Multi-Criteria Service Selection Agent for Federated Cloud

S. Sudhakar, B. L. Radhakrishnan, P. Karthikeyan, K. Martin Sagayam, and Dac-Nhuong Le

Abstract—**Federated cloud interconnects small and medium-sized cloud service providers for service enhancement to meet demand spikes. The service bartering technique in the federated cloud enables service providers to exchange their services. Selecting an optimal service provider to share services is challenging in the cloud federation. Agent-based and Reciprocal Resource Fairness (RRF) based models are used in the federated cloud for service selection. The agent-based model selects the best service provider using Quality of Service (quality of service). RRF model chooses fair service providers based on service providers' previous service contribution to the federation. However, the models mentioned above fail to address free rider and poor performer problems during the service provider selection process. To solve the above issue, we propose a Multi-criteria Service Selection (MCSS) algorithm for effectively selecting a service provider using quality of service, Performance-Cost Ratio (PCR), and RRF. Comprehensive case studies are conducted to prove the effectiveness of the proposed algorithm. Extensive simulation experiments are conducted to compare the proposed algorithm performance with the existing algorithm. The evaluation results demonstrated that MCSS provides 10% more services selection efficiency than Cloud Resource Bartering System (CRBS) and provides 16% more service selection efficiency than RPF.**

Index terms—**federated cloud, multi-factor service selection, multi-provider service selection, QoS, free rider.**

I. Introduction

Federated cloud has gained huge popularity among the Cloud Service Providers (CSPs) as it facilitates service sharing ability to meet dynamic demand. Small and medium-sized CSPs are unable to handle service requests efficiently due to the varying number of service request. CSPs have two solutions to handle the varying number of service request: (a) Increase the amount of available resources, (b) Reject the resource request. The solution (a) may lead to underutilization of resources, and the solution (b) affects the customer base of the service provider. Besides, both solutions (a) and (b) may affect the trustworthiness and reputation of the CSPs in the market.

Manuscript received October 12, 2021; revised January 18, 2022. Date of publication May 31, 2022. Date of current version May 31, 2022.

S. Sudhakar is with the SRM Institute of Science and Technology, Kattankulathur, Tamilnadu, India (e-mail: sudhakar.sree@gmail.com).

B. L. Radhakrishnan is with the Karunya Institute of Technology and Sciences, Coimbatore, India (e-mail: blradhakrishnan@gmail.com).

P. Karthikeyan is with the National Chung Cheng University,Taiwan- (e-mail: nrmkarthi@gmail.com).

K. Martin Sagayam is with the Karunya Institute of Technology and Sciences, Coimbatore, India (e-mail: martinsagayam.k@gmail.com).

Dac-Nhuong Le is with the Faculty of Information Technology, Haiphong University, Haiphong, Vietnam (e-mail: nhuongld@dhhp.edu.vn) (Corresponding author).

Digital Object Identifier (DOI): 10.24138/jcomss-2021-0148

Cloud federation interconnects service providers and aggregates unused services of the federation to meet dynamic demand. The small and medium-sized service providers join the federation to meet dynamic service demand. Cloud federation provides a monetary based solution to the problem mentioned above. However, monetary based service exchange is frightening for the CSPs with limited capital investment [1-3].

Bartering is a technique used to exchange services for other services without monetary benefits. Service bartering is an alternative to the monetary based service exchange that significantly increases CSPs' service capacity and utilization [4, 5]. Direct and indirect are the two service bartering trading methods. Both direct trade (peer to peer) and indirect trade (agent-based) are used in the federated cloud. Direct trade is provider to provider, and no third party is involved. Indirect trade is among multiple providers with the help of a third-party agent. Avoiding poor performers and free riders are significant challenges in the development of the service bartering-based system. QoS value determines service provider performance in service selection process. A poor performer is a CSP who has low QoS value. A free rider is a user who avails services but does not share services with others [6], [7].

Whitewasher is a CSP who utilizes the services of one service provider entirely but does not share any services. Majorly, reputation-based system and ring-based incentive mechanism were used to handle the white washer problem [8-10]. In reputation-based system and ring-based incentive mechanism, rating collection, information aggregation, source validation, recording performance, and calculating the reputation value of service providers create computational overhead. Resource contribution-based service selection method solves the free rider problem, but it selects poor performers. Additionally, many CSPs offer similar services that create challenges while selecting an optimal CSP [11]. For example, Amazon, IBM, Microsoft, and Google offer similar services like compute, storage, database, migration, mobile services, and analytics. Majority of the method fail to address problem of computational overhead, poor performer selection, whitewashing, and optimal service selection.

The proposed MCSS system uses three parameters, such as QoS, PCR, and RRF, to select the best CSP for service sharing. QoS value is calculated from CSPs' service history. PCR is the ratio between QoS of service provided and quoted service cost. RRF is the ratio between resources consumed and resources shared by the service provider. MCSS presents CSP selection algorithm that comprises of factors such as QoS, PCR, and RRF, which solves the problem of free riders, poor performer selection, whitewashing, and optimal service selection. MCSS selects the best CSP by neglecting poor performers and free riders.

The significant contributions of the paper are outlined as follows:

1. A novel multi-criteria service selection agent has been proposed for federated cloud.
2. Multi-factor Service Selection (MFSS) algorithm and Multi-provider Service Selection (MPSS) algorithm have been implemented to select a service provider effectively
3. Comprehensive case studies are conducted to prove the effectiveness of the proposed algorithm.
4. The extensive simulation experiments are conducted to compare the proposed algorithm performance with existing algorithm.

The rest of paper is organized as follows; section II discusses the related work of multi-criteria service selection agent for federated cloud. The proposed system and algorithm is described in section III. The proposed algorithm performance is evaluated using simple case study in section IV. Section V deals with performance evaluation, and section VI consists of the conclusion and future work.

II. Related Works

Many service selection models have been proposed to solve free riders, whitewashers, poor performers, and optimal service selection problems in the federated cloud. Agent-based service selection, QoS-based service selection, and RRF-based service selection in peer-to-peer environment are discussed in subsections, respectively.

A. Agent-based Service Selection

An automated resource bartering system proposed by ZarAfshan Goher *et al.* [12] is a multi-agent e-bartering system that uses utility value to select services from available CSPs. The automated resource bartering system makes price adjustment based on utility values and prolongs free riders to consume services until they clear their previous debts. When a new provider joins federation, agent assigns zero to QoS parameter that causes starvation in the service provider selection process. The proposed model is not handling free riders.

Agent-based e-barter system developed by Demirkol *et al.* [13] is a multi-agent e-bartering system that utilizes ontology-based comparison for bid matching. The agent-based e-barter system performs service selection using consumers' maximum buying price, providers' maximum selling price, and their tolerance value (for negotiation). Also, the agent-based e-barter system makes a price adjustment. However, systems mentioned above fail to address free-riding, resource contribution, and whitewashing problems.

Zhao *et al.* reported the service agent for simulation in cloud manufacturing. Cloud manufacturing application stage, administration agent can assist endeavors with discovering accomplices quicker with all the more precisely. Be that as it may, in the cloud producing reenactment stage, the administration agent can mimic the conduct of service, the plan of action, the administration procedure [14]. Al-Sayed *et al.* implemented the intelligent cloud service framework by considering functional and nonfunctional features of the cloud service providers to share the resource. Intelligent system as it applies most of the fundamental AI techniques, such as knowledge representation, knowledge inference, knowledge discovery, and NLP. The developed framework depends on a comprehensive cloud service ontology that has been constructed to provide a standardized semantic specification of services [15].

