A Bayesian game-enhanced auction model for federated cloud services using blockchain

Shi, Zeshun; Zhou, H.; de Laat, C.; Zhao, Z.

DOI
10.1016/j.future.2022.05.017

Publication date
2022

Document Version
Final published version

Published in
Future Generation Computer Systems

Future Generation Computer Systems 136 (2022) 49–66

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A Bayesian game-enhanced auction model for federated cloud services using blockchain

Zeshun Shi [a], Huan Zhou [b,*], Cees de Laat [a], Zhiming Zhao [a,*]

[a] Informatics Institute, University of Amsterdam, Amsterdam, 1098 XH, The Netherlands
[b] School of Computer Science, National University of Defense Technology, Changsha, 410073, China

ARTICLE INFO

Article history:
Received 13 November 2021
Received in revised form 18 May 2022
Accepted 20 May 2022
Available online 28 May 2022

Keywords:
Blockchain
Auction
Federated cloud service
Quality of service
Game theory
Trustworthy

ABSTRACT

Industrial applications often require federated cloud services from multiple providers to improve reliability and flexibility. Traditional selection methods through auctions usually involve a centralized auctioneer to coordinate the auction procedure. Blockchain and smart contracts provide a decentralized mechanism to automate the cloud auction process; however, existing solutions fail in the selection of the most suitable providers and the violation detection of the signed auction agreements, which are also known as service-level agreements (SLAs). To tackle these problems, we propose an integrated auction model using Bayesian game theory and blockchain techniques. The proposed model is enhanced with two Bayesian Nash Equilibriums (BNEs); the first BNE enables the selection of cost-effective providers to construct the federated cloud services, while the second BNE ensures consistent and trustworthy monitoring of federated SLAs. Moreover, a timed message submission (TMS) algorithm is proposed to protect the auction privacy during the message submission phase. This paper validates the equilibrium results of two BNEs and implements the proposed model on the Ethereum blockchain. The analytical and experimental results demonstrate the feasibility, trustworthiness, and cost-effectiveness of our model.

1. Introduction

Cloud computing provides on-demand elastic IT services via the Internet and is playing an increasingly important role in the development and operations of business applications. When an application is highly distributed or has high requirements for data availability, a single service provider is often not sufficient. This leads to a situation where multiple service providers may form a group, namely a cloud federation, to provide services together [1,2]. Compared with traditional centralized cloud services, this new cloud paradigm has many benefits. For example, a cloud federation allows customers to choose the best combination of providers considering service flexibility, availability, and cost to meet their specific business or technical requirements. It also enables IT companies to distribute workloads on a global scale and allows users to access cloud resources in a safe and reliable way.

Since providers on the market are diverse in terms of service quality, price, and reputation, it is usually difficult and time-consuming to choose suitable providers that meet specific application requirements. An auction is one of the effective and fair solutions to this problem. It is a sale activity in which potential buyers make competitive bids for assets or services [3]. In the field of cloud computing, the auction-based pricing strategy can effectively reflect potential trends in cloud resource demand and supply. Thus it is an effective way to allocate resources and satisfy both buyers and sellers [4]. Auction-based cloud pricing strategies have been developed rapidly in the past few years. Large cloud service providers (e.g., AWS, Azure, and Google) have supported spot instance pricing for users to bid for unused capacity in a cloud data center. Some users can even save up to 90% of the cost compared with the traditional on-demand instance pricing.[1] However, it is still challenging to apply auction models in a federated cloud service scenario, which is mainly because:

- **There is a lack of a trustworthy platform** for users to auction cloud services from multiple providers. Most existing cloud auction solutions have the vendor lock-in issue; the provider also acts as an auctioneer, which may lead to bias and untrustworthiness.
- **There is a lack of an automated and cost-effective mechanism** to enforce the service lifecycle from auction agreement generation to service delivery; traditional auction houses or auctioneers are cumbersome and expensive.

* Corresponding authors.
E-mail addresses: z.shi2@uva.nl (Z. Shi), huanzhou@nudt.edu.cn (H. Zhou), delaat@uva.nl (C. de Laat), z.zhao@uva.nl (Z. Zhao).

[1] https://aws.amazon.com/ec2/spot/.

https://doi.org/10.1016/j.future.2022.05.017

- **There is a lack of a fair mechanism to detect the auction agreement violation** (e.g., the cloud service is not delivered as agreed) without bias. The provider has more power in the current model to verify service violations and decide whether to compensate the customer.

In recent years, blockchain has attracted tremendous attention as an enabling technology for building decentralized systems. In general, blockchain is a decentralized ledger system that combines existing technologies such as distributed data storage, peer-to-peer networking, consensus mechanisms, and cryptographic algorithms. The ledger is maintained by all nodes participating in the system and is therefore decentralized, tamper-proof, transparent, and secure [5]. Blockchain was originally introduced as the underlying technology for Bitcoin [6]. Now, with smart contract technology bringing powerful programmability, it is widely believed that blockchain can be applied to build decentralized systems in various application scenarios, e.g., healthcare, finance, energy trading, wireless communication, service allocation, electronic voting, and supply chain management [7–10].

Blockchain and smart contract technologies bring new opportunities to deal with the challenges of federated cloud auctions. Due to its immutable and verifiable properties, blockchain has proven to be a promising tool for auction usage without requiring a centralized trusted third party (TTP) [11,12]. In 2018, for the first time in the world, million-dollar artworks from Andy Warhol had been tokenized and auctioned on the blockchain.[2] This mechanism of bidding on item ownership with cryptocurrencies and smart contracts might become a future trend. However, since most auction objects are not digital, blockchain and smart contracts cannot handle their ownership directly. The trustworthy enforcement of auction agreements still remains a challenge.

In this paper, an integrated auction model using Bayesian game theory and blockchain is proposed for federated cloud services. Specifically, blockchain and smart contracts are leveraged to build a decentralized, secure, and trustworthy auction platform. Bayesian games are leveraged to model incomplete information sharing among different auction participants, and to enhance the effectiveness and trustworthiness of the cloud auction. Our model considers both the effective bidding and the trustworthy enforcement of the auction agreements. In brief, the main contributions of this paper are summarized as follows:

- A novel blockchain-based federated cloud auction model where a new role, called auction witness, is introduced to support the monitoring of service delivery.
- Two unique Bayesian Nash Equilibriums (BNEs) are derived to select cost-effective providers, and to monitor federated service-level agreements (SLAs) in a consistent and trustworthy way.
- A timed message submission (TMS) algorithm is designed to protect auction privacy during the message submission phase.
- A prototype system based on the Ethereum blockchain is fully developed and tested.[3]

The remainder of the paper is organized as follows. Section 2 briefly reviews the related works on blockchain-based cloud auction solutions. Section 3 introduces the overall architecture and process of the proposed federated cloud auction model. Section 4 dives into the key techniques, including a federated cloud partition model, two Bayesian games with the analysis of BNEs, and the TMS algorithm. Section 5 presents the evaluation and implementation details. Section 6 discusses the design choices and concerns of the proposed model. Finally, the paper is concluded in Section 7.

2. Related work

Auctions have attracted research attention in many fields due to their effective and fair transaction properties. Classic auction models, e.g., double auction, combinatorial auction, and Vickrey–Clarke–Groves (VCG) auction, have been widely leveraged to optimize cloud resource allocation in a federated cloud environment [13–16]. From a practical point of view, most of the existing solutions are centralized and need to rely on an auctioneer to manage the auction process. However, in practice, an auctioneer is not always trustworthy and can conspire with any auction parties [17]. The high commission fees also limit the promotion of the above auction models in real industries. Recently, blockchain has been demonstrated to be a promising tool for decentralized auctions, which is mainly because: (1) every transaction executed on the blockchain is public, verifiable, and immutable. Therefore blockchain can be leveraged as an audit device, and participants cannot cheat during the auction process; (2) there is no need for a specific auctioneer to manage the auction due to the help of smart contracts, which greatly reduces the auction cost [11]; and (3) blockchain cryptocurrencies can be used to complete auction payments due to their simple and secure transaction properties. Recent studies [17,18] have demonstrated the great potential of using blockchain as decentralized auction platforms.

There are already several frameworks that target cloud services and resources allocation using blockchain and auction models. For example, DeCloud [19] is a secure and decentralized auction system specifically built for edge/cloud service trading. It integrates a truthful double auction model and a bidding language to match highly heterogeneous edge resources with different service requests. CloudAgora [20] is a platform that enables low-cost storage and computing access based on blockchain and auctions, where prices are determined through an auction game. ChainFaaS [21] is designed to run serverless tasks using the computing power of personal computers. AStERISK [22] is a sealed-bid auction platform on the blockchain. It automatically determines the best price for cloud services and assigns customers to the most suitable provider. Similarly, Chen et al. [23] introduced a blockchain-based auction and trading model for cloud virtual machine (VM) allocation. Their model can achieve fairness in auction transactions by implementing commitment-based state mechanisms, smart contracts, and cryptocurrency technologies. In [24], the authors paid attention to the cloud storage problem and proposed VCG auction-based resource trading models for distributed cloud storage. Besides, Debe et al. [25] demonstrated a blockchain-based reverse auction solution for public fog service allocation. Yu et al. [26] also leveraged the reverse auction model and presented a blockchain-based edge crowdsourcing service system. Specifically, a changeable auction algorithm is designed so that each request from the user will find a winner that can provide the appropriate edge service.

It should be noted that most of the above models focus on designing on-chain auction algorithms to improve allocation efficiency and privacy; unfortunately, none of them consider the execution of auction contracts. iExec [27] and Golem [28] are two popular projects that are closely related to our model. The former aims to build a blockchain-based distributed cloud environment, while the latter tries to build a worldwide supercomputer. They both propose that oracles are needed to define whether a transaction is successful and to trigger the execution of the smart contract when the conditions are met. However, most currently

[2] https://finance.yahoo.com/news/andy-warhol-multi-million-dollar-162928721.html.

[3] The code repository is open sourced in: https://github.com/ZeshunShi/SC4CloudAuction.

Table 1
Comparison with existing blockchain-based cloud auction models.

Ref.	Topic	Auction Algorithm	Auction Privacy	Auction Enforcement
DeCloud [19]	Edge/Cloud service trading	●	●	○
CloudAgora [20]	Cloud storage and computing sharing	●	●	○
ChainFaaS [21]	Serverless computing	◐	○	○
AStERISK [22]	Shared economy service allocation	●	●	○
Chen et al. [23]	Cloud VM allocation	●	●	○
Gu et al. [24]	Cloud storage resource trading	●	○	○
Debe et al. [25]	Fog service trading	●	○	○
Yu et al. [26]	Edge service crowdsensing	●	○	○
iExec [27]	Decentralized cloud computing	◐	●	◐
Golem [28]	Worldwide supercomputer	◐	●	◐
Our model	Federated cloud services trading	●	●	●

Notes: Filled (or half-filled) circles indicate that the properties are (partially) addressed, while empty circles mean that properties are not considered.

existing oracle services are provided by third-party companies (e.g., Chainlink, Provable, and Witnet) [29] and are therefore subject to single points of failure. To handle this issue, Matsushima and Noda [30] designed a self-monitoring mechanism to enforce digital court smart contracts. However, this mechanism may lead to biased results and lack the prototype system's verification. In our previous work [31], we introduced the idea of a decentralized monitoring mechanism for blockchain using the complete information game theory. However, the proposed model assumes that players possess full information about their opponents, which is unrealistic. Besides, the model can only implement naive incentives to determine whether the service is violated through a boolean value without the ability to monitor federated SLAs at the same time.

In summary, there is an urgent need to establish a secure, trustworthy, and cost-effective auction model in the federated cloud services scenario. Although blockchain-based decentralized auction models have great potential to tackle this problem, most existing solutions only focus on optimizing bidding processes regardless of the auction agreement enforcement. A comparison of our model with related studies is shown in Table 1. Our work is among the first to combine bidding and auction enforcement with blockchain, and enhance the auction effectiveness and trustworthiness through Bayesian game-based incentive mechanisms.

3. Federated cloud auction model

Based on current research gaps, we propose our federated cloud auction model in detail. In practice, cloud customers usually use cloud services from federated service providers to improve fault tolerance and reliability. There are many providers with similar functions on the market. In contrast to the spot instance pricing that requires customers to bid for resources, providers need to offer flexible pricing strategies and submit bids to sell their services in our model.

3.1. Architecture overview

Fig. 1 shows the architecture overview of the proposed federated cloud auction model. Generally, there are three roles involved: (1) a cloud customer who pays and consumes the federated cloud services. In the auction model, the customer works as a service purchaser and publishes auction invitations; (2) multiple cloud providers that collaborate to provide federated cloud services. They act as bidders in the auction model and fight for the right to sell services through bidding activities; and (3) a new role called auction witnesses is introduced to monitor the federated SLAs and ensure the successful delivery of the auctioned cloud services.[4] There are two types of smart contracts involved. The auction smart contract is the main contract to manage the entire auction process (e.g., setting parameters, auction rules, and execution orders). SLA smart contracts are sub-contracts generated by the auction contract to specify the service details between the customer and each provider. Here SLA smart contracts work as the auction agreements in traditional auction activities, and contain the terms and obligations for both sellers and buyers to finish the deal. Smart contracts can be used as audit evidence by both parties, making the service terms immutable, open, and transparent. Besides, there are two types of fees involved in the auction payment. Witness fees are leveraged to encourage potential blockchain users to join the monitoring activity. They also act as deposits to ensure that the customer and providers cannot easily deny their obligations. By contrast, service fees are the costs of providing cloud services, which are generated by the providers' bids and paid by the customer in the form of cryptocurrency. It should be noted that blockchain cannot act as an enforcement device unless the money (i.e., tokens) is transferred to the smart contract in advance. Therefore, these two types of fees require prepayments to enforce the money transfer at the right time.

