etc., are monitored. Al-Masri [12] presented microservices Quality of Service Management (mQoSM) framework which measures the overall quality of microservices based Industrial Internet of Things (IIoT) applications. Bhamare *et al.* [13] handled the problem of scheduling microservices across different clouds according to user level and service level agreements with

specific focus on cost and latency.

The global approaches such as [14, 15] suffer from the exponential time complexity. Senivongse and Wongsawangpanich [16] employs genetic algorithm and suffers from poor time characteristics for large number of services. In another research work, Mardukhi *et al.* [17], artificial neural network is used to predict the QoS of candidate service instances so that the genetic algorithm can select the best predicted service. Similarly, Ding *et al.* [18] handles the web services selection based on global QoS using genetic algorithm. Yuan *et al.* [19] have used global selection method for web services. QoS global optimization and dynamic re-planning web service selection algorithm has been proposed to select optimal execution plan for web services while meeting the QoS requirements globally [20].

Alrifai and Risse [21] decompose the given global QoS constraints using Mixed Integer Programming (MIP), the time characteristics of these methods become poor with respect to number of services. The method proposed by Qi *et al.* [22] uses a heuristics based service composition where the range of extreme values of QoS attributes are divided in many levels and candidate services are selected from different levels to enumerate the possible combinations to produce near-to-optimal solutions. Here when the number of levels is large, the method will suffer from long computation time. The method presented by Guang *et al.* [23] uses an adaptive adjustment method based on fuzzy logic to decompose the global constraints into local constraints. It uses empirical utility. The decomposition discussed in [24] uses genetic method to decompose the global constraints. The local selection model proposed by Ye *et al.* [25] presented a method to increase web service composition by using enhanced decomposition and greedy algorithm which have weak connection with QoS values of candidate services. Rodriguez-Mier *et al.* [26] proposed a shortest path algorithm to the service match which consists of many valid composition patterns. Here also, finding suitable composition pattern according to the QoS is NP hard problem and authors applied a set of admissible optimizations to reduce the search space. Deng *et al.* [27] proposed QoS-aware service pruning method based on the correlations among QoS attributes to prune non-optimal candidate services for composition. Alrifai *et al.* [28] proposed a skyline approach to reduce the number of candidate services involved in composition, according to their QoS. Chattopadhyay and Banerjeem [29] proposed a pareto based method to identity optimal solutions from services retrieved using Input-Output Model along with two other heuristics based methods.

III. Proposed Method of QoS Based Microservices Selection

At first the tasks involved in a business process has to be determined and represented as workflow. The workflow exhibits tasks and the specific pattern of execution of tasks. The workflow may of simple sequential kind as in Fig. 3 or it may of a combinational one as in Fig. 4.

As any combinational workflow can be reduced into a sequential pattern [21, 30], only sequential workflows are considered in this work. The block diagram of the proposed method is shown in Fig. 5. There are 4 steps, namely, querying step in which the QoS demands and preferences are captured.

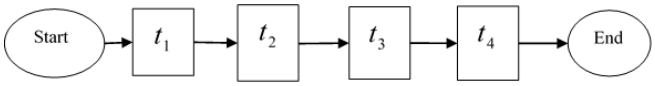

Fig. 3 Sequential workflow.

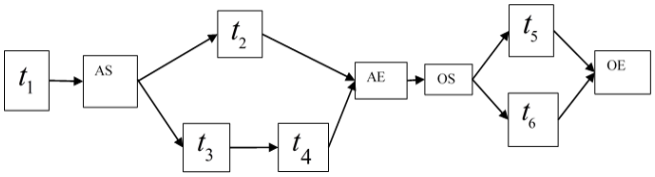

Fig. 4. Combinational workflow.

