# CoMCLOUD: Virtual Machine Coalition for Multi-Tier Applications Over Multi-Cloud Environments

Sourav Kanti Addya, *Senior Member, IEEE*, Anurag Satpathy, *Member, IEEE*, Bishakh Chandra Ghosh, *Student Member, IEEE*, Sandip Chakraborty, *Member, IEEE*, Soumya K. Ghosh, *Senior Member, IEEE*, and Sajal K. Das, *Fellow, IEEE*

**Abstract**—Applications hosted in commercial clouds are typically multi-tier and comprise multiple tightly coupled virtual machines (VMs). Service providers (SPs) cater to the users using VM instances with different configurations and pricing depending on the location of the data center (DC) hosting the VMs. However, selecting VMs to host multi-tier applications is challenging due to the trade-off between cost and quality of service (QoS) depending on the placement of VMs. This paper proposes a multi-cloud broker model called *CoMCLOUD* to select a sub-optimal VM coalition for multi-tier applications from an SP with minimum coalition pricing and maximum QoS. To strike a trade-off between the cost and QoS, we use an ant-colony-based optimization technique. The overall service selection game is modeled as a first-price sealed-bid auction aimed at maximizing the overall revenue of SPs. Further, as the hosted VMs often face demand spikes, we present a parallel migration strategy to migrate VMs with minimum disruption time. Detailed experiments show that our approach can improve the federation profit up to 23% at the expense of increased latency of approximately 15%, compared to the baselines.

**Index Terms**—Cloud computing, data center, virtual machine, migration, ant colony optimization, game theory

## 1 Introduction

The majority of the commercial cloud service providers (SPs) use region-based pricing for infrastructure-as-a-service (IaaS) provisioning by offering virtual machine (VM) prices depending on the location of the server [1], [2]. The price depends on the costs associated with the local data-center (DC) management, electricity, hardware pricing, etc. For example, the typical prices of hosting a VM of different configurations at three SPs (Microsoft Azure, Amazon EC2, and Google cloud) are shown in Table 1 for disparate regions as of June 2019. It is evident from the table that the prices of VMs depend on the DC region hosting the VM, even if the configurations remain the same. Assuring cost minimization and quality of service (QoS) together at the same time is difficult for large SPs like Azure of EC2, as they follow a user-centric approach where the end-user decides the target server [3]. On the other hand, small SPs and cloud exchanges like OnApp [4] or Red Hat OpenShift face challenges in selecting the minimum-cost DC to satisfy the QoS of users, as they follow an SP-centric approach where the SPs dynamically select the target server.

The existing literature [5], [6], [7], [8], [9] rely on the concept of cloud federation, where multiple SPs (mostly the small SP) come to a federation level agreement (FLA) for IaaS provisioning for the end-users by sharing their resources while optimizing the total cost of operation. A cloud broker typically manages a cloud federation. The broker receives the end-users application requests and allocates resources to the cloud SPs by optimizing the cost and the application QoS. The commercial cloud exchanges like OnApp [4], Arjuna's Agility Framework [10], and EGI Federated Cloud [11] are based on this principle. However, the existing broker-based services ideally work on a single VM, whereas many cloud applications are multi-tier. For example, a typical cloud-based web server follows three tiers: (i) *Application tier* hosting the business logic, (ii) *Web tier* hosting the presentation layer, and (iii) *Database tier* hosting the data layer [12]. Different layers are deployed over different VMs with disparate configurations and variable memory access patterns to ensure sufficient fault tolerance and ease of maintenance. Moreover, as Fig. 1 shows, the VMs often communicate with each other and serve the users.

- *Sourav Kanti Addya is with the Department of Computer Science & Engineering, National Institute of Technology Karnatak, Surathkal, Karnataka 575025, India. E-mail: kanti.sourav@gmail.com.*
- *Anurag Satpathy is with the Department of Computer Science & Engineering, National Institute of Technology, Rourkela, Odisha 769008, India. E-mail: anurag.satpathy@gmail.com.*
- *Bishakh Chandra Ghosh, Sandip Chakraborty, and Soumya K. Ghosh are with the Department of Computer Science & Engineering, Indian Institute of Technology, Kharagpur, West Bengal 721302, India. E-mail: {ghoshbishakh, sandipchkraborty}@gmail.com, skg@iitkgp.ac.in.*
- *Sajal K. Das is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409 USA, and also with the VAJRA Faculty, IIT Kharagpur, West Bengal 721302, India. E-mail: sdas@mst.edu.*

*Manuscript received 23 April 2020; revised 5 August 2021; accepted 9 October 2021. Date of publication 26 October 2021; date of current version 8 March 2023.*
*The authors are grateful to the editor and anonymous reviewers for insightful comments which helped us improve the technical quality and presentation of the manuscript significantly. The work of S. K. Das was partially supported by the NSF under Grants OAC-2104078 and OAC-1725755.*
*(Corresponding author: Sourav Kanti Addya.)*
*Recommended for acceptance by Y. Yang.*
*Digital Object Identifier no. 10.1109/TCC.2021.3122445*

TABLE 1
Region Based VM Pricing for Three SPs as of June 2019

| Type | Region 1 | Region 2 | Region 3 | Region 4 |
|---|---|---|---|---|
| Category (VM) | Central US (\$/hr) | India west (\$/hr) | Sydney (\$/hr) | Tokyo (\$/hr) |
| **Microsoft Azure** | | | | |
| $t_1$ (A1) | 0.012 | 0.014 | 0.015 | 0.015 |
| $t_2$ (A2) | 0.048 | 0.054 | 0.059 | 0.059 |
| $t_3$ (A3) | 0.023 | 0.027 | 0.029 | 0.029 |
| $t_4$ (A4) | 0.094 | 0.11 | 0.12 | 0.12 |
| **Amazon EC2** | | | | |
| $t_1$ (Small) | 0.023 | 0.0248 | 0.0292 | 0.0304 |
| $t_2$ (Medium) | 0.0464 | 0.0496 | 0.0584 | 0.0608 |
| $t_3$ (Large) | 0.0928 | 0.0992 | 0.1168 | 0.1216 |
| $t_4$ (2Exlarge) | 0.3712 | 0.3968 | 0.4672 | 0.4864 |
| **Google Cloud** | | | | |
| $t_1$ (Standard-1) | 0.0475 | 0.057 | 0.0674 | 0.061 |
| $t_2$ (Standard-2) | 0.0950 | 0.1141 | 0.1348 | 0.122 |
| $t_3$ (Standard-3) | 0.19 | 0.2282 | 0.2697 | 0.244 |
| $t_4$ (Standard-4) | 0.38 | 0.4564 | 0.5393 | 0.488 |

*Challenges.* Hosting multi-tier applications over the broker-based federated cloud has the following research challenges.

**(1)** As different VMs of a multi-tier application have varying requirements in terms of VM instances, a single DC under an SP may not be able to serve all the VMs. However, placing different VMs at different DCs may cause *VM dispersion* when the DCs are geographically separated [13]. Typically, the VMs in a multi-tier application needs to communicate with each other, implying an adverse impact on the performance due to *VM dispersion*. As a consequence, the objective of selecting the target SP for placing a multi-tier application is multi-criteria in nature. Two different parameters need to be optimized simultaneously – (*i*) the cost of hosting VMs, and (ii) VM dispersion cost (inter-VM communication latency).

**(2)** The cloud broker (CB) may not be the best candidate to run the above optimization requiring DC-level information, as the CB maintains only SP level and federation-related information. Maintaining DC level information has a significant overhead for the CB as the information like DC load changes very frequently. Therefore, the SPs must collectively solve the optimization in a decentralized platform.

**(3)** A federated cloud should maximize the profit of SPs and incentivize them to be part of the federation. Achieving this is challenging when the decision-making is decentralized without involving the CB.

**(4)** The hosted VMs often face increased demands and spikes [14]. The inability to meet such needs can have harmful impacts on service quality and application performance. Increased resource demand by the VMs can trigger VM migration, thereby necessitating developing a suitable VM migration strategy.

*Our Contributions.* This paper proposes a novel multi-cloud broker model, called *CoMCLOUD*, that aids the users in VM selection for multi-tier applications. While *CoMCLOUD* follows a broker-centric approach for cloud federation, the CB is only responsible for collecting user application requests and initiating the allocation process by bundling

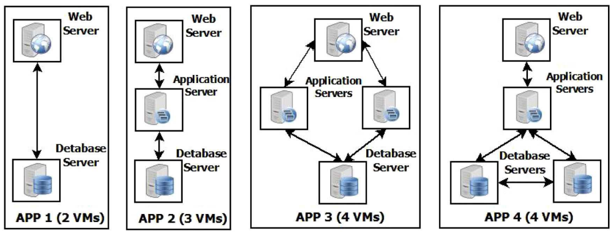


Fig. 1. Multi-tier applications: Every tier is deployed on a different VM; the VMs communicate with each other to provide end services.

them into batches. The allocation process executes in a distributed fashion. Every SP first computes the pricing for optimal service coalition (a set of VMs hosted in different DCs following the placement criteria) for a multi-tier application request. In *CoMCLOUD*, the price calculation for optimal service coalition is modeled as a bi-objective optimization, which we solve using a meta-heuristic based on *Ant Colony Optimization* (ACO).

Based on the pricing for optimal service coalition of a multi-tier application over an SP, the interactions among the SPs for resource allocation and VM placement are modeled using a first-price sealed-bid auction game, called a *service coalition selection game*. The motivation behind the auction game is attributed to the fact that different VMs have variable pricing. It requires considerable effort to determine the accurate valuation of an SP for hosting a multi-tier application. Further, the first-price sealed-bid auction is selected because of its transparency; in other words, one cannot manipulate the final price as the winner pays the amount equal to his bid. The service coalition selection game considers the previously mentioned challenges of hosting a multi-tier application over federated clouds. *CoMCLOUD* also considers the dynamic workload of VMs after deployment, thus incorporating a dynamic VM migration strategy for multi-tier applications while maintaining the performance objectives. We have implemented *CoMCLOUD* using CloudSim 3.0 simulator [15]. Experimental results show that our proposed model can reduce the aggregate price but may sometimes lead to slightly higher latency due to *VM dispersion*. Nevertheless, the latency in *CoMCLOUD* shows to be upper bounded by a threshold. More precisely, *CoMCLOUD* is able to improve the federation profit by $\sim 23\%$ at the expense of increased latency of approximately $\leq 15\%$ compared to the best-performing baseline technique.

This work is the first that aims to optimize the cost and latency of placing multi-tier applications across different DCs for different SPs to the best of our knowledge. A preliminary version of this work was published in a conference [16] where we proposed a basic framework for modeling the interactions among SPs for hosting multi-tier applications over a federated cloud. The current journal version is a significant extension that provides the end-to-end framework with extensive experimental evaluation and performance comparison with the baselines.

The rest of the paper is organized as follows. Section 2 reviews relevant literature while Section 3 provides an overview of *CoMCLOUD* architecture. Section 4 describes the procedure for selecting the optimal service coalition and Section 5 presents the service coalition selection game. The Migration procedure for multi-tier applications is explored

TABLE 2
Comparison of Different Approaches and *CoMCLOUD*

| **Work** | **VM Provisioning Cost** | **Communication Latency** | **Service Provider Profit** | **Federation Profit** | **Multi-Tier App Support** | **VM Migration** | **VM Coalition** | **QoS** | **Broker Profit** |
|---|---|---|---|---|---|---|---|---|---|
| Li *et al.* [7] | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| N. Samaan [5] | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Mashayekhy *et al.* [6] | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Konstanteli *et al.* [17] | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| Rubio-Montero *et al.* [18] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| Pillai and Rao [19] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ |
| Quarati *et al.* [20] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| Mei et al. [11] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| Nesmachnow *et al.* [21] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| Larumbe and Sansò [22] | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| **CoMCLOUD** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

in Section 6 while the performance evaluation of *CoMCLOUD* is presented in Section 7. A overall discussion on the *CoMCLOUD* is presented in Section 8. Finally, Section 9 concludes the paper with directions of future research.

## 2 Related Works

This section reviews the relevant literature on resource provisioning over multi-cloud architecture from two different aspects – *federation-centric* and *broker-centric*.