3.2. Model process

In Fig. 1, the numbers next to process flows represent the sequence of the action. Different letters along with the same number (e.g., 1a and 1b) represent that those events can happen at the same time. The entire auction process can then be described as follows. First, an agent (usually the customer) deploys the auction smart contract on the blockchain. Before the auction starts, the customer needs to consider how many providers are needed to form the cloud federation. An off-chain federated cloud partition model is therefore used to accomplish this goal. After that, the customer can publish the bid invitation on the auction smart contract. This invitation contains a detailed description of the service requirement, e.g., provider numbers, VM specifications, uptime, throughput, and the reserve price. A certain amount of witness fees need to be prepaid at the same time. When providers notice the auction invitation posted by the customer, they can register as bidders to participate in the auction. Providers can start bidding when the number of registered providers meets the requirement within a specified time window. Similar to the customer, providers need to prepay part of the witness fee when submitting bids.[5] When the sum

[4] For simplicity, customer, provider, and witness are used in the following text.

[5] The total amount of the witness fee is shared by both the customer and selected providers.

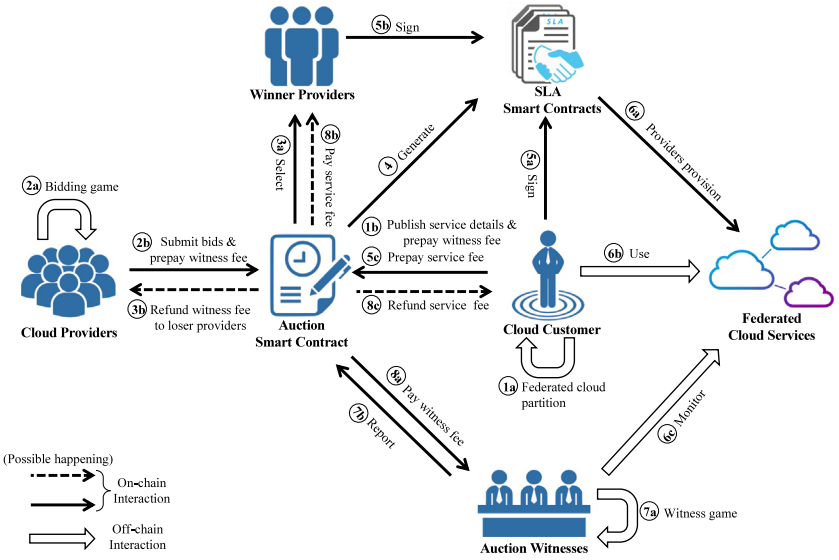


Fig. 1. Architecture overview of the federated cloud auction model.

of k (the number of providers required by the customer) bids is less than the customer's reserve price, the requirements from both parties are met. The auction smart contract then judges this auction as a valid auction and selects k providers with the lowest bids as the winning providers. Otherwise, the auction is invalid (the customer's reserve price is too low or the providers' bids are too high) and the witness fee prepaid by both parties will be refunded.

Once the winning providers are selected, the auction smart contract generates k SLA smart contracts and waits for both parties to sign. Next, the customer needs to prepay the service fee generated by the bidding of providers. After all SLA smart contracts are signed, a witness registration window opens and allows normal blockchain users to register as auction witnesses to earn commission fees. SLA smart contracts become active only when a sufficient number of witnesses are registered. The providers then start to provide the corresponding services in accordance with the SLA smart contracts, and the witnesses start to monitor the services. An example of how a witness can monitor the service is as follows. The federated providers provide a cluster of on-demand VMs and notify the customer and all witnesses of the public IP address via the service details field in the SLA smart contracts. Therefore, the customer can use this cluster, and each witness starts to "ping" VMs' IP addresses constantly. If the violation happens during the service time, i.e., the IP address is not accessible, the witness can record this violation. However, this is a naive example with a simple monitoring task. If more advanced monitoring tasks are required, then certain permissions need to be granted to allow witnesses to deploy probes in the service for monitoring. In real SLAs of more complex scenarios, the service monitoring component can be negotiated and provided by the customer and providers. Besides, this component can be delivered in the form of containers, which are lightweight and portable. Then, the witness is able to download the container and query it to detect service violations. However, as with any technology, there are trade-offs when using containers for this monitoring component. Despite the flexibility of the container infrastructure, serious security risks can arise if containers are not properly configured. Therefore, container security tools and policies must be considered to ensure that monitoring activities in the container are operating as expected. Some popular container security solutions and tools, including scanning (e.g., Docker Bench for Security), monitoring (e.g., Prometheus), and firewall configuration (e.g., Cilium), can partially address security concerns [32]. After the federated cloud services are completed, witnesses need to report the monitoring results. The auction smart contract then calculates and pays the witness fee to witnesses and finalizes the ownership of the service fee based on the reported SLAs violation result; if there is no violation, the service fees are paid to all providers. Otherwise, the part of the prepaid service fee will be refunded to the customer if any of the sub-services are in violation.

4. Key techniques

In this section, we describe three key techniques in our federated cloud auction model in detail. This model enables the automatic enforcement of the auction, the results of which can convince both auction parties. Specifically, we first model the partition of federated cloud services as a graph partition problem (steps 1a in Fig. 1). Then, in our model, the payoff functions for different stakeholders are hardcoded in the smart contract on the blockchain. This means that when providers and witnesses notice these payoff functions, they are motivated to act according to the derived BNEs to maximize their profits. Thus, the effectiveness and trustworthiness of the auction are guaranteed through two BNEs among providers and witnesses (steps 2a and 7a in Fig. 1). Finally, a timed message submission (TMS) algorithm is proposed to protect the auction privacy during the message submission phase. The algorithm also allows the model to satisfy the basic assumption of the static Bayesian game (steps 2b and 7b in Fig. 1). The notions and symbols used in this paper are listed in Table 2 for easy reference.

4.1. Federated cloud partition

We consider a problem scenario where a customer used to use the cloud service from a single provider. Now, this customer wants to switch to a federated cloud solution to improve flexibility and reliability. Before using auction models to select different providers, the customer needs to consider how many providers are needed and how to allocate existing resources to those providers. We assume that the total budget and VM topology are fixed. In this case, the original VM topology can be simulated as an infrastructure graph.

Table 2
The list of symbols and notations used in this paper.

Symbol/Notation	Description
$G = (M, E)$	An undirected weighted infrastructure graph for a specific cloud service.
$G' = G_1 \cup G_2 \cup \cdots \cup G_k$	A partitioned infrastructure graph such that k blocks are disjoint and have (nearly) balanced *VMSize* and *Budget*.
$M = \{m_1, m_2, \ldots, m_M\}$	A set of cloud VMs and each $m_i \in M$ represents a VM.
$p(m_i)$	The time unit service price of VM m_{i}.
E	A set of edges and each $e = \{i, j\} \in E$ represents the network communication cost between m_i and m_j.
J	A vector of resource attributes for a cloud VM.
$\mathbf{r} = \left(r_1, r_2, \ldots, r_{\lvert J\rvert}\right)$	The resource capacity of the VM is a vector, where r_h is a value of the hth resource attribute in J.
$c_{uv} \in \{0, 1\}$	The binary decision variable for edges of the G, which is 1 if $e = \{u, v\}$ is a cut edge, otherwise it is 0.
$x_{v,k} \in \{0, 1\}$	The binary decision variable for vertices of the G, which is 1 if v is in block k, otherwise it is 0.
VMSize	The VM size requirement of each partition.
Budget	The budget requirement of each partition.
b_{i}	The bidding function of provider i is $b_{\mathrm{i}} \geq 0$, which is monotonically increasing and differentiable.
v_{i}	The expected value of provider i for the service to be auctioned.
$b_{\mathrm{i}}^{*}(v_{\mathrm{i}})$	The BNE of the provider i in the BBG.
BBG	An n-player static Bayesian game of incomplete information for bidding on the federated cloud services.
$N = \{1, 2, \ldots, n\}$	A set of players. Each player is a provider who can offer bids for the cloud service that the customer needs.
$T = \left[v^{\min}, v^{\max}\right]$	The continuous type space of different providers. v_{i} is independently and identically distributed in this interval.
$A = [0, \infty)$	The non-negative continuous action space (bid) of each provider.
p	A common belief of all providers that v_{i} is independently and identically distributed in $\left[v^{\min}, v^{\max}\right]$.
$u = (u_1, u_2, \ldots, u_{\mathrm{n}})$	A set of payoff functions for service providers, where $u_{\mathrm{i}} : A \times T \rightarrow R$.
$\mathrm{C_P}$	A provider needs to pay a blockchain transaction fee $\mathrm{C_P}$ to submit new blockchain transactions.
$s = (s_1, s_2, \ldots, s_{\mathrm{n}})$	The strategy profile of the Bayesian game.
WBG	An m-player static Bayesian game of incomplete information for monitoring the service j.
$M = \{1, 2, \ldots, m\}$	A set of players. Each player is a witness and all the witnesses form the witness committee for service j.
$WT = \{H, D, R\}$	The type space of witness members.
$WA = [0, 1]$	The continuous action space for witnesses to monitor the service.
P	A common belief of all witness members that there are three types of witnesses, and the proportions are p_{H}, p_{D}, and p_{R}.
$U = (U_1, U_2, \ldots, U_{\mathrm{m}})$	A set of payoff functions for witnesses, where $U_{\mathrm{i}} : WT \times WA \rightarrow R$.
$w_{\mathrm{i}}^{\mathrm{j}}$	The monitoring report given by witness i for the SLA j.
$w_{\mathrm{R}}, w_{\mathrm{H}}, w_{\mathrm{H}}$	The monitoring results of three types of witnesses.
$SLA_{\mathrm{violate}}^{\mathrm{j}}$	The SLA violation indicator for service j.
$\varphi\left(w_{\mathrm{i}}^{\mathrm{j}}\right)$	The penalty function of witness i for monitoring service j.
ε	The intensity factor of the penalty function $\varphi\left(w_{\mathrm{i}}^{\mathrm{j}}\right)$.
$f\left(w_{\mathrm{i}}^{\mathrm{j}}\right)$ and $g\left(w_{\mathrm{i}}^{\mathrm{j}}\right)$	The intrinsic psychological cost functions for honest and dishonest witnesses.
h and l	The preference intensity coefficients for honest and dishonest witnesses.
$\mathrm{C_W}$	A witness needs to pay a blockchain transaction fee $\mathrm{C_W}$ to submit new blockchain transactions.
$F_{\mathrm{witness_i}}^{\mathrm{j}}$	The witness fee of witness i for monitoring service j.
α	The growth factor of the witness i's blockchain transaction fee.

Definition 1 (*Infrastructure Graph*)**.** Let $G = (M, E)$ be an undirected weighted infrastructure graph for a specific cloud service, where M is a set of vertices and E is a set of edges.

In a federated infrastructure graph, $M = \{m_1, m_2, \ldots, m_M\}$ denotes a set of cloud VMs. Each VM m_i is assigned to a value $p(m_i)$ that indicates the unit price of this VM regarding usage time. E is a set of edges and each $e = \{i, j\} \in E$ represents the network communication cost between m_i and m_j.

Definition 2 (*Resource Capacity*)**.** Let J denote a vector of resource attributes for a cloud VM. Thus, the resource capacity of this VM is a vector $\mathbf{r} = \left(r_1, r_2, \ldots, r_{|J|}\right)$, where r_h is the value of the hth resource attribute in J.

A cloud VM can have several resource attributes to determine its pricing, e.g., vCPU, memory, instance storage, and network bandwidth. Suppose there are two VMs with similar functions. Their resource capacities are (2, 500, 100) and (1, 1000, 200), and the three values represent vCPU, memory, and storage, respectively. In this case, the first VM has more vCPU processors, whereas the second VM has higher memory and storage attributes. It is difficult to compare which one is more valuable. In order to evaluate the value of VMs based on their available resources, we use the Simple Additive Weighting (SAW) [33] method to convert the different resource attributes into a single value.

First, the values of different resource attributes can be normalized to eliminate incompatibility. Let the resource capacity of m_i be $\mathbf{r}_i = \left(r_{i,1}, r_{i,2}, \ldots, r_{i,|J|}\right)$, then any $r_{i,h} > 0$ can be normalized by the following function:

$$\beta\left(r_{i,h}\right) = \begin{cases} \frac{r_{i,h} - r_{i,h}^{\min}}{r_{i,h}^{\max} - r_{i,h}^{\min}} & \text{if } r_{i,h}^{\max} \neq r_{i,h}^{\min} \\ 1 & \text{if } r_{i,h}^{\max} = r_{i,h}^{\min} \end{cases} \tag{1}$$

Here $r_{i,h}^{\max}$ and $r_{i,h}^{\min}$ are the maximum and minimum values of the hth resource attribute on m_i. Then, the time unit price of m_i can be converted into a single value, forming the attribute of each node:

$$p(m_i) = \sum_{h=1}^{|J|} \left(\omega_h \times \beta\left(r_{i,h}\right)\right) \tag{2}$$

Here $\omega_h \in (0, 1)$ is the price coefficient that determines the unit price of the hth resource attribute. In this way, Eq. (2) can represent the total value of the VM with different resource attributes. It should be noted that ω_h could differ in different providers. However, as a partition user (i.e., the customer), it is impossible to know this value for specific providers until the auction is completed (because the providers are not yet selected). Here, the cloud partition model aims to estimate how many providers are most beneficial based on the existing market pattern and total budget. Therefore, the user can use an expected value of the possible price coefficients among all candidate cloud providers.