B. QoS-based Service Selection

Ranking CSPs based on QoS attributes mentioned in the CSMIC Service Measurement Index (SMI) version 2.1 is a complex task as it has many KPIs, attributes, and sub-attributes. Zeleny [16] defined this problem as Multi-Criteria Decision Making (MCDM) problem. Moreover, Grguevic [17] has found that choice of MCDM techniques depends on the kind of service offered by CSPs. Context-aware cloud service selection approach implemented by Lie *et al.* [18] is a subjective and objective assessment system based on cloud providers' QoS value. Service selection algorithm in multi-cloud environment proposed by Farokhi [19] is a Service Level Agreement (SLA) based service allocation method, which is suitable for SaaS. Service selection algorithm in a multi-cloud environment proposed by Farokhi *et al.* [20] is a prospect theory based service selection method that compares SLA while selecting a service provider. Integrated multiple criteria decision-making method implemented by Kumar *et al.* [21] is an Analytical Hierarchical Process (AHP) and TOPSIS based service selection method. SELCLOUD proposed by Jatoth *et al.* [22] is a hybrid service provider selection method that uses AHP, TOPSIS, and gray theory. A Neutrosophic Multi-Criteria Decision Analysis (NMCDA) approach proposed by Abdel-basset *et al.* [23] is a framework for evaluating CSPs' QoS value. The above-mentioned methods consider only QoS value for service selection that solves poor performer problem but fails to solve free rider, whitewasher, and optimal service selection.

C. RRF-based Service Selection in Peer-to-Peer Environment

RRF-based service selection systems select service providers by considering past service contribution [24], [25]. Zhang *et al.* [8] proposed a novel Cluster-Based Incentive Mechanism (CBIM) that provides incentives to the contributing service provider that solves free riding and whitewashing problems. However, the CBIM is suitable for the CSPs who consume a type of resource and shares another type of resource in the federation simultaneously. The CBIM is not suitable for the CSPs who needs to consume resources when there is a demand, contributes resources in the near future and inversely. Falcao *et al.* [10], [24] proposed a Decentralized Fairness Driven Network of Favors (FD-NoF) model that focuses service contribution fairness based service selection in peer to peer service sharing. FD-NoF has high computational overhead because peers calculate and update fairness value, debit details for every transaction.

Zhang *et al.* [26] implemented a lightweight reputation-based incentive system that gives solution to free riding. Lightweight reputation-based incentive system computes simple reputation value and goodness factor based on local transaction data. Then, the ultimate reputation value is calculated from the simple reputation value and goodness factor. Peers who have low ultimate reputation value in the suspect table are considered as cooperative peers, and others are considered as rational peers or malicious free riders. Lightweight reputation-based incentive system eliminates free-riding problems; however, the final decision is individual peer's decision and not a collective one. RRF based resource selection approach is a multitenant cooperative system which is suitable for cooperative users but not suitable for federated cloud users The methods mentioned above are not automated bartering systems. Rosa et al. present the computational resource and cost prediction service for scientific workflows in federated clouds (CRCPs). CRCPs allows users to pick between high-performance, low-budget executions or to automatically and transparently define how much to pay and how long to complete a procedure [27]. Chauhan et al developed the broker-based resource allocation model for federated. Kumar et al discuss the performance-based risk driven trust secured service sharing in peer-to-peer federated cloud. This work mainly focusses on the secure service selection and it is not considering the free riding issues in the federated cloud [28], [29].

In federated cloud, service selection deals with optimal service selection, free riding, whitewashing, and poor performer problems. The existing systems adopted two different approaches for service selection: a) QoS-based selection, and b) RRF-based selection. However, the approach (a) has free riding and whitewashing problems, and (b) has poor performer problem. The proposed MCSS algorithms select optimal CSPs by considering the QoS, PCR, and RRF. Besides, MFSS and MPSS algorithms solve free riding, whitewashing, and poor performer problems. Various models, discussed above consider the factors such as QoS, service contribution, and fairness individually while selecting services. The models do not consider a combination of these factors while selecting services. Additionally, none of the models discussed service distributions equitably if more than one service provider meets the service requirement. Our MCSS algorithms consider a combination of the factors while selecting the service provider and provides fair service selection efficiency than traditional methods.

III. Multi-Criteria Service Selection Agent for Federated Cloud

Cloud Federation indicates to the unionization of programming, framework and stage administrations from divergent organizations that can be gotten to by a customer through the web. It is vital to take note of that united distributed computing administrations actually depend on the presence of actual server farms. The union of cloud assets permits customers to upgrade venture IT administration conveyance. The league of cloud assets permits a customer to pick the best cloud administrations supplier, as far as adaptability, cost and accessibility of administrations, to meet a specific business or innovative need inside their association. Organization across various cloud asset pools permits applications to run in the most suitable foundation conditions. The union of cloud assets additionally permits a venture to disperse responsibilities all over the planet, move information between dissimilar organizations and execute creative security models for client admittance to cloud assets.

One shortcoming that exists in the league of cloud assets is the trouble in expediting availability between a customer and a given outer cloud supplier, as they each have their own remarkable organization tending to plot. To determine this issue, cloud suppliers should give customers the authorization to indicate a tending to conspire for every server the cloud supplier has stretched out to the web. This furnishes clients with the capacity to get to cloud administrations without the requirement for reconfiguration when utilizing assets from various specialist organizations. Cloud league can likewise be carried out behind a firewall, furnishing customers with a menu of cloud administrations given by at least one confided in substances.

We developed a multi-criteria service selection agent for federated cloud that performs optimal service selection by using QoS, performance-cost ratio, and reciprocal resource fairness value.

A. Mathematical Preliminaries

The list of mathematical terms used is given below for better understanding of the paper.

TABLE I
THE LIST OF MATHEMATICAL TERMS

Symbol	Description	Symbol	Description
a	cost adjust value	av	available service
c	service cost	CSP	cloud service provider set
CSP_s	selected service provider	d	duration
in	number of instances	n(S)	number of service provider in s
pcr	performance cost ratio	q	qos value
r	required service	rc	service count
rrf	reciprocal resource fairness	s	csp score
S	service request satisfying provider set	st	service type
w_{pcr}	weight of the performance cost ratio	w_q	weight of the qos
w_{rrf}	weight of the reciprocal resource fairness		

B. Service Selection Architecture

Service selection architecture is shown in Fig. 1. consists of three components: CSPs, service selection agent, and service monitoring database. CSPs are service providers who share services to federation or consume services from federation.

Service provider score calculation component calculates service provider's score to rank services. Service provider score is calculated using equation

$$\mathrm{CSP}_i.\mathrm{s} = \mathrm{CSP}_i.\mathrm{q} \times w_q + \mathrm{CSP}_i.\mathrm{pcr} \times \mathrm{wpcr} + \mathrm{CSP}_i.\mathrm{rrf} \times \mathrm{w_{rrf}} \quad (1)$$

where CSP_i is ith service provider where i = 1, 2… n, q – QoS, pcr – performance-cost ratio, rrf – reciprocal resource fairness, W_q, W_{pcr}, and W_{rrf} are relative weight assigned to QoS, PCR, and RRF, respectively.

The agent sort score in descending order and selects a high score service provider using the calculated score. Service aggregation and negotiation component do the service negotiation and service aggregation if the required service is not available with one service provider. Otherwise, the agent eliminates top scored CSP from the list and selects next top scored CSP for service negotiation, and this step iterates till the service negotiation is successful. After the successful negotiation, service control is transferred to service requesting CSP. Service monitoring component monitors selected services' and records the performance in service monitoring database. Service history update component records complete transaction details in service monitoring database.

