



Hierarchical Service Composition via Blockchain-enabled Federated Learning

Li Huang¹ · Lu Zhao² · Yansong Liu³ · Yao Zhao⁴

Received: 11 February 2024 / Revised: 17 June 2024 / Accepted: 31 July 2024 / Published online: 24 August 2024
© The Author(s) 2024

Abstract

In recent years, the transformative evolution of cloud computing has reshaped organizational practices by enabling the outsourcing of web service applications. This shift has led to the emergence of the cloud environment, characterized by the involvement of Cloud Service Providers (CSPs) and intelligent applications. Cloud Service Composition (CSC) has become pivotal in this context, playing a crucial role in enhancing efficiency, Quality of Service (QoS), and customer satisfaction through the aggregation of diverse Cloud Services (CSs) to create composite services. However, the vast array of available CSs presents a challenge in efficiently addressing specified QoS requirements, turning CSC into a recognized *NP*-hard problem. Existing solutions, often involving third-party brokers, struggle with scalability in large-scale systems and overlook crucial security concerns. To address these limitations, we propose the Hierarchical Service Composition (HSC) approach, leveraging blockchain and federated learning to minimize computational complexity. The integration of Blockchain-enabled Federated Learning (BFL) facilitates machine learning model training with decentralized data, ensuring practicality and fairness. HSC comprises an initialization phase and two selection layers. The first selection layer enables each CSP to efficiently select services using a pre-trained model, while the second selection layer employs a blockchain-based QoS-aware mechanism for the final composition result, addressing privacy concerns. HSC introduces a novel framework, collaborative service selection methods, and a smart selection algorithm, demonstrating remarkable composition efficiency in extensive simulations compared to the baseline approach.

Keywords Service composition · Blockchain · Federated learning · Model-based

1 Introduction

In recent years, cloud computing has undergone a transformative evolution in the realm of information technology-based organizations. This evolution has enabled organizations to outsource web service applications, leading to a paradigm shift in established practices [1]. The cloud environment has redefined the deployment of web services, involving Cloud Service Providers (CSPs) and intelligent applications distributed over the internet. Several compelling factors drive the preference for cloud computing over traditional server-based computing, including cost efficiency, high speed, global scalability, enhanced productivity, superior performance, and reliability [2]. The multi-cloud environment involves the simultaneous utilization of multiple Cloud Services (CSs) offered by various CSPs and equipped with different settings, delivering a diverse set of services through a process known as Cloud Service Composition (CSC). Take Fig. 1 as an example: five CSPs collaborate to create a cloud-based

✉ Yao Zhao
cnr@deakin.edu.au

Li Huang
huangli713@126.com

Lu Zhao
luzhao@njupt.edu.cn

Yansong Liu
liuys221@163.com

¹ Jiangsu Open University, Nanjing, China

² College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China

³ Department of Computer Science, RWTH Aachen University, Aachen, Germany

⁴ School of Information Technology, Deakin University, Burwood, VIC, Australia

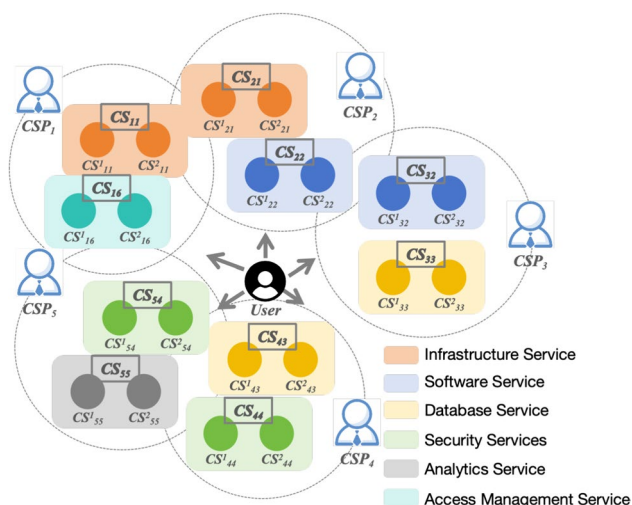


Fig. 1 Real example of cloud service composition. Five CSPs, labelled from CSP_1 to CSP_5 , offer various concrete services. For example, CSP_1 provides CS_{11} and CS_{16} , where CS_{11} provides two configurations CS_{11}^1 and CS_{11}^2 , while CS_{16} are also available in two configurations CS_{16}^1 and CS_{16}^2