*Federation-Centric Approaches.* The works in this category have focused on various aspects of cloud federation, such as profit maximization, resource allocation, resource sharing, etc. In [7], the authors have presented a double auction-based mechanism for inter-cloud VM trading and scheduling in a cloud federation. A novel model has been proposed in [5] for capacity sharing in a federation of cloud infrastructure service providers. The interactions among the SPs are modeled as a repetitive game of VM outsourcing to offer all unused capacity in the spot market. A game theory-based cloud federation model is proposed in [6] to enhance the SPs' dynamic resources to fulfill user's demands. Konstanteli *et al.* [17] proposed a federated approach among SPs to allocate resources for dynamic services in a virtualized setup. On the other hand, Rubio-Montero *et al.* [18] presented a framework for distributed computation in federated clouds supporting multi-environment and fair-sharing for several users executing legacy applications. Furthermore, a game theory-based resource allocation model has been proposed in [19] through coalition formation among cloud SPs. Following the centralized broker-based federation architecture, various commercial prototypes (e.g., OnApp [4], Arjuna's Agility framework [10]) have been developed cloud federation for better utilization of IaaS resources.

*Broker-Centric Approaches.* Apart from the federation-centric architecture, several works in the literature also talk about broker-based multi-cloud systems. In [20], a hybrid cloud-broker (CB) is proposed to allocate services to private resources or public clouds; if the service is executed on private resources, the revenue of the CB is the sum of the brokering service price and resource provisioning price; otherwise, the CB receives only the brokering service price, while the

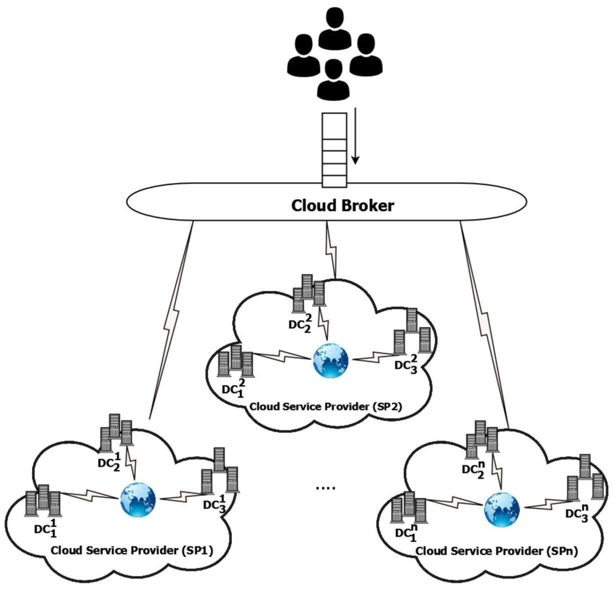


Fig. 2. CoMCLOUD architecture: A multi-cloud broker system.

resource provisioning price is paid to the public clouds. In [11], a broker-centric architecture is proposed to minimize the cost incurred by short-term users. The broker rents reserved instances with discounted prices and rents them to short-term users with smaller billing time units (BTU) at a significantly low price. The authors in [21] proposed a new kind of broker for cloud computing, whose business relies on outsourcing VMs to its customers. More specifically, the broker owns several reserved instances of different VMs from several cloud providers and offers them to its customers on demand. Finally, an energy-aware VM placing broker is proposed in [22] to minimize the operational expenditures while respecting constraints on the QoS, power consumption, and $CO_2$ emissions.

From the above discussions, it is evident that federated cloud architecture and multi-cloud broker based platforms have been investigated in the existing literature, and commercial systems have been developed to cater to the end-users for cost optimization purpose. However, existing studies mostly look into the profit maximization for the broker while ensuring the end-users' budgetary constraints. The overall objective of *CoMCLOUD* is to maximize the revenue of the SPs in the federation with an adequate allocation of user applications requests satisfying their resource demands. Table 2 provides a comparative analysis of the literature based on different criteria for multi-cloud broker architecture.

## 3 CoMCLOUD Architecture

This section provides an overview of the *CoMCLOUD* system architecture and its functionalities (see Fig. 2).

### 3.1 Federation Architecture in CoMCLOUD

*CoMCLOUD* follows a broker-based architecture where application requests from the end-users arrive at the CB which initiates the resource reservation process. The SPs register themselves with the CB. However, in contrast to typical broker-based multi-cloud models, the CB in *CoMCLOUD* does not directly take part in the resource reservation decision, thereby maintaining the trust in the system. In *CoMCLOUD*, we consider that every SP has multiple DCs with different resource availability. A complete allocation of a multi-tier application depends on successfully deploying the corresponding VMs over the DCs. We assume that a single SP handles an application, although various VMs corresponding to that application may be deployed under different DCs of the SP. Consequently, we design and solve the following two decision problems in *CoMCLOUD*.

1) The CB initiates scheduling application requests among the SPs. These may be multi-tier in nature and therefore may require multiple VMs with different resource requirements. Such requirement is batched together and a single SP handles it completely. An application request's allocation among the SPs must (*i*) satisfy the resource demands of the VMs, (*ii*) maximize the incentives for the SP, (*iii*) bound the VM dispersion within a limit, and (*iv*) maximize the revenue of the federation structure.
2) After an SP agrees to serve an application, the VMs corresponding to it is allocated over the DCs under that SP. Furthermore, to accommodate new applications, fast migration of VMs with minimum service interruption needs to be ensured. The VM allocation and migration among the DCs of an SP must (*i*) minimize the communication latency among the VMs of a multi-tier application and (*ii*) maximize the migration bandwidth for fast VM migration.

Next, we provide an overview of *CoMCLOUD* architecture to address the above decision problems.

### 3.2 CoMCLOUD Functionalities

Let $B = \{\mathcal{SP}_1, \mathcal{SP}_2, \ldots, \mathcal{SP}_l\}$ be a broker with $l$ number of SPs. An SP, denoted as $\mathcal{SP}_i = \{\mathcal{DC}_1^i, \mathcal{DC}_2^i, \ldots, \mathcal{DC}_p^i\}$, consists of $p$ geographically distributed DCs under $\mathcal{SP}_i$. For different SPs, the value of $p$ may be different, thus providing heterogeneity to our architecture. Any $\mathcal{DC}_j^i$, $1 \leq j \leq p$, has $m$ hosts represented as $\mathbb{H} = \{\mathcal{H}_j^1, \mathcal{H}_j^2, \ldots, \mathcal{H}_j^m\}$. Each host $\mathcal{H}_j^k$ has its own capacity ($\mathcal{H}_j^{k.C}$) in terms of CPU, memory, disk, etc. (where $C$ denotes any dimension). Here, the capacity denotes the current available capacity of the host. In our model, we consider different VM instances and their variable pricing (based on their location) for different SPs as shown in Table 1. We assume that multi-tier application requests in a cloud data center arrive at the CB, $B$, in batches [23] following a Poisson distribution [24] with exponentially distributed service times. The CB processes the incoming applications in batches. If $n$ number of SPs are registered with the CB, it fetches at most $n$ application requests from the arrival queue and broadcasts them among the SPs. The *CoMCLOUD* federation then executes the following four steps.

*Determine a Feasible Service Coalition at each SP for Every Application Request.* Once an $SP_i$ receives the set of application requests, it processes them individually to determine whether it can meet the resource demands of the VMs corresponding to that request. Let $\mathcal{U}_q$ be such an application with a VM instance request set, $\mathcal{U}_q^s = \{u_q^{t1}, u_q^{t2}, \ldots, u_q^{tg}\}$, where $u_q^{tx}$ is an instance of a VM type required to host the application $\mathcal{U}_q$. In this model, we assume that a multi-tier application can demand a maximum of one instance of a VM type [12]. *CoMCLOUD* can cater to the multiple instance requests. However, to keep the modeling simple, we restrict an application to one instance of a VM type. If $SP_i$ wants to bid for an application $\mathcal{U}_q$, it needs to determine a feasible *service coalition* by selecting a set of DCs under it that can host a VM of each instance type requested by $\mathcal{U}_q$. Formally, a feasible service coalition is defined as follows.

**Definition 1 (Feasible Service Coalition).** *Each application $\mathcal{U}_q$ has a corresponding VM instance request set $\mathcal{U}_q^s = \{u_q^{t1}, u_q^{t2}, \ldots, u_q^{tg}\}$. A* feasible service coalition *at $SP_i$ with the set of DCs $\mathcal{SP}_i = \{\mathcal{DC}_1^i, \mathcal{DC}_2^i, \ldots, \mathcal{DC}_p^i\}$, is defined as the set of tuples $\{\langle u_q^{tx}, \mathcal{DC}_j^i \rangle, \forall u_q^{tx} \in \mathcal{U}_q^s, \exists \mathcal{DC}_j^i \in \mathcal{SP}_i\}$ where $\mathcal{DC}_j^i$ has capacity to host $u_q^{tx}$.*

*Extract Optimal Service Coalition at each SP for Bidding a Application Request.* In *CoMCLOUD*, every SP bids for an application for which it can find out a feasible service coalition. To bid for an application $\mathcal{U}_q$, an $SP_i$ calculates its revenue $\Re(\mathcal{U}_q)$ which is a function of the total price for hosting the instance types $\mathcal{U}_q^s = \{u_q^{t1}, u_q^{t2}, \ldots, u_q^{tg}\}$ over the DCs under $SP_i$ corresponding to an *optimal service coalition*. Among all the feasible service coalitions for an application $\mathcal{U}_q$ over $SP_i$, an optimal service coalition is defined as one for which the total price for hosting the services is minimized, and the VM dispersion is minimum, i.e., the entire communication latency among all the VMs corresponding to the instance types under $\mathcal{U}_q^s$ is minimum.

We show that determining an optimal service coalition is NP-hard; also, it is a bi-objective decision-making problem where the solution space follows a Pareto-optimal frontier. To this end, *CoMCLOUD* solves this problem with the help of an *Ant Colony Optimization* (ACO) based meta-heuristic, as detailed in Section 4. The revenue corresponding to this optimal service coalition is used in the next step to bid for the application $\mathcal{U}_q$.

*Allocate Application Requests Among the Bidding SPs.* The aim is to find out an allocation of applications among the SPs using a decentralized approach. The overall service selection, referred to as *service coalition selection game* (see Section 5), is formulated as first price sealed bid auction game to attain efficient allocation. In this auction game, the objective is to maximize the total overall revenue of the federation. The game converges with an allocation for the current batch among SPs. This result is then communicated to the CB, $B$ for accountability and meeting the FLA. The *service coalition selection game* is detailed in Section 5.

*Handle Dynamic Workloads of VMs.* Based on a parallel migration strategy (see Section 6), *CoMCLOUD* supports VM migration for handling dynamic workload at VMs for multi-tier applications to ensure minimum migration overhead (in terms of migration time). This strategy allows the SPs to dynamically provision resources for the VMs with minimum interruption of services.

## 4 Selection of Optimal Service Coalition

As mentioned earlier, the *CoMCLOUD* broker $B$ forwards the application requests in batches to the SPs. To bid for applications, the SPs, in turn, compute individual revenues of the optimal coalition for each. An efficient coalition takes care of end-user's preferences, such as: (1) ensure that requisite VMs for applications are provisioned, (2) minimize the price for provisioning VMs under multi-tier applications depending on the hosting price at different DCs under the SP, and (3) minimize the VM dispersion for ensuring minimum latency for inter-VM communications. Let us first formally define the constraints for finding an efficient service coalition and formulate an optimization problem that yields the above three objectives.

## 4.1 Constraints

The constraints corresponding to a feasible service coalition are as follows.

### 4.1.1 *Completeness of Allocation*

A VM $\mathcal{V}_{tx}^{\omega}$ corresponds to an instance $u_q^{tx}$ of $\omega$-tier of a multi-tier application request $\mathcal{U}_q$. Here, $\omega \in \{ap, wb, db\}$ corresponds to a specific tier from the set of tiers comprising application ($ap$), web ($wb$), and database ($db$) for a multi-tier application [25]. Let $\mathcal{I}(\mathcal{V}_{tx}^{\omega}, \mathcal{DC}_j^i)$ be an indicator variable defined as follows

$$\mathcal{I}(\mathcal{V}_{tx}^{\omega}, \mathcal{DC}_j^i) = \begin{cases} 1 & \text{If } \mathcal{DC}_j^i \text{ under } SP_i \text{ can host } \mathcal{V}_{tx}^{\omega} \\ 0 & \text{otherwise} \end{cases}. \quad (1)$$

Let $\mathbb{V}_{\mathcal{U}_{\text{II}}}$ be the set of requisite VM instances for application $\mathcal{U}_q$. Then, the following condition defines the completeness of allocation

$$|\mathbb{V}_{\mathcal{U}_{\text{II}}}| = \sum_{\forall \mathcal{V}_{tx}^{\omega} \in \mathbb{V}_{\mathcal{U}_{\text{II}}}} \min\left(1, \sum_{\forall \mathcal{DC}_j^i \in SP_i} \mathcal{I}(\mathcal{V}_{tx}^{\omega}, \mathcal{DC}_j^i)\right). \quad (2)$$

Here $|.|$ denotes the cardinality of the set. The above constraints indicate that $SP_i$ is eligible to host the application $\mathcal{U}_q$ only if it can support all the instances under $\mathcal{U}_q^s$.