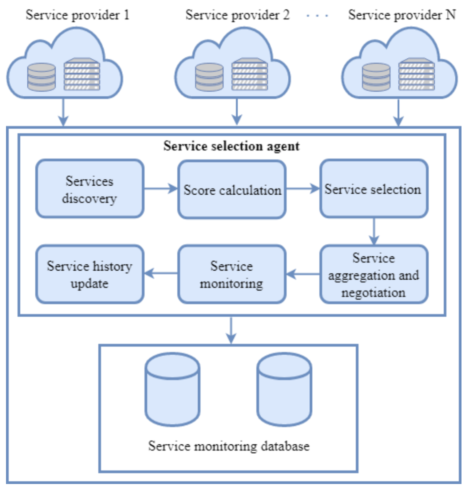

Fig. 1. Service selection architecture

C. Multi-factor Service Selection

Multi-factor service selection (MFSS) makes the service provider selection using input attributes such as service type, QoS, number of instances, duration, service cost, and cost adjustment value. A registered CSP submits a service request to the federation to avail service. Agent executes MFSS algorithm to find best CSP that meet the submitted service requests. MFSS algorithm performs two functions service discovery and service selection.

C.1 Multi-factor Service Discovery

Service Discovery (SD) algorithm takes attributes in the service request as inputs and returns set of services (*S*) that satisfy the service request. Initially, S is an empty set containing no service. SD algorithm takes services from service database in sequential order to check service request is met or not. If service meets the service request, then the selected service is added to S and weighted score of the service is calculated using eq. (1). Weighted sum technique is a widely used multi-factor decision-making technique for evaluating a number of alternatives. Otherwise, service is not added to *S*. This process repeats for all the services exist in the service database. Finally, *S* contains a set of services that meets service request.

```
**Service_Discovery** (CSP, r, in, q, d)

Input:   r: required service
         in: number of instances
         q: QoS, d: duration
         CSP: List of cloud service providers
Output:  S: set of services satisfying r, in, q, and d

Initialization      $S= \phi$; i=1;
begin
1.  foreach ($CSP_i$ in CSP) do
2.    if($CSP_i$.st == r^q<=$CSP_i$.q ^ in<= $CSP_i$.av ^ d<= $CSP_i$.d)
3.        $CSP_i.s=CSP_i.q*w_q+CSP_i.pcr*w_{pcr}+ CSP_i.rrf*w_{rrf}$;
4.        S = S U $CSP_i$
5.      return S;
end
```

C.2 Service Selection

MFSS has two sub-algorithms, namely SD and service selection algorithm. Initially, MFSS calls SD algorithm to find a set of matching services (S) for the given request. Next, MFSS calls the service selection algorithm to find the best service from the set S. Service selection algorithm performs two tests based on number of elements in the set S. When $n(S) = 0$, service selection algorithm returns failure and agent informs the failure to the requesting service provider. When $n(S) > 0$, service selection algorithm selects the service provider that has the highest weighted score in set S. If the cost of the service (CSP_i.c) is less than or equal to cost of service (c) given in the request, then selected service provider is returned. Otherwise, price negotiation process starts. During the negotiation process, cost difference (CSP_i.c – c) is calculated. If cost difference is less than or equal to service provider's cost adjustment value (CSP_i.a) or cost adjustment value given in the service request, then negotiation succeeds, and the selected service provider is returned. Otherwise, the selected service provider is removed from the set S and repeat from step 2 of service selection algorithm.

```
**Service_selection** (CSP, r, in, q, d, c, a)

Input:   CSP: list of cloud service providers;
         r: required service, in: number of instances;
         q: QoS, d: duration, c: service cost;
         a: cost adjustment value
Output:  $CSP_i$: selected service provider or a failure;
begin
    1.   S = SD (CSP, r, in, q, d));
    2.   if ( n(S) == 0)     return failure;
    3.   else
    4.     i=findMaxIndex(CSP[].s);
    5.     if ( $CSP_i$.c <= c)  return $CSP_i$;
    6.     else
    7.         if ($CSP_i$.a >= $CSP_i$.c – c ∨ a >= $CSP_i$.c – c)
    8.           return $CSP_i$;
    9.         else
    10.          S = S – $CSP_i$;
    11.          repeat step2;
end
```

D. Multi-provider Service Selection

Multi-provider service selection (MPSS) is called, when one service provider of the federation does not have sufficient services to meet the submitted service requests. MPSS aggregates services from multiple service provider of the federation to meet the submitted service requests.

D.1 Multi-provider Service Discovery

The submitted service request has multiple attributes such as service type, QoS, number of instances, duration, service cost, and cost adjustment value. Service Discovery1 (SD1) algorithm takes all attributes in the service request as inputs except number of instances required (in) and returns set of services (*S*) that satisfy the service request. Initially, S is an empty set containing no service. SD1 algorithm takes services from service database in sequential order to check service request is met or not. If service meets the service request, then the selected service is added to S and weighted score of the service is calculated using eq. (1). Otherwise, service is not added to *S*. This process repeats for all the services that exist in the service database. Finally, *S* contains a set of services that meet all attributes except the number of instances required.

Service_Discovery_1 (CSP, r, q, d)

```
Input:    r: required service;
          q: QoS, d: duration;
Output:  S: List of services satisfying r, q, and d

Initialization        $S= \phi$; i=1;
begin
  1.    foreach ($CSP_i$ in CSP) do
  2.      if($CSP_i.st == r \wedge q<=CSP_i.q \wedge d<= CSP_i.d$)
  3.        $CSP_i.s=CSP^i.q*w_q+ CSP_i.pcr*w_{pcr}+ CSP_i.rrf*w_{rrf}$;
  4.        $S = S \cup CSP_i$
  5.    return S;
end
```

D.2 Service Selection

Service_selection_1 algorithm executes three functions orderly to select a set of service providers for the given service request that is shown in Fig. 2. Firstly, service selection1 algorithm executes service discovery by calling SD1 algorithm to find matching services (S) for the given request.

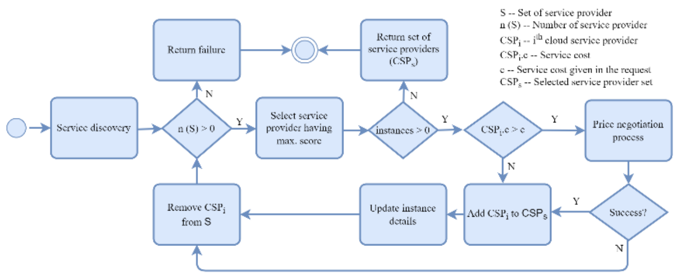
S -- Set of service provider
n (S) -- Number of service provider
CSP$_i$ -- ith cloud service provider
CSP$_i$.c -- Service cost
c -- Service cost given in the request
CSP$_s$ -- Selected service provider set
Return failure
Return set of service providers (CSP$_s$)
Service discovery
n (S) > 0
Select service provider having max. score
instances > 0
CSP$_i$.c > c
Price negotiation process
Success?
Add CSP$_i$ to CSP$_s$
Update instance details
Remove CSP$_i$ from S
Y
N

Fig. 2. State diagram for service selection algorithm 1

Secondly, Service_selection_1 algorithm examines the number of elements in the set S. When *n(S)* = 0, service selection1 algorithm returns failure and agent informs the failure to the requesting service provider. When n(S) > 0, service selection algorithm1 selects the service provider that has the highest weighted score in set S.