customer relationship management system, with each CSP offering two types of specific services. For instance, CSP_1 provides Infrastructure Service CS_{11} and Access Management Service CS_{16} , each tailored with distinct resource configurations to meet varying user requirements. This setup is common in practice. For example, Alibaba Cloud¹ offers Object Storage Service with different storage capacities ranging from 40 GB to 3 PB. The CSC process begins with submitting a user workflow containing various tasks, each associated with Quality of Service (QoS) constraints such as response time, reliability, availability, and throughput. As a result, CSC algorithms play a pivotal role in improving efficiency, QoS, and customer satisfaction by aggregating numerous CSs to create composite services that meet specific requirements.

However, given the extensive array of available Cloud Services (CSs) that may fulfill specified Quality of Service (QoS) requirements, the resulting solution space becomes vast, presenting a challenge that cannot be efficiently addressed within polynomial time. Consequently, Cloud Service Composition (CSC) is acknowledged as an *NP*-hard problem [3]. Numerous innovative techniques, such as multi-criteria decision-making [4], fuzzy logic [5], and meta-heuristic algorithms [6], have been proposed to maximize QoS and tackle the CSC problem. However, these QoS-driven approaches face challenges in large-scale systems due to their computation-intensive nature and the *NP*-hardness of

problem-solving [7]. Furthermore, existing solutions often involve a third-party broker for CSC without adequately addressing potential security issues. In practice, relying on third-party brokers introduces trust concerns, as they may be motivated by additional financial or service-related gains, leading to the deliberate assertion of sub-optimal composition results [8].

To overcome the identified limitations, we propose a Hierarchical Service Composition (HSC) approach that leverages blockchain and Federated Learning (FL). HSC comprises an initialization phase and two selection layers, aiming to enhance efficiency by minimizing computational complexity. In the initialization phase, executed only once, two smart contracts are established, and a service selection model is trained using Blockchain-enabled Federated Learning (BFL). The integration of BFL is justified by its capability to facilitate machine learning model training with decentralized data, reducing communication overhead in local data transfer. Given the heterogeneity and security concerns, blockchain technology ensures practicality and fairness in FL model training, since no single CSP serves as a central authority. By leveraging blockchain technology, cloud service composition benefits from increased trust, transparency, security, and efficiency, resulting in more reliable and resilient cloud-based solutions. In the first selection layer, each CSP utilizes the pre-trained service selection model for efficient service selection, eliminating the need for complex optimization algorithms for CS selection. Considering that each selected CS may involve multiple resource settings, the second selection layer is adopted to achieve the final composition result (with specific resource settings) without involving third parties by using a blockchain-based QoS-aware selection mechanism, addressing potential privacy concerns. The incorporation of these layers significantly reduces overall computational overhead, leading to a substantial improvement in service composition efficiency.

The key contributions of this paper can be outlined as follows:

- We introduce a novel hierarchical service composition framework named HSC, which integrates BFL for large-scale systems. The primary objective of this framework is to efficiently select suitable services for creating composite services, resulting in a substantial improvement in service composition efficiency.
- To bolster the security of HSC, we devise a collaborative service selection method and a smart selection algorithm to attain final service composition results. This strategic approach ensures the adaptability of HSC in various operational scenarios.
- We conduct a comprehensive evaluation of the proposed HSC framework through extensive simulations. In comparison to the baseline algorithm, HSC consistently

¹ <https://eu.alibabacloud.com/>.

exhibits a remarkable increase in composition efficiency. These findings affirm the effectiveness of HSC in composing services.

The rest of this paper is organized as follows: Sect. 2 provides a review of related work. The system model is introduced in Sect. 3. Section 4 details the proposed algorithm, and its performance is extensively evaluated in Sect. 5. Finally, Sect. 6 concludes the paper with a summary and outlines future directions.