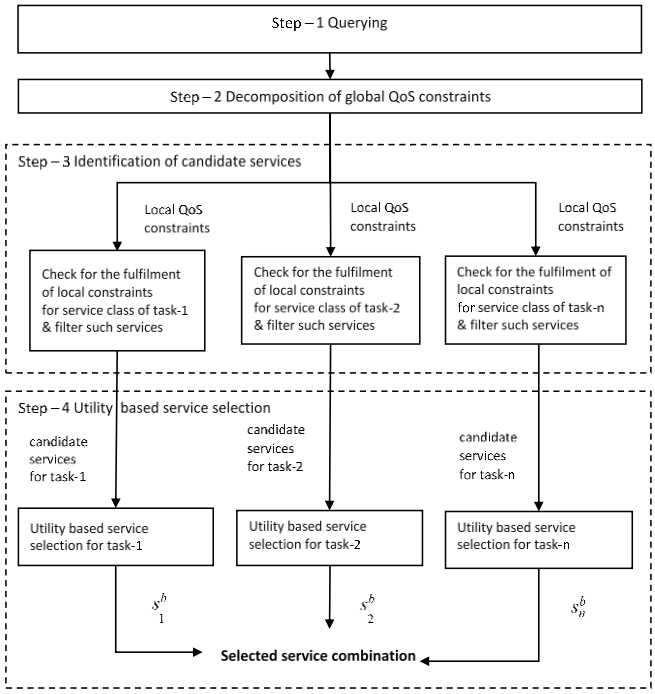

Fig. 5. QoS based microservices selection based on local constraints.

A. Querying Step

QoS requirements vary from consumer to consumer and it basically arise based on the nature of application. In querying step, the QoS requirements of service consumers are captured. QoS requirements include QoS constraints and QoS preferences. QoS constraints refer to the conditions on the values of the attributes that to be satisfied whereas QoS preferences refer to the consumer's priority over different QoS attributes. Both QoS constraints and QoS preferences are meant for the entire workflow. A typical query would be as given below:

Find a travel_plan_service with
QoS constraints (response time=12 seconds, cost=300$ & availability=90%) and
QoS preferences (40% priority to response time, 10% priority to cost & 50% priority to availability)

B. Decomposition Step

The given global QoS constraints are decomposed equally to all participating tasks and the local constraints of QoS constraints are derived from the standard aggregation formulae used to compute the QoS attributes of a composite service.

Consider a workflow having m number of service classes denoted by $sc(1), sc(2), sc(3), \ldots, sc(m)$. Let $s(i, j)$

denote the j^{th} service in i^{th} service class. Let n_i denote the number of services in i^{th} service class. Let cs denote a composite service formed by combining a particular service, say, l_i from each service class i. It is denoted as

$$cs = s(1, l_1), s(2, l_2), s(3, l_3), \ldots, s(m, l_m) \quad (1)$$

Let $rt(s(i,j))$, $c(s(i,j))$, $a(s(i,j))$, $rel(s(i,j))$, $tp(s(i,j))$ and $rp(s(i,j))$ denote the response time, cost, availability, reliability, throughput and reputation of $s(i,j)$. Let $rt(cs)$, $c(cs)$, $a(cs)$, $rel(cs)$, and $tp(cs)$ denote the response time, cost, availability, reliability and throughput of the composite service. Now the QoS values of composite service are computed using Eqs. (2)–(6) [17, 25, 28].

$$rt(cs) = \sum_{i=1}^{m} rt(s(i,l_i)) \quad (2)$$

$$c(cs) = \sum_{i=1}^{m} c(s(i,l_i)) \quad (3)$$

$$a(cs) = \prod_{i=1}^{m} a(s(i,l_i)) \quad (4)$$

$$rel(cs) = \prod_{i=1}^{m} rel(s(i,l_i)) \quad (5)$$

$$tp(cs) = \min\{tp(s(i,l_i)) | 1 \le i \le m\} \quad (6)$$

In this work, it is proposed to divide the given global constraints equally to all service classes involved in the workflow. It is proposed to divide the global constraints according to the aggregations formulae used for the computations of QoS attributes of composite service. When the local constraints are derived from the aggregation formulae, the service combination that will be obtained using local selection method will give guarantee to meen the given global constraints. Secondly, the proposed method is mathematically derived one. Thus, in the proposed method, the given global constraints are equally divided into those service classes which are involved in the process

For any business process, certain quality attributes, namely, response time, cost, availability, reliability, throughput are very important in deciding the performance of a process. Let rt^{global}, c^{global}, a^{global}, rel^{global} and tp^{global} denote the global QoS constraints of response time, cost, availability, reliability and throughput. Let $rt^{local}(sc(i))$, $c^{local}(sc(i))$, $a^{local}(sc(i))$, $rel^{local}(sc(i))$ and $tp^{local}(sc(i)$ denote the local QoS constraints of response time, cost, availability, reliability, throughput and reputation any i^{th} service class involved in the workflow. Now, the given global constraints are equally divided into various service classes as given in Table 1.