Thirdly, service selection1 algorithm checks the number of instances required (in) from the service request. If instances required is 0, then service selection1 algorithm returns a set of service providers (CSPs). Otherwise, service selection1 algorithm verifies the service cost with service cost given in the request. Next, service selection1 algorithm goes for the price negotiation process when the service cost is higher than the service cost given in the request. Finally, MPSS examines the criteria (a) and (b). (a) If the service cost is less than or equal to service cost given in the request or price negotiation process succeeds then CSP_i is added to CSP_s. Subsequently, MPSS updates the instance count details in the service database, and CSP_i is removed from the set *S*. (b) If the price negotiation process fails in the third step, CSP_i is removed from the set *S*. This process continues till the service selection algorithm returns either set of service providers for the given request or failure.

Service_selection_1 (CSP, r, in, q, d, c, a)

```
Input:     r: required service;
           in: number of instances;
           q: QoS, d: duration;
           c: resource cost, a: cost adjustment value;
           CSP: list of cloud service providers;
Output:  $CSP_s$: selected service provider set or a failure;
begin
  1.    $CSP_s = \phi$;
  2.    S:= Discovery (CSP, r, in, q, d))
  3.    if ( n(S) == 0)
  4.      return failure;
  5.    else
  6.      i=findMaxIndex(CSP[].s);
  7.      if (in >0)
  8.        if ($CSP_i.c > c$)
  9.          if ($CSP_i.a >= CSP_i.c - c \vee a >= CSP_i.c - c$)
  10.           $CSP_s = CSP_s \cup CSP_i$;
  11.           if (in >= $CSP_i$.a)
  12.             in = in – $CSP_i$.a;
  13.             $CSP_i.rc = CSP_i.a$;
  14.             $CSP_i$.a=0;
  15.             $S = S - CSP_i$;
  16.           else
  17.             $CSP_i.a = CSP_i.a - in$;
  18.             $CSP_i$.rc = in;
  19.             in = 0;
          else
  20.         S:= S – $CSP_i$;
  21.         repeat step3;
  22.      else
  23.        $CSP_s = CSP_s \cup CSP_i$;
  24.        if (in >= $CSP_i$.a)
  25.          in = in – $CSP_i$.a;
  26.          $CSP_i.rc = CSP_i.a$;
  27.          $CSP_i$.a=0;
  28.          $S = S - CSP_i$;
  29.        else
  30.          $CSP_i.a = CSP_i.a - in$;
  31.          $CSP_i$.rc = in;
  32.          in = 0;
  33.   else
  34.   return $CSP_s$;
end
```

IV. CASE STUDY: CSP SELECTION USING MCSS

Consider CSP1, CSP2 … and CSP10 have advertised their service (VM) sharing details with the federation, which is given in Table II. Agent at federation maintains CSPs' service sharing details in the service database. Agent calculates service providers' QoS, PCR, and RRF values using the weighted sum method [30]. The calculated CSPs' QoS, PCR, and RRF values

TABLE II
CSPs SERVICE SHARING DETAILS

CSP / Service details	*Instance details*												*Cost adj. (in %)*
	Small	*Instance cost /hour (in $)*	*Duration (in days)*	*Medium*	*Instance cost/hour (in $)*	*Duration (in days)*	*Large*	*Instance cost / hour (in $)*	*Duration (in days)*	*X-large*	*Instance cost / hour (in $)*	*Duration (in days)*	
CSP1	200	0.191	60	80	0.367	30	22	0.739	45	2	1.498	90	10
CSP2	415	0.192	45	60	0.369	45	10	0.743	90	4	1.506	30	5
CSP3	50	0.191	30	78	0.367	60	24	0.739	60	5	1.498	60	12
CSP4	130	0.193	30	30	0.371	45	16	0.747	60	3	1.514	30	10
CSP5	260	0.192	60	20	0.369	30	12	0.743	45	6	1.506	45	18
CSP6	75	0.191	75	42	0.367	45	18	0.739	30	5	1.498	90	10
CSP7	20	0.191	60	60	0.367	30	6	0.739	60	0	1.498	45	12
CSP8	360	0.193	90	85	0.371	30	15	0.747	60	7	1.514	75	12
CSP9	30	0.194	30	55	0.373	45	0	0.751	90	3	1.522	45	10
CSP10	80	0.188	30	100	0.361	60	4	0.727	45	8	1.474	60	12

TABLE III
CSPS' QOS, PCR, AND RRF DETAILS

CSP / Year	*CY*			*CYM1*			*CYM2*			*CYM3*			*Weighted Sum*		
	QoS	*PCR*	*RRF*	*QoS*	*PCR*	*RRF*	*QoS*	*PCR*	*RRF*	*QoS*	*PCR*	*RRF*	*QoS*	*PCR*	*RRF*
CSP1	89.95	470.95	121.67	87.35	457.33	108.89	71.91	376.49	88.89	87.33	457.23	95.56	85.30	446.60	108.67
CSP2	67.60	352.08	101.25	69.63	362.65	108.57	69.73	363.18	91.43	68.60	357.29	91.43	68.73	357.99	100.50
CSP3	87.72	459.27	115.38	86.63	453.56	109.30	89.47	468.43	90.70	89.99	471.15	104.88	87.97	460.58	107.57
CSP4	96.90	502.07	80.00	90.34	468.08	85.71	99.95	517.88	100.00	94.37	488.96	93.62	95.29	493.73	87.08
CSP5	66.86	348.25	90.00	68.46	356.56	110.00	64.44	335.63	105.00	65.91	343.28	95.00	66.76	347.72	99.50
CSP6	82.24	430.59	115.00	84.41	441.94	90.00	80.31	420.47	106.45	81.11	424.66	90.00	82.39	431.38	103.29
CSP7	72.42	379.18	116.67	74.77	391.47	106.25	73.76	386.18	106.12	73.45	384.55	96.08	73.50	384.80	109.37
CSP8	77.44	401.24	89.29	79.98	414.40	93.75	79.35	411.14	105.41	80.45	416.84	121.21	78.88	408.73	97.04
CSP9	92.83	478.50	75.00	88.63	456.86	82.05	90.55	466.75	107.14	91.91	473.76	110.34	91.02	469.19	87.08
CSP10	86.12	458.08	80.00	88.55	471.01	120.00	89.19	474.41	110.00	88.99	473.35	142.86	87.75	466.75	104.29

TABLE IV
VIRTUAL MACHINE CONFIGURATION DETAILS

Specification	*Virtual machine configuration*				
	Micro	*Small*	*Medium*	*Large*	*X-Large*
RAM	1 - 4 GigB	4 - 8 GiB	8 - 16 GiB	16 - 32 GiB	32 - 64 GiB
Storage Space	20 – 60 GB	80 – 240 GB	320 – 800 GB	1 – 1.5 TB	2 TB
VCPUs	1	1 - 2	1 - 4	2 - 8	4 - 16

are given in Table III. VM standard instance configuration details are given in Table IV.

Let's assume a CSP in federation submits a service request as <X- large, 4, 80, 45 days, $1.35, 5>. Firstly, the agent executes SD algorithm and selects CSP3, CSP6, and CSP10 that meets the given service request and rejects CSP1, CSP2, CSP4, CSP5, CSP7, CSP8, and CSP9 that do not meet the given service request. Reason for rejecting service providers CSP1, CSP2, CSP4, CSP5, CSP7, CSP8, and CSP9 is given in Table V.