2 Related Work

Cloud service composition (CSC) plays a pivotal role in integrating individual cloud services into more intricate and valuable composite services, enabling the creation of comprehensive solutions tailored to meet specific user requirements. In distributed environments, where diverse independent cloud services collaborate to deliver enhanced functionality, CSC becomes crucial for achieving customization, efficiency, scalability, and innovation in cloud computing. It empowers users to craft bespoke solutions, optimize resource usage, and respond adeptly to dynamic business environments. Recently, numerous research efforts have focused on enhancing the CSC process, proposing various meta-heuristic optimization methods for this complex problem. In this study, with a particular emphasis on efficiency and security, especially when utilizing a diverse set of cloud services across cloud networks, we concentrate on two key aspects: (1) QoS-aware service composition and (2) secure-guarantee service composition. These aspects are further elaborated upon below.

2.1 QoS-aware Service Composition

To date, multiple works address the QoS-aware service composition under the inter-service constraints. For example, Tang et al. utilized a local optimizer to enhance individual quality and a knowledge-based crossover operator to address individuals violating constraints between services [9]. Gao et al. employed mathematical expressions to articulate constraints among services, employing integer programming to achieve service composition [10]. In addition, Lu et al. explored a semantic web-based scheme for cloud service composition in a dynamic manufacturing environment [11]. Zhou et al. put forth a hybrid differential artificial bee colony algorithm for cloud service composition [12]. Li et al. modeled the service composition problem as a multi-objective optimization problem, incorporating constraints between services and utilizing a multi-objective genetic algorithm for problem resolution [13].

Moreover, Ai et al. employed a penalty function method to penalize infeasible individuals violating constraints, enabling the elimination of infeasible individuals in the next generation [14]. Another approach by Ai et al. utilized domain knowledge to repair infeasible individuals, minimizing the number of services violating constraints [15]. Chen et al. presented a flexible QoS-aware web service composition method using multi-objective optimization in cloud manufacturing [16]. Wu et al. framed the problem as a multi-objective optimization issue, using a genetic algorithm (GA) for resolution. They also applied the penalty function method to penalize individuals violating constraints between services and calculated the similarity between individuals to maintain population diversity [17].

All these methods view service composition as the task of choosing one service from each service pool to create a composite service through multi-objective optimization methods. In general, due to the abundance and heterogeneity of services in the pool, service composition presents a challenging and complex *NP*-hard multi-objective optimization problem.

2.2 Secure-Guarantee Service Composition

The QoS-aware service composition problem has often overlooked security as a critical QoS factor. Many studies focusing on security have concentrated on filtering users' private data, with service composition not being their main objective. The emergence of Blockchain technology coincided with the introduction of the first-ever cryptocurrency, Bitcoin, in 2008 [18]. Bitcoin serves as a platform for value transformation, enabling anonymous individuals to directly transfer any amount of value without the need for intermediaries like national banks, governments, or notaries. Motivated by such benefits, blockchain is a promising technology for ensuring secure cloud service composition.

Cloud service composition architectures based on blockchain is proposed to improve information transparency and decentralization in cloud manufacturing systems. To name a few, Aghamohammadzadeh et al. improved solution feasibility by assigning the service composition task to the service composition provider and leveraging the distributed characteristics of blockchain [19]. Yet, they overlook the drawback of significant computational overhead associated with blockchain PoW and solving the *NP*-hard service composition problem, diminishing the scheme's practicality. In addition, Yu et al. proposed a blockchain-based service composition architecture, primarily focusing on enhancing transparency and dispersion of QoS information [20]. However, the service composition process is still managed by cloud data centers. Several studies have proposed PoW mechanisms.

Table 1 Key Notations

Notations	Definitions
α	QoS attribute weight, $0 < \alpha < 1$
λ	Penalty function adjustment factor
CS	Cloud service
CSP	Cloud service provider
CSC	Cloud service composition
HSC	Hierarchical service composition
BFL	Blockchain-enabled federated learning
MLP	Multi-Layer Perceptron
QoS	Quality of service
SSA	Smart selection algorithm
<i>AggSC</i>	Smart contract for model aggregation
<i>SelSC</i>	Smart contract running smart selection algorithm
CSP_i	<i>i</i> th CSP
Q_k	<i>k</i> th QoS attribute value
$Q(cs)$	Fitness function
CS_{ij}	<i>j</i> th CS in CSP_i
CS_{ij}^k	<i>k</i> th setting for <i>j</i> th CS in CSP_i
CS_{ij}^{k*}	Optimal <i>k</i> th setting for <i>j</i> th CS in CSP_i