### 4.1.2 *DC Capacity Constraint*

To place a VM of type $t1$ on the $k$th host at the $j$th DC, the following constraint needs to be satisfied. Note that the assignment of VMs to servers is independent of the tier to which it corresponds and is entirely dependent on the resource availability

$$\begin{gathered} u_q^{t1.C} \leq \mathcal{H}_j^{k.C} \\ \forall C \in \{CPU, memory\}; \quad \forall u_q^{tx} \in \mathcal{U}_q^s, \end{gathered} \quad (3)$$

where $\mathcal{H}_i^{k.C}$ refers to the available capacity of some host $\mathcal{H}_j^k$ of $\mathcal{DC}_j$ along a dimension. This constraint guarantees that the host where the VM is instantiated has enough resources along all dimensions.

## 4.2 Objectives for Optimal Service Coalition

The optimal service coalition is expressed as a bi-objective optimization problem as follows.

### 4.2.1 *Objective 1: Minimizing VM Dispersion*

Although the VMs supporting the applications are instantiated under the same SP, they might reside in different DCs under that SP. Consequently, *CoMCLOUD* may place the VMs on separate DCs for cost savings which may lead to increased inter-VM communication latency for executing the applications. Therefore, we first model the inter-VM communication latency, one of the significant components for optimal service coalition. We first introduce a $|\mathcal{U}_q^s| \times |\mathcal{U}_q^s|$ communication matrix $\mathcal{M}_{\mathcal{U}_q}$ for a multi-tier application $\mathcal{U}_q$. An entry $m_{y,z} \in \mathcal{M}_{\mathcal{U}_q}$ with a value of 1 indicates a communication dependency between VMs $\mathcal{V}_y^{\omega}$ and $\mathcal{V}_z^{\omega^*}$ such that $\omega, \omega^* \in \{ap, wb, db\}$, and is set to 0 otherwise. Note that the entries in $\mathcal{M}_{\mathcal{U}_q}$ are user-dependent and can vary from one application to another based on the defined architectural constraints. For simplicity, we ignore the type of VMs as they have no impact on the latency. Let $\Phi^0[y,z]$ denote the *average communication latency* between VMs $\mathcal{V}_y^{\omega}$ and $\mathcal{V}_z^{\omega^*}$ such that $m_{y,z} = 1$ and they are placed on the same DC; this is indeed the minimum average latency for inter-VM communication over an SP. However, to utilize the price difference at different DCs, *CoMCLOUD* may also place the VMs on separate DCs, leading to higher communication overhead due to the bottleneck bandwidth connecting the DCs. Let $\Phi^{(s,d)}[y,z]$ be the average communication latency between $\mathcal{V}_y^{\omega}$ and $\mathcal{V}_z^{\omega^*}$ such that $m_{y,z} = 1$ and they are placed on two different DCs, $\mathcal{DC}_s^i$ and $\mathcal{DC}_d^i$, under the same $SP_i$. Let $\mathcal{Q}_{\mathcal{U}_q}^{SP_i}$ denote a feasible service coalition for the application $\mathcal{U}_q$ over $SP_i$. Considering multiple VMs under a multi-tier application $\mathcal{U}_q$, the *aggregate latency* (AL) experienced by $\mathcal{U}_q$ over $SP_i$ due to VM dispersion over a service coalition $\mathcal{Q}_{\mathcal{U}_q}^{SP_i}$ is expressed as follows

$$\Lambda(\mathcal{Q}_{\mathcal{U}_q}^{SP_i}) = \sum_{\forall \mathcal{V}_y^{\omega} \in \mathcal{U}_q^s} \left( \sum_{\forall \mathcal{V}_z^{\omega^*} \in \mathcal{U}_q^s}^{\mathcal{V}_y^{\omega} \neq \mathcal{V}_z^{\omega^*}} \left( \Phi^{(s,d)}[y,z] - \Phi^0[y,z] \right) \times m_{y,z} \right). \quad (4)$$

For optimal service coalition for an application $\mathcal{U}_q$ over $SP_i$, the objective is to obtain a feasible coalition $\mathcal{Q}_{\mathcal{U}_q}^{SP_i'}$ such that $\Lambda(\mathcal{Q}_{\mathcal{U}_q}^{SP_i'})$ is minimum among all possible feasible service coalitions.

### 4.2.2 *Objective 2: Minimizing the Price for VM Hosting*

Let $\Theta_y^j$ denote the price for placing $\mathcal{V}_y^{\omega}$ over $\mathcal{DC}_j$. For a feasible coalition $\mathcal{Q}_{\mathcal{U}_q}^{SP_i}$ of $\mathcal{U}_q$ over $SP_i$, the total price for hosting the service is given as follows

$$\Psi(\mathcal{Q}_{\mathcal{U}_q}^{SP_i}) = \sum_{\langle \mathcal{V}_y^{\omega}, \mathcal{DC}_j \rangle \in \mathcal{Q}_{\mathcal{U}_q}^{SP_i}} \Theta_y^j. \quad (5)$$

### 4.2.3 *Bi-Objective Optimization for Optimal Service Coalition*

Let $\mathbb{C}_{\mathcal{U}_q}^{SP_i}$ bet the set of all feasible service coalitions for the application $\mathcal{U}_q$ over $SP_i$. Summarizing the objectives and the constraints discussed above, the optimal service coalition problem can be expressed as follows. Eqs. (6a) and (6b) represent the overall objective of *CoMCLOUD*, i.e., minimization of latency and cost for hosting multi-tier applications. Constraint 6c enforces the coalition generated by *CoMCLOUD* must belong to the set of feasible allocations and should be able to allocated all VMs of a multi-tier application. Constraint 6d makes sure that an individual VM can only be hosted on a server if it has sufficient resources across different dimensions, i.e., CPU and memory. Finally, (6e) and (6f) depict the feasible set of values of the variables

$$\min_{\forall \mathcal{Q}^{SP_i}_{\mathcal{U}_q} \in \mathbb{C}^{SP_i}_{\mathcal{U}_q}} \Lambda(\mathcal{Q}^{SP_i}_{\mathcal{U}_q}) \tag{6a}$$

$$\min_{\forall \mathcal{Q}^{SP_i}_{\mathcal{U}_q} \in \mathbb{C}^{SP_i}_{\mathcal{U}_q}} \Psi(\mathcal{Q}^{SP_i}_{\mathcal{U}_q}) \tag{6b}$$

$$\text{s.t.} \quad \forall \mathcal{Q}^{SP_i}_{\mathcal{U}_q} \in \mathbb{C}^{SP_i}_{\mathcal{U}_q},$$

$$|\mathbb{V}_{\mathcal{U}_{\text{II}}}| = \sum_{\forall \mathcal{V}^{\omega}_{tk} \in \mathbb{V}_{\mathcal{U}_{\text{II}}}} \min\left(1, \sum_{\forall \mathcal{DC}^i_j \in SP_i} \mathcal{I}(\mathcal{V}^{\omega}_{tx}, \mathcal{DC}^i_j)\right) \tag{6c}$$

$$u_q^{t1.C} \leq \mathcal{H}_j^{k.\,C} \tag{6d}$$

$$\forall C \in \{CPU, memory\}; \;\; \forall u_q^{tx} \in \mathcal{U}_q^s \tag{6e}$$

$$\forall k \in \{1, 2, \ldots, |\mathcal{H}^i_j|\}; \;\; \forall\, \omega \in \{ap, wb, db\}; \tag{6f}$$

Next, we show that the above objectives under the constraints is a hard problem to solve using a polynomial time algorithm.

**Theorem 1.** *The optimal service coalition problem for multi-tier applications is $\mathcal{NP}$-hard.*

**Proof.** We first show that the *Bin Packing Problem* [26] is polynomial-time reducible to the optimal service coalition problem. The *Bin Packing Problem* is defined as follows: Given a set of bins $\mathcal{B}_1, \mathcal{B}_2..\mathcal{B}_p$, of same size $W$, and a list of objects $o_1, o_2, .., o_g$ with sizes $a_1, a_2, .., a_g$ respectively, find the smallest $K$ partition (minimize $K$) $\mathcal{B}_1 \cup .. \cup \mathcal{B}_K$ of the objects, such that $\sum_{o_i \in \mathcal{B}_j} a_i \leq W for\ all\ j = 1, .., K$. In order to reduce the *Bin Packing Problem* to the optimal service coalition problem, we map each bin $\mathcal{B}_j$ to a data center $\mathcal{DC}^i_j$ of the service provider $\mathcal{SP}_i$. Consider each data center has only one host $\mathcal{H}^1_j$ with capacity $\mathcal{H}^{1.C}_j = W$ (only one-dimensional capacity). Each item $o_j$ is mapped to $u^{tj}_q$, which is a requested VM instance type of $\mathcal{U}^s_q$. Now, we set $\Phi^0[y, z] < \Phi^{(s,d)}[y, z] \;\; \forall\ combinations\ of\ y, z\ and\ \mathcal{DC}_s, \mathcal{DC}_d$. That is, the average communication latency between any two VMs will always be less in case they are placed in the same DC, compared to when they are placed in different DCs. For simplicity, we assume $\Phi^0[y, z] = 0\ and\ \Phi^{(s,d)}[y, z] = 1 \;\; \forall\ combinations\ of\ y, z\ and\ \mathcal{DC}_s, \mathcal{DC}_d$.

We also simplify the optimization problem to minimize only the aggregate latency $\Lambda(\mathcal{Q}^{SP_i}_{\mathcal{U}_q})$ of a feasible service coalition $\mathcal{Q}^{SP_i}_{\mathcal{U}_q}$, for application $\mathcal{U}_q$. So the threshold for the total cost of the service coalition is set to $\Psi(\mathcal{Q}^{SP_i}_{\mathcal{U}_q}) = \infty$.

It is evident from the chosen values of $\Phi^0[y, z]\ and\ \Phi^{(s,d)}[y, z]$ that in order to minimize the aggregate latency $\Lambda(\mathcal{Q}^{SP_i}_{\mathcal{U}_q})$, more VMs need to be placed in the same DC. Therefore, the optimal solution will place all the requested VMs using the least number of DCs possible so that the aggregate latency is as low as possible. This behavior results in an optimal solution with a minimum number of DCs, which corresponds to the optimal solution with a minimum number of bins in the *Bin Packing Problem*.

The reduction is possible in polynomial time by mapping each bin, and each object to DCs and the requested VM instance type, respectively. The overall time complexity is $\mathcal{O}(p) + \mathcal{O}(g)$, where $p$ is the number of bins, and $g$ is the number of objects.

As the Bin Packing problem is known to be $\mathcal{NP}$-hard [26], we can conclude that the optimal service coalition problem for multi-tier applications is $\mathcal{NP}$-Hard. □

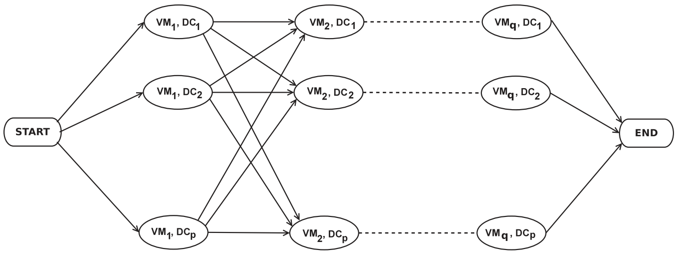


Fig. 3. Construction graph for ant-colony based solution.