TABLE V
SERVICE PROVIDER REJECTION SET

Service Provider	*Reason for rejection*
CSP1	Insufficient instances
CSP2	Duration did not match
CSP4	Insufficient instances
CSP5	QoS value did not match
CSP7	Insufficient instances
CSP8	QoS value did not match
CSP9	Insufficient instances

The selected service provider set (S) details are shown in Table VI. SD algorithm uses a relative weight-based score calculation method to rank service providers in the set S that is given in Eq. (1). Let us consider relative weight assigned for QoS, PCR, and RRF are 0.4, 0.3, and 0.3, respectively.

TABLE VI
SERVICE PROVIDER SELECTION SET (S)

CSP / Service details	*X-Large*	*Instance cost/hour (in $)*	*Duration (in days)*	*Cost adj. (in %)*	*QoS*	*PCR*	*RRF*
CSP3	5	1.498	60	12	87.97	460.58	107.57
CSP6	5	1.498	90	10	82.39	431.38	103.29
CSP10	8	1.474	60	12	87.75	466.75	104.29

Service providers in set S are arranged as per the calculated score that is shown in Table VII.

TABLE VII
SERVICE PROVIDERS' SCORE

CSP / Service details	*QoS*	*PCR*	*RRF*	*Score*
CSP10	87.75	466.75	104.29	206.41
CSP3	87.97	460.58	107.57	205.63
CSP6	82.39	431.38	103.29	193.36

V. PERFORMANCE EVALUATIONS

The simulation environment was made to evaluate the efficacy of the MCSS algorithm. The results of the simulation of MCSS are compared against two popular techniques: (a) Cloud Resource Bartering System (CRBS) and (b) RRF.

A. Evaluation Setup

Federated cloud environment was created using CloudSim4.0. We have used Simulated Cloud Service QoS Dataset, the dataset used for this simulation experiments were based on SMI attributes [31-35]. The data values were synthesized based on quantitative attributes such as response time, cost, and availability. The datasets were prepared according to the dataset used in the existing federated cloud literature. On every dataset, iterative evaluations were conducted to find the performance of MCSS.

The entire dataset was divided into two sets. The first set contains service advertisement details, and the second set contains service request details. Datasets were further categorized into micro, small, medium, and large classes. Micro datasets contain 25 service providers whereas small, medium and large contain 50, 100 and 150 respectively. Dataset categorization aimed to produce test cases that calculate MCSS performance under different conditions. First, test cases were executed on MCSS, and the outcome was logged.

B. Comparative Methods

The proposed method performance analyzed with the existing methods, such Cloud Resource Bartering System (CRBS) and Reciprocal Resource Fairness (RRF) and each work is discussed in detail as follows:

Cloud Resource Bartering System (CRBS): It dynamically extends the capacity of cloud providers. It encourages cloud providers to work together to meet spikes in resource demand. It increases overall service availability without requiring additional infrastructure.

Reciprocal Resource Fairness (RRF): RRF is a revolutionary resource allocation technique that allows numerous tenants in new-generation cloud systems to share several types of resources fairly. Inter-tenant resource trading and intra-tenant weight adjustment are two complimentary and hierarchical strategies for resource sharing that RRF provides.

C. Handling Free Riders

Free-riders can take a number of steps to get around the barriers that are in the way of their activities while also maximizing resource consumption in the ecosystem. Identifying and studying these behaviors can aid in the development of anti-free-rider systems. The existence of an uncooperative service provider who does not follow federation guidelines and increase his revenue affects trust in the federation. Free riders are selfish service providers focusing only on revenue generation, as discussed in section 1. Service provider selection in CRBS is based on their QoS value, but frequently free riders get selected in this method.

MCSS agent maintains service contribution and service consumption history of all the registered service providers. Additionally, MCSS employs QoS, PCR, and RRF as service selection factors for selecting a service provider. A service provider who has low RRF value gets an overall low score in service provider score calculation. Service providers with low score are rarely get selected during service provider selection. RRF is one of the selection criteria to avoid free riders. In order to get selected for service sharing at the federation, the corresponding service provider has to improve RRF (service contribution and service consumption) value.

In the dataset, a random number of free riders were included to verify the MCSS performance against the free riders. Two distinct sets of experiments were conducted to test the efficiency of MCSS. In the first experiment, the number of service request was fixed as 100, whereas the number of service providers was varied as 25, 50, 75, and 100. Among 100 service requests, 80 requests are made by genuine service providers, and 20 requests are made by free riders. In the second experiment, the number of service request was changed as 25, 50, 100, and 150, whereas the service providers made fixed as 50.

The evaluation results proved that CRBS permits a considerable number of free riders to avail services from the federation (shown in Fig. 3, and Fig. 5). MCSS rarely permits free riders to consume services from federation (shown in Fig. 4 and Fig. 6). Preventing free riders created a significant impact on the number of successful transactions; however, preventing free riders made enough service availability for genuine service providers.

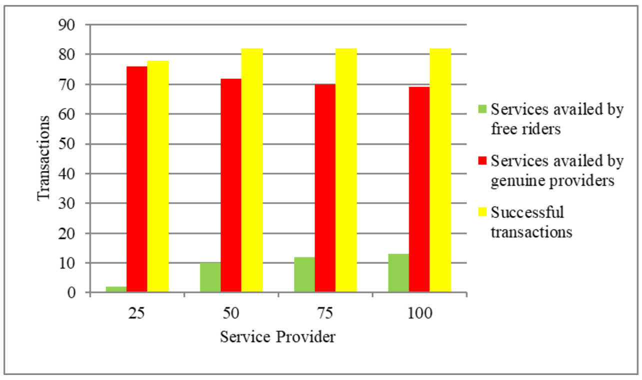

Fig. 3. Free riders' prevention in CRBS (No. of service request fixed as 100)

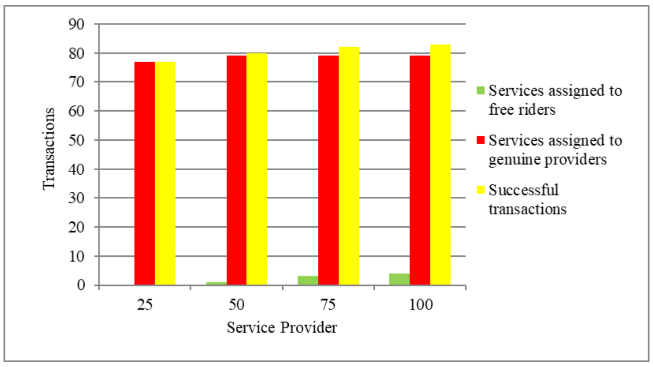

Fig. 4. Free riders' prevention in MCSS (No. of service request fixed as 100)

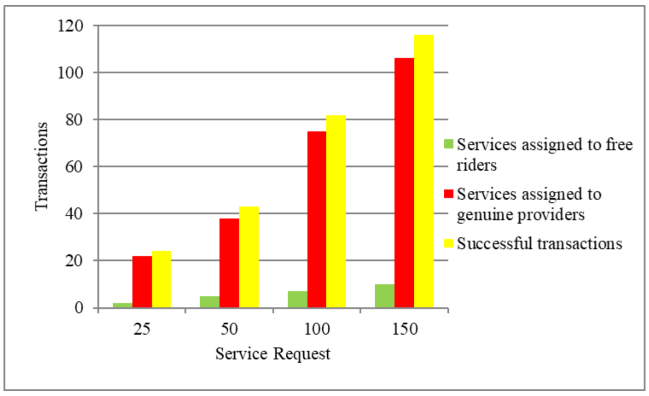

Fig. 5. Free riders' prevention in CRBS (No. of service provider fixed as 50)