For instance, Haouari et al. [21] introduced a proof-of-useful-work mechanism for a blockchain storing transportation transactions, demonstrating the applicability of a real-world *NP*-hard problem in PoW. However, none of these solutions address the efficiency issue, reducing the effectiveness of the schemes. Additionally, there is a notable absence of research solutions for constructing proof-of-useful-work mechanisms tailored for dynamic and multi-objective service composition in cloud computing. Overall, despite these advantages, researchers in blockchain technology have overlooked its application as a facilitator for CSC.

In summary, service composition plays a crucial role in achieving enhanced functionality, efficiency, and

adaptability in distributed computing environments. An effective solution is urgently needed for the success of cloud computing and other related paradigms.

3 System Model

We propose the basic composition workflow and formulate the CSC problem in this section. Key notations are listed in Table 1 for ease of reference.

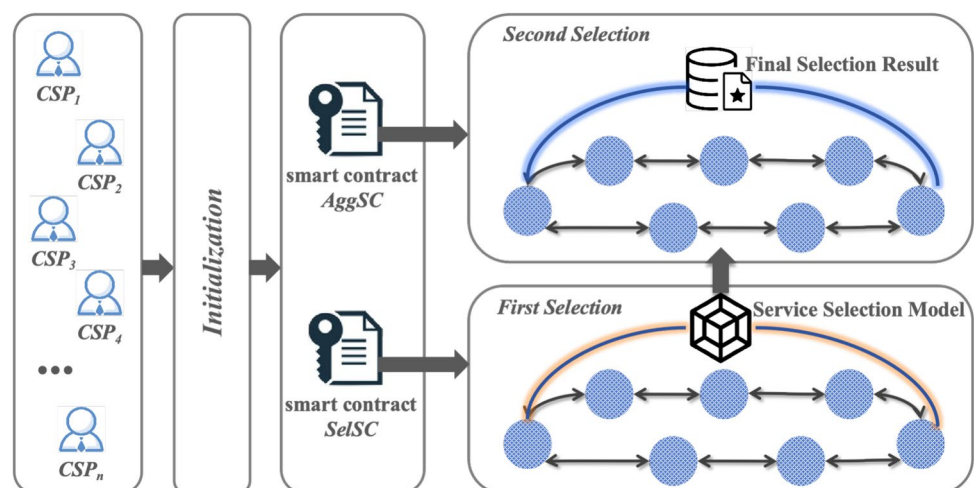
3.1 Composition Workflow

As depicted in Fig. 1, the Cloud Service Providers (CSPs) publish their Cloud Services (CSs) on the blockchain for users to access. When a user intends to utilize a combined services which are provided by various CSs, it submits a request to the blockchain to identify the most suitable CSs based on their requirements. Subsequently, the blockchain furnishes the necessary details for invoking the requested CSs. Assuming that there are multiple CSPs, denoted by $CSP_i \in CSP$ ($i \in \{1, 2, \dots\}$), each of which can offer one or more CSs denoted by CS_{ij} ($j \in \{1, 2, \dots\}$). In real world scenarios, CSP_i has multiple settings for each CS_{ij} , denoted by CS_{ij}^k . From the CSs provided by CSPs, one of the most matched services, CS_{ij}^{k*} , is selected for service composition.

The comprehensive composition workflow of our proposed HSC is depicted in Fig. 2, encompassing an initialization phase and two subsequent layers: the first selection layer and the second selection layer.

- *Initialization*: CSPs engage in negotiations for two smart contracts, namely *AggSC* and *SelSC*, which are subsequently deployed on the blockchain network. The global

Fig. 2 Comprehensive composition workflow with three phases: Initialization, First Selection, and Second Selection



service selection model is then collaboratively generated through the synergy of Federated Learning (FL) and blockchain, with distribution to each CSP. The incorporation of FL is motivated by the imperative to ensure privacy. Given CSPs' reluctance to disclose operational logs due to privacy concerns, FL facilitates distributed learning without compromising privacy. Furthermore, the integration of blockchain within the HSC system plays a crucial role in mitigating security vulnerabilities associated with central entities, such as central servers, during the FL training process.