Table 1. Collection of test services in different service classes along with their QoS values

S.No.	Ticket booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$	Hotel booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$	Cab booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$
1	*s(1,1)*	6	50	90	*s(2,1)*	5	150	90	*s(3,1)*	12	200	90
2	*s(1,2)*	7	90	89	*s(2,2)*	8	130	92	*s(3,2)*	10	180	96
3	*s(1,3)*	8	180	95	*s(2,3)*	9	145	94	*s(3,3)*	11	190	98
4	*s(1,4)*	4	60	98	*s(2,4)*	10	65	98	*s(3,4)*	9	100	99
5	*s(1,5)*	3	140	97	*s(2,5)*	3	70	99	*s(3,5)*	8	120	100
6	*s(1,6)*	4	130	98	*s(2,6)*	4	80	100	*s(3,6)*	7	130	98
7	*s(1,7)*	2	120	95	*s(2,7)*	6	90	95	*s(3,7)*	6	160	92
8	*s(1,8)*	10	70	94	*s(2,8)*	7	200	93	*s(3,8)*	5	90	91
9	*s(1,9)*	15	80	93	*s(2,9)*	14	100	91	*s(3,9)*	4	100	98
10	*s(1,10)*	9	120	92	*s(2,10)*	12	150	90	*s(3,10)*	3	135	89

$$rt^{local}(sc(i)) = \frac{rt^{global}}{m}, i = 1,2,3,\ldots,m \quad (7)$$

$$c^{local}(sc(i)) = \frac{c^{global}}{m}, i = 1,2,3,\ldots,m \quad (8)$$

$$a^{local}(sc(i)) = (a^{global})^{1/mm},\ i = 1,2,3,\ldots,m \quad (9)$$

$$rel^{local}(sc(i)) = (rel^{global})^{1/mm},\ i = 1,2,3,\ldots,m \quad (10)$$

$$tp^{local}(sc(i) \ge tp^{global}, i = 1,2,3,\ldots,m \quad (11)$$

The computed QoS values of the individual services would form as the task level constraints, called as local constraints. The local constraints involved of any j^{th} service class is denoted as in Eq. (12).

$$c_i = (rt^{local}(sc(i)), c^{local}(sc(i)), a^{local}(sc(i)), rel^{local}(sc(i)), tp^{local}(sc(i)) \quad (12)$$

These local constraints will be used in the identification of candidate services.

C. Identification of Candidate Services

In this step, every service in a service class is checked for the fulfilment of the local constraints and the services which satisfy the local constraints are filtered out as candidate services for further computation. While performing QoS matching between the query and a service, one has to normalize the values of QoS attributes of both query and the service. Because, different service providers would have used different units and measurements. So, it is essential to normalize the values in the range 0 to 1. With respect to QoS parameters, there are two kinds of parameters; certain parameters like availability, scalability, are required to be maximized whereas there as other parameters such as response time, cost are required to be minimized. For each task in the workflow, as mentioned earlier, there would be many services available which are functionally similar but have different QoS. So, for a given service class, for each QoS

parameter, there would be a range of values will be available with minimum and maximum.

Let $q_k(s(i,j))$ denote k^{th} QoS attribute of j^{th} service in i^{th} service class. Let $Q_{\max}(i,k)$ denote the maximum value of k^{th} QoS attribute for i^{th} service class. Let $Q_{\min}(i,k)$ denote minimum of k^{th} QoS attribute for i^{th} service class. Now the normalized value of $q_k(s(i,j))$ is computed using

$$normalized\_q_k(s(i,j)) = \frac{(Q_{\max}(i,k) - q_k(s(i,j))}{(Q_{\max}(i,k) - Q_{\min}(i,k))} \quad (13)$$

{for attributes that are required to be minimized}

$$normalized\_q_k(s(i,j)) = \frac{q_k(s(i,j)) - Q_{\min}(i,k))}{(Q_{\max}(i,k) - Q_{\min}(i,k))} \quad (14)$$

{for attributes that are required to be maximized}

The QoS constraints of user's query are also should be normalized using the above equation before performing the QoS matching. Thus, the services which comply all the local constraints are identified as candidate services

D. Service Selection

In this step the utility of candidate services is computed, taking into consideration the QoS preferences of the consumers. The key point to be noted here is that the QoS preferences of each task is same as that of the workflow. In addition, there are two cases, (i) a service consumer may give only the QoS constraints; and (ii) a service consumer may give both QoS constraints and preferences