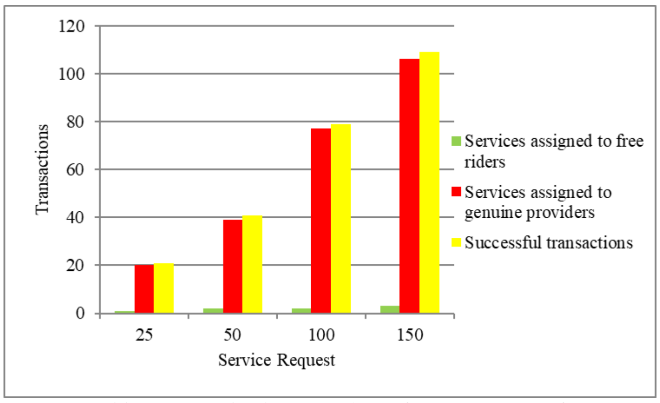

Fig. 6. Free riders' prevention in MCSS (No. of service provider fixed as 50)

D. Handling Poor Performing Service Providers

The presence of poor-performing service providers in federation affects its reputation. In RRF, a service provider can acquire services from federation based on their service contribution. Selecting service providers based on RRF value selects a poorly performing service provider. In MCSS, Service provider score calculation takes three factors such as QoS, PCR, and RRF while calculating service provider weighted score. Service provider having low QoS value gets a low score in the score calculation. Score based service selection provides more chances to the high scored service provider at the same time provides less chance to the low scored service providers. Service providers should improve their QoS, PCR, and RRF value to get an increased score. Selection of service provider based on multiple factors eliminates poor performer selections.

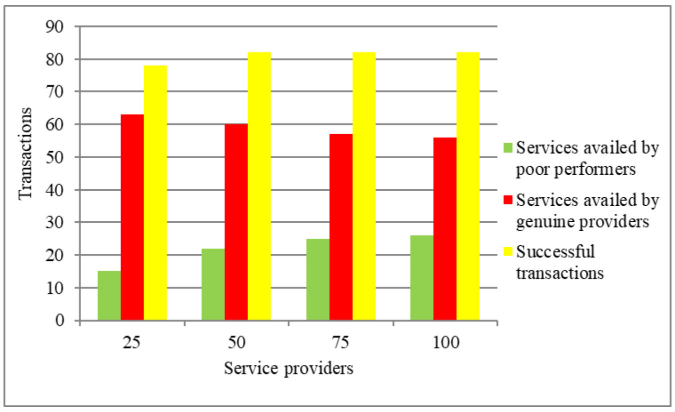

Fig. 7. Poor performers negotiation in RRF (Number of service request fixed as 100)

In the dataset, a random number of poor performers were included to verify the MCSS performance against the poor performer selection problem. Two distinct sets of experiments were conducted to test the efficiency of MCSS discussed in Section 5.2. Test cases were executed on CRBS, and results were recorded. The same set of test cases was executed on MCSS, and results were recorded for comparative studies. The evaluation results showed that MCSS rarely selects poor performing service provider that is shown in Fig. 7, and Fig. 9. Whereas RRF based service selection selects a considerable number of poor performers in service selection that is shown in Fig. 8 and Fig 10.

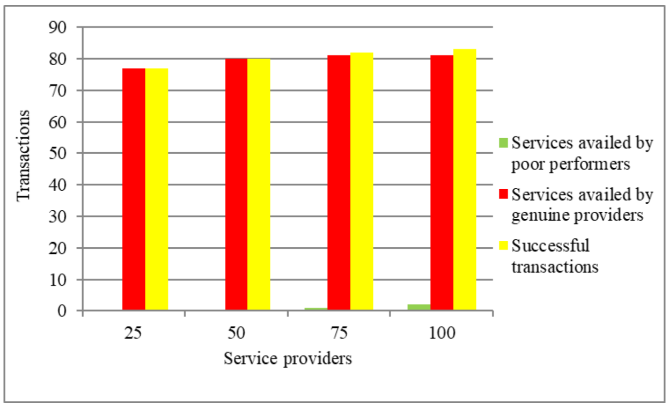

Fig. 8. Poor performers negotiation in MCSS (No. of service request fixed as 100)

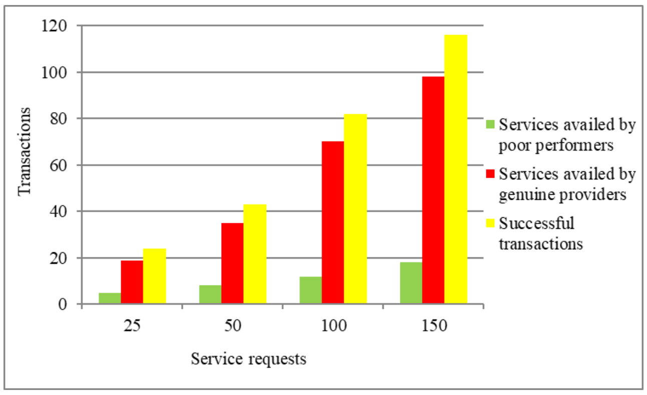

Fig. 9. Poor performers negotiation in RRF (No. of service provider fixed as 50)

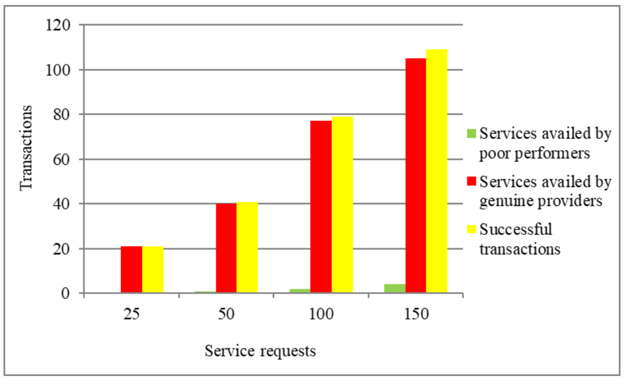

Fig. 10. Poor performers negotiation in MCSS (No. of service provider fixed as 50)

E. Service Selection Ability

Service selection ability of agent-based systems is determined by the number of genuine providers' transactions. Many test cases were prepared to determine the ability of MCSS under different conditions. In the first experiment, the number of service providers in the federation was fixed as 25, and service requests were given as 25, 50, 100, and 150. In the second experiment, the number of service request was fixed as 25, and service providers were set as 25, 50, 75, and 100.

The performance of MCSS for a fixed number of service providers against the variable number of service requests was recorded. Similarly, the performance of MCSS for a fixed number of service requests against the variable number of service providers was recorded. The same set experiments were conducted on CRBS and RRF based systems, and the results were recorded. The experiment results demonstrated that service selection ability of MCSS outperformed CRBS and RRF based systems. Also, the service selection efficiency of MCSS was not decreasing when increasing the number of service requests. Whereas, service selection efficiency of CRBS and RRF based systems were decreasing when increasing number of service requests. The comparison of MCSS, CRBS, and RRF based systems is given in Fig. 11.