- **First Selection:** In this phase, each CSP executes the deployed service selection model and transmits the unique IDs of selected CSs to smart contract *SelSC* for the second selection to determine the final resource settings. This layer significantly reduces the CSC problem complexity.
- **Second Selection:** This selection stage is introduced to further refine the selection of cloud services, thereby obtaining the final CSC result. For this purpose, a blockchain-based selection mechanism is employed. This mechanism achieves the second selection by leveraging the smart contract *SelSC* to collaborate on the selection process based on the service Quality of Service (QoS).

3.2 Problem Formulation

In the second selection layer, CSPs collaborate to produce the final selection result using the smart contract *SelSC* with a QoS-aware mechanism. To achieve this, we first model the QoS score for each CS (denoted by *cs*) to measure its corresponding QoS value by using a fitness function. Assume that a composite service consists of n abstract services. Each abstract service has a candidate service set that contains m concrete services (with the same functional attributes and different QoS attributes). Each concrete service has QoS attributes. CSC is to select a specific service from each set of candidate services to form a composite service to meet the requirements and restrictions put forward by users. Please note that a single CS provided by a particular CSP may come with various resource configurations, as introduced in Sect. 1. These configurations can be considered as distinct CSs in the second selection layer.

CSC has four types of structures, including Sequence, Switch, Cycle, and Parallel. The latter three types can all be converted into sequential types so we only discuss sequential types in this paper. We adopt the following four QoS attributes to describe service quality, namely: response time (*rt*), reliability (*re*), availability (*av*), throughput (*th*), which can be changed comprehensive expression of service characteristics. Since different QoS attribute values may differ by many orders of magnitude, the QoS value range

is normalized to $[0, 1]$. Specifically, for positive attributes (e.g., availability, reliability, and throughput), the larger the attribute value, the better the service quality, and the normalization process is performed according to Eq. (1); for the negative attributes (e.g., response time), the smaller the attribute value, the better the service quality, and the normalization process is carried out according to Eq. (11).

$$Q_k = \begin{cases} \frac{Q^k - \min Q^k}{\max Q^k - \min Q^k} & \max Q^k \neq \min Q^k \\ 1 & \max Q^k = \min Q^k \end{cases} \quad (1)$$

$$Q_k = \begin{cases} \frac{\max Q^k - Q^k}{\max Q^k - \min Q^k} & \max Q^k \neq \min Q^k \\ 1 & \max Q^k = \min Q^k \end{cases} \quad (2)$$

where $\max Q^k = \max\{Q^1, Q^2, Q^3, \dots\}$ denotes the maximum value of the QoS attribute across all CSs, while $\min Q^k = \min\{Q^1, Q^2, Q^3, \dots\}$ represents the minimum value of the QoS attribute of across CSs. Additionally, Q^k represents the value of the k th QoS attribute.

According to the characteristics of different QoS attributes, the quality of service QoS aggregation function has the following three types: sum, multiply, and minimum operation characteristics.

1) **Sum:** Some QoS attributes of the combined service are the accumulation of the QoS attributes of the specific services or the average value of the accumulation, e.g., Response Time. Specifically, the response time attribute refers to the time interval between the user making a service request and delivering the result to the user, including the total service processing time. The combined service response time performance function is:

$$ResponseTime = Q_{rt}(cs) = \sum_{i=1}^n q_{rt}(i). \quad (3)$$

2) **Multiply:** Some QoS attributes of the composite service, such as reliability and availability, are multiplied by the QoS attributes of its specific services.