Now it is possible to derive a Pareto-optimal frontier for solving the optimal service coalition problem, as the VM dispersion may arise when minimizing the value of $\Psi(.)$. On the other hand, minimizing $\Lambda(.)$ may yield a solution that is not optimized in terms of $\Psi(.)$. An example can be a situation when the minimum-priced DC can host only a subset of the VMs under $\mathcal{U}^s_q$. Therefore, we develop a meta-heuristic based on ACO, where the value of $\Psi(.)$ is minimized while keeping the value of $\Lambda(.)$ within a pre-defined threshold. The threshold value depends on the tolerable latency for an application and is considered as an additional parameter in *CoMCLOUD*. The details follow in the following subsection.

## 4.3 A Meta-Heuristic Based Solution

In order to map the optimal coalition selection problem to the ACO problem, we define a construction graph $G^{Con}(N^{Con}, L^{Con})$ as shown in Fig. 3. The assignment of VMs to DCs at different times are represented as tuples $\langle \mathcal{V}^{\omega}_y, \mathcal{DC}_j \rangle$ in the graph indicating that $\mathcal{DC}_j$ has sufficient resources to place the VM, $\mathcal{V}^{\omega}_y$. The link between any two vertices in the graph indicates the likelihood that an artificial ant would move from one to another. This likelihood is computed based on two factors: *heuristic information* and *pheromone trails*. The former indicates the partial contribution of each movement towards the objective function, whereas the latter guides the ants towards a better solution. The nest of the ants is denoted by a start vertex, the food, and an end vertex. A trail from the start vertex to the end vertex corresponds to a feasible solution, i.e., mapping all VMs.

### 4.3.1 Initializing the Pheromone Trail

The initial pheromone trail $\tau^0$ is calculated as

$$\tau^0 = \frac{1}{|\mathcal{U}^s_q|(\Psi(\mathcal{S}^0) + \Lambda(\mathcal{S}^0))}, \tag{7}$$

where $\mathcal{S}^0$ is the initial solution generated by the following policy. We sort the DCs based on their price and then place the maximum possible VMs in the minimum priced DC, followed by the second minimum priced DC, and so on.

### 4.3.2 Computation of the Heuristic Information

The heuristic information, represented by $\eta^{\mathcal{V}^{\omega}_y}_{\mathcal{DC}_j}$, indicates the desirability of $\mathcal{V}^{\omega}_y$ to move to $\mathcal{DC}_j$. This can be calculated as

$$\eta_{\mathcal{DC}_j}^{\mathcal{V}_y^{\omega}} = \frac{1}{\Theta_y^j} + \frac{1}{\sum_{\substack{\forall \mathcal{V}_z^{\omega^*} \in \mathcal{U}_q^s, \\ \langle \mathcal{V}_z^{\omega^*}, \mathcal{DC}_{j'} \rangle \in \mathcal{Q}_{\mathcal{U}_q}^{SP_i}}}^{\mathcal{V}_y^{\omega} \neq \mathcal{V}_z^{\omega^*}} \left( \Phi^{(j,j')}[y,z] - \Phi^0[y,z] \right)}. \tag{8}$$

The above equation indicates that the affinity of $\mathcal{V}_y^{\omega}$ towards $\mathcal{DC}_j$ is inverse of the price to host $\mathcal{V}_y^{\omega}$ over $\mathcal{DC}_j$, and the overall latency for communicating with other VMs of $\mathcal{U}_q$ such that $\forall\ m_{y,z} \in \mathcal{M}_{\mathcal{U}_q}, m_{y,z} = 1$ and are in the feasible coalition $\mathcal{Q}_{\mathcal{U}_q}^{SP_i}$.

### 4.3.3 Computation of the Pheromone Trail

Every ant contributes to the local pheromone trail as follows. It updates the local pheromone for the incoming edge of a vertex corresponding to the tuple $\langle \mathcal{V}_y^{\omega}, \mathcal{DC}_j \rangle$ by indicating its contribution for visiting that node. Let $\mu_{lc}$ denote the local pheromone evaporation rate. Then the local pheromone $\tau^{\delta}_{(\mathcal{V}_y^{\omega}, \mathcal{DC}_j)}$ for placing $\mathcal{V}_y^{\omega}$ over $\mathcal{DC}_j$ at iteration $\delta$ is updated as

$$\tau^{\delta}_{(\mathcal{V}_y^{\omega}, \mathcal{DC}_j)} = (1 - \mu_{lc}) \tau^{\delta-1}_{(\mathcal{V}_y^{\omega}, \mathcal{DC}_j)} + \mu_{lc} \times \tau^0. \tag{9}$$

Additionally, the global pheromone trail for every incoming edge of the tuple $\langle \mathcal{V}_y^{\omega}, \mathcal{DC}_j \rangle$ is updated periodically, depending on the affinity of the previous solution towards a new solution in every iteration $\delta$. Let $\mathcal{S}^{\delta}$ be the solution obtained in iteration $\delta$, and $\mu_{gl}$ be the evaporation rate of the global pheromone. Then the global pheromone trail $\mathcal{T}^{\delta}$ is computed as

$$\mathcal{T}^{\delta} = (1 - \mu_{gl}) \mathcal{T}^{\delta-1} + \frac{\mu_{gl}}{|\mathcal{U}_q^s| (\Psi(\mathcal{S}^{\delta}) + \Lambda(\mathcal{S}^{\delta}))}. \tag{10}$$

Finally, the total pheromone contribution of an incoming edge corresponding to the tuple $\langle \mathcal{V}_y^{\omega}, \mathcal{DC}_j \rangle$ at iteration $\delta$ is calculated as

$$\mathcal{T}^{\delta}_{(\mathcal{V}_y^{\omega}, \mathcal{DC}_j)} = \tau^{\delta}_{(\mathcal{V}_y^{\omega}, \mathcal{DC}_j)} + \mathcal{T}^{\delta}. \tag{11}$$

### 4.3.4 An Iterative Execution to Find Out the Optimal Solution

The iterative execution of ACO for optimal coalition formation works as follows. Every iteration is considered as an ant traversing from the start vertex to the end vertex by visiting a path in the construction graph depending on the *heuristic information* and *pheromone trails* across the edges of the path. At the beginning of every iteration $\delta$, the value of $\mathcal{T}^{\delta}_{(\mathcal{V}_y^{\omega}, \mathcal{DC}_j)}$ is updated for every incoming edge corresponding to the tuple $\langle \mathcal{V}_y^{\omega}, \mathcal{DC}_j \rangle$. Initiating from the start vertex, a path in the construction graph is selected by every ant as follows. Consider an ant already visited the tuple $\langle \mathcal{V}_y^{\omega}, \mathcal{DC}_j \rangle$. Next, the task is to select $\mathcal{DC}_{j'}$ such that it can move from $\langle \mathcal{V}_y^{\omega}, \mathcal{DC}_j \rangle$ to $\langle \mathcal{V}_{y+1}^{\omega^*}, \mathcal{DC}_{j'} \rangle$, indicating that the next VM $\mathcal{V}_{y+1}^{\omega^*}$ can be placed at $\mathcal{DC}_{j'}$. Let $\Omega_{\mathcal{V}_{y+1}^{\omega^*}}$ be the set of DCs where $\mathcal{V}_{y+1}^{\omega^*}$ can be placed ensuring the constraint in Eq. (6d). $\mathcal{DC}_{j'}$ is selected based on the following equation

$$\mathcal{DC}_{j'} = \begin{cases} \arg\max_{\mathcal{DC}_m \in \Omega_{\mathcal{V}_{y+1}^{\omega^*}}} \mathcal{W} & \text{if } r \leq r_0 \\ \text{Any random } \mathcal{DC}_m \in \Omega_{\mathcal{V}_{y+1}^{\omega^*}} & \text{otherwise} \end{cases}, \tag{12}$$

where $\mathcal{W} = \beta \times \eta_{\mathcal{DC}_m}^{\mathcal{V}_{y+1}^{\omega^*}} + (1 - \beta) \times \mathcal{T}^{\delta}_{(\mathcal{V}_{y+1}^{\omega^*}, \mathcal{DC}_m)}$; $0 < \beta < 1$ and $0 < r_0 < 1$ are two hyper-parameters; and $r_0$ is a random number between 0 and 1. This introduces randomness in every iteration so that the maximum solution paths are visited. The solution converges at iteration $\delta$ when $(\mathcal{T}^{\delta} - \mathcal{T}^{\delta-1}) \leq \epsilon$ where $\epsilon$ is a threshold indicating no significant change in the global pheromone contribution in two successive iterations. We obtain (near)-optimal service coalition once the ACO terminates.

### 4.3.5 Working of Meta Heuristics Solution

To illustrate the working of the meta-heuristic ACO, we refer to the traveling salesman problem (TSP), were given a set of interconnected cities; a salesman needs to find the shortest route visiting every city exactly once. As TSP is polynomial-time unsolvable, Dorigo *et al.* [27] discussed an ACO-based solution where artificial agents called ants placed at a start city visit a new city (vertex in a graph) in every iteration to build a global solution. While constructing the solutions, artificial ants show cooperative behavior by exchanging information via pheromone deposited on graph edges. Analogous to the TSP and referring to the construction graph in Fig. 3, we describe the working of ACO as follows: (1) artificial ants are placed at the start vertex, (2) at every iteration, each ant traverses a new vertex in the construction graph proceeding towards the end vertex. Each vertex traversed depicts an assignment of a VM of a multi-tier application. (3) After repeated such traversals, as an ant reaches the end vertex, it terminates its traversal. The nodes traversed in the path of an ant denote a feasible allocation for VMs of the multi-tier application. The detailed working of ACO can be found in [27].

## 5 Service Coalition Selection Game

Once the price for the optimal service coalition is decided by the SPs, the SP executes the service coalition selection game to allocate the application requests among themselves. A game-theoretic approach is used to obtain a stable allocation avoiding the broker, so that the SPs are satisfied with their respective allocations from which they have no incentive to deviate. Consequently, our objective is to maximize the revenue of the federation.

A game typically comprises of conceivable strategies of each player and corresponding payoffs [28]. For an application request to be placed in its own DCs, an SP can impose some strategy. The more the number of applications an SP can allocate, the more profit it can gain. *CoMCLOUD* models the service coalition selection game as a *First Price Auction* [29], [30]. The motivation behind a bidding-based allocation even after the valuation of individual VMs is due to the fact that it is arduous to determine the valuation for a distributed allocation of a group of VMs across different providers with variable pricing. Hence, a first price sealed bid auction strategy finds the actual valuation of a given service comprising a multi-tier application.

For every $SP_i$, the bid value is obtained from the most optimal coalition $\mathbb{Q}_{\mathcal{U}_q}^{SP_1}$ and is decided by individual providers. Given First Price Auction, the set of bids from all participating SPs are given by

$$\Psi = [\Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_1}), \ldots, \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}), \ldots, \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_l})]. \tag{13}$$

## 5.1 Bidding Mechanism

The *Service Coalition Selection Game* must ensure that each SP's bids are kept secret from the other SPs. Even after the completion of the auction, the bids should be kept secret until the result is revealed. At the same time, the auction result must be trusted and accepted by each of the participating SPs. In our CloudSim implementation for *CoMCLOUD* (see Section 7), this bidding process is carried out by the CB as a trusted third-party auctioneer. However, to further ensure the trustworthiness of the system, the entire bidding mechanism in *CoMCLOUD* can be implemented in a decentralized process without relying on any single trusted entity such as the CB. Such distributed implementation of the bidding process will reduce dependency on the CB while increasing democracy among SPs without affecting the service coalition selection game. In Section 8 (Point 4), we discuss one such decentralized implementation based on blockchain, using Hyperledger Fabric [31].

## 5.2 Payoff Calculation

Given the above (real) bid sequence in Eq. (13), let $SP_i$ be the winner with its bid $\Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})$, which includes the cost for placement of VMs and the profit value for SP. Therefore, the payoff is $\Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) - \Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})$, where $\Upsilon(\cdot)$ is the deployment cost of $SP_i$ for placing $\mathcal{U}_q$ application request in its DCs. Formally,

**Definition 2 (Payoff).** *The payoff function $\mathbb{P}_i$ for the player $SP_i$ in the game associated with first price auction can be defined as*

$$\mathbb{P}_i(\Psi) = \begin{cases} \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) - \Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) & if \;\; i = argsmax\ \Psi \\ 0 & \qquad\qquad otherwise \end{cases},$$

*where $\Psi$ is the vector of bids.*

If the following three conditions are satisfied, the service coalition game with bids $\Psi$ attains Nash Equilibrium [32].