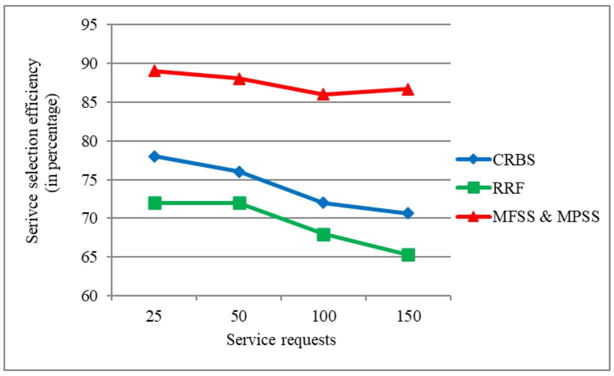

Fig. 11. Comparison of service selection efficiency

F. Average Service Cost

Request making service provider gets satisfied when selected service cost is lesser. An effective service matching system matches services at a lesser cost. Average service cost is taken as an evaluation parameter to evaluate low-cost service matching ability of the MCSS system. MCSS' successful transactions average cost was calculated for 25, 50, 100, and 150 service requests. Besides, CRBS' and RRF' successful transactions average cost was calculated for the same number of service requests. Average service cost comparison proved that the average cost of MCSS' transactions were less than the average cost of CRBS' and RRF' transactions. Also, CRBS' and RRF' average cost were fluctuating whereas MCSS' cost was steady, that is shown in Fig. 12.

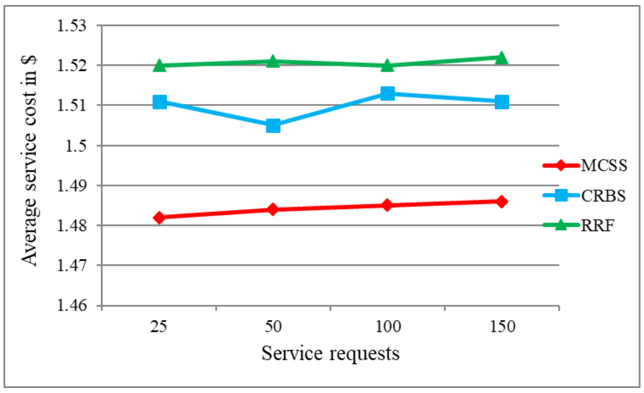

Fig. 12. Comparison of selected service average cost

VI. Conclusion and Future Direction

Multi-criteria Service Selection (MCSS) has been used to select a service provider effectively in the federated cloud environment. MCSS system uses QoS, Performance-Cost Ratio

(PCR), and RRF values for selecting service providers. Reciprocal resource fairness ensures that service providers actively participate in federation. MCSS handles non-cooperative providers such as free rider, poor performers, and white washer very effectively by selecting service providers with good QoS, PCR, and RRF value than the traditional algorithm. Service selection efficiency of MCSS is high compared to the existing CRBS and RRF based systems. MCSS selects low-cost services, whereas CRBS and RRF based systems select high-cost services.

MCSS algorithms suggested deal with service requests that only include one service type at a time. In the future, these algorithms may be expanded to handle service requests that combine numerous service types into a single request. Another research might be conducted with the goal of dynamically balancing service charges based on the services provided in the federation. Dynamically balancing service charges may incentivize service providers to supply additional services while also preventing free riding issues. Data analytics services may be used to forecast service provider behavior and weed out non-cooperative service providers. Data analytics may also be used to forecast service demand increases.

References

[1] K. Woongsup and J. Mvulla, “Reducing Resource Over-Provision Using Workload Sharing for Energy Cloud Computing,” *Appl. Math. Inf. Sci.*, vol. 7, no. 5, pp. 2097–2104, 2013.

[2] S. S. Chauhan, E. S. Pilli, R. C. Joshi, G. Singh, and M. C. Govil, “Brokering in interconnected cloud computing environments: A survey,” *J. Parallel Distrib. Comput.*, vol. 133, pp. 193–209, 2019.

[3] C. Labba, N. Bellamine Ben Saoud, and J. Dugdale, “A predictive approach for the efficient distribution of agent-based systems on a hybrid-cloud,” *Futur. Gener. Comput. Syst.*, vol. 86, pp. 750–764, 2018.

[4] O. C. D. Anejionu *et al.*, “Spatial urban data system: A cloud-enabled big data infrastructure for social and economic urban analytics,” *Futur. Gener. Comput. Syst.*, vol. 98, pp. 456–473, 2019.

[5] C. T. Do, N. H. Tran, E. N. Huh, C. S. Hong, D. Niyato, and Z. Han, “Dynamics of service selection and provider pricing game in heterogeneous cloud market,” *J. Netw. Comput. Appl.*, vol. 69, pp. 152–165, 2015.

[6] M. Liaqat *et al.*, “Federated cloud resource management: Review and discussion,” *J. Netw. Comput. Appl.*, vol. 77, pp. 87–105, 2017.

[7] A. Levin, D. Lorenz, G. Merlino, A. Panarello, A. Puliafito, and G. Tricomi, “Hierarchical load balancing as a service for federated cloud networks,” *Comput. Commun.*, vol. 129, pp. 125–137, 2018.

[8] K. Zhang and N. Antonopoulos, “A novel bartering exchange ring based incentive mechanism for peer-to-peer systems,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 1, pp. 361–369, 2013.

[9] A. Comi, L. Fotia, F. Messina, G. Pappalardo, D. Rosaci, and G. M. L. Sarné, “A reputation-based approach to improve QoS in cloud service composition,” *Proc. - 2015 IEEE 24th Int. Conf. Enabling Technol. Infrastructures Collab. Enterp. WETICE 2015*, pp. 108–113, 2015.

[10] E. de L. Falcão, F. Brasileiro, A. Brito, and J. L. Vivas, “Enhancing fairness in P2P cloud federations,” *Comput. Electr. Eng.*, vol. 56, pp. 884–897, Nov. 2016.

[11] Q. Duan, “Cloud service performance evaluation: status, challenges, and opportunities – a survey from the system modeling perspective,” *Digit. Commun. Networks*, vol. 3, no. 2, pp. 101–111, 2017.

[12] S. ZarAfshan Goher, P. Bloodsworth, R. Ur Rasool, and R. McClatchey, “Cloud provider capacity augmentation through automated resource bartering,” *Futur. Gener. Comput. Syst.*, vol. 81, pp. 203–218, 2018.

[13] S. Demirkol, S. Getir, M. Challenger, and G. Kardas, “Development of an agent based e-barter system,” in *2011 International Symposium on* Innovations *in Intelligent Systems and Applications*, 2011, pp. 193–198.

[14] C. Zhao, X. Luo, and L. Zhang, “Modeling of service agents for simulation in cloud manufacturing,” *Robot. Comput. Integr. Manuf.*, vol. 64, no. November 2019, p. 101910, 2020.

[15] M. M. Al-Sayed, H. A. Hassan, and F. A. Omara, “An intelligent cloud service discovery framework,” *Futur. Gener. Comput. Syst.*, vol. 106, pp. 438–466, 2020.

[16] M. ZELENY, “Multiple Criteria Decision Making (MCDM): From Paradigm Lost to Paradigm Regained,” vol. 18, no. 2, pp. 77–89, 2011.

[17] I. Grgurević, “Multi-criteria Decision-making in Cloud Service Selection and Adoption,” pp. 8–12, 2017.

[18] L. Qu, Y. Wang, M. A. Orgun, L. Liu, H. Liu, and A. Bouguettaya, “CCCloud: Context-aware and credible cloud service selection based on subjective assessment and objective assessment,” *IEEE Trans. Serv. Comput.*, vol. 8, no. 3, pp. 369–383, 2015.