- **Reliability.** Reliability properties refer to the ability of a composite service to correctly execute user requests. The reliability performance function of the combined service is:

$$Reliability = Q_{re}(cs) = \prod_{i=1}^n q_{re}(i). \quad (4)$$

- **Availability.** The availability attribute refers to the probability that the composite service can operate normally. The availability performance function of the composite service is:

$$Availability = Q_{av}(cs) = \prod_{i=1}^n q_{av}(i). \quad (5)$$

3) *Minimum*: Some QoS attributes of the composite service are obtained by taking the minimum value of the QoS attributes of the specific service. For example, the throughput attribute refers to the number of successfully executed combined services per unit time. The throughput performance function of the composite service is:

$$Throughput = Q_{th}(cs) = \min\{q_{th}(i)\}_{i=1}^n. \quad (6)$$

Using a simple weighting method, the total performance function of the combined service can be expressed as:

$$Q(cs) = \sum_{k \in \{rt, re, av, th\}} \alpha_k Q_k(cs), \quad (7)$$

where k is a certain QoS attribute; α is the weight of different QoS attribute values ($0 < \alpha < 1$). The larger the value of $Q(cs)$, the better the overall quality of the combined service.

To facilitate global optimization, a penalty function is introduced, thereby transforming the constrained optimization problem into an unconstrained optimization problem. The fitness function is:

$$Q(cs) = \lambda \times \sum_{k \in \{rt, re, av, th\}} \left(\frac{\Delta Q_k}{Q_k^{max} - Q_k^{min}} \right)^2 - Q(cs), \quad (8)$$

where λ is the penalty function adjustment factor, determined by empirical values; Q_k^{max} is the maximum value for the constraint conditions; Q_k^{min} is the minimum value for the constraint conditions; ΔQ_k is defined as follows.

$$\Delta Q_k = \begin{cases} Q_k - Q_k^{max} & Q_k \geq Q_k^{max} \\ 0 & Q_k^{min} < Q_k < Q_k^{max} \\ Q_k^{min} - Q_k & Q_k \leq Q_k^{min} \end{cases} \quad (9)$$

To sum up, it can be seen that the goal of CSC is to select a specific service from each candidate service set to form a composite service, so as to minimize the transformed $Q(cs)$ shown in Eq. (8).

4 HSC Framework

In this section, we offer a detailed explanation of the proposed framework for hierarchical cloud service composition, referred to as HSC.

Input: CSP set CSP

Output: selected cloud service set S

```

// Initialization
1:  $SS \leftarrow \emptyset$ , initialize  $w_0$ 
2: build and install two smart contracts  $AggSC$  and  $SelSC$ 
   into blockchain.
3: for each training round  $t = 1, 2, \dots$  do
4:   for each  $CSP \in CSP$  in parallel do
5:      $w_{t+1}^k \leftarrow Local\_Update(k, w_t)$ 
6:   end for
7:    $w_{t+1} \leftarrow \sum_{k=1}^{|SS|} \frac{n_k}{n} w_{t+1}^k$ 
8: end for
9:  $Model \leftarrow Service\_Selection(w_{t+1})$ 
// First Selection
10: for  $cs_j \in CS$  do
11:    $SS \leftarrow SS \cup Model(cs_j, data_j)$ 
12: end for
// Second Selection
13: for  $ss_i \in SS$  do
14:    $S \leftarrow$  QoS-aware selection mechanism
15: end for
16: return  $S$ 

```

Procedure 1 Service Composition Framework: HSC

The core process of HSC is depicted in Procedure 1, involving an initialization phase and two subsequent layers: the first selection layer and the second selection layer. In the initialization phase, essential parameters and two smart contracts, $AggSC$ and $SelSC$, are negotiated and deployed onto the blockchain. $AggSC$ oversees FL training, while $SelSC$ manages the final selection. Blockchain-based FL, coordinated by smart contract $AggSC$, trains a service selection model $Model$ by leveraging metadata from local logs to identify suitable cloud services for users (lines 3–9). In the first selection layer, each CSP independently executes the pre-trained global model $Model$ to securely identify the most suitable local cloud services CS (lines 10–12). It is important to note that model training and aggregation methods can be chosen flexibly according to the usage environment. In the second selection layer, smart contract $SelSC$ performs the final selection from $ss_i \in SS$ to generate the ultimate CSC result S (lines 13–15). Finally, the set of ultimately selected services S is returned (line 16). Each layer is explained in detail as follows.