1) $\Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) \leq \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})$, i.e., no *overbidding* condition.
2) $min_{i' \neq i} \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) \geq \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_{i'}})$, i.e., *sufficient high bid* offered by the winner.
3) $\Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) \neq min_{i' \neq i} \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_{i'}})$, i.e., no more than one player submitted the same bid as $SP_i$.

**Theorem 2.** *If the proposed game model with players' valuation (deployment cost) is $\Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})$, then $\Psi$ is always in Nash Equilibrium (NE) iff $\forall\ i = argsmax\ \Psi$.*

**Proof.**

1) If $\Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) > \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})$, then the payoff of player $SP_i$ is negative as he bids more than his valuation, and it can be increased to 0 if the bid of $SP_i$ is $\Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})$. Hence, the player's incentive is to reduce his bid.
2) If $min_{i' \neq i} \Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) > \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})$, then the player $SP_{i'} \ni \Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_{i'}})$ can be a *winner* whose bid is in the open interval $(\Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}),\ \Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_{i'}}))$, and consequently his utility can be improved. Hence, this situation incentivizes the players to deviate from their bid and disturbs the stability of the allocation,
3) If $\Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) > min_{i' \neq i} \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_{i'}})$, then the player $SP_i$ can bid with an increment in the open interval $(max_{i' \neq i} \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_{i'}}),\ \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}))$. Hence, the players have an incentive to deviate from their initial bid, yet maximizing their utility (i.e., revenue).

Therefore, by contradiction, we have shown that $\Psi$ is not in NE if any of the above three conditions (1) to (3) is violated. Let us now consider that if any bid $\Psi$ holds all three conditions, then player $SP_i$ is the *winner* with non-negative payoff. On the other hand, the payoff of player $SP_{i'}$ is 0 iff he bids more but violates condition (2). Therefore, his payoff must be negative and $\Psi$ is in NE. □

In a single broker multi-cloud SP scenario, the SPs, the users and the broker have their own individual preferences. The SPs want to host maximum number of applications, thus resulting in the maximization of their profit while balancing the overall investment. We have assumed that there is no overbidding condition. The users, on the other hand, have different QoS and price specifications. All such preferences have an impact on the broker's own preferences, which leads to the stable multi-cloud broker system and its responses. The revenue earned by an SP by hosting certain applications can be calculated as

$$\mathrm{R}ev_i = \sum_{\forall \mathcal{U}_q \in \mathcal{SP}_i} \Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}) - \Upsilon(\mathbb{Q}_{\mathcal{U}_q}^{SP_i}). \tag{14}$$

Here, $Rev_i$ denotes the overall revenue of $\mathcal{SP}_i$ for hosting many user application. In the context of *CoMCLOUD*, the revenue earned by individual SPs is the cost of hosting a given set of a multi-tier applications. The revenue is calculated by excluding a fixed deployment cost from the overall cost of hosting such applications. The primary agenda of *CoMCLOUD* is to maximize the individual profit of service providers while respecting the latency constraints of the multi-tier applications. Furthermore, the maximization of social welfare of *CoMCLOUD* architecture implies maximization of overall revenue, which in turn depends on the revenue generated by the individual SPs for a fixed deployment cost (maintenance cost + operational cost). This is adequately captured by Eq. (14).

It is well established that in a first-price auction of a single item, a Nash Equilibrium is always efficient. Thus both the *Price of Anarchy (PoA)* and the *Price of Stability (PoS)* are always 1.

### 5.2.1 Working of Service Coalition Selection Game

The service coalition game outputs a sub-optimal assignment for multi-tier applications over a Geo-distributed infrastructure. It proceeds as follows: (1) as soon as a multi-tier application request arrives, it is broadcast to all the registered SPs. (2) SPs having enough resources construct a sub-optimal allocation plan for VMs that best suit the objectives. This optimal solution (coalition of VMs) is obtained using ACO. (3) After each SP obtains the allocation, it quotes its price as a bid to the cloud broker. Note that the SPs distributively calculate their prices. (4) Once a broker receives all price quotations, a first price sealed bid auction is conducted by the broker. The SPs that quotes the highest price emerges as the winner of the auction. This implies the winner hosts that particular multi-tier application. The remaining players wait for the next application, and the entire process is repeated.

## 6 Migration for Multi-Tier Applications

The VMs supporting multi-tier applications may become overloaded due to the demand spikes or increased requirements [14] [33]. The SPs can respond to the demand spikes of VMs in two ways, i.e., (1) horizontal scaling and (2) vertical scaling. In the former strategy, more VMs are instantiated to deal with the excess demand, whereas the latter scale up resources of individual VMs facing additional demand. In *CoMCLOUD*, we prefer to vertically scale the resources owing to the following: (1) management of VMs is simpler, (2) lower maintenance costs, (3) minimum/no changes to the implementation, and (4) reduces communication dependencies as less VMs are instantiated. However, in some situations, the servers hosting the VMs may not have the resources to deal with excess demands, thereby requiring relocating of VMs. VM migration provides with seamless relocation of VMs, hence addressing this issue [34]. Migrating a VM involves copying the disk, the memory images, and the CPU state; a live migration enables the VMs to continue execution during the migration process [35], [36]. Important parameters dictating the performance of any migration technique are *migration time* and *application downtime* [24]. The migration time refers to the total duration of migration, while the downtime is when the running application within the VM is suspended. In *CoMCLOUD*, we use pre-copy-based live migration [37], in which the migration time for a VM involves a disk copy phase followed by an iterative memory-copy phase. The obvious option to reduce the migration overheads is to conduct VM migrations in parallel. Further, multi-tier applications cannot resume their normal execution until all VMs have been successfully migrated. Therefore, the migration strategy works in two phases: first, we determine the most feasible path to conduct migration, i.e., the one with the highest residual bandwidth; second, we perform multiple such migrations.

The VM image is generally copied in a single round, and the memory of the VMs is copied iteratively over several rounds [38]. Eqs. (15) and (16) denote the disk and memory migration times, respectively. Let $\mathbb{O}$ denote the set of all overloaded VMs in a multi-tier application, and let $n_{max}$ indicate the maximum number of pre-copy rounds. We also define $\mathcal{V}_m^{img}$ and $\mathcal{V}_m^{mem}$ as the image and memory of a VM $\mathcal{V}_m^{\omega}$ to be migrated. For simplicity of notations, we ignore the tier at which the VM is operating as it has no relevance in the migration process. Let $\mathcal{L}_x$ denote the migration bandwidth available for the VM $\mathcal{V}_m$. Additionally, let $dr_x$ and $\mathcal{V}_m^{th}$ respectively refer to the ratio of dirtying rate to the transmission rate and the memory threshold to stop the pre-copy rounds

$$\mathbb{T}_{img}^{m} = \frac{\mathcal{V}_m^{img}}{\mathcal{L}_x} \tag{15}$$

$$\mathbb{T}_{mem}^{m} = \frac{\mathcal{V}_m^{mem}}{\mathcal{L}_x}\left(\frac{1 - dr_x^{n_{itr}(x)+1}}{1 - dr_x}\right). \tag{16}$$

The number of iterations $n_{itr}(x)$ for parallel migration of a VM is derived as follows

$$n_{itr}(x) = \min\left(\left\lceil \log_{dr_x}(\mathcal{V}_m^{th}/\mathcal{V}_m^{mem})\right\rceil,\, n_{\max}\right);\ \ 0 < dr_x < 1. \tag{17}$$

Hence, the total migration time for migrating all VMs in $\mathbb{O}$ is defined as

$$\mathbb{T}_{mig}^{p} = \max_{\forall\, \mathcal{V}_m \in \mathbb{O}} (\mathbb{T}_{img}^{m} + \mathbb{T}_{mem}^{m}). \tag{18}$$

The downtime of a single VM is the time taken by the stop-and-copy phase of the memory pre-copy phase added with an intrinsic delay $\mathbb{T}_r$ taken to resume the execution at the destination. Hence, the downtime for migrating a single VM is calculated as

$$T_{\text{down}}^{\text{m}} = \frac{\mathcal{V}_m^{mem}}{\mathcal{L}_x} dr_x^{n_{itr}(x)} + \mathbb{T}_r. \tag{19}$$

Downtime for migrating VMs in the overloaded set is calculated as the difference between the downtime of the VM that is last to complete its migration to the one suspended at the very first. It is derived as follows:

$$\mathbb{T}_{down}^{p} = \max_{\forall \mathcal{V}_m \in \mathbb{O}} (T_{\text{down}}^{\text{m}}) - \min_{\forall \mathcal{V}_z \in \mathbb{O}} (T_{\text{down}}^{\text{z}} - \mathbb{T}_r). \tag{20}$$

In *CoMCLOUD*, the VM migration works in the following way. For an overloaded VM, $\mathcal{V}_m$, we first try to assign the additional resource on the same server where the VM was initially assigned. If the initial server is incapable of providing additional resources, we migrate the VM to another server that satisfies the updated resource demands. In order to migrate $\mathcal{V}_m$, we determine a path $p_{\mathcal{V}_m}^{j} \in p_{\mathcal{V}_m}$, where $p_{\mathcal{V}_m}$ is the set of all possible end-to-end paths to migrate $\mathcal{V}_m$. The available end-to-end migration bandwidth can be computed as follows:

$$\mathcal{L}_c = \min_{e_v \in p_{\mathcal{V}_m}^{j}} (e_v). \tag{21}$$

Finally, we select the path with the highest available migration bandwidth, $\mathcal{L}_x$ that is set as follows:

$$\mathcal{L}_x = \max_{\forall p_{\mathcal{V}_m}^{j} \in p_{\mathcal{V}_m}} (\mathcal{L}_c). \tag{22}$$

Since VMs of multi-tier applications experience demand spikes, a parallel migration strategy enables *CoMCLOUD* to reconfigure the VMs with minimum service degradation by reducing the migration time and the service downtime.

## 7 Performance Evaluation

We have implemented the *CoMCLOUD* architecture and VM allocation strategy at the CB using CloudSim 3.0 simulator [15]. The readers can find the source code for implementation at [39]. The details are presented in this section, along with experimental results.

### 7.1 Simulation Environment

Table 3 summarizes the parameters used for simulation. As mentioned in Section 1, we used four different VM instances and their respective pricing models from Amazon EC2 [40], Microsoft Azure [41], and Google Cloud [42]. Table 4 summarizes the configurations used in our analysis.

The multi-tier applications are generated randomly as per Table 6. For simulation, we have considered 3 SPs, each having 4 DCs representing four different regions (see Table 1).

TABLE 3
Simulation Parameters

| Sl. No. | Category | Value |
|---|---|---|
| 1 | Number of cloudlets | 250 |
| 2 | Cloudlet Length | 40000 |
| 2 | Input File size | 500 Bytes |
| 3 | Output File Size | 500 Bytes |
| 4 | Number of Hosts | 500 |
| 5 | Number of APPS | 50 - 250 |

TABLE 4
VM Type Configurations Used in Simulation [40], [41], [42]

| Sl. No. | Category(VM) | Core | RAM(GB) |
|---|---|---|---|
| **Microsoft Azure** | | | |
| 1 | $t_1$ (A1) | 1 | 2 |
| 2 | $t_2$ (A2) | 2 | 4 |
| 3 | $t_3$ (A3) | 4 | 8 |
| 4 | $t_4$ (A4) | 8 | 16 |
| **Amazon EC2** | | | |
| 1 | $t_1$ (Small) | 1 | 2 |
| 2 | $t_2$ (Medium) | 2 | 4 |
| 3 | $t_3$ (Large) | 2 | 8 |
| 4 | $t_4$ (X Large) | 4 | 16 |
| **Google cloud** | | | |
| 1 | $t_1$ (Standard-1) | 1 | 3.75 |
| 2 | $t_2$ (Standard-2) | 2 | 7.5 |
| 3 | $t_3$ (Standard-4) | 4 | 15 |
| 4 | $t_4$ (Standard-8) | 8 | 30 |

Each DC has a fat-tree topology [24] generated using the BRITE topology generator [43]. The Top of the Rack (ToR) links (connecting the servers and switches) are set to be 1 Gbps links, and the switch to switch (both aggregate and core) links are set to a capacity of 10 Gbps. The interconnect links between different DCs are assumed to be high capacity 40 Gbps links. Each DC under an SP comprises 32 physical servers. The physical servers are chosen randomly from a pool of three configurations as shown in Table 5.