We now consider partitioning the *Infrastructure Graph* into different blocks with a mixed-integer linear programming (MILP) model. A balanced partition target for both VM size and budgets is set so that all the sub-blocks can be regarded as similar auction objects. At the same time, the communication costs across different providers should be minimized since these costs are usually considered the most expensive ones [34].

Definition 3 (*K-Partitioned Infrastructure Graph*)**.** Let $G' = G_1 \cup G_2 \cup \cdots \cup G_k$ denote a partitioned infrastructure graph such that k blocks are disjoint and have (nearly) balanced VM size and budget. Meanwhile, the total weight of cut edges[6] is minimized.

In order to get G', we first introduce binary decision variables for edges and vertices of the G. More precisely, for each edge $e = \{u, v\} \in E$, we introduce the variable $c_{uv} \in \{0, 1\}$, which is 1 if e is a cut edge, otherwise it is 0. In addition, for each $v \in V$ and block k, we introduce the variable $x_{v,k} \in \{0, 1\}$, which is 1 if v is in block k, otherwise it is 0. There are upper and lower bounds on VM size and budget. Users can adjust the imbalance rate of VM size and budget for different partitions based on $VMSize_{max}$, $VMSize_{min}$, $Budget_{max}$, and $Budget_{min}$ parameters. Finally, $t(v)$ is the service time of VM v. We assume that the budget is proportional to the service time $t(v)$ and the time unit price $p(v)$. Thus, the objective of an infrastructure graph partition model can be described as:

$$\text{Minimize} \sum_{\{u,v\} \in E} c_{uv} \cdot e(\{u, v\}) \tag{3}$$

$$\text{Subject to:} \forall \{u, v\} \in E, \forall k : c_{uv} \geq x_{u,k} - x_{v,k} \tag{4}$$

$$\forall \{u, v\} \in E, \forall k : c_{uv} \geq x_{v,k} - x_{u,k} \tag{5}$$

$$\forall k : \sum_{v \in V} x_{v,k} \leq VMSize_{max} \tag{6}$$

$$\forall k : \sum_{v \in V} x_{v,k} \geq VMSize_{min} \tag{7}$$

$$\forall k : \sum_{v \in V} x_{v,k} p(v) t(v) \leq Budget_{max} \tag{8}$$

$$\forall k : \sum_{v \in V} x_{v,k} p(v) t(v) \geq Budget_{min} \tag{9}$$

$$\forall v \in V : \sum_{k} x_{v,k} = 1 \tag{10}$$

Objective Eq. (3) expresses the goal of this model. Constraints Eq. (4) & Eq. (5) ensure that c_{uv} satisfies the basic assumption of a cut edge. Constraints Eq. (6) & Eq. (7) guarantee that partitioned blocks are balanced regarding the VM size. Constraints Eq. (8) & Eq. (9) make sure the block budget does not exceed the upper/lower bounds. Finally, constraint Eq. (10) sets that each node can only be set to one block.

An example of a federated cloud partition is illustrated in Fig. 2. The original infrastructure graph with nine VMs is partitioned into three balanced blocks, and each block has the same VM size and budget boundary. Meanwhile, the total communication cost across different providers (red dotted lines) is minimized. After the partition, the customer should know how many providers are needed to form the federated cloud services. The customer can then post an auction request on the blockchain. After the winning providers are selected, the customer needs to prepay the service fee using cryptocurrencies. This fee will be automatically enforced at the end of the auction. It should be noted that here we consider a balanced infrastructure graph partition. In fact, our auction model is still applicable when dealing with an unbalanced partition scenario. The difference is that in an unbalanced partition, the auction needs to be completed in each sub-block independently since the requirements of blocks are different.

4.2. Bayesian game-based auction enhancement

Bayesian games are often used to model economic situations where players may not know various features of the environment. In this section, two Bayesian games are leveraged to model

[6] A cut edge is an edge that connects different partitions of the graph. Therefore, cut edges can be an approximation of the cross-cloud communication cost incurred by the partitioning.

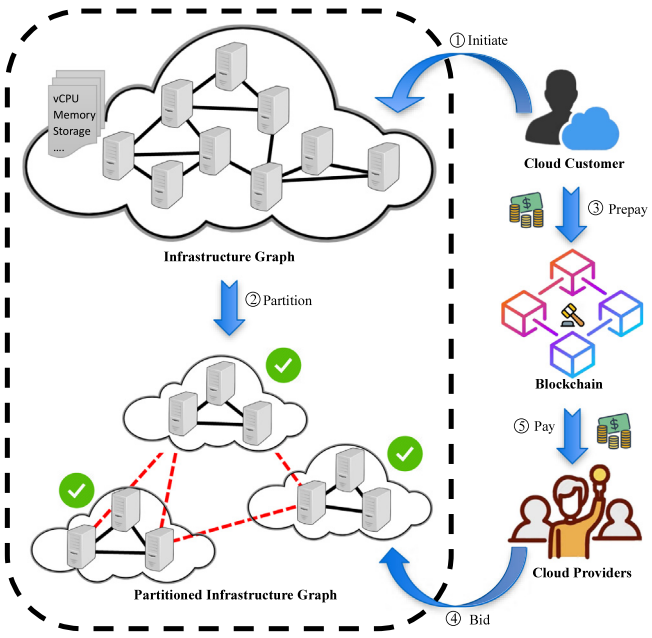


Fig. 2. An illustration of the federated cloud partition and auction.

incomplete information sharing among different cloud providers and auction witnesses.

4.2.1. Bidding Bayesian game

Some basic assumptions for the bidding Bayesian game are described as follows. There are n providers simultaneously bidding on the cloud services. Since they do not know the types of other players, it is impossible to know the true bids of others. The bidding function of provider i is $b_i \geq 0$, which is monotonically increasing and differentiable. Moreover, v_i is the expected value of provider i for the service to be auctioned. It is the private information of provider i and can be regarded as the provider's type. We further define v_i is independently and identically distributed[7] on $\left[v^{\min}, v^{\max}\right]$, the cumulative density function is F, and its probability density function is f. Thus, we can formulate this bidding problem as a static Bayesian game of incomplete information.

Definition 4. Let BBG = (N, T, A, p, u) denote an n-player static Bayesian game of incomplete information for bidding on the federated cloud services, where:

- $N = \{1, 2, \ldots, n\}$ is a set of players. Each player is a provider who can offer bids for the cloud service that the customer needs.
- $T = \left[v^{\min}, v^{\max}\right]$ is the continuous type space of different providers. Providers only observe their own types.
- $A = [0, \infty)$ is the non-negative continuous action (i.e., bid) space of each provider.
- p is a common belief of all providers that v_i is independently and identically distributed on $\left[v^{\min}, v^{\max}\right]$ with cumulative density function F.
- $u = (u_1, u_2, \ldots, u_n)$ is a set of payoff functions for service providers, where $u_i : A \times T \to R$.

[7] In practice, the distribution of v_i for different providers may be different. This brings system noises and increases the difficulty of estimating the provider's bids. Therefore, we assume that bidders are symmetric in our model, which is the same assumption as in most auction studies.

We propose that in our BBG model, the providers who submit the k-lowest bids win this auction. The rewards they can get are their own bidding prices, and their utilities are the differences between the bids and the internal expected values. In practice, a provider usually does not submit the bid as the expected value v_i. Instead, a higher bid is often submitted to maximize profits. At the same time, providers must consider submitting bids lower than other $n - k$ providers' to ensure they will be selected. A provider also needs to pay a blockchain transaction fee C_P to submit new blockchain transactions.[8] Thus the utility function u_i of provider i can be described as:

$$u_i(b_i, v_i) = \begin{cases} b_i - v_i - C_P & \text{if } \sum_j \mathbb{1}\left\{b_i < b_j, \forall j \neq i\right\} > n - k \\ -C_P & \text{otherwise} \end{cases} \tag{11}$$

Definition 5. Let the strategy of player i be function $s_i : T \to A$, thus the strategy profile $s = (s_1, s_2, \ldots, s_n)$ is a BNE if for any $i \in N$, s_i assigns an optimal action that maximizes player i's expected payoff.

Theorem 1. *If the bid of each provider satisfies the following function, the strategy profile* $s = \left(b_1^*(v_1), b_2^*(v_2), \ldots, b_n^*(v_n)\right)$ *is a unique BNE for the BBG.*

$$b_i^*(v_i) = v_i + \frac{\int_{v_i}^{v^{\max}} (1 - F(v))^{n-k} dv}{[1 - F(v_i)]^{n-k}} \tag{12}$$

Proof of Theorem 1. Assume the BNE of the BBG can be expressed as $b_i^*(v_i) = b(v_i)$ for any $i \in N$, where b is an increasing and differentiable function. A specific provider i must submit a bid which is lower than other $n - k$ players. Thus, the expected utility of provider i can be described as:

$$\mathbb{E}[u_i(b_i, v_i)] \tag{13}$$

$$= (b_i - v_i - C_P) \cdot \Pr\left\{\sum_j \mathbb{1}\left\{b_i < b_j^*(v_j) = b(v_j), \forall j \neq i\right\} > n - k\right\} + (-C_P) \cdot \left(1 - \Pr\left\{\sum_j \mathbb{1}\left\{b_i < b_j^*(v_j) = b(v_j), \forall j \neq i\right\} > n - k\right\}\right) \tag{14}$$

$$= (b_i - v_i) \cdot \Pr\left\{\sum_j \mathbb{1}\left\{v_j > b^{-1}(b_i), \forall j \neq i\right\} > n - k\right\} - C_P \tag{15}$$

$$= (b_i - v_i) \cdot \left[1 - F\left(b^{-1}(b_i)\right)\right]^{n-k} - C_P \tag{16}$$

Therefore, the objective is to obtain the expression of b_i that maximizes the expected utility. Using the first order condition, we get:

$$(v_i - b_i) \cdot f\left(b^{-1}(b_i)\right) \cdot (n - k)\left[1 - F\left(b^{-1}(b_i)\right)\right]^{n-k-1} \cdot \left[b^{-1}(b_i)\right]' + \left[1 - F\left(b^{-1}(b_i)\right)\right]^{n-k} = 0 \tag{17}$$

Bringing in the equilibrium point of provider i, we get $b_i = b_i^*(v_i)$. Since $b_i^*(v_i) = b(v_i)$, we can replace b_i with $b(v_i)$ in the above equation and get:

$$(v_i - b(v_i)) \cdot f(v_i) \cdot (n - k) \cdot [1 - F(v_i)]^{n-k-1} / b'(v_i) + [1 - F(v_i)]^{n-k} = 0 \tag{18}$$

[8] We define C_P as a constant since the transaction fees are similar among different providers. The same setting is applied to C_W.

This equation can be rearranged into:

$$\frac{d}{dv_i} b(v_i)\cdot[1-F(v_i)]^{n-k} = -v_i\cdot f(v_i)\cdot(n-k)\cdot[1-F(v_i)]^{n-k-1} \quad (19)$$

By integrating both sides of the equation from v_i to v^{max}, we get:

$$b(v_i) = \frac{\int_{v^{max}}^{v_i} v_i\cdot\left[(1-F(v_i))^{n-k}\right]' dv_i}{[1-F(v_i)]^{n-k}} \quad (20)$$

By simplifying the above equation, the proof of Theorem 1 is finished.

From Eq. (12) we know that when n and k are fixed, $b_i^*(v_i)$ depends on v_i, v^{max}, and $F(v_i)$. Since we assume that all the providers are rational, they will bid according to the BNE points of Eq. (12) to maximize their utility. Consequently, our auction model receives a set of bidding prices where all bidders' utilities are maximized. The most suitable providers (with the k lowest v_i) are selected and the effectiveness of the auction is guaranteed.

Next, we discuss the case of different distributions of v_i, which is often assumed to be uniform, normal, or log-normal distribution in related research [35]. In order to show the explicit equation, we suppose all v_i follows a uniform distribution in $[v^{min}, v^{max}]$. Using the cumulative density function of uniform distribution $F(v_i) = \frac{v_i - v^{min}}{v^{max}-v^{min}}$, Eq. (12) can be further simplified as:

$$b_i^*(v_i) = \frac{n-k}{n-k+1} v_i + \frac{1}{n-k+1} v^{max} \quad (21)$$

The above equation is obtained when the number of target providers k is known before different providers start to bid. In this case, the bid of provider i only needs to be lower than any other $n-k$ providers' to ensure the successful bidding. By contrast, in a first-price auction model where only one provider (with the lowest bid) can be selected, provider i's bid must be lower than all other $n-1$ providers' bids. The customer can also select k providers by performing a k-round first-price auction. At this time, provider i's BNE is:

$$b_{i'}^*(v_i) = \frac{n-1}{n} v_i + \frac{1}{n} v^{max} \quad (22)$$

In fact, both bidding strategies (Eqs. (21) and (22)) can select k providers to form the cloud federation. However, here we only choose Eq. (21) in our auction model, which is mainly because: (1) the provider's utility in Eq. (21) is higher than that in Eq. (22) (as shown in Eq. (23)), so providers are more willing to participate; and (2) Eq. (22) requires multiple loops of the auction process, which is time-consuming and expensive to execute on the blockchain.

$$b_i^*(v_i) - b_{i'}^*(v_i) = \frac{k-1}{n(n-k+1)}\cdot\left(v^{max} - v_i\right) \geq 0 \quad (23)$$

4.2.2. Witness Bayesian game

Once the customer and federated providers reach an agreement on the auction detail, they start to follow and execute it. Providers need to provide the federated cloud services according to the requirements in SLAs, and the customer needs to pay the corresponding service fees. However, any individual can violate the previous agreement. The providers may not provide the quality of service (QoS) they promised, and the customer may also refuse to pay service fees using the excuse of service violations. Therefore, a trustworthy witness mechanism is proposed to monitor and control the enforcement of federated SLAs. We assume the witness committee monitors different cloud services independently; their reporting results for one service do not affect the results for others. Thus we can model the monitoring process of the service j as follows.