[19] S. Farokhi, “Towards an SLA-based service allocation in multi-cloud environments,” *Proc. - 14th IEEE/ACM Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2014*, pp. 591–594, 2014.

[20] S. Farokhi, F. Jrad, I. Brandic, and A. Streit, “HS4MC: Hierarchical SLA-Based Service Selection for Multi-Cloud Environments,” *Proc. CLOSER’14*, pp. 722–734, 2014.

[21] R. R. Kumar, S. Mishra, and C. Kumar, “Prioritizing the solution of cloud service selection using integrated MCDM methods under Fuzzy environment,” *J. Supercomput.*, vol. 73, no. 11, pp. 4652–4682, 2017.

[22] C. Jatoth, G. R. Gangadharan, U. Fiore, and R. Buyya, “SELCLOUD: a hybrid multi-criteria decision-making model for selection of cloud services,” *Soft Comput.*, vol. 23, no. 13, pp. 4701–4715, 2019.

[23] M. Abdel-Basset, M. Mohamed, and V. Chang, “NMCDA: A framework for evaluating cloud computing services,” *Futur. Gener. Comput. Syst.*, vol. 86, pp. 12–29, 2018.

[24] E. D. L. Falcão, F. Brasileiro, A. Brito, and J. L. Vivas, “Enhancing P2P Cooperation through Transitive Indirect Reciprocity,” *Proc. - 2016 IEEE 36th Int. Conf. Distrib. Comput. Syst. Work. ICDCSW 2016*, pp. 189–198, 2016.

[25] H. Liu and B. He, “Reciprocal Resource Fairness: Towards Cooperative Multiple-Resource Fair Sharing in IaaS Clouds,” *Int. Conf. High Perform. Comput. Networking, Storage Anal. SC*, vol. 2015-January, no. January, pp. 970–981, 2014.

[26] K. Zhang and N. Antonopoulos, “A lightweight reputation system for bartering exchange based incentive mechanisms in Peer-To-Peer systems,” *Proc. - 2nd Int. Conf. Intell. Netw. Collab. Syst. INCOS 2010*, pp. 443–448, 2010.

[27] Rosa, M.J., Ralha, C.G., Holanda, M. and Araujo, A.P., 2021. Computational resource and cost prediction service for scientific workflows in federated clouds. *Future Generation Computer Systems*, *125*, pp.844-858.

[28] Chauhan, Sameer Singh, Emmanuel S. Pilli, and R. C. Joshi. "BGSA: Broker Guided Service Allocation in Federated Cloud." *Sustainable Computing: Informatics and Systems* 32 (2021): 100609.

[29] Kumar, Rakesh, and Rinkaj Goyal. "Performance based Risk driven Trust (PRTrust): On modeling of secured service sharing in peer-to-peer federated cloud." *Computer Communications*, 183 (2022): 136-160.

[30] Sudhakar, N. S. Nithya, and B. L. Radhakrishnan, “Fair service matching agent for federated cloud,” *Comput. Electr. Eng.*, vol. 76, pp. 13–23, 2019.

[31] Le DN, Bhateja V, Nguyen GN. A parallel max-min ant system algorithm for dynamic resource allocation to support QoS requirements. In *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON*) 2017 Oct 26 (pp. 697-700). IEEE.

[32] Le DN. A New Ant Algorithm for Optimal Service Selection with End-to-End QoS Constraints. *Journal of Internet Technology*. 2017 Sep 1;18(5):1017-30.

[33] Le DN. Applied MMAS Algorithm to Optimal Resource Allocation to Support QoS Requirements in NGNs. In *International Wireless Internet Conference* 2014 Nov 13 (pp. 209-216). Springer, Cham.

[34] Le DN. Evaluation of pheromone update in min-max ant system algorithm to optimizing QoS for multimedia services in NGNs. In *Emerging ICT for Bridging the Future-Proceedings of the 49th Annual Convention of the Computer Society of India CSI* Volume 2 2015 (pp. 9-17). Springer, Cham.

[35] Ezenwoke, Azubuike; Daramola, Olawande; Adigun, Matthew (2018), “Simulated Cloud Service QoS Dataset”, *Mendeley Data*, V1, doi: 10.17632/5vffs75j85.

Dr. S Sudhakar obtained the Bachelor of Technology in Information Technology from Anna University, Chennai, and Tamil nadu, India in 2005 and received his Master of Engineering (M.E,) in Computer Science and Engineering from Anna University, Coimbatore India in 2011. He has completed a Ph.D. degree in Anna University, Chennai in 2020. He is currently faculty in the Department of Data Science and Business Systems, School of Computing, SRM Institute of Science and Technology, India. Skilled in developing projects and carrying out research in the area of Federated Cloud and Blockchain.

B. L. Radhakrishnan is currently faculty in the Department of Computer Science and Engineering, Karunya Institute of Technology and Sciences (KITS), India. He is currently pursuing his Ph.D. in Computer Science and Engineering, KITS. Received his M.Tech in Computer Science and Engineering from Dr.MGR University, India in 2008, and did his B.E in Computer Science and Engineering from Anna University, India in 2006. He holds Industry Certifications such as Certified OpenStack Administrator, SUSE Certified Engineer in Enterprise Linux and Google Certified Associate Cloud Engineer. His research interests include remote sleep monitoring, cloud computing and Blockchain.

Dr P. Karthikeyan obtained his Bachelor of Engineering (B.E.,) in Computer Science and Engineering from Anna University, Chennai, and Tamil nadu, India in 2005 and received his Master of Engineering (M.E,) in Computer Science and Engineering from Anna University, Coimbatore India in 2009. He has completed a Ph.D. degree in Anna University, Chennai in 2018. Skilled in developing projects and carrying out research in the area of Cloud computing and Data science with the programming skill in Java, Python, R and C. He published more than 20 International journals with a good impact factor and presented more than 10 International conferences. He was the reviewer of Elsevier, Springer, Inderscience and reputed Scopus indexed journals. He is acting as editorial board members in EAI Endorsed Transactions on Energy Web, The International Arab Journal of Information Technology and Blue Eyes Intelligence Engineering and Sciences Publication journal.

K. Martin Sagayam, completed his PhD in the field of Signal and Image Processing using Machine Learning Algorithms from Karunya Institute of Technology and Sciences (Deemed to be University), India. Currently, he is working as an Assistant Professor in the Department of ECE, Karunya Institute Technology and Sciences, Coimbatore, India. He has authored/ co-authored a greater number of referred International Journals and conferences. He has three Indian patents and two Australian patents for his innovations. He published 2 edited books, 2 authored books, book series and more than 15 book chapters with reputed international publishers. He is an active member of IEEE.

Dac-Nhuong Le has an MSc and PhD in computer science from Vietnam National University, Vietnam in 2009, and 2015, respectively. He is an Associate Professor on Computer Science, Deaon of Faculty of Information Technology, Haiphong University, Vietnam. He has a total academic teaching experience of 20+ years in computer science. He has more than 100+ publications in the reputed international conferences, journals, and book chapter contributions (Indexed by SCIE, SSCI, ESCI, Scopus). His areas of research are in the field of intelligence computing, multi-objective optimization, network security, cloud computing, virtual reality/argument reality. Recently, he has been on the technique program committee, the technique reviews, the track chair for international conferences under Springer-ASIC/LNAI/CISC Series. Presently, he is serving on the editorial board of international journals and edited/authored 20+ computer science books published by Springer, Wiley, CRC Press, IET, and Bentham Publishers.