A multi-tier application request consists of multiple VM instance types. The instance-wise composition of such applications is neither made public by SPs nor is readily available in the literature. Therefore, we consider applications as shown in Table 6 with different VM instances in our simulation set up to approximate the real-world closely. These VM instance configuration and pricing are according to the real-world setting as defined in Tables 1 and 4 and are made public by different SPs.

The dirty memory threshold for stopping the per-copy iteration is set at 0.1 times the VM size. The ratio of dirtying rate to the transmission rate, i.e., $r$ of each VM, is generated following a uniform distribution in the range of U [0,1]. The maximum number of iterations $n_{max}$ to prematurely stop the pre-copy rounds is set to 8. The fraction of overloaded VMs is set to 40% per application. Finally, we considered 50 to 250 applications at an interval of 50 per observation. We also assume that the appearance of applications at the broker queue follows a Poisson distribution. The reason for modeling the arrival using a Poisson process is threefold [24]: First, the multi-tier application requests generally arrive one at a time. Second, the probability that a multi-tier application request arrives at any time is independent of when other application requests come. Third, the likelihood that a multi-tier request arrives at a given time is independent of the time. Moreover, these properties make Poisson distribution an apt choice for modeling the arrival of multi-tier application requests [44]. The appearance of applications at the broker closely resembles real-world setups such as supermarkets, banks, telephone exchanges, customer service call centers, and other retail establishments and are also modeled using Poisson distribution.

TABLE 5
Host Configuration Used in Simulation

| MIPS Rating | Pes | RAM | Disk Size | VMM |
|---|---|---|---|---|
| 100000 | 16 | 32 GB | 1 TB | Xen |
| 125000 | 16 | 32 GB | 800 TB | Xen |
| 135000 | 32 | 64 GB | 1.5 TB | Xen |

TABLE 6
Application Set Configuration Used in Simulation

| Application ID | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|
| APP1 | ✓ | ✗ | ✓ | ✓ |
| APP2 | ✗ | ✓ | ✓ | ✗ |
| APP3 | ✓ | ✓ | ✗ | ✓ |

## 7.2 The Baseline Algorithms

To compare the performance of *CoMCLOUD*, we consider a modified version of the heuristic-based algorithm due to Sun *et al.* [24] (referred to as VDC(modi)) and Metwally *et al.* [1] (referred to as Metwally) as baselines. The authors in [24] aimed to reduce the virtual data center (VDC) re-embedding cost considering the cost of hosting VMs on the servers and virtual links over the substrate paths. For comparison, we consider only the VM embedding cost while embedding the VDCs, and ignore the virtual link mapping cost as a parameter for evaluation. The remaining constraints on the virtual links, such as the delay and capacity, are considered in our mapping.

On the other hand, the authors in [1] aimed at maximizing the revenue of geo-distributed DCs for hosting IaaS requests (in the form of VDCs) using a double auction strategy. To compare its performance with *CoMCLOUD*, we make slight modifications to the proposal; similar to VDC(modi), we ignore the virtual link mapping cost, and only focus on the VM mapping. We consider each VM as an individual entity in the auction and its bid corresponds directly to the number of resources requested.

For comparative evaluation of our proposed *CoMCLOUD* approach against the baselines, the total cost and average latency are considered as the performance metrics.

## 7.3 Experimental Results

The performance of *CoMCLOUD* is evaluated from three different perspectives: (1) ability to schedule different multi-tier

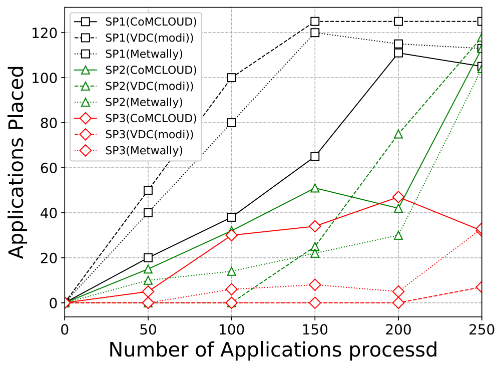


Fig. 4. Applications placed versus processed.

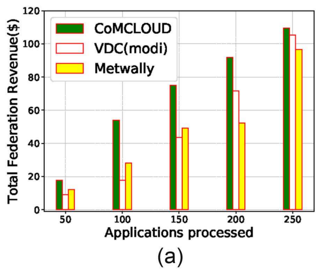
applications; (2) trade-off between the revenue and latency incurred in hosting multiple applications; and (3) necessity of parallel migration strategy under demand spikes and increased resource demands.

### 7.3.1 *CoMCLOUD Performances*

Fig. 4 shows the relationship between the number of applications placed versus the number of applications processed over the three SPs. It is observed from the plots that the majority of applications are hosted at $SP_1$ and $SP_2$ for both the baselines. This assignment can be attributed to the fact that $SP_1$ and $SP_2$ charge were less for different VM instance types as observed from Table 1. Though *CoMCLOUD* facilitates a more distributed placement of VMs, it generates a higher aggregate revenue in comparison with the baselines as can be inferred from Fig. 5a. However, the increased revenue comes at the cost of a marginal increase in average latency, as illustrated in Fig. 5b.

Moreover, Table 1 demonstrates that the difference in price for the same VM instance type at different geographical locations is less for $SP_1$ and $SP_2$ in comparison with $SP_3$, implying a large number of applications is hosted at $SP_1$ and $SP_2$. However, for the VDC(modi) baseline approach, a greedy selection of DCs based on a lower price, the DCs having lower costs are filled up first for hosting different applications. On the other hand, the Metwally baseline shows similar behavior as VDC(modi). This behavior is attributed to the fact that users set their bids (assuming truthful bidding) as per Table 1 and they get assigned to the DCs with minimum cost till the capacity is not exhausted.

*CoMCLOUD* can achieve a more balanced placement of applications. Table 8 shows the average utilization of resources (CPU and memory) for four different DCs at each of the three SPs. The utilization of resources is highlighted in Table 8. We



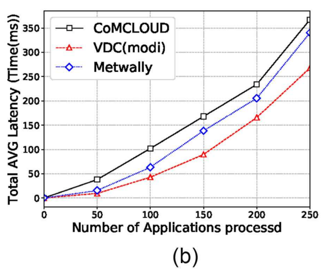


Fig. 5. (a) Total revenue($) versus Applications processed; (b) Total average latency versus applications processed.

TABLE 7
Comparison With Optimal Solution

| | Total Revenue ($) | | | Total Avg Latency (ms) | | |
|---|---|---|---|---|---|---|
| # Applications | 10 | 20 | 30 | 10 | 20 | 30 |
| CoMCLOUD Optimal | 6.14 | 12.43 | 18.81 | 2.44 | 4.89 | 7.33 |
| *CoMCLOUD* | 4.14 | 7.13 | 11.97 | 7.57 | 15.37 | 21.72 |
| VDC(modi) | 1.88 | 3.56 | 5.81 | 1.89 | 3.79 | 6.17 |
| Metwally | 2.45 | 4.91 | 7.36 | 3.1 | 6.19 | 9.23 |

can observe that *CoMCLOUD* achieves a more load-balanced placement as compared to the baseline algorithms, which suffer in starvation (highlighted in red) during load distribution.

Finally, we also compare *CoMCLOUD*'s meta-heuristic solution and other baseline algorithms with an exhaustive search based optimal solution implementation of *CoMCLOUD* (source code :[39]) for small test cases. The results in Table 7 depicts the total federation revenue, and total average latency for hosting 10, 20, and 30 multi-tier application requests. *CoMCLOUD* achieves near-optimal revenue compared to all other baseline techniques. However, considering the total average latency *CoMCLOUD* suffers. This behavior is because for the small test cases considered, both baseline algorithms, i.e., VDC(modi) and Metwally, place the VMs of multi-tier applications on the same DCs, leading to reduced communication latency while hampering revenue. The difference between the latency subsequently decreases for test cases with more applications as the DCs have limited resources, thereby forcing a dispersed allocation, and this behavior is captured in Fig. 5b.

### 7.3.2 *Revenue and Latency Trade-Off*

Next, we analyze the revenue and the QoS (here, average latency). The overall performance of *CoMCLOUD* in comparison with the baselines is displayed in Fig. 5. In particular, Figs. 5a and 5b show the overall revenue and average latency across the DCs for hosting 50-250 applications obtained by all three SPs. We observe that the overall revenue is higher for the proposed *CoMCLOUD* model, but the latency suffers slightly as compared to the baselines. However, the latency increases as the VMs are dispersed to minimize the aggregate cost. The VDC(modi) and Metwally approach end up placing the VMs on the same DCs. Thus, the aggregate latency remains lower as compared to *CoMCLOUD*, similar to the baselines.

As discussed earlier, *CoMCLOUD* maintains a trade-off between the revenue and latency, providing a Pareto-optimal solution. Figs. 6a and 6b are Pareto graphs for hosting 50 and 100 applications, respectively, on the cloud. It shows the trade-off between the latency and revenue (aggregate cost) incurred for hosting 50 and 150 applications.

Finally, we observe the rate of convergence of *CoMCLOUD* in Fig. 7, which shows the efficiency concerning the number of iterations. Efficiency is calculated by combining the effects of the cost and latency for placing 50, 150, and 250 applications. This combination is a linear summation of the reciprocal of aggregate cost and aggregate latency for hosting such applications. We observe from the plot that the convergence rate of *CoMCLOUD* is good, and it quickly converges to an optimal solution (close to 375 iterations). Moreover, Figs. 4 and 5a demonstrate that although maximum number of applications have been hosted on $SP_1$, the

TABLE 8
Utilization (%) in Different DCs

| APPs | DC1 | | | DC2 | | | DC3 | | | DC4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Metwally | VDC(modi) | CoMCLOUD | Metwally | VDC(modi) | CoMCLOUD | Metwally | VDC(modi) | CoMCLOUD | Metwally | VDC(modi) | CoMCLOUD |
| | SP1 | | | | | | | | | | | |
| 50 | 92.66 | 98.37 | 21.34 | 54.59 | 55.33 | 19.74 | 9.66 | 0 | 13.24 | 0 | 0 | 15.67 |
| 100 | 89.37 | 97.88 | 3.58 | 69.37 | 96.36 | 30.86 | 57.36 | 94.67 | 21.65 | 2.16 | 12.05 | 18.37 |
| 150 | 94.33 | 99.12 | 51.56 | 87.39 | 98.56 | 48.36 | 84.14 | 97.68 | 36.89 | 75.63 | 98.33 | 31.88 |
| 200 | 93.58 | 98.65 | 79.89 | 91.25 | 98.77 | 75.68 | 88.7 | 99.01 | 58.36 | 78.7 | 95.65 | 44.38 |
| 250 | 95.46 | 99.1 | 88.25 | 93.66 | 96.11 | 78.36 | 88.9 | 98.94 | 66.58 | 89.7 | 97.66 | 66 |
| | SP2 | | | | | | | | | | | |
| 50 | 36.55 | 0 | 16.35 | 11.25 | 0 | 14.56 | 2.52 | 0 | 9.65 | 0 | 0 | 8.69 |
| 100 | 18.36 | 0 | 26.57 | 10.32 | 0 | 25.66 | 5.7 | 0 | 18.59 | 0 | 0 | 12 |
| 150 | 38.369 | 96.38 | 28.65 | 33.78 | 0 | 23.54 | 26.38 | 0 | 18 | 17.66 | 0 | 11 |
| 200 | 47.19 | 97.21 | 32.35 | 26.36 | 98.36 | 28.36 | 36.59 | 33.148 | 22 | 14.62 | 0 | 16.36 |
| 250 | 96.53 | 98.67 | 90.6 | 90.16 | 97.78 | 80.7 | 82.47 | 98.82 | 76.3 | 87.16 | 98.33 | 65.36 |
| | SP3 | | | | | | | | | | | |
| 50 | 0 | 0 | 11.52 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| 100 | 17.7 | 0 | 28.89 | 11.57 | 0 | 21 | 0 | 0 | 19.65 | 0 | 0 | 18.66 |
| 150 | 31.85 | 0 | 27.66 | 17.36 | 0 | 19.98 | 11.29 | 0 | 18.67 | 6.52 | 0 | 16 |
| 200 | 18.63 | 0 | 38.33 | 15.37 | 0 | 31.25 | 5.23 | 0 | 25.26 | 11.7 | 0 | 22 |
| 250 | 56.95 | 26.45 | 28.23 | 45.29 | 0 | 22.58 | 36.89 | 0 | 21.6 | 24.92 | 0 | 11 |

aggregate revenue is actually less. It implies that maximum VMs hosted on $SP_1$ incur a lesser cost.