Definition 6. Let WBG = (M, WT, WA, P, U) denote an m-player static Bayesian game of incomplete information for monitoring the service j, where:

- $M = \{1, 2, \ldots, m\}$ is a set of players. Each player is a witness and they form the witness committee for service j.
- $WT = \{H, D, R\}$ is the type space of witness members, where H, D, and R represent honest, dishonest, and rational witness, respectively. Witnesses only observe their own types.
- $WA = [0, 1]$ is the continuous action space for witnesses to monitor the service. Specifically, we denote w_i^j as the monitoring result given by witness i for the SLA j.
- P is a common belief of all witness members that there are three types of witnesses in total, and the proportions are p_H, p_D, and p_R, respectively.
- $U = (U_1, U_2, \ldots, U_m)$ is a set of utility functions for witnesses, where $U_i : WT \times WA \to R$ is the payoff function determining the rewards of witness i.

Definition 7. Based on the monitoring result of service j, the SLA violation is confirmed only when the majority of witnesses report violations. Otherwise, it is treated as no violation happens. Since $w_i^j \in [0, 1]$, the witness i reports the violation of SLA j when $w_i^j > 1/2$.

$$SLA_{violate}^j = \begin{cases} 0 & \text{if } \sum_i \mathbb{1}\left\{0 \leq w_i^j \leq \frac{1}{2}\right\} > \frac{m}{2} \\ 1 & \text{if } \sum_i \mathbb{1}\left\{\frac{1}{2} < w_i^j \leq 1\right\} > \frac{m}{2} \end{cases} \quad (24)$$

Here $SLA_{violate}^j = \{0, 1\}$ is used to denote the finial result of whether SLA j is violated; $SLA_{violate}^j = 1$ means SLA j is violated while $SLA_{violate}^j = 0$ means not. The above definition shows that when most witnesses report violations, the model determines that the SLA j is violated. Next, we further design that witnesses should be penalized if their reports fail to match the reports of others. Thus the penalty function of witness i can be described as:

$$\varphi\left(w_i^j\right) = \frac{\varepsilon}{n-1}\sum_{i\neq k}\left(w_i^j - w_k^j\right)^2 \quad (25)$$

Here ε is defined as the intensity factor of the penalty function. Since each type of witness will give out the same report based on their strategy profile, the penalty function $\varphi\left(w_i^j\right)$ can be further described as:

$$\varphi\left(w_i^j\right) = \varepsilon\left[(1-p_H-p_D)\left(w_i^j - w_R\right)^2 + p_H\left(w_i^j - w_H\right)^2 + p_D\left(w_i^j - w_D\right)^2\right] \quad (26)$$

Here w_R, w_H, and w_D are used to denote the monitoring results from three types of witnesses. We further assume honest and dishonest witnesses have a psychological cost to tell a lie/truth, while rational witnesses have no psychological burden. The quadratic functions $f\left(w_i^j\right)$ and $g\left(w_i^j\right)$ can well represent the psychological burden of two types of witnesses.

$$f\left(w_i^j\right) = \begin{cases} h\cdot\left(1-w_i^j\right)^2 & \text{if } SLA_{violate}^j = 1 \\ h\cdot(w_i^j)^2 & \text{if } SLA_{violate}^j = 0 \end{cases} \quad (27)$$

$$g\left(w_i^j\right) = \begin{cases} l\cdot\left(1-w_i^j\right)^2 & \text{if } SLA_{violate}^j = 0 \\ l\cdot(w_i^j)^2 & \text{if } SLA_{violate}^j = 1 \end{cases} \quad (28)$$

Here $f\left(w_i^j\right), g\left(w_i^j\right) : [0,1] \rightarrow R_+$. h and l are the preference intensity coefficients for honest and dishonest witnesses. We design that honest and dishonest witnesses have inherent psychological costs, which means that when honest witnesses tell lies or dishonest witnesses tell the truth, their psychological burden will increase. Especially, we set $h = l = 1$ in the following text to simplify the calculation. Thus, we can describe the utility function of three types of witnesses as follows, where C_W is the blockchain transaction fee and $F^j_{\text{witness}_i}$ is the witness fee of witness i for service j. We specify that the witness fee is large enough so that the witness's utility is always positive ($U_i > 0$). In this way, witnesses always have an incentive to participate in our model and receive a reward.

$$U_i\left(w_i^j, SLA^j_{\text{violate}}, R\right) = F^j_{\text{witness}_i} - \varphi\left(w_i^j\right) - C_W \tag{29}$$

$$U_i\left(w_i^j, SLA^j_{\text{violate}}, H\right) = F^j_{\text{witness}_i} - \varphi\left(w_i^j\right) - f\left(w_i^j\right) - C_W \tag{30}$$

$$U_i\left(w_i^j, SLA^j_{\text{violate}}, D\right) = F^j_{\text{witness}_i} - \varphi\left(w_i^j\right) - g\left(w_i^j\right) - C_W \tag{31}$$

Theorem 2. *In a WBG, when the monitoring result of each type of witness satisfies the following equations, the strategy profile* $s = \left(w_R^*, w_H^*, w_D^*\right)$ *constitutes a unique BNE.*

$$w_R^* = \begin{cases} \frac{p_H}{p_H+p_D} & \text{if } SLA^j_{\text{violate}} = 1 \\ \frac{p_D}{p_H+p_D} & \text{if } SLA^j_{\text{violate}} = 0 \end{cases} \tag{32}$$

$$w_H^* = \begin{cases} 1 - \frac{\varepsilon \cdot p_D}{(\varepsilon+1)(p_H+p_D)} & \text{if } SLA^j_{\text{violate}} = 1 \\ \frac{\varepsilon \cdot p_D}{(\varepsilon+1)(p_H+p_D)} & \text{if } SLA^j_{\text{violate}} = 0 \end{cases} \tag{33}$$

$$w_D^* = \begin{cases} 1 - \frac{p_H+(\varepsilon+1)\cdot p_D}{(\varepsilon+1)(p_H+p_D)} & \text{if } SLA^j_{\text{violate}} = 1 \\ \frac{p_H+(\varepsilon+1)\cdot p_D}{(\varepsilon+1)(p_H+p_D)} & \text{if } SLA^j_{\text{violate}} = 0 \end{cases} \tag{34}$$

Proof of Theorem 2. Since the situation about SLA violation is symmetric, we only consider $SLA^j_{\text{violate}} = 1$ to prove Theorem 2. Here the iterated elimination of strictly dominated strategies (IESDS) method [36] is leveraged to narrow down and solve the BNE of the WBG. Specifically, with IESDS the original game can be divided into an n-round game to remove dominated strategies. In each round, witness i wants to maximize the utility by reporting different w_i^j. The monitoring result of round $t+1$ is determined by round t when the payoff is maximized.

$$w_R(t+1) = \arg \max_{w_i^j \in w_R(t)} \left[F^i_{\text{witness}} - \varphi\left(w_i^j\right) - C_W\right] \tag{35}$$

$$w_H(t+1) = \arg \max_{w_i^j \in w_H(t)} \left[F^i_{\text{witness}} - \varphi\left(w_i^j\right) - f\left(w_i^j\right) - C_W\right] \tag{36}$$

$$w_D(t+1) = \arg \max_{w_i^j \in w_D(t)} \left[F^i_{\text{witness}} - \varphi\left(w_i^j\right) - g\left(w_i^j\right) - C_W\right] \tag{37}$$

Specifically, the utility would be maximized when the first-order condition is satisfied. For three types of witnesses, let $w_i^j = w_R$, $w_i^j = w_H$, and $w_i^j = w_D$, respectively. We get:

$$w_R(t+1) = \frac{p_H}{p_H+p_D} w_H(t) + \frac{p_D}{p_H+p_D} w_D(t) \tag{38}$$

$$w_H(t+1) = \frac{\varepsilon \cdot (1-p_H-p_D)}{\varepsilon - \varepsilon \cdot p_H + 1} w_R(t) + \frac{\varepsilon \cdot p_D \cdot w_D(t) + 1}{\varepsilon - \varepsilon \cdot p_H + 1} \tag{39}$$

$$w_D(t+1) = \frac{\varepsilon \cdot (1-p_H-p_D)}{\varepsilon - \varepsilon \cdot p_D + 1} w_R(t) + \frac{\varepsilon \cdot p_H}{\varepsilon - \varepsilon \cdot p_D + 1} w_H(t) \tag{40}$$

From the above combined equations we can see that the maximum/minimum value of w_R in round $t+1$ is determined by the maximum/minimum value of w_H and w_D in round t. The cases of w_H and w_D are similar. We then use the proof by contradiction to show the uniqueness of the Nash Equilibrium. Assume in the IESDS process, w_R^*, w_H^*, w_D^* finally converge to an interval $[\min(s^*), \max(s^*)]$ separately instead of a single point. Thus, $\min\left(w_R^*\right)$, $\max\left(w_R^*\right)$, $\min\left(w_H^*\right)$, $\max\left(w_H^*\right)$, $\min\left(w_D^*\right)$, and $\max\left(w_D^*\right)$ should all satisfy the combined equations. At the Nash equilibrium state the witnesses should report the same result at round t and $t+1$. However, given the fixed ε, p_H, and p_D, the above combined equations only have one unique set of solution, as shown in equation Eq. (41), Eq. (42), and Eq. (43).

$$w_R^* = \frac{p_H}{p_H+p_D} \tag{41}$$

$$w_H^* = 1 - \frac{\varepsilon \cdot p_D}{(\varepsilon+1)(p_H+p_D)} \tag{42}$$

$$w_D^* = 1 - \frac{p_H+(\varepsilon+1)p_D}{(\varepsilon+1)(p_H+p_D)} \tag{43}$$

This means $w_R^* = \min\left(w_R^*\right) = \max\left(w_R^*\right)$, $w_H^* = \min\left(w_H^*\right) = \max\left(w_H^*\right)$, and $w_D^* = \min\left(w_D^*\right) = \max\left(w_D^*\right)$ in the Nash Equilibrium. The same conclusion can also be obtained in the situation when $SLA_{\text{violate}} = 0$. Therefore, the original assumption (i.e., the strictly dominated strategies converge to an interval) does not hold. Theorem 2 is proved.

In the above modeling, witness i only monitors the SLA j. When k federated SLAs need to be monitored together, the overall payoff of witness i then changes to Eq. (44) (with the rational type). Since we assume the witness committee monitors different sub-cloud services independently, the BNE of the monitoring result for a specific provider does not change.

$$U_i\left(w_i, SLA_{\text{violate}}, R\right) = k \cdot F^j_{\text{witness}_i} - \sum_{j \in J} \varphi\left(w_i^j\right) - \alpha \cdot C_W \tag{44}$$

Here α is the growth factor of the witness i's blockchain transaction fee. It is also noted that the witness committee monitors each sub-service separately in the above modeling process. This means that the monitoring result of each witness is a k-dimension vector, and each value of the vector indicates whether a sub-cloud service is violated. By contrast, each witness can also make a general judgment about whether the whole federated cloud service is violated or not. In this case, the monitoring result is a value from 0 to 1 instead of a vector, and the federated cloud services formed by k providers are actually regarded as one service. However, this design may be unfriendly to providers who do not violate the SLA when the whole judgment is a violation.

4.3. Timed Message Submission (TMS) algorithm

In the BBG and WBG of the proposed federated cloud auction model, we assume that both providers and witnesses submit messages (bids and reports) to the blockchain simultaneously. The simultaneity and data privacy during the submission phase are crucial to:

- Satisfy the basic assumption of game theory. The static Bayesian game requires all players to submit messages to the blockchain simultaneously. However, this is very difficult to achieve in reality.
- Protect bid privacy and avoid possible plagiarism among submitters. Since data on the blockchain is public and transparent, if users who submit later can see the predecessors' message, their judgment may be affected.

Algorithm 1 TCS algorithm – Phase 1

Input:
Length of the registered witnesses array: $len(RW)$;
Length of the SLA array: $len(SLAs)$;
Sealed message array of witness i: $sealedMessages_i$;
Submission deposit of the witness i: $Deposit_i$.
Output:
Sealed message map for all witnesses: *sealedMap*;
// Phase 1: sealed message array submission
function SUBMITMESSAGES($sealedMessages_i$)
  require ($now < submitEnd$)
  require ($RW[address_i].registered == true$)
  require ($msg.value >= Deposit_i$)
  if $len(sealedMessages_i) == len(SLAs)$ **then**
    $submissionDeposit[address_i] = msg.value$
    $sealedMap[address_i] \leftarrow sealedMessages_i$
    $witnessNum++$
    if $witnessNum == len(RW)$ **then**
      return *sealedMap*
    end if
  end if
end function

In this context, we propose the TMS algorithm to handle the above-mentioned challenges. Our algorithm consists of two phases: (1) sealed message array submission; and (2) message array reveal and deposit refund. Correspondingly, there are two function interfaces named "submitMessages" and "RevealMessages" in the auction smart contract. The proposed algorithm can be leveraged by both providers and witnesses. Here we use the witness's case as an example to explain the algorithm details.