Case 1—When only QoS constraints are given: In this case, the utility of j^{th} service in i^{th} service class, denoted by $u(s(i,j))$ is computed using Eq. (15)

$$u(s(i,j)) = \frac{\sum_{k=1}^{n} normalized\_q_k(s(i,j))}{N} \quad (15)$$

In Eq. (15) N denote the number of QoS attributes

Case 2—When both QoS constraints and preferences are given: In this case, the utility of j^{th} service of i^{th} service class is computed using

$$u(s(i,j)) = \sum_{k=1}^{N} normalized\_q_k(s(i,j) \times w_k \quad (16)$$

In Eq. (16), w_k denotes the weight of k^{th} qos attributes. Also, $\sum_{k=1}^{N} w_k = 1$. Here, the preferences included while computing utility of the services using Simple Additive Weighting method [31]. Now, for each i^{th} service class the service having highest utility is selected as the best service and it is denoted by s_i^b. Thus, best services selected for all service classes would form the most appropriate service combination for implementing the given business process. By invoking the best service for each task according to the execution pattern of the workflow, the given business process will get implemented.

Further the proposed method has been implicated and evaluated with a case study in the subsequent section.

IV. Evaluation of the Proposed Method

The proposed method of QoS based microservices selection has been evaluated using a case study. The details of case under study are described below. Consider a microservices based travel-plan process as the case under study. Consider that the case is represented as a sequential workflow consisting of three abstract tasks, t1, t2 and t3 as shown Fig. 6. The task, t1 books air tickets for a travel trip from a source location to a destination location on the desired dates given by the user. The task t2 reserves rooms for the dates of halt in the place of visit. The task-3 books cab for different sight-seeing locations. Here, it is obvious that only after the confirmation of air tickets only, the task t2 can be executed. Similarly, after the execution of t2 only, the task t3 will be executed. So, the above tasks are executed in sequential manner as in Fig. 6.

Fig. 6. Travel_plan process.

The proposed method is applied to the case under study as follows.

A. Querying Step

Consider that a service consumer is querying for *travel_plan process* with QoS demands and preferences as:

QoS constraints (response time=12 secs, availability = 90% and cost 300$).

QoS preferences (40% priority to response time, 10% priority to cost & 50% priority to availability)

B. Decomposition Step

Let rt^{global}, c^{global} and a^{global} denote the global constraint of response time, cost and availability. Let $rt^{local}(sc(i))$, $c^{local}(sc(i))$ and $a^{local}(sc(i))$ denote the local constraints of response time, cost and availability of j^{th} service class

$$rt^{local}(sc(i)) = \frac{rt^{global}}{m}, i = 1,2,...,m \quad (17)$$

$$c^{local}(sc(i)) = \frac{c^{global}}{m}, i = 1,2,...,m \quad (18)$$

$$a^{local}(sc(i)) = (a^{global})^{1/mm},\ i = 1,2,\ldots,m \quad (19)$$

In the above example, the number of tasks in the workflow, $m = 3$. Now, response time of i^{th} service class is computed as

$$rt^{local}(sc(i)) = \frac{rt^{global}}{m} = \frac{12}{3} = 4,\ i = 1,2,3,\ldots,m \quad (20)$$

Similarly, the local cost constraint i^{th} service class is computed using

$$c^{local}(sc(i)) = \frac{c^{global}}{m} = \frac{300}{3} = 100, i = 1,2,3,...m \quad (21)$$

The value of $c^{local}(sc(i))$ is computed as 100. In contrast to response time and cost, availability is a multiplicative attribute. In the given example, the local constraint of availability of jth service class is computed as

$$a^{local}(sc(i)) = (a^{global})^{\frac{1}{m}} = \frac{0.9^{1/3}}{3} \quad (22)$$

$$a^{local}(sc(i)) = 0.965, i = 1,2,3,\ldots,m \quad (23)$$

Now, the local constraints of any i^{th} service class is formulated as

$$C_j = (rt^{local}(sc(i)), c^{local}(sc(i)), a^{local}(sc(i))) \quad (24)$$

$$C_j = (4,100,96.5) \quad (25)$$

C. Identification of Candidate Services

To illustrate this step, a collection of functionally similar services corresponding to the three different tasks has been generated, with service IDs and QoS values as in Table 1. For simplicity, 10 test services have been created in each service class. Local QoS constraints of the query are matched against the available services. The services which satisfy the local constraints are filtered out as candidate services. The candidate services which satisfy the given constraints are given in Table 2. The QoS constraints of user's query are also should be normalized using the above equation before performing the QoS matching. The normalized values are given in Table 3.