4.1 Initialization

In this subsection, we explain how to generate a service selection model using the intrinsic attributes of users' activity data with the smart contract $AggSC$. Notably, the model training occurs only once during the initialization stage, allowing it to be used for selecting services across multiple rounds within a

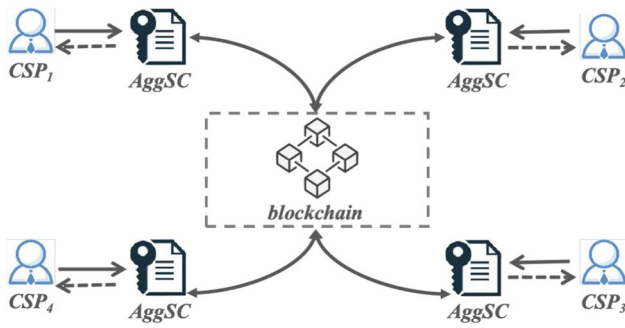


Fig. 3 Blockchain-based Federated Learning. Four CSPs from CSP_1 to CSP_4 collaboratively train a global model by using smart contract AggSC

specified period without additional computational overhead. This one-time computation cost is acceptable for long-term usage. Primarily, model-based selection methods, such as statistical [22–24], clustering [25–27], and classification-based [28–30] mechanisms, construct a profile for each data instance. Given the unique characteristics of cloud environments and the imperative for privacy and security, we integrate FL and blockchain to achieve an efficient model generation approach, as depicted in Fig. 3.

Before commencing model training, it is essential to define crucial parameters such as the number of communication rounds, pre-defined loss function, initial gradients, convergence criterion, learning rate, and model aggregation method. To facilitate scalable parameter negotiation, we employ a distributed parameter negotiation mechanism. In this mechanism, clients communicate and collaboratively determine these parameters, considering their individual capabilities and the prevailing network conditions [31]. As demonstrated in [31], this negotiation mechanism scales effectively with an increasing number of participating CSPs, ensuring a seamless and efficient initialization phase for blockchain-based FL in cloud environments.

Each CSP is treated as an FL client. Assuming that each CSP possesses its on-device metadata dataset $D_{CSP} = \{x_h, y_h\}_{h=1}^{|D_{CSP}|}$, where $|D_{CSP}|$ indicates the total number of data samples in the dataset on CSP, x_h denotes the h th sample, and y_h is the corresponding label [32]. D_{CSP} is derived from local logs, capturing the actions performed by served users during a specific timeframe. Without loss of generality, the FL training process aims to minimize the global loss function, i.e.,

$$\min_{x \in \mathbf{R}^d} F_{CSP}(x) = \frac{1}{D_{CSP}} \sum_{h \in D_{CSP}} \mathbf{E}_{z_h} (f(x; z_h) + \lambda h(x)), \quad (10)$$

where $f(\cdot; \cdot)$ represents the local loss function for CSP, $\forall \lambda \in [0, 1]$, $h(\cdot)$ is a regularizer function for CSP, and $\forall h \in [1, \dots, |D_{CSP}|]$, z_h is sampled from the local dataset D_{CSP} on CSP.

In the model update phase, each CSP_i updates its local model $w_i(t)$ in round t according to the stochastic gradient descent method [33]. The update process in CSP is expressed as follows.

$$w_i(t) = w_i(t-1) - \eta \nabla f_{CSP}(w_i(t-1); D_{CSP}(t-1)), \quad (11)$$

where $w_i(t-1)$ is the local model in round $t-1$ and η is a learning rate.

In the local model aggregation phase, smart contract AggSC employs Federated Averaging [34] in a decentralized manner (referred to as Dec-FedAVG) to derive a new global model $\mathcal{W}$ for the next iteration. Dec-FedAVG is shown in Procedure 2.

Input: local model set W

Output: aggregated global model $\mathcal{W}$

```

1: for  $w \in W$  do
2:    $\mathcal{W} = \sum_{i=1} \alpha_i w$ 
    $\alpha_i \geq 0$  is the weighted average coefficient, satisfying
    $\sum_{i=1}^n \alpha_i = 1$ 
3: end for
4: return  $\mathcal{W}$ 
```

Procedure 2 Dec-FedAVG (run by AggSC)

CSPs iterate through the mentioned process until the global model converges. Since various scenarios may require different integration approaches involving blockchain and FL, we do not provide an exhaustive implementation guide for this integration. A comprehensive description of BFL can be found in [35]. It is worth noting that research [36] has validated that only 0.1% of gradients with the largest absolute value are useful for gradient aggregation. Thus, gradient compression mechanisms [37] can be applied to reduce communication overhead during FL training. After establishing the global model, each CSP can utilize it as a service selection model, performing the first selection for efficient CSC.