Algorithms 1 and 2 demonstrate the two phases of the TMS algorithm. In the first phase, only registered witnesses can submit the sealed message array. This array contains k values, and each value is a hash of the witness's judgment and the private key. When the current time is in the submission time window and the size of the message array meets the requirement, the witness can submit a deposit and store the sealed message array in *sealedMap*. After all witnesses submit their sealed message arrays, the function outputs a sealed message map, where keys are the addresses of witnesses, and values are the arrays of hash values. Considering the irreversibility of the hash function, it is impossible for witnesses to detect the true messages of others in this phase.

Then, the algorithm comes to the second phase. At this time, each witness needs to submit a message array (with true values instead of hashes) and a right witness key to reveal and verify the previously submitted sealed message array. Only witnesses who have submitted the sealed message in the first stage can call this function. In this stage, when the length and range of the real message array entered by the witness meet the requirement, the function checks whether the hash of each message value and witness key is equal to the sealed one submitted in the previous phase. We design that the smart contract can confiscate the deposit as a penalty if a submitter does not disclose the information within a specific time window. In this way, the submitter is forced to open the sealed message in time. When all values are revealed successfully, the real message array of the witness is stored in the smart contract for further processing. Then, when all witnesses are verified successfully, the algorithm ends and the "revealedMap" is exported. With these two steps, data privacy and simultaneity submission are guaranteed within a time window. The algorithm further discourages irresponsible submitters from causing losses to others by not revealing the message in time.

Algorithm 2 TCS algorithm – Phase 2

Input:
Message array of witness i: $realMessages_i$;
Private key of witness i: $witnessKey_i$;
Sealed message map obtained in the previous phase: *sealedMap*;
Output:
Revealed message map for all witnesses: *revealedMap*
// Phase 2: message array reveal and deposit refund
function REVEALMESSAGES($realMessages_i, witnessKey_i$)
  require ($now > submitEnd$ && $now < revealEnd$)
  require ($sealedMap[address_i] != null$)
  require ($len(realMessages_i) == len(SLAs)$)
  for $j = 0$; $j < len(SLAs)$; $j++$ **do**
    require ($realMessages_i[j] \in [0, 1]$)
    if $Hash(realMessages_i[j], witnessKey_i) == sealedMap[address_i][j]$ **then**
      $SLAsNum++$
    end if
  end for
  if $SLAsNum == len(SLAs)$ **then**
    $revealedMap[address_i] \leftarrow realMessages_i$
    $address_i.transfer(submissionDeposit[address_i])$
    $submissionDeposit[address_i] = 0$
    $revealNum++$
    if $revealNum == len(RW)$ **then**
      return *revealedMap*
    end if
  end if
end function

5. Evaluation and implementation

In this section, we design and implement experiments to test the proposed federated cloud auction model. Our evaluation is in the following parts. We first evaluate each of the three key techniques introduced in Section 4, namely the cloud partition model, the BNE strategies of two Bayesian games, and the TMS algorithm. Then, the entire smart contract implementation and the cost details are presented.

5.1. Cloud partition benchmark

To benchmark the proposed federated cloud partition model, we prepared four graph datasets. Table 3 shows the statistical information of the selected graphs, in which both synthetic and real-world datasets are used. We first used three types of synthetic workflows provided by the workflow generator in the Pegasus community [37]. These datasets simulate real scientific applications in seismology (CyberShake), biology (Genomes), and astronautics (Montage) fields. The datasets provide information about the application patterns and task dependencies performed on benchmark workstations, and are therefore suitable for testing our graph partition model. We also used a real-world dataset that contains cluster traces from the Alibaba Cluster Trace Program [38]. The trace is sampled from a real production cluster with long-running applications and batch workloads on each machine.

The cut edge proportion is used as a metric to evaluate the graph partitioning performance. We use Gurobi 8.1.1 as the MILP solver to find the optimal solution of the partitioned combinations. In addition, a popular graph partitioning tool called METIS is used as the benchmark for comparison. METIS could partition unstructured graphs into user-specified parts using either recursive bisection or k-way partitioning algorithms, both of which produce high-quality partitions [39].

Fig. 3 consists of 12 plots, where each column represents the result of one dataset. The four plots above show the performance

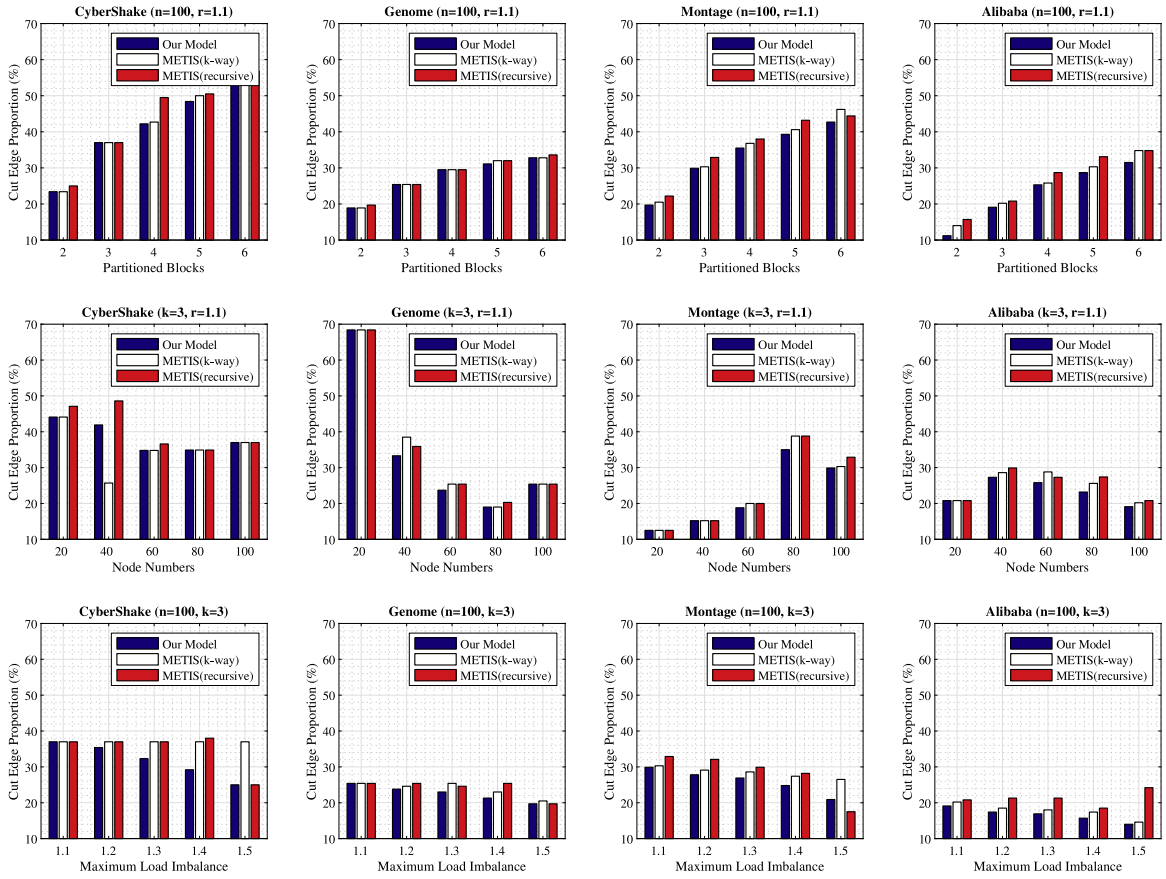


Fig. 3. Performance analysis of the federated cloud partition model.

Table 3
Statistics of the tested graph datasets.

Graph Dataset	Vertices	Edges	Average Degree	Category
CyberShake	100	192	3.84	Synthetic (Pegasus)
Genomes	100	122	2.44	Synthetic (Pegasus)
Montage	100	234	4.68	Synthetic (Pegasus)
Alibaba	100	178	3.56	Real-world (Clusters trace 2018)

with different partition blocks when the number of nodes and the load imbalance are fixed (n = 100, r = 1.1). It can be concluded that the proportion of cut edges tends to increase linearly with the increase in the number of partitioned blocks. Our model performs better in the CyberShake, Montage, and Alibaba datasets than in the Genome case. The middle four plots present the result of increasing the node numbers when the number of partitions and the load imbalance are fixed (k = 3, r = 1.1). Although an increase in graph nodes can increase the total number of edges, there is no such an increasing trend in the proportion of cut edges. Finally, the four plots below show the performance when the nodes and partition blocks are fixed (n = 100, k = 3). In general, when relaxing the constraint of the maximum load imbalance, our model obtains better performance to a linear trend. The case of METIS is more complex, and both partitioning methods' performance fluctuates under different configurations. It is worth noting that in the two special cases (i.e., CyberShake (n = 40, k = 3, r = 1.1) and Montage (n = 100, k = 3, r = 1.5)) the METIS method outperforms our model. After analysis, we found that METIS could not obtain feasible solutions in both cases; despite the lower value of the cut edge proportion, the load imbalance actually exceeds the maximum limit and thus leads to partition failures. This also proves the advantage of our model from the side. In summary, the number of nodes and partitions, the maximum load imbalance, and the model solving methods all affect the partition performance. Our model shows different degrees of improvement in different datasets and experimental settings compared to the two METIS approaches. The maximum performance improvement in CyberShake, Genome, Montage, and Alibaba datasets are 12%, 5.2%, 5.6%, and 10.2%, respectively.

Another finding is from Eq. (12) and Fig. 3, the fewer partitioned blocks result in lower bids and less cut edges. Therefore, the cloud customer seems to prefer to partition fewer blocks. However, in real life, the customer's decision on partitioning can be very complex and depends on many factors. Choosing fewer partition blocks means putting all their eggs in fewer baskets, which increases the risk of single points of failure. More partitioned blocks may also have advantages, such as better flexibility, reliability, and scalability — which is why federated cloud services are needed. Therefore, customers need to consider the trade-off between partitioning cost and QoS requirements.

Besides, we notice that the execution time is acceptable (within a few minutes) for most of the tested graphs. However, when dealing with large-scale graphs (e.g., more than several hundred nodes or when the graph density is very high), the model may take more than several hours to solve. This is mainly because the proposed MILP model is an NP-hard problem aimed at obtaining an optimal solution, which is a trade-off compared to METIS where fast partitioning results (but not optimal) can be obtained within seconds. The goal of the proposed federated cloud partition model is to find an optimal solution that helps customers to choose the right number of providers. Based on this consideration, we value partition quality as a more important

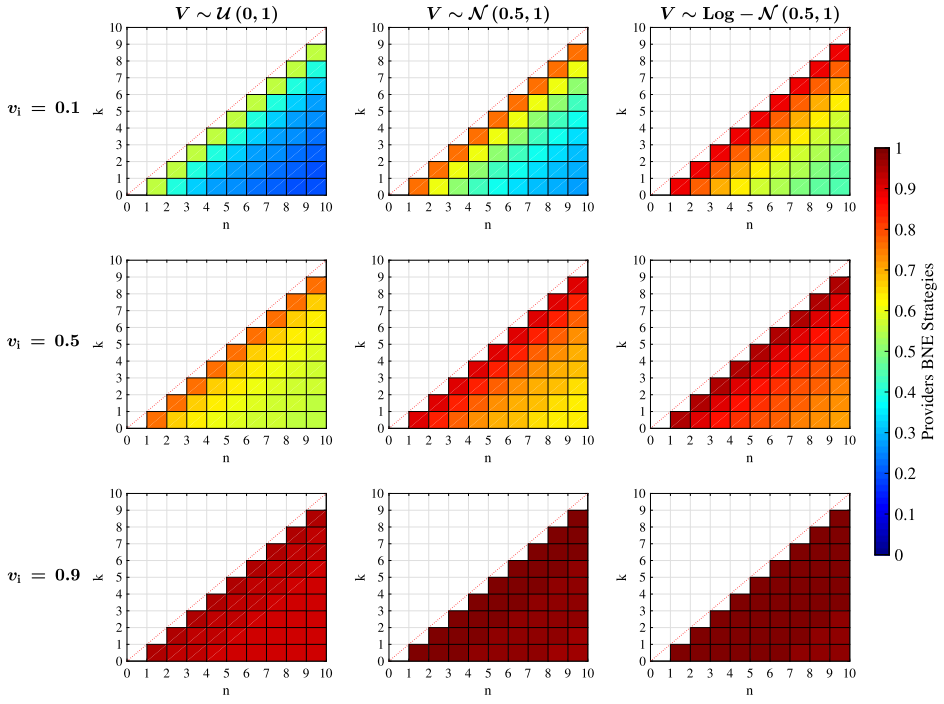


Fig. 4. Equilibrium strategies of the providers.

metric than execution time in the current model. We leave the algorithmic optimization of the model execution time to future work.

5.2. Bayesian Nash equilibrium analysis

In this section, we analyze and validate BNE strategies of the BBG and WBG in our model. Fig. 4 shows the equilibrium strategies of providers (bidders) when the types and distributions are different. There are three rows of plots, representing three types of bidders, namely $v_i = 0.1$, $v_i = 0.5$, and $v_i = 0.9$. Different bidders will submit bids based on their own expected values. Usually, bidders with higher v_i can offer higher bidding prices. The three columns correspond to three different distributions of V, namely $V \sim \mathcal{U}(0,1)$, $V \sim \mathcal{N}(0.5,1)$, and $V \sim log - \mathcal{N}(0.5,1)$. The x-axis of each plot represents the number of bidding providers (n), and the y-axis represents the number of providers to be selected (k). The color of the square is the final bid $b_i^*(v_i)$ submitted by provider i at the equilibrium point, where the redder the color, the higher the bid. It should be noted that there is no equilibrium when $k > n$.