D. Service Selection

Case 1—When only QoS constraints are given by consumers, the utility of candidate services is computed using Eq. (15). The utility values of candidate services are given in Table 4. From the Table 4, it is clear that the service combination (s(1,4), s(2,5), s(3,9)) would be selected with total utility of (0.926), which is computed as the average value of utility of selected services.

Case 2—When consumers have both QoS constraints and QoS preferences for different QoS attributes, it is considered that the task level QoS preferences are the same as QoS preferences and the given preferences are taken into account as weights of the attributes. In this case, the utility will be computed using Eq. (16). The computed utility values are given in Table 5.

Table 2. Candidate services for different service classes

S.No.	Ticket booking task	*q1(s(i,j))*	*q2(s(i,j))*	*q3(s(i,j))*	Hotel booking task	*q1(s(i,j))*	*q2(s(i,j))*	*q3(s(i,j))*	Cab booking task	*q1(s(i,j))*	*q2(s(i,j))*	*q3(s(i,j))*
1	*s(1,4)*	4	60	98	*s(2,5)*	3	70	99	*s(3,9)*	4	100	99
2					*s(2,6)*	4	80	100				

Table 3. Normalized QoS attributes of candidate services

S.No	Ticket booking task	Normalized QoS			Hotel booking task	Normalized QoS			Cab booking task	Normalized QoS		
		q1(s(i,j))	*q2(s(i,j))*	*q3(s(i,j))*		*q1(s(i,j))*	*q2(s(i,j))*	*q3(s(i,j))*		*q1(s(i,j))*	*q2(s(i,j))*	*q3(s(i,j))*
1	*s(1,4)*	0.85	0.92	1	*s(2,5)*	1	0.96	0.9	*s(3,9)*	0.89	0.91	0.91
2					*s(2,6)*	0.91	0.89	1				

Table 4. Utility values of candidate services

S.No	Ticket booking task		Hotel booking task		Cab booking task	
	Candidate Service	Utility	Candidate Service	Utility	Candidate Service	Utility
1	*s(1,4)*	0.923	*s(2,5)*	0.953	*s(3,9)*	0.903
2			*s(2,6)*	0.933		

Table 5. Utility values obtained with weightage to QoS attributes

S.No	Ticket booking task		Hotel booking task		Cab booking task	
	Candidate Service	Utility	Candidate Service	Utility	Candidate Service	Utility
1	*s(1,4)*	0.9352	*s(2,5)*	0.946	*s(3,9)*	0.902
2			*s(2,6)*	0.953		

From Table 5, it is understood that the service combination (s(1,4), s(2,6), s(3,9)) would be selected with total utility of (0.930). Here, the total utility is computed as the average value of utility of selected services.

V. Results and Discussion

There are two aspects to be considered in the discussion. One is related to the satisfaction of given global constraints. The other is detection efficiency of the method. The proposed method of local selection is simple and straight forward from their corresponding aggregation formulae for different QoS attributes. Since the local constraints are set based on the aggregating functions, it is 100% ensured that the selected service combination ensures 100% guarantee to fulfil the given global constraints. Secondly with local selection the number of combinations involved is m×n where m refers to the number of tasks or service classes and n refers to the number of services in each service class. It is greatly reduced with compared to the global method which is n^m.

Despite the above features, there may be situations where the method may not be able to detect an appropriate service combination even if it exists. The results obtained using global and local selection methods for the case study are given in Table 6.

Table 6. Comparison of proposed method with global method

Proposed method	Global method
(*s(1,4), s(2,5), s(3,9)*)	(*s(1,4), s(2,5), s(3,9)*)
(*s(1,4), s(2,6), s(3,9)*)	(*s(1,4), s(2,6), s(3,9)*)
	(*s(1,6), s(2,5), s(3,9)*)
	(*s(1,7), s(2,5), s(3,9)*)
	(*s(1,7), s(2,6), s(3,9)*)

From Table 6, it is found that the global method produced 5 solutions whereas the proposed method has produced 2 solutions. Here, one may argue that the local method is not able to detect or identify the all the possible solutions when compared to global method. One has to look very carefully the time involved in global method. Despite the efficiency of global methods, it has a serious limitation that cannot be put for practical use when there are numerous microservices. Another important thing is that the local method identified two feasible solutions which guarantees the given global constraints. It is adequate to realize the process. But there may be situations where the method may not find any solution for one or more tasks. In such cases, the local constraints are relaxed as discussed in the subsequent section.