### 7.3.3 Migration Performance

Figs. 8a and 8b show the migration time and downtime, respectively, for migrating overloaded VMs for different number of applications. We use serial and parallel migration strategies for VMs associated with a given application. Migrating VMs implies finding a suitable path from the source to the destination DC, shared by the migrating VMs belonging to the same application. Serial migration means each VM will individually get the total share of the residual bandwidth for migration. In contrast, in parallel migration, the residual bandwidth is shared among all migrating VMs of the application. A parallel migration strategy is more suitable for multi-tier applications as it incurs minimum service disruption (in terms of downtime) [24].

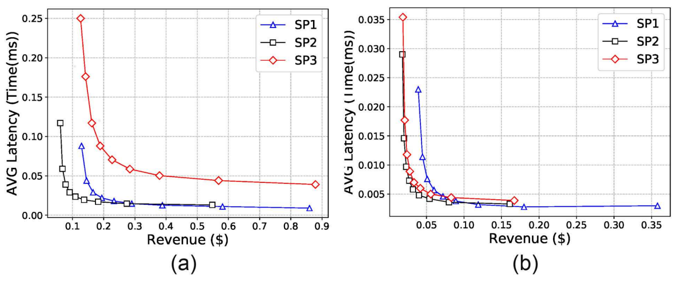


Fig. 6. (a) Average latency *versus* revenue for 50 apps; (b) Average latency *versus* revenue for 100 apps.

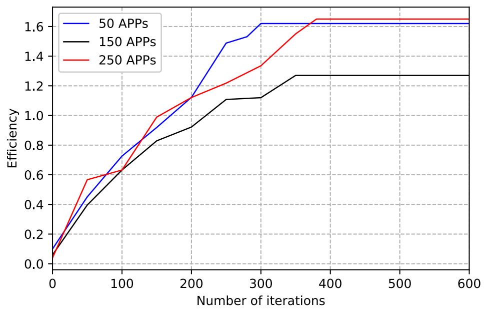


Fig. 7. Convergence rate.

It is observed from Fig. 8b that the average downtime is high for serial migration due to fewer pre-copy iterations, which is observed from Fig. 9. It leads to a higher amount of memory copied in the stop-and-copy phase, increasing the downtime. Furthermore, Fig. 8 exhibits that the increase in the downtime for serial migration is so high that it negatively impacts the average migration time.

## 8 Discussion

It is evident that *CoMCLOUD* successfully implements all key functionalities required in a broker-based multi-cloud environment with some specialized features. In this section, the salient features of the proposed *CoMCLOUD* architecture are presented concisely.

*(1) Reduced Dependency on CB.* To maintain the trust, the CB in *CoMCLOUD* does not actively take part in the resource assignment. In contrast, in a typical broker-based multi-cloud model, all placement decisions are taken by the CB. In *CoMCLOUD* the CB only takes care of a few things that include: maintaining log register for SPs, initiating resource reservation process, and acting as a gateway for in/out bound application requests.

*(2) Geo-Separated VM Coalition.* In contrast to traditional broker based multi-cloud model, *CoMCLOUD* allows SPs to form coalition to host inter dependent VMs for servicing a multi-tier application request in a Geo-distributed setting. Geo-separated DCs across the globe for an SP supports this VM coalition. This unique feature of *CoMCLOUD* helps in achieving a load-balanced architecture for an SP. This feature

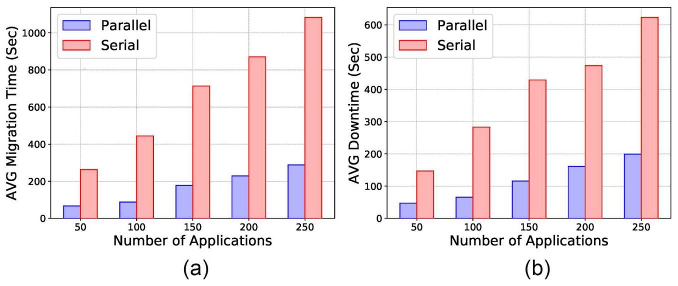


Fig. 8. (a) Migration time for migrating different number of applications; (b) Downtime for migrating different number of applications.

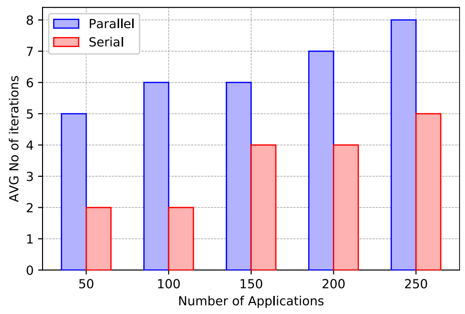


Fig. 9. Number of iterations for different number of applications.

also prevents a single point failure for a multi-tier application requests.

*(3) Democracy of SPs.* In *CoMCLOUD*, as dependency on CB is reduced, democracy among SPs is increased. The SPs can independently calculate the VM coalition price, and placement decisions are at the SP's disposal. Each SP can use meta-heuristics based on ACO based on its parameter setting for this optimal decision. After deciding the optimal price and coalition, each SP can participate in a first-price sealed-bid auction game with their strategy (cost) to maximize their payoff.

*(4) Decentralized Bidding Mechanism.* Towards eliminating centralization and dependence on CB, a blockchain-based decentralized bidding mechanism can be implemented for the *Service Coalition Selection Game*. Blockchain-based techniques are effective in designing decentralized cloud federations [45], where the individual SPs carry out the placement decisions through smart contracts. The entire auction process of the *Service Coalition Selection Game* in *CoMCLOUD* can be carried over a blockchain network, in two phases: *sealed bidding phase*, and *revealing phase* through smart contracts as follows.

In the *sealed bidding* phase, each participating $SP_i$ submits its bid to the blockchain by submitting only the signed cryptographic hash of its actual bid value. Therefore, $SP_i$ submits $< Hash(\Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})) >_{\sigma_{SP_i}}$, where $Hash(.)$ denotes the cryptographic hash function and $<.>_{\sigma_{SP_i}}$ denotes the message signed by the private key of $SP_i$. Once committed, this signed hash cannot be changed due to the immutable property of the blockchain distributed ledger. By using cryptographic hash functions to hash the bids, we can ensure that the actual bid cannot be determined by any other SPs within the limited duration of the bidding phase. On the other hand, committing the bids on the blockchain ensures that these are public to all the participants and cannot be changed in future. Lastly, the signature is required for authentication and non-repudiation.

In the *revealing* phase, each $SP_i$ reveals its bid by posting the actual bid value $<\Psi(\mathbb{Q}_{\mathcal{U}_q}^{SP_i})>_{\sigma_{SP_i}}$ on the blockchain. This bid is only valid if its hash matches the hash of the bid posted in the bidding phase. Thus, as all the SPs reveal their bids, each participant can determine the auction winner individually and then come to a consensus on the same. If some SP refuses to reveal its actual bid, it is simply ignored, implying it cannot be the winner.

The SP that bids the highest wins the auction and hosts the multi-tier application. The entire bidding mechanism can be implemented as a permissioned blockchain using smart contracts. Since the system is a closed one comprising only SPs in a federation, a permissioned blockchain is preferred as it offers higher throughput and lower transaction commitment latency [46]. The bidding protocol can be implemented using Hyperledger Fabric platform [31] with chaincodes (smart contracts) for (*i.*) initializing an auction, (*ii.*) bidding on an existing auction, (*iii.*) revealing bids, (*iv.*) computing validating results of an auction. Every auction, when initiated, will have its *initiation time* and *auction duration* (a fixed parameter) within which all bids have to be placed. All transactions (i.e., initiate, bid, reveal) will go through Hyperledger Fabric's endorsement policy-based consensus protocol having endorsements of the majority of the participants in a federation. This ensures the immutable property of the blockchain, which is essential for the bidding mechanism.

*(5) Migration to Handle a Sudden Load Spikes.* As all coalitions in *CoMCLOUD* host multiple dependent VMs which are often exposed to demand spikes, hardware failures etc. Therefore in any such scenario a migration protocol is put in place that help is seamless movement of a VM from one server/DC to another server/DC within a SP. These movements can have deleterious impacts on the service availability, hence a live migration based parallel migration is used in *CoMCLOUD*, to reduce migration overheads and provide services with minimal interruption.

*(6) Performance Comparison.* To sum up, we discuss the performance of *CoMCLOUD* against the baselines considering CPU and memory intensiveness. With regards to CPU usage, Sun *et al.* [24] consumes the maximum CPU as it performs exhaustive comparisons. However, comparing *CoMCLOUD* and Metwally *et al.* [1], *CoMCLOUD* consumes more CPU resources as its based on ACO meta-heuristic which involves more computations in comparison to the double auction game. Considering memory usage the relationship is similar as the approach of Sun *et al.* involves the exhaustive storage of all solutions, which are later compared to obtain the best solution. In *CoMCLOUD*, we use a separate construction graph that adds to the memory used. *CoMCLOUD* consumes more CPU and memory resources in comparison to Metwally *et al.* [1]. Although *CoMCLOUD* performs better than Metwally *et al.* considering revenue but suffers from higher latency in hosting multi-tier applications.

*(7) Asymptotic Analysis.* Here, we discuss the asymptotic bounds of techniques used for comparison.

(i.) *CoMCLOUD*: The running time of *CoMCLOUD* comprises the meta-heuristic ant-colony-optimization (ACO) local to individual SPs followed by a centralized bidding process at the broker. Computing the asymptotic time complexity of ACO is *extremely challenging* and is still an *open problem* as shown by the findings of [47] [48]. Although efforts have been made to estimate the time complexity of ACO, they roughly approximate the behavior of ACO and are not applicable for all scenarios. Moreover, different variants of ACO adds to the complexity of such analysis. The closest finding that resembles the overall working of the ACO meta-heuristic in *CoMCLOUD* is provided by [49]. The time complexity of ACO is dependent on the number of iterations executed to reach a sub-optimal solution. It is computed to be in the $O(\frac{1}{\rho} * mn^2 * logn))$, where $\rho$ is the evaporation rate, $n$ and $m$ are the number of vertices and edges of a directed-cyclic graph (DAG) [49]. Note that we exclude the impact of the heuristic information in the analysis as it makes it even more challenging. Considering *CoMCLOUD*, the overall problem of placing a multi-tier application is represented in the form of a DAG

$G^{Con}(N^{Con}, L^{Con})$ (refer to Fig. 3 at Page-6). The number of vertices of $G^{Con}$ is computed to be $p * q + 2$, where $p$ is the average number of data centers (DCs) at a service provider (SP), and $q$ is the average number of virtual machines (VMs) corresponding to a multi-tier application. Two additional vertices are added depicting the start and endpoints. The number of edges at the first level and the last level connecting the start vertex and end vertex is $p$, whereas, at the intermediate levels, the number of links at each level is $p^2$. Hence, the total number of edges in the graph is $2p + (q-1)p^2$. Therefore, the total time in reaching a sub-optimal assignment for a multi-tier application given the eligibility of $l$ SPs to host the given application is $O(l * \frac{1}{\rho} * 2p + (q-1)p^2 * (qp)^2 * log(qp))$. Therefore, for $n$ number of multi-tier applications the total asymptotic time complexity of ACO is $O(n * l * \frac{1}{\rho} * 2p + (q-1)p^2 * (qp)^2 * log(qp))$. Followed by ACO, a first-price-sealed bid auction is conducted in $O(n)$ time. Therefore, the overall time complexity of *CoMCLOUD* is $O(n * l * \frac{1}{\rho} * 2p + (q-1)p^2 * (qp)^2 * log(qp)) + O(n)$.

(ii.) *Metwally et al.* [1]: The asymptotic time complexity is $O(n * k)$ + $O(n)$, where $n$ is the total number of multi-tier applications and $k$ is the number of SPs. The first term depicts the time complexity of performing a match between the application and the regional coordinator (RC) using a double auction. The second term depicts the time incurred in matching rejected applications from the first step using a single auction.

(iii.) *VDC (modi)* [24]: It has a time complexity $O(n * k!)$, where $n$ is the number of multi-tier applications and $k$ is the number of DCs. The mapping procedure for each multi-tier application consumes $O(k!)$, as it exhaustively selects the least cost mapping by considering all possible mapping of VMs onto DCs.