First, when v_i and k are fixed, $b_i^*(v_i)$ will decrease with the increase of n. This is because the bidding becomes more competitive when n increases, and bidders must submit a lower bid to defeat their competitors. Similarly, when n and v_i are fixed, $b_i^*(v_i)$ will increase as k increases due to the bidding competition is weakened when k becomes larger. Thus, bidders can increase profits by submitting higher bids. When k approaches to n, $b_i^*(v_i)$ reaches its highest point. Besides, in the BBG bidders needs to detect other bidders' types from the current distribution of V to determine their own bids. For example, when $v_i = 0.1$ for bidder i, the probabilities of the other bidders' v_j locating between 0 to 0.1 for uniform, normal, and log-normal distribution are 10%, 3.6%, and 0.25%, respectively. The smaller probability means that potential competitors (people with lower bids than bidder i) are less likely to appear. So bidder i can submit a higher bid to increase profits.

Fig. 5 consists of 12 plots showing the equilibrium strategies of three witnesses types ($WT_i = R$, $WT_i = H$, and $WT_i = D$). The x-axis of each plot is the proportion of honest witnesses p_H, while the y-axis is the proportion of dishonest witnesses p_D. The color in the square represents the monitoring result w_i^* reported by witness i at the equilibrium point. The equilibrium strategy does not exist when $p_H + p_D > 1$. Besides, The first two columns indicate the situation of an SLA violation, while the last two columns indicate there are no violations. We set the penalty function factor $\varepsilon = 1$ in the first and third columns, and $\varepsilon = 10$ in the second and fourth columns. When $SLA^j_{\text{violate}} = 1$, it can be observed that the w_i^* of rational witnesses are only influenced by the current p_H and p_D. As p_H increases, they tends to report the true result where $SLA^j_{\text{violate}} > 1/2$. Otherwise when p_D is larger, they will report $SLA^j_{\text{violate}} < 1/2$. Next, the w_i^* of honest and dishonest witnesses are affected by both their intrinsic psychological cost and penalty function factor ε. When $\varepsilon = 1$, the penalty function is not enough to restrict their behaviors. Therefore, the honest/dishonest witnesses will follow their nature to tell the truth/lie. However, when ε is 10, witnesses' reports need to be consistent with others' to reduce the huge penalty. In conclusion, when ε is large enough and $p_H > p_D$, all three types of witnesses will report the consistent true SLA violations in the Nash equilibrium in order to improve their utility. The consistency and trustworthiness of monitoring, in this way, are guaranteed.

5.3. TMS algorithm evaluation

In Section 4.3 we proposed that the TMS algorithm can be applied to both providers and witnesses for submitting bids and monitoring results while protecting privacy. In this section, we evaluate the performance of the TMS algorithm using the case

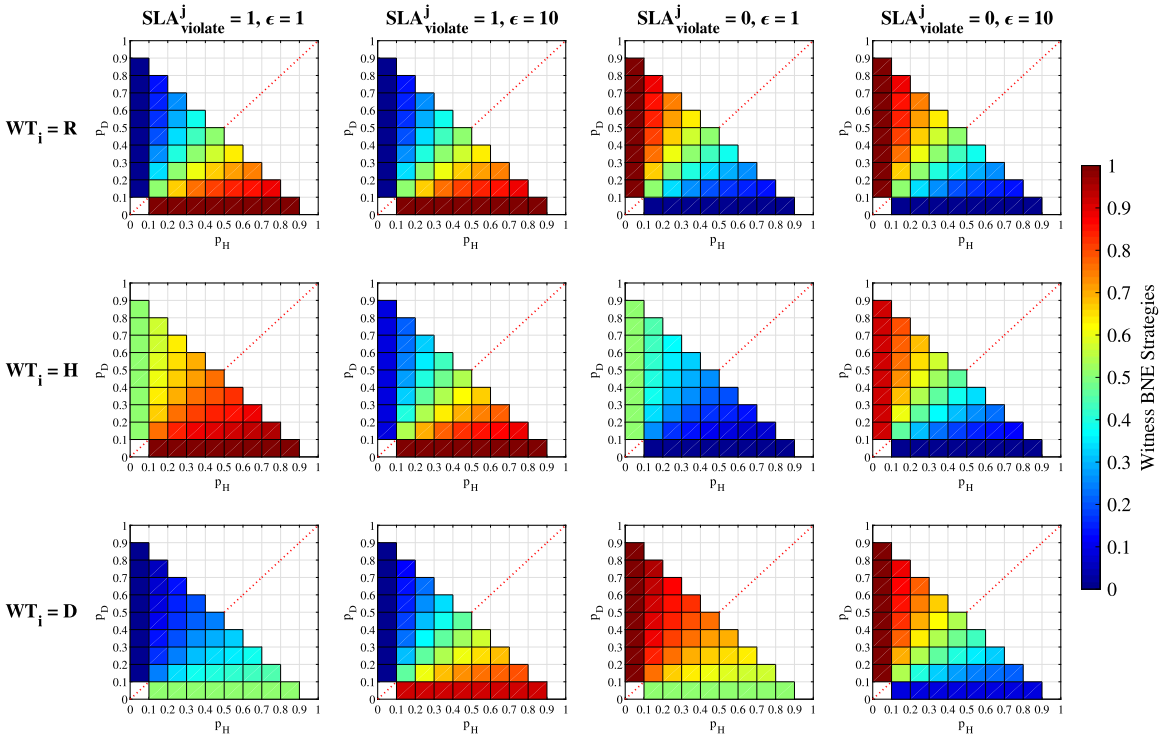


Fig. 5. Equilibrium strategies of the auction witnesses.

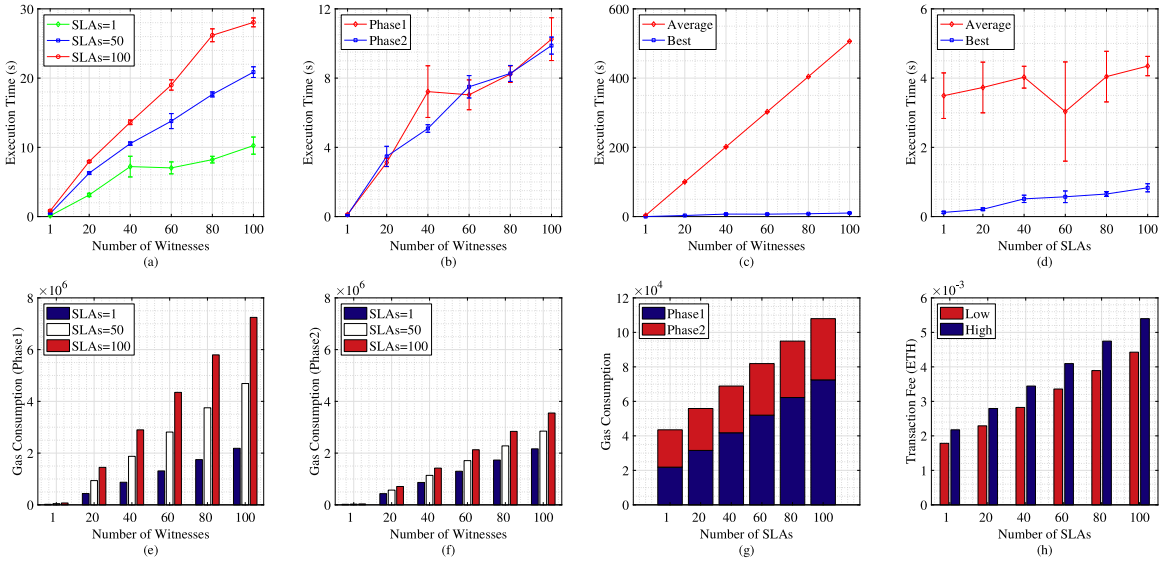
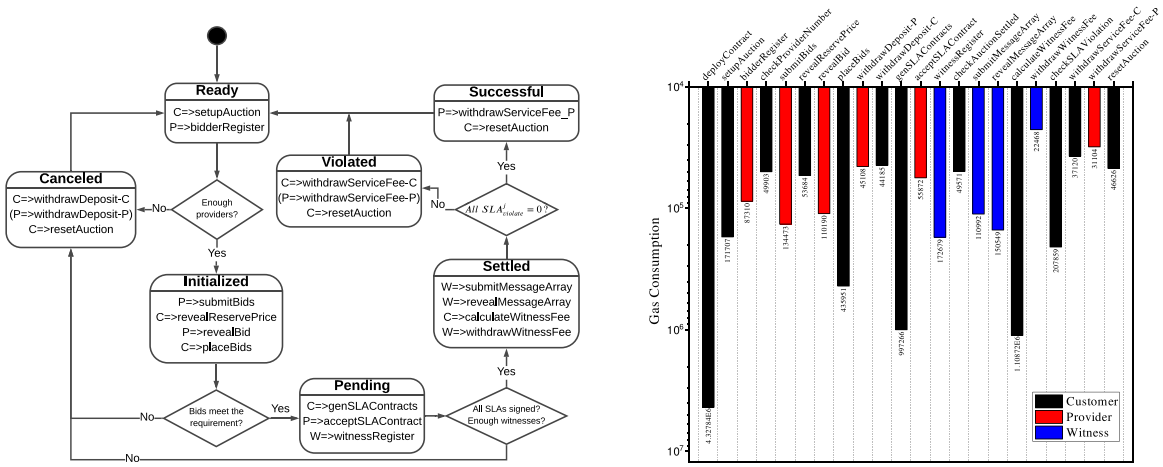


Fig. 6. Execution time and cost evaluation of the TMS algorithm.

of witnesses. The on-chain execution overhead, including execution time and cost, is tested in a local Ethereum blockchain. Specifically, time is calculated as the difference between block timestamps [40], and the cost is the gas consumption to perform transactions on the Ethereum blockchain. Different numbers of witnesses, as well as the SLAs they need to monitor, are tested. Two mining network congestion situations of the blockchain are simulated: "Best" means that there are enough miners who will process the transactions in time, while "Average" means that mining is congested and there is a delay in transaction processing.

Fig. 6 consists of eight plots. In the four plots above, plots (a) and (d) show that the execution time of the TMS algorithm increases linearly with the growth of witnesses and SLA numbers. Specifically, the algorithm execution time is less affected by the number of SLAs compared to the number of witnesses. When the witness number increases to 100, the execution time increases significantly. In contrast, when the SLA is increased by a factor of 100, the execution time increases only a little. Besides, plot (b) shows that the execution time of phases 1 and 2 are similar. Plot (c) indicates that the congestion of the blockchain mining network plays a critical role in the algorithm performance; it takes only a few seconds when the network is in the "best" condition but can last for several minutes when the network is congested (i.e., "average" condition).

The lower four plots of Fig. 6 demonstrate the total cost of the TMS algorithm and the average cost per user. First, plots (e) and (f) show that the total gas consumption for both phases of the TMS algorithm increases linearly with the number of witnesses



(a) State transition diagram (b) Gas consumption

Fig. 7. Implementation and evaluation of the auction smart contract.

and SLAs. By comparing the two graphs, it can be seen that when the number of witnesses is very high, the gas consumption of phase 1 is about twice that of phase 2. When the number of SLAs increases, the total cost of phase 1 increases more significantly. Next, plot (g) shows a linear increase in the average cost per user as the number of SLAs changes. When the SLA increases to 100 times, the gas consumption increases only by less than three times. In addition, plot (h) shows the predicted transaction costs of the TMS algorithm on the Ethereum main chain for two transaction speed cases.[9] The results show that the transaction fee increases by 21.95% when the requested transaction speed is changed to "high". However, the amount of Ether spent per user remains low, and in the worst case (i.e., SLA number is 100 and transaction speed is high), the algorithm spends less than $3. In summary, we argue that the execution overhead of the TMS algorithm is acceptable for each user compared to its improvement in model trustworthiness.

5.4. Smart contract implementation and evaluation

According to the architecture and payoff functions of our federated cloud auction model, we implement a prototype system based on the Ethereum blockchain with Solidity programming language.[10] Fig. 7(a) shows the state transitions of the auction smart contract. The rectangles in the figure represent different composite states, where the upper part shows the current state of the auction, and the lower part are the actions that can occur in this state. The *C*, *P*, and *W* next to the actions represent the customer, provider, and witness. We design different function interfaces to be initiated by different auction roles, and the initiator is the beneficiary with the greatest benefit at the current stage. The deployer of the smart contract (usually the customer) needs to check the current conditions and determine the confirmation of the state transfer. For money refunding, instead of automatically transferring money from the smart contract to users, we design that users need to withdraw the money by themselves. This is because when transferring money to multiple addresses, an attacker can trap the contract into an unusable state. In contrast, in the "withdrawal" mode, an attacker can only cause his or her own withdrawals to fail without affecting the rest of the contract's work [41,42]. Also, we omit the introduction of states in the SLA smart contracts for simplicity. SLA smart contracts are sub-contracts generated by the auction smart contract and are not the focus of this paper on the auction problem. Their functions and state machines can be customized by users themselves.