VI. Relaxation of Local Constraints

If no candidate service is found for a task, then the constraints of tasks are relaxed as discussed below. The basic idea behind this relaxation is based on the fact that there may be situations, where the maximum value of a QoS attribute for a service class may be less than the local constraint obtained by equal distribution method. In that case, the maximum value of the QoS attribute itself will be assigned as constraint for that attribute, for the service class and the balance between the maximum value and local constraint computed by equal distribution will be utilized for a service class whose minimum value of QoS attribute itself is greater than the computed constraint of that attributes. The proposed idea for relaxation of local constraints is illustrated with response time attribute using three service classes as shown in Fig. 7.

From Fig. 7, it is clear that, minimum response time, maximum response time of service class-1 denoted by *min_rt(sc(1)), max_rt(sc)1)* respectively are less than the constraint of response time computed by equal distribution method, denoted by (rt^{global}/m).

For service class-2, minimum response time denoted by *min_rt(sc(2))* is less than (rt^{global}/m) but the maximum response time denoted by *max_rt(sc(2))* is larger than (rt^{global}/m).

For service class-3, both minimum response time, maximum response time of service class-3 denoted by *min_rt(sc(3)), max_rt(sc(1))* is larger than (rt^{global}/m)

In the constraint relaxation method, at first, for service class-1, since the $max\_rt(sc(1)) < rt^{local}(sc(1))$, *max_rt(sc(1))* has been fixed as constraint for response time for service class-1.

Here the difference, $((rt^{global}/m) - max\_rt(sc(1)))$ will be used for a service class which is in need of extra amount of constraint.

For service class 2, since *max_rt(sc(2))* is greater than (rt^{global}/m), then, (rt^{global}/m) will be fixed as the constraint of service class-2.

Now, for service class 3, *min_rt(sc(3))* itself is greater than (rt^{global}/m). The difference $((rt^{global}/m) - max\_rt(sc(1)))$ will be used to provide the extra amount of constraint required for service class-3.

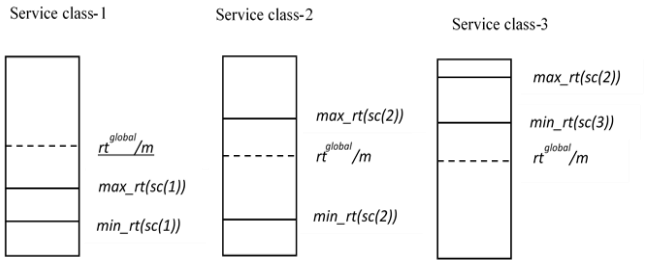

Fig. 7. Relaxation of local constraints (e.g., response time).

Consider a very small collection of services as shown in Table 7 for discussion.

Let the consumer's global constraints be (15, 300, 90).

When the above constraints are decomposed using the given method, the local constraints obtained are (5, 100, and 96.5).

The candidate services obtained by the proposed method of equal distribution is shown in Table 8.

From Table 8, it is found that there is no candidate service found for cab booking task. Now relaxation of constraints is applied. For service class 1 and service class 2, the value of maximum value of response time is 4 and it is less than the constraint of response time computed using the proposed method which is 5. Now, allot the maximum value of response time as constraint for service classes 1 and 2, the, a balance response time of two will be added with 5 and a constraint value of 7 is allotted to service class 3.

Results obtained with relaxation of constraints is shown in Table 9. As illustrated in the above example, relaxation of constraints helps in effective extraction of feasible service combinations. Relaxing constraints increases the ability of the proposed method in identifying feasible solutions.

Table 7. A small collection of services chosen for illustrating relaxation of constraints

S.no	Ticket booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$	Hotel booking task	q1(s(i,j))	q2(s(i,j))	q3(s(i,j))	Cab booking task	q1(s(i,j))	q2(s(i,j))	q3(s(i,j))
1	*s(1,4)*	4	60	98	*s(2,4)*	10	65	98	*s(3,4)*	9	100	99
2	*s(1,5)*	3	140	97	*s(2,5)*	3	70	99	*s(3,5)*	8	120	100
3	*s(1,6)*	4	130	98	*s(2,6)*	4	80	100	*s(3,6)*	7	130	98
4	*s(1,7)*	2	120	95	*s(2,7)*	6	90	95	*s(3,7)*	6	160	92

Table 8. Candidate services obtained using proposed method with local constraints (5, 100, 96.5)

S.no	Ticket booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$	Hotel booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$	Cab booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$
1	*s(1,4)*	4	60	98	*s(2,5)*	3	70	99	No candidate service			
2					*s(2,6)*	4	80	100				