## 9 Conclusion

In this paper, we proposed a framework called *CoMCLOUD* for multi-tier VM placements in a multi-cloud federation. Each eligible SP locally computes the most optimal coalition of VMs using an ant colony optimization (ACO) meta-heuristic that optimizes two distinct user-centric parameters, i.e., hosting cost and inter-VM communication latency. Simulation results demonstrate that the proposed approach can generate higher federation revenue and achieve a decent trade-off between the cost of hosting an application and its QoS in terms of latency. Additionally in *CoMCLOUD*, we implemented a parallel migration strategy to migrate overloaded VMs of multi-tier applications with minimum impact on the services in terms of degradation time.

As a future work, we plan to extend the model to host multi-tier applications in a federated SP environment. Moreover, to eliminate the pitfalls of a centralized bidding architecture, we also aim to extend the bidding model to a decentralized blockchain-based secure framework.

## References

[1] K. Metwally, A. Jarray, and A. Karmouch, “A distributed auction-based framework for scalable IaaS provisioning in geo-data centers,” *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 647–659, Third Quarter 2018.

[2] M. Tortonesi and L. Foschini, “Business-driven service placement for highly dynamic and distributed cloud systems,” *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 977–990, Fourth Quarter 2018.

[3] A. Amokrane, R. Langar, M. F. Zhani, R. Boutaba, and G. Pujolle, “Greenslater: On satisfying green SLAs in distributed clouds,” *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 3, pp. 363–376, Sep. 2015.

[4] Onapp-Federation. Accessed: Aug. 4, 2021. [Online]. Available: https://onapp.com/federation

[5] N. Samaan, “A novel economic sharing model in a federation of selfish cloud providers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 12–21, Jan. 2014.

[6] L. Mashayekhy, M. M. Nejad, and D. Grosu, “Cloud federations in the sky: Formation game and mechanism,” *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 14–27, First Quarter 2015.

[7] H. Li, C. Wu, Z. Li, and F. C. M. Lau, “Virtual machine trading in a federation of clouds: Individual profit and social welfare maximization,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1827–1840, Jun. 2016.

[8] L. Mashayekhy, M. M. Nejad, and D. Grosu, “A trust-aware mechanism for cloud federation formation,” *IEEE Trans. Cloud Comput.*, 2019, doi: 10.1109/TCC.2019.2911831.

[9] U. Ahmed, I. Raza, and S. A. Hussain, “Trust evaluation in cross-cloud federation: Survey and requirement analysis,” *ACM Comput. Surv.*, vol. 52, no. 1, 2019, Art. no. 19.

[10] Arjuna’s Agility framework. Accessed: Aug. 4, 2021. [Online]. Available: http://www.arjuna.com/federation

[11] J. Mei, K. Li, Z. Tong, Q. Li, and K. Li, “Profit maximization for cloud brokers in cloud computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 190–203, Jan. 2018.

[12] H. Liu and B. He, “VMbuddies: Coordinating live migration of multi-tier applications in cloud environments,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 1192–1205, Apr. 2015.

[13] A. Atrey, G. V. Seghbroeck, H. Mora, F. D. Turck, and B. Volckaert, “SpeCH: A scalable framework for data placement of data-intensive services in geo-distributed clouds,” *J. Netw. Comput. Appl.*, vol. 142, pp. 1–14, 2019.

[14] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, “Application-aware virtual machine migration in data centers,” in *Proc. IEEE INFOCOM*, 2011, pp. 66–70.

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Softw. Pract. Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[16] S. K. Addya, A. Satpathy, S. Chakraborty, and S. K. Ghosh, “Optimal VM coalition for multi-tier applications over multi-cloud broker environments,” in *Proc. 11th Int. Conf. Commun. Syst. Netw.*, 2019, pp. 141–148.

[17] K. Konstanteli, T. Cucinotta, K. Psychas, and T. A. Varvarigou, “Elastic admission control for federated cloud services,” *IEEE Trans. Cloud Comput.*, vol. 2, no. 3, pp. 348–361, Third Quarter 2014.

[18] A. Rubio-Montero, E. Huedo, and R. Mayo-García, “Scheduling multiple virtual environments in cloud federations for distributed calculations,” *Future Gener. Comput. Syst.*, vol. 74, pp. 90–103, 2016.

[19] P. S. Pillai and S. Rao, “Resource allocation in cloud computing using the uncertainty principle of game theory,” *IEEE Syst. J.*, vol. 10, no. 2, pp. 637–648, Jun. 2016.

[20] A. Quarati, A. Clematis, A. Galizia, and D. D’Agostino, “Hybrid clouds brokering: Business opportunities, QoS and energy-saving issues,” *Simul. Model. Pract. Theory*, vol. 39, pp. 121–134, 2013.

[21] S. Nesmachnow, S. Iturriaga, and B. Dorronsoro, “Efficient heuristics for profit optimization of virtual cloud brokers,” *IEEE Comput. Intell. Mag.*, vol. 10, no. 1, pp. 33–43, Feb. 2015.

[22] F. Larumbe and B. Sansò, “Green cloud broker: On-line dynamic virtual machine placement across multiple cloud providers,” in *Proc. 5th IEEE Int. Conf. Cloud Netw.*, 2016, pp. 119–125.

[23] J. Panneerselvam, L. Liu, N. Antonopoulos, and Y. Bo, “Workload analysis for the scope of user demand prediction model evaluations in cloud environments,” in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput.*, 2014, pp. 883–889.

[24] G. Sun, D. Liao, D. Zhao, Z. Xu, and H. Yu, “Live migration for multiple correlated virtual machines in cloud-based data centers,” *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 279–291, Mar./Apr. 2018.

[25] J. N. Khasnabish, M. F. Mithani, and S. Rao, “Tier-centric resource allocation in multi-tier cloud systems,” *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 576–589, Third Quarter 2017.

[26] “Bin-Packing,” *Comb. Optim.: Theory Algorithms*, Berlin, Heidelberg: Springer, 2006, pp. 426–441. [Online]. Available: https://doi.org/10.1007/3-540-29297-7_18, doi: 10.1007/3-540-29297-7_18.

[27] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: Optimization by a colony of cooperating agents,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.
[28] R. Gibbons, “A primer in game theory,” Birmingham, U.K.: Harvester Wheatsheaf, 1992.
[29] C. A. Holt Jr, “Competitive bidding for contracts under alternative auction procedures,” *J. Political Economy*, vol. 88, no. 3, pp. 433–445, 1980.
[30] K. R. Apt, “A primer on strategic games,” *CoRR*, 2011. [Online]. Available: http://arxiv.org/abs/1102.0203
[31] E. Androulaki *et al.*, “Hyperledger fabric: A distributed operating system for permissioned blockchains,” in *Proc. 13th EuroSys Conf.*, 2018, pp. 1–15. [Online]. Available: https://doi.org/10.1145/3190508.3190538
[32] K. R. Apt. Lecture notes: Strategic games. Accessed: Aug. 4, 2021. [Online]. Available: https://homepages.cwi.nl/~apt/stra/
[33] Y. S. Patel, A. Page, M. Nagdev, A. Choubey, R. Misra, and S. K. Das, “On demand clock synchronization for live VM migration in distributed cloud data centers,” *J. Parallel Distrib. Comput.*, vol. 138, pp. 15–31, 2020.
[34] W. Zhang, S. Han, H. He, and H. Chen, “Network-aware virtual machine migration in an overcommitted cloud,” *Future Gener. Comput. Syst.*, vol. 76, pp. 428–442, 2017.
[35] U. Mandal, P. Chowdhury, M. Tornatore, C. U. Martel, and B. Mukherjee, “Bandwidth provisioning for virtual machine migration in cloud: Strategy and application,” *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 967–976, Fourth Quarter 2018.
[36] H. Wang, Y. Li, Y. Zhang, and D. Jin, “Virtual machine migration planning in software-defined networks,” *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 1168–1182, Fourth Quarter 2018.
[37] F. Callegati and W. Cerroni, “Live migration of virtualized edge networks: Analytical modeling and performance evaluation,” in *Proc. IEEE SDN Future Netw. Services*, 2013, pp. 1–6.
[38] M. F. Bari, M. F. Zhani, Q. Zhang, R. Ahmed, and R. Boutaba, “CQNCR: Optimal VM migration planning in cloud data centers,” in *Proc. IFIP Netw. Conf.*, 2014, pp. 1–9.
[39] Comcloud source code. Accessed: Aug. 4, 2021. [Online]. Available: https://github.com/souravaddya/CoMCLOUD
[40] Amazon EC2, VM Configuration and Pricing. Sep. 2018. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/
[41] Microsoft Azure, VM Configuration and Pricing, Sep. 2018. [Online]. Available: https://azure.microsoft.com/en-in/pricing/details/virtual-machines/linux/
[42] Google cloud, VM Configuration and Pricing, Sep. 2018. [Online]. Available: https://cloud.google.com/compute/pricing
[43] Accessed: Aug. 4, 2021. [Online]. Available: https://www.cs.bu.edu/brite/
[44] J. Sztrik *et al.*, “Basic queueing theory,” *Univ. Debrecen, Fac. Inform.*, vol. 193, pp. 60–67, 2012.
[45] B. C. Ghosh, T. Bhartia, S. K. Addya, and S. Chakraborty, “Leveraging public-private blockchain interoperability for closed consortium interfacing,” in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
[46] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, “Performance analysis of private blockchain platforms in varying workloads,” in *Proc. 26th Int. Conf. Comput. Commun. Netw*, 2017, pp. 1–6.
[47] F. Neumann and C. Witt, “Runtime analysis of a simple ant colony optimization algorithm,” in *Proc. Int. Symp. Algorithms Comput.*, 2006, pp. 618–627.
[48] Y. Zhou, “Runtime analysis of an ant colony optimization algorithm for tsp instances,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1083–1092, Oct. 2009.
[49] N. Attiratanasunthron and J. Fakcharoenphol, “A running time analysis of an ant colony optimization algorithm for shortest paths in directed acyclic graphs,” *Inf. Process. Lett.*, vol. 105, no. 3, pp. 88–92, 2008.

**Sourav Kanti Addya** (Senior Member, IEEE) received the PhD degree in CSE from NIT Rourkela, India and worked as a postdoctoral fellow with the Department of CSE, IIT Kharagpur, India. He is currently an assistant professor with the Department of CSE, NITK Surathkal, India. His research interests include Cloud System, Serverless computing, IoT, Blockchain. He is a Member of the ACM.

**Anurag Satpathy** (Member, IEEE) received the BTech degree in information technology from IIIT Bhubaneswar, India, in 2014, and MTech degree in CSE from BIT Mesra, Ranchi, India, in 2017. He is currently a research scholar with the Department of Computer Science and Engineering, NIT Rourkela, India. His research interests includes cloud computing, Internet of Things, and distributed systems.

**Bishakh Chandra Ghosh** (Student Member, IEEE) received the BTech degree in information technology from NIT Durgapur, India. He is a doctoral research student in CSE at IIT Kharagpur. His research interests includes cloud computing, blockchain and distributed systems.

**Sandip Chakraborty** (Member, IEEE) received the PhD degree in CSE from IIT Guwahati. Currently, he is an associate professor in CSE with IIT Kharagpur. He is working as an area editor of Elsevier’s Ad Hoc Networks and Pervasive and Mobile Computing journals. His research interests include the intersections of computer systems, distributed systems and human conputer interaction.

**Soumya K. Ghosh** (Senior Member, IEEE) received the PhD degree in CSE from the IIT, Kharagpur. Currently, he is currently a professor with the Department of CSE, IIT Kharagpur. Prior to joining IIT Kharagpur, he was at the ISRO in the area of satellite remote sensing and GIS. His research interests include cloud computing, spatial data science, and IoT.

**Sajal K. Das** (Fellow, IEEE) is currently a professor of computer science and daniel st. clair endowed chair with the Missouri University of Science and Technology, Rolla. His research interests include wireless and sensor networks, mobile and pervasive computing, cyber-physical systems and IoT, smart environments (smart city, smary grid, smart health, smart agriculture), parallel and cloud computing, mobile crowdsensing, cybersecurity, biological and social networks, and applied game theory. He has published extensively in these areas. He is the editor-in-chief or Elsevier’s Pervasive and Mobile Computing journal, and associate editor of *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Dependable and Secure Computing*, and *ACM Transactions on Sensor Networks*, among others.