The auction smart contract can be expressed in seven states: "Ready", "Initialized", "Pending", "Settled", "Violated", "Successful", and "Canceled". The "Ready" state is automatically enabled when the auction smart contract is deployed on the blockchain. Within this state, the customer can set up an auction and promote the required services. Providers can then register as bidder candidates; when enough bidders register, the auction is "Initialized". In this state, registered providers can submit and reveal their bids. The customer also needs to reveal the reserve price and place all the bids by order. If there are enough bids to meet the customer's requirement, the auction comes into the "Pending" state. This means that the bidding phase has finished and the auction enforcement is pending. The customer must now invoke the interface to automatically generate SLA smart contracts for winning providers and wait for their acceptance. The auction is only "Settled" when all SLA smart contracts are signed and there are enough registered witnesses to monitor the SLAs. In this stage, witnesses monitor and submit their results and the customer calculates the witness fee for each witness according to the payoff functions. Finally, if all SLAs are performed as agreed, the auction state changes to "Successful". The providers can withdraw their own service fees. By contrast, if there are any violations occur, the auction state then converts to "Violated", and the customer can withdraw the prepaid service fee for specific violated providers. It should be noted that if any of the above auction conditions are not met, the sale is "Canceled". To retrieve the prepaid deposit, the customer and providers can use the "withdrawDeposit" interface. The customer can also use the "resetAuction" interface to reset the auction state to the "Ready" stage and wait for the next auction round.

It should be noted that the above process may cause a waste of resources for the customer and providers when they have already reached an agreement and there are not enough witnesses. However, we argue that such design is necessary and reasonable; if we let witnesses register in advance, this may also result in a waste of resources for witnesses, i.e., one may register as a witness (with a transaction fee on the blockchain) but not perform the monitoring task to win profits because the bidding fails. We chose the current design because the trustworthy enforcement of our model is based on witnesses' monitoring, and thus more witnesses should be incentivized to participate. To reduce the possibility of the waste of resources for the customer

9 https://etherscan.io/gastracker.

10 https://solidity.readthedocs.io/en/latest/.

Table 4
Transaction fee of each auction participant in a specific auction event.

Participant	Gas Consumption	Transaction Fee	USD
Customer	3202593 Gas	0.0342 ETH	$91.00
Provider	464057 Gas	0.0048 ETH	$12.76
Witness	456688 Gas	0.0050 ETH	$13.25

and providers in the current model, some measures can be taken. For example, an option could be added to extend the witness registration window to wait for more witnesses to join. The original witness fee can also be upgraded to provide an incentive for more witnesses to join within a specified time window.

The smart contract is tested using the Kovan[11] testnet, which is one of the most famous Ethereum testnets in the community. To evaluate the cost of a real cloud auction, we create several accounts, deposit some tokens (Ethers) in advance, and then simulate a federated cloud auction scenario with four providers and six witnesses. Fig. 7(b) reveals the detailed gas consumption of each interface, while Table 4 shows the total transaction fee of each auction participant (converted to US dollars). In general, the customer needs to invoke the largest number of interfaces (more than 50%) and consume the largest amount of gas, which is in line with our expectations. The customer is the beneficiary and initiator of the service auction and therefore should bear more commission charges. In fact, the customer not only acts as the auction publisher in the model, but also assumes the tasks of deploying contracts (consuming the most gas) and triggering some functions related to auction management, e.g., "setupAuction", "placeBids", and "genSLAContract". These tasks are performed by auctioneers or auction platforms in traditional auctions. Besides, since all the auction rules are openly hard-coded in the smart contract, any specific third party can initiate the smart contract while ensuring the fairness and credibility of the auction. It can also be found from Table 4 that the transaction fees of each witness and provider are not expensive (around $13), which ensures both parties have sufficient motivation to participate in the auction.[12]

We further compare the commission fees of our model with popular online auction platforms (eBay and eBid), as shown in Fig. 8. The eBay auction fee is 12.9% of the total sale ($7500 in maximum) plus an insertion fee ($0.35) and an additional handling fee per order ($0.30) [43]. In contrast, eBid's base fee is 5% of total sales [44]. The difference between the two is that while eBid is cheaper, eBay is generally considered to have a stronger market share and buyer base. As can be seen from the plot, the fees for eBay and eBid increase exponential as the service time and resource units increases. This is based on the fact that their fee mechanism is determined in proportion to the final sale price of the auction. In contrast, the customer, as the initiator of the auction, only pays a fixed commission fee (i.e., $91.00 in this case) in our blockchain-based model. This fee is independent of the final price of the auctioned services.

Fig. 9 shows the comparison of the fees when auctioning different applications and services. We selected five real scientific workflow applications provided by the Pegasus workflow gallery; their computational resources and running time can be found

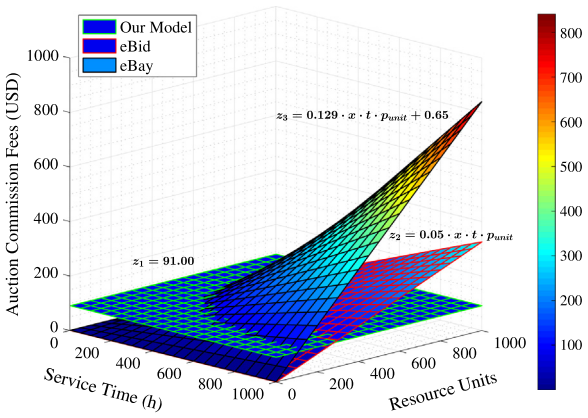


Fig. 8. Theoretical comparison of auction commission fees. The unit price of resources in the figure is illustrated using the cost of the smallest and cheapest Amazon EC2 instance t2.nano ($0.0058/h).

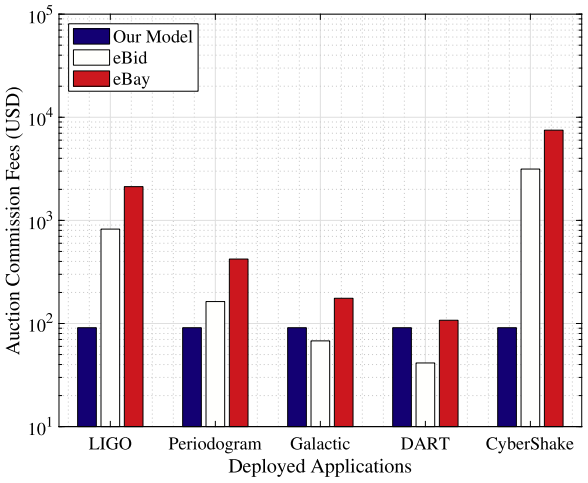


Fig. 9. Experimental comparison of auction commission fees.

in [45]. As observed from the figure, our model has different price advantages compared to eBay and eBid when deploying LIGO, Periodogram, and CyberShake applications. In the case of CyberShake, due to the need for a massive computing cluster and service time, our commission fees are just 2.89% and 1.12% of eBay and eBid, respectively. Whereas in Galactic and DART applications, choosing eBid would be more economical because of the small final auction price. In fact, when the price of the auctioned cloud service is less than the commission fee, our model will not be applicable to public blockchains like Ethereum. At this time, a fee-free permissioned blockchain can be used as an alternative, and the whole proposed model is still valid.

Finally, it is important to note that the price of Ether is volatile, and there is a payment risk for users. We thus consider the fluctuations in the history of Ether to USD. Fig. 10 shows the changing trend in fees for the three actors in our model over the last year. It can be observed that the provider and witness only pay a small fee with small changes. In November 2021, the transaction fee reached its maximum, making it more expensive for customers to initiate an auction. However, the recent trend of a significant decrease indicates that our model is more cost-effective compared to other online auction platforms.

11 https://kovan-testnet.github.io/website/.

12 The exchange rate between ETH and USD is changing continuously. When collecting the data, the exchange rate is 1 Ether = $2662 .08. Here the transaction fee is determined by both gas consumption and gas price. The difference in gas prices caused a larger transaction fee for witnesses than providers.

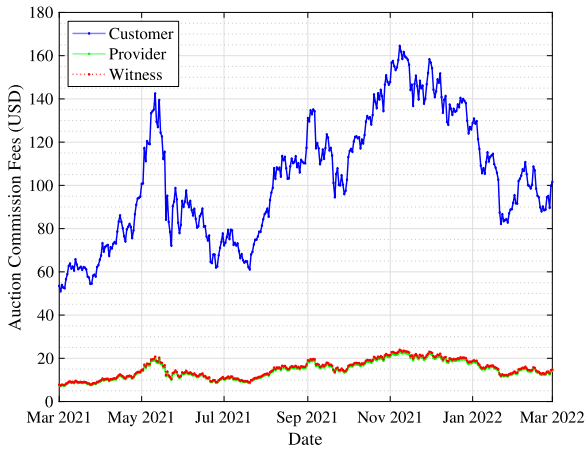


Fig. 10. Historical changes in auction transaction fees on Ethereum.

6. Discussion

In this section, we discuss the design choices and concerns of the proposed federated cloud auction model.

6.1. Model design concerns

In our federated cloud auction model, an off-chain balanced graph partition process is required at the beginning of the auction. This process guarantees each sub-provider bid on the same sub-service regarding price and VM size; therefore, the multi-object auction (services for multiple providers) can be implemented in just one round. We believe that this design is effective and can save extra execution costs on the blockchain. It should be noted that our model is also applicable in the case of an unbalanced partition; the only difference is that each sub-service needs to be auctioned independently, and therefore a k-round auction is required. Besides, a reverse sealed bid auction mechanism among federated cloud providers is used to conduct efficient and fair service auctions. We argue that such a reverse auction can motivate sellers' competition and bring substantial cost savings to buyers. It also helps streamline the auction process; auction time is saved because buyers do not need to send requests to different sellers one by one [46]. Other auction models (e.g., forward auction and open-cry auction) can also be integrated into the current model. However, those options increase the model complexity and require more execution costs on the blockchain [47].

6.2. Penalty and witness fee settings

The witnesses' penalty function factor ε directly affects the trustworthiness of auction agreement violation detection. According to the analysis results shown in Section 5, when ε is too small, the inherent psychological costs of witnesses dominate their utility equations. In this way, both honest and dishonest witnesses will make decisions according to their intrinsic types, e.g., honest witnesses tell truths and dishonest witnesses tell lies. This causes an inconsistent and untrustworthy monitoring result. To ensure that all witnesses tell the truth consistently, the value of ε must be large enough. At the same time, one basic assumption is that $F^{j}_{\text{witness}_i}$ and $\varphi_{\left(w_i^j\right)}$ are weakly balanced and $F^{j}_{\text{witness}_i} \geq \varphi_{\left(w_i^j\right)} = \frac{\varepsilon}{n-1}\sum_{i\neq k}\left(w_i^j - w_k^j\right)^2$. Otherwise, witness i may have a negative utility and cannot be motivated to join the monitoring. Thus the maximum value of ε also depends on the current witness fee $F^{j}_{\text{witness}_i}$. Normally we believe that setting the witness fee as 10%–30% of the service fee is a reasonable option and can be accepted by all auction participants.

6.3. Blockchain and cryptocurrency choices

There are many alternative blockchain technologies and platforms on the market. We design and implement our federated cloud auction model using the permissionless Ethereum blockchain to ensure that the auction process is completely decentralized and trustworthy. At the same time, the cryptocurrency Ether can be leveraged to execute the auction and witness fee payment. Some permissioned blockchain platforms (e.g., Hyperledger Fabric) have better scalability compared with permissionless ones, but their trust attributes are compromised (because of partial decentralization) and there is no stable cryptocurrency support.[13] We believe that the cryptocurrency itself has a volatility issue. Some emerging cryptocurrencies have tiny trading markets, so they may lack liquidity or may not be considered equivalent to fiat money in the auction. This is the main reason why we chose Ethereum and Ether to design the auction model.

6.4. Possible attacks

Blockchain technology is generally considered to be highly secure, but it may also suffer from some attacks, e.g., 51% attack and Sybil attack. Our blockchain-based federated cloud auction model suffers from those attacks as well. Taking the Sybil attack as an example, any parties of the auction (provider or customer) may try to control the auction/monitoring result by registering a large number of fake bidder/witness users. In response to this issue, we set a registration threshold (e.g., a non-refundable registration fee and a minimum reputation value) in the smart contract to limit arbitrary blockchain users from joining the auction. This mechanism partly guarantees that no party is able to register many malicious accounts because such an activity requires a large amount of money. Besides, the unbiased sortition algorithm proposed in our previous work [31] can also be used to select bidders/witnesses in a random and independent way, and to avoid possible unfairness or collusion.

6.5. Practical issues

Bayesian games are leveraged to build the proposed model and solve a practical problem (i.e., federated cloud auction). Therefore, our model is also subject to the limitations of game theory. For example, there is a basic assumption that players within the game will instinctively strive to maximize their payoffs. Bayesian games also assume that players have incomplete information about other players. However, these assumptions may be difficult to satisfy in real life since the player's decision is affected by complex factors, e.g., personal relationships and experiences. Therefore, our model may still have gaps when fully applied to reality. However, examples like the FCC auction model[14] have proved that game theory can indeed play a key role in guiding auction practices. We believe that our model could offer some new ideas for the current cloud auction research and industrial practices.

[13] https://cryptobriefing.com/hyperledger-fabric-forbes-50-blockchain/.
[14] https://www.nobelprize.org/prizes/economic-sciences/2020/popular-information/.