Table 9. Candidate services obtained with relaxed constraints (4, 100, 96.5)

S.no	Ticket booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$	Hotel booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$	Cab booking task	$q_1(s(i,j))$	$q_2(s(i,j))$	$q_3(s(i,j))$
1	*s(1,4)*	4	60	98	*s(2,5)*	3	70	99	*s(3,6)*	7	130	98
2					*s(2,6)*	4	80	100	*s(3,7)*	6	160	98

VII. Conclusion

In this work, a method has been proposed for QoS-aware microservices selection for composition. The method initially decomposes the given global constraints by equal distribution method and selects services from different service classes according to local constraints. In case candidate service is not found for one or more service classes, the method will relax the local constraints without violating the given global constraints and look for candidate services. Also, in case the method of relaxation of local constraints does not help in identifying candidate services, then selection of services would be done based on an efficient global selection method which uses indexing scheme for quick retrieval and the method is currently in its implementation stage. In the indexing scheme, separate indices are being created to maintain the values of different QoS attributes of services. It is proposed to perform the matching of QoS of services during the functional matching process itself. The key point is that separate indices are maintained for inputs, outputs and QoS values. This will certainly retrieve the available services that meet both functional and QoS requirements of users. Here the computation time is made effective with the help of previously computed indices.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Chellammal Surianarayanan carried out the full work and prepared the entire document. Manikandan Sethunarayanan Ramasamy helped in the formulation of mathematical representation of the concept. Pethuru Raj Chelliah helped in verifying the concept. Baby Nirmala M helped in editing. All authors had approved the final version.

References

[1] A. AlSedrani and A. Touir, "Web service composition processes: A comparative study," *International Journal on Web Service Computing (IJWSC)*, vol. 7, no. 1, pp. 1–21, 2016. doi: 10.5121/ijwsc.2016.7101

[2] Y. Zhang, W. Hua, Z. Zhou, G. E. Suh, and C. Delimitrou, "Sinan: ML-based and QoS-aware resource management for cloud microservices," in *Proc. the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 167–181. doi: 10.1145/3445814.3446693

[3] K. Fu *et al.*, "QoS-aware and resource efficient microservice deployment in cloud-edge continuum," in *Proc. 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 932–941. doi: 10.1109/IPDPS49936.2021.00102

[4] B. Stevant, J.-L. Pazat, and A. Blanc, "QoS-aware autonomic adaptation of microservices placement on edge devices," in *Proc. the 10th International Conference on Cloud Computing and Services Science (CLOSER 2020)*, 2020, pp. 237–244.

[5] S. Pallewatta, V. Kostakos, and R. Buyya, "QoS-aware placement of microservices-based IoT applications in Fog computing environments," *Future Generation Computer Systems*, vol. 131, pp. 121–136, 2022. https://doi.org/10.1016/j.future.2022.01.012

[6] M. R. Hossen, M. A. Islam, and K. Ahmed, "Practical efficient microservice autoscaling with QoS assurance," in *Proc. the 31st International Symposium on High-Performance Parallel and Distributed Computing*, 2022, pp. 240–252. https://doi.org/10.1145/3502181.3531460

[7] M. Caporuscio, M. De Toma, H. Muccini, and K. Vaidhyanathan, "A machine learning approach to service discovery for microservice architectures," in *Proc. the 2021 European Conference on Software Architecture*, Springer, Cham., 2021, pp 66–82. https://doi.org/10.1007/978-3-030-86044-8_5

[8] M. D'Angelo, M. Caporuscio, V. Grassi, and R. Mirandola, "Decentralized learning for self-adaptive QoS-aware service assembly," *Future Generation Computer Systems*, vol. 108, pp. 210–227, 2020.

[9] H. Chang, M. Kodialam, T. V. Lakshman, and S. Mukherjee, "Microservice Fingerprinting and Classification using Machine Learning," in *Proc. 2019 IEEE 27th International Conference on Network Protocols (ICNP)*, 2019, pp. 1–11. doi: 10.1109/ICNP.2019.8888077

[10] Z. Ding, S. Wang, and M. Pan, "QoS-constrained service selection for networked microservices," *IEEE Access*, vol. 8, pp. 39285–39299, 2020. doi: 10.1109/ACCESS.2020.2974188

[11] P. Štefanic, M. Cigale, A. Jones, and V. Stankovski, "Quality of service models for microservices and their integration into the SWITCH IDE," in *Proc. 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2017, pp. 215–218. doi: 10.1109/FAS-W.2017.150