7. Conclusion

In this paper, a Bayesian game and blockchain-based auction model for federated cloud services is proposed. We leverage incomplete information game theory to analyze the bidding of providers and the SLA monitoring of witnesses, with two unique BNEs generated for two respective groups. The first BNE enables the selection of cost-effective and suitable service providers to construct the federated cloud services, while the second BNE ensures that the witnesses can report the truth about service violations consistently, which further makes the auction enforcement trustworthy. Finally, we evaluate the key techniques of the proposed model, and a prototype system using the Ethereum blockchain is fully implemented and verified. Our experimental results show that the prototype implementation is consistent with our model design expectations. Although we are targeting a cloud service auction scenario, the model can be easily extended to other service types as well.

The future work can be mainly targeted at two directions: smart contract and game theory. For the auction smart contract, we will continue to expand more functions and optimize the contract code to reduce transaction costs. In addition, we will try to improve the contract security to cope with possible attacks. Regarding the game theory, more complicated models, such as dynamic games and Perfect Bayesian Equilibriums (PBEs), will also be considered for integration with the proposed model.

CRediT authorship contribution statement

Zeshun Shi: Conceptualization, Methodology, Writing – original draft, Visualization. **Huan Zhou:** Methodology, Writing – review & editing, Supervision. **Cees de Laat:** Conceptualization, Resources, Supervision. **Zhiming Zhao:** Writing – review & editing, Methodology, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is funded by the European Union's Horizon 2020 research and innovation program under grant agreements 825134 (ARTICONF project), 824068 (ENVRI-FAIR project), and 862409 (BLUECLOUD project). The National Natural Science Foundation of China under Grant No. 62102434 and No. 62002364. The research is also supported by the China Scholarship Council, and EU LifeWatch ERIC.

References

[1] A.I. Middya, B. Ray, S. Roy, Auction based resource allocation mechanism in federated cloud environment: TARA, IEEE Trans. Serv. Comput. (2019) http://dx.doi.org/10.1109/TSC.2019.2952772, Early Access.

[2] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, M. Kunze, Cloud federation, in: International Conference on Cloud Computing, GRIDs, and Virtualization, IARIA, 2011, pp. 32–38.

[3] P. Klemperer, Auctions: Theory and Practice, Princeton University Press, 2004, http://dx.doi.org/10.2139/ssrn.491563.

[4] H. Zhang, H. Jiang, B. Li, F. Liu, A.V. Vasilakos, J. Liu, A framework for truthful online auctions in cloud computing with heterogeneous user demands, IEEE Trans. Comput. 65 (3) (2015) 805–818, http://dx.doi.org/10.1109/TC.2015.2435784.

[5] I. Bashir, Mastering Blockchain: Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, Packt Publishing Ltd, 2018.

[6] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008, https://bitcoin.org/bitcoin.pdf, (Accessed 01 November 2021).

[7] K.M. Khan, J. Arshad, M.M. Khan, Investigating performance constraints for blockchain based secure e-voting system, Future Gener. Comput. Syst. 105 (2020) 13–26, http://dx.doi.org/10.1016/j.future.2019.11.005.

[8] F. Casino, T.K. Dasaklis, C. Patsakis, A systematic literature review of blockchain-based applications: Current status, classification and open issues, Telemat. Inform. 36 (2019) 55–81, http://dx.doi.org/10.1016/j.tele.2018.11.006.

[9] A. Lei, Y. Cao, S. Bao, D. Li, P. Asuquo, H. Cruickshank, Z. Sun, A blockchain based certificate revocation scheme for vehicular communication systems, Future Gener. Comput. Syst. 110 (2020) 892–903, http://dx.doi.org/10.1016/j.future.2019.03.039.

[10] N. Kabra, P. Bhattacharya, S. Tanwar, S. Tyagi, MudraChain: Blockchain-based framework for automated cheque clearance in financial institutions, Future Gener. Comput. Syst. 102 (2020) 574–587, http://dx.doi.org/10.1016/j.future.2019.08.035.

[11] C. Braghin, S. Cimato, E. Damiani, M. Baronchelli, Designing smart-contract based auctions, in: International Conference on Security with Intelligent Computing and Big-Data Services, Springer, 2018, pp. 54–64, http://dx.doi.org/10.1007/978-3-030-16946-6_5.

[12] M. Kadadha, H. Otrok, R. Mizouni, S. Singh, A. Ouali, SenseChain: A blockchain-based crowdsensing framework for multiple requesters and multiple workers, Future Gener. Comput. Syst. 105 (2020) 650–664, http://dx.doi.org/10.1016/j.future.2019.12.007.

[13] X. Zhang, C. Wu, Z. Li, F.C. Lau, A truthful $(1-\epsilon)$-optimal mechanism for on-demand cloud resource provisioning, IEEE Trans. Cloud Comput. 8 (3) (2018) 735–748, http://dx.doi.org/10.1109/TCC.2018.2822718.

[14] S. Hosseinalipour, H. Dai, A two-stage auction mechanism for cloud resource allocation, IEEE Trans. Cloud Comput. 9 (3) (2021) 881–895, http://dx.doi.org/10.1109/TCC.2019.2901785.

[15] K. Metwally, A. Jarray, A. Karmouch, A distributed auction-based framework for scalable iaas provisioning in geo-data centers, IEEE Trans. Cloud Comput. 8 (3) (2018) 647–659, http://dx.doi.org/10.1109/TCC.2018.2808531.

[16] L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi, M. Huang, TCDA: Truthful combinatorial double auctions for mobile edge computing in industrial Internet of things, IEEE Trans. Mob. Comput. (2021) http://dx.doi.org/10.1109/TMC.2021.3064314, Early Access.

[17] S. Thakur, B.P. Hayes, J.G. Breslin, Distributed double auction for peer to peer energy trade using blockchains, in: International Symposium on Environment-Friendly Energies and Applications, EFEA, IEEE, 2018, pp. 1–8, http://dx.doi.org/10.1109/EFEA.2018.8617061.

[18] S. Xia, F. Lin, Z. Chen, C. Tang, Y. Ma, X. Yu, A Bayesian game based vehicle-to-vehicle electricity trading scheme for blockchain-enabled internet of vehicles, IEEE Trans. Veh. Technol. 69 (7) (2020) 6856–6868, http://dx.doi.org/10.1109/TVT.2020.2990443.

[19] A. Zavodovski, S. Bayhan, N. Mohan, P. Zhou, W. Wong, J. Kangasharju, DeCloud: Truthful decentralized double auction for edge clouds, in: Proc. IEEE ICDCS, Dallas, USA, 2019, pp. 2157–2167.

[20] K. Doka, T. Bakogiannis, I. Mytilinis, G. Goumas, CloudAgora: Democratizing the cloud, in: International Conference on Blockchain, Blockchain-2019, Atlanta, USA, 2019, pp. 142–156.

[21] S. Ghaemi, H. Khazaei, P. Musilek, ChainFaas: An open blockchain-based serverless platform, IEEE Access 8 (2020) 131760–131778.

[22] A. Sonnino, M. Król, A.G. Tasiopoulos, I. Psaras, AStERISK: Auction-based shared economy ResolutIon system for blockchain, 2019, arXiv:1901.07824.

[23] Z. Chen, W. Ding, Y. Xu, M. Tian, H. Zhong, Fair auction and trade framework for cloud VM allocation based on blockchain, 2020, arXiv:2001.00771.

[24] Y. Gu, D. Hou, X. Wu, J. Tao, Y. Zhang, Decentralized transaction mechanism based on smart contract in distributed data storage, Information 9 (11) (2018) 286.

[25] M. Debe, K. Salah, M.H.U. Rehman, D. Svetinovic, Blockchain-based decentralized reverse bidding in fog computing, IEEE Access 8 (2020) 81686–81697.

[26] B. Yu, Y. Chen, S. Fu, W. Yu, X. Guo, Building trustful crowdsensing service on the edge, in: Proc. Int. Conf. WASA, Hawaii, USA, 2019, pp. 445–457.

[27] iExec, iExec: Blockchain-based decentralized cloud computing, 2022, https://iex.ec/, (Accessed 01 November 2021).

[28] Golem, Golem: A decentralized cloud computing network, 2022, https://www.golem.network/, (Accessed 01 November 2021).

[29] Ethereum Community, Oracle services, 2021, https://ethereum.org/en/developers/docs/oracles/, (Accessed 01 November 2021).

[30] H. Matsushima, S. Noda, Mechanism Design with Blockchain Enforcement, 2020, http://dx.doi.org/10.2139/ssrn.3554512, Available at SSRN 3554512.

[31] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, Z. Zhao, A blockchain based witness model for trustworthy cloud service level agreement enforcement, in: IEEE International Conference on Computer Communications, INFOCOM, IEEE, 2019, pp. 1567–1575, http://dx.doi.org/10.1109/INFOCOM.2019.8737580.
[32] David Strom, How to improve container security, 2022, https://www.csoonline.com/article/3388025/how-to-improve-container-security.html, (Accessed 01 November 2021).
[33] A. Afshari, M. Mojahed, R.M. Yusuff, Simple additive weighting approach to personnel selection problem, Int. J. Innov. Technol. Manag. 1 (5) (2010) 511, http://dx.doi.org/10.7763/IJIMT.2010.V1.89.
[34] A. Henzinger, A. Noe, C. Schulz, ILP-based local search for graph partitioning, J. Exp. Algorithmics 25 (2020) 1–26, http://dx.doi.org/10.1145/3398634.
[35] S. Yuan, J. Wang, B. Chen, P. Mason, S. Seljan, An empirical study of reserve price optimisation in real-time bidding, in: ACM International Conference on Knowledge Discovery and Data Mining, SIGKDD, ACM, 2014, pp. 1897–1906, http://dx.doi.org/10.1145/2623330.2623357.
[36] S. Tadelis, Game Theory: An Introduction, Princeton University Press, 2013, http://dx.doi.org/10.1002/9781118547168.
[37] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, K. Vahi, Characterization of scientific workflows, in: 2008 Third Workshop on Workflows in Support of Large-Scale Science, 2008, pp. 1–10, http://dx.doi.org/10.1109/WORKS.2008.4723958.
[38] Alibaba Open Source, Alibaba Cluster Trace Program.
[39] G. Karypis, V. Kumar, METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, 2013, http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/manual.pdf, (Accessed 01 November 2021).
[40] N. Tapas, F. Longo, G. Merlino, A. Puliafito, Experimenting with smart contracts for access control and delegation in IoT, Future Gener. Comput. Syst. 111 (2020) 324–338, http://dx.doi.org/10.1016/j.future.2020.04.020.
[41] Solidity Documentation, Withdrawal from contracts, 2022, https://docs.soliditylang.org/en/v0.4.24/common-patterns.html#withdrawal-from-contracts, (Accessed 01 November 2021).
[42] Solidity Documentation, Security considerations, 2022, https://docs.soliditylang.org/en/v0.4.24/security-considerations.html#security-considerations, (Accessed 01 November 2021).
[43] eBay, Selling fees, 2022, https://www.ebay.com/help/selling/fees-credits-invoices/selling-fees?id=4822, (Accessed 01 November 2021).
[44] eBid, Fees and charges, 2022, https://www.ebid.net/us/help/fees-and-charges/, (Accessed 01 November 2021).
[45] Pegasus, Workflow gallery, 2022, https://pegasus.isi.edu/workflow_gallery/index.php, (Accessed 01 November 2021).
[46] I.A. Omar, H.R. Hasan, R. Jayaraman, K. Salah, M. Omar, Implementing decentralized auctions using blockchain smart contracts, Technol. Forecast. Soc. Chang. 168 (2021) 120786, http://dx.doi.org/10.1016/j.techfore.2021.120786.
[47] Z. Shi, C. de Laat, P. Grosso, Z. Zhao, When blockchain meets auction models: A survey, some applications, and challenges, 2021, Available at arXiv:2110.12534.

Zeshun Shi is currently pursuing his Ph.D. degree in the MultiScale Networked Systems (MNS) group, University of Amsterdam (UvA), Netherlands. He received his Master's degree with a focus on semantic web and ontology modeling from Beijing Normal University (BNU), China, in 2018. Since then, he has been working on research topics using blockchain technology and auction models for quality-critical cloud applications to provide trustworthy services. His current research interests include blockchain, auction theory, cloud computing, DevOps, and optimization solutions for infrastructure services.

Huan Zhou received the Ph.D. degree in computer science from the University of Amsterdam in 2019. He is currently a lecturer in School of Computer, National University of Defense Technology. His research mainly focuses on Cloud computing and network. He is specifically working on Cloud infrastructures seamless programming and control for orchestrating Cloud applications, as well as blockchain enhanced Cloud/Fog/Edge service management and secure network communications.

Cees de Laat is the chair of the System and Network Engineering laboratory at the University of Amsterdam. The SNE lab conducts research on leading-edge computer systems of all scales, ranging from global-scale systems and networks to embedded devices. His own work focuses on Secure Trusted Distributed Data Processing Systems. Prof. de Laat served on the Lawrence Berkeley Laboratory Policy Board for ESnet, the scientific advisory board of SURF, was (co-)founder of the Global Lambda Integrated Facility (GLIF), GRIDforum.nl, and CineGrid.org. He is a member of the Advisory Board Internet Society Netherlands. See: http://delaat.net/.

Zhiming Zhao received his Ph.D. in computer science in 2004 from the University of Amsterdam (UvA). He is currently a senior researcher in the MultiScale Networked Systems (MNS) group at UvA. He coordinates research efforts on quality critical systems on programmable infrastructures in the context of EU H2020 projects of ARTICONF, SWITCH, etc. His research interests include blockchain, SDN, workflow management systems, multi-agent systems, and big data research infrastructures.