[12] E. Al-Masri, "QoS-aware IIoT microservices architecture," in *Proc. 2018 IEEE International Conference on Industrial Internet (ICII)*, 2018, pp. 171–172. doi: 10.1109/ICII.2018.00030

[13] D. Bhamare, M. Samaka, A. Erbad, R. Jain, and L. Gupta, "Exploring microservices for enhancing internet QoS," *Journal of Transactions on Emerging Telecommunications Technologies (ETT)*, vol. 29, issue 11, e3445, 2018. https://doi.org/10.1002/ett.3445

[14] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition," in *Proc. the 12th International Conference on World Wide Web*, ACM, 2003, pp 411–421.

[15] L. Zeng and B. Benatallah, "A QoS-aware middleware for web service composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[16] T. Senivongse and N. Wongsawangpanich, "Composing services of different granularity and varying QoS using genetic algorithm," in *Proc. World Congress on Engineering and Computer Science*, 2011, vol. I, pp. 388–393.

[17] F. Mardukhi, N. N. Bakhsh, K. Zamanifar, and A. Barati, "QoS decomposition for service composition using genetic algorithm," *Applied Soft Computing*, vol. 13, issue 7, pp. 3409–3421, July 2013.

[18] Z. J. Ding, J. J. Liu, Y. Q. Sun, C. J. Jiang, and M. C. Zhou, "A transaction and QoS-aware service selection approach based on genetic algorithm," *IEEE Trans. Syst. Man Cybern. Syst*., vol. 45, pp. 1035–1046, 2017.

[19] Y. Yuan, W. Zhang, X. Zhang, and H. Zhai, "Dynamic service selection based on adaptive global QoS constraints decomposition," *Symmetry*, vol. 11, no. 3, 403, 2019. https://doi.org/10.3390/sym11030403

[20] N. Zhang, "Service discovery and selection based on dynamic QoS in the internet of things," *Complexity*, 6642514, 2021. https://doi.org/10.1155/2021/6642514

[21] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," in *Proc. the 18th International Conference on World Wide Web*, ACM, 2009, pp. 881–890.
[22] L. Qi, Y. Tang, W. Dou, and J. Chen, "Combining local optimization and enumeration for QoS-aware web service composition," in *Proc. the International Conference on Web Services*, IEEE Computer Society, 2010, pp. 34–41.
[23] S. Guang, Q. Sun, and F. Yang, "Web service dynamic selection by the decomposition of global QoS constraints," *Journal of Software*, vol. 22, no. 7, pp. 1426–1439, 2011. doi: 10.3724/SP.J.1001.2011.03842
[24] F. Mardukhi, N. N. Bakhsh, K. Zamanifar, and A. Barati, "QoS decomposition for service composition using genetic algorithm," *Applied Soft Computing*, vol. 13, issue 7, pp. 3409–3421, July 2013. https://doi.org/10.1016/j.asoc.2012.12.033
[25] H. Ye, T. Li, and C. Jing, "Decomposition of global constraints for QoS-aware web service composition," *International Journal of Innovative Computing, Information and Control*, vol. 12, no. 6, pp. 2053–2066, 2016.
[26] P. Rodriguez-Mier *et al*., "Hybrid optimization algorithm for large scale QoS-aware service composition," *IEEE Transactions on Services Computing*, vol. 10, issue 4, pp. 547–559, 2017.
[27] S. Deng, H. Wu, D. Hu, and J. L. Zhao, "Service selection for composition with QoS correlations," *IEEE Transactions on Services Computing*, vol. 9, no. 2, pp. 291–303, 2016. doi: 10.1109/TSC.2014.2361138
[28] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for QoS-based web service composition," in *Proc. the 19th International Conference on World Wide Web*, April 2010, pp. 11–20. https://doi.org/10.1145/1772690.1772693
[29] S. Chattopadhyay and A. Banerjeem, "QoS aware automatic web service composition with multiple objectives," *ACM Transactions on the Web*, vol. 14, issue 3, pp. 1–38, 2020.
[30] J. Cardoso, J. Miller, A. Sheth, and J. Arnold, "Quality of service for workflows and web service processes," *Journal of Web Semantics*, vol. 1, no. 3, pp. 281–308, 2004.
[31] K. P. Yoon and C. L. Hwang, *Multiple Attribute Decision Making: An Introduction (Quantitative Applications in the Social Sciences)*, Sage Publications, 1995.