4.2 First Selection

Each CSP employs the service selection model pre-trained during the initialization stage. This enables the identification of potentially suitable services without incurring complex computational overhead. Importantly, utilizing a pre-trained model for service selection is computationally more efficient compared to employing optimization methods directly.

4.3 Second Selection

In the second selection phase, we utilize smart contract *SelSC* to run a Smart Selection Algorithm (SSA), aiming to select the most suitable cloud services for users, forming a composite service [8].

Input: cloud service set CS

Output: selected service set SS

```

1:  $Q \leftarrow \emptyset$ 
   \ \ QoS score calculation
2: for  $cs_i \in CS$  do
3:    $Q \leftarrow Q \cup Q(cs_i)$ 
4: end for
   \ \ cloud service selection
5: function FINDOPTIMAL( $Q, u, v, w$ )
6:    $temp = \text{randint}(u, v)$ 
7:    $Q[|Q|], Q[temp] = Q[temp], Q[|Q|]$ 
8:    $a = b = u$ 
9:   while  $b < |Q|$  do
10:    if  $Q[b] > Q[|Q|]$  then
11:       $Q[a], Q[b] = Q[b], Q[a]$ 
12:       $a+ = 1$ 
13:    end if
14:     $b+ = 1$ 
15:  end while
16:   $Q[a], Q[b] = Q[b], Q[a]$ 
17:  if  $|Q| > w + a - 1$  then
18:    return FINDOPTIMAL( $Q, a + 1, |Q|, w$ )
19:  else if  $|Q| < w + a - 1$  then
20:    return FINDOPTIMAL( $Q, 0, a - 1, w - (|Q| - a + 1)$ )
21:  else
22:    return  $Q[a : ]$ 
23:  end if
24: end function
25:  $SS = \text{FINDTOP}(Q, 0, |Q|, 1)$ 
26: return  $SS$ 

```

Procedure 3 Smart Selection Algorithm: SSA (run by *SelSC*)

SSA is outlined in Procedure 3. It begins by initializing set Q to cache intermediate variable (line 1). Subsequently, it calculates the Quality of Service (QoS) score $Q(cs_i)$ for the services chosen from the first layer and cache them in Q (lines 2-4). In lines 5-24, the algorithm employs the randomized-select algorithm to identify the most suitable services with the minimum QoS score for each user and stores the corresponding unique IDs of the services in the selected service set SS . Specifically, it starts by generating a random number $temp$ within the range $[u, v]$, where u and v denote the start and end positions of the set Q , respectively. Subsequently, the set Q is partitioned into two subsets, possibly empty, denoted as $[u, temp - 1]$ and $[temp + 1, v]$. In this partition, each element in $[u, temp - 1]$ is less than or equal to $Q[temp]$, and, conversely, each element is less than every element in $[temp + 1, v]$. The algorithm then checks whether $Q[temp]$ represents the smallest element. If so, it returns the

unique IDs of the smallest elements. Otherwise, it determines whether the smallest element lies in the low subset $[u, temp - 1]$ or the high subset $[temp + 1, v]$. If $|Q| > a$, indicating that the expected element is in the high subset, the algorithm chooses it from the high subset recursively. On the other hand, if $|Q| < a$, signifying that the expected element is in the low subset, the algorithm selects it from the low subset.

Theorem 1 Compared to skipping the first selection, the possibility of getting the optimal composite service can be increased by n^k times through both the first and second selections.

Proof Notably, the first selection identifies potentially suitable services, while the second selection determines the specific configuration of these services. Supposing a composite service involving k concrete services, where there are n CSPs providing the same concrete service, each equipped with m configurations. Therefore, each concrete service presents $n \cdot m$ options. The possibility Pr of achieving the optimal composite service is expressed by

$$Pr = \left(\frac{1}{n \cdot m} \right)^k. \quad (12)$$

When the user performs the first selection, the possibility $\overline{Pr}$ of attaining the optimal composite service through both selections is given by

$$\overline{Pr} = \left(\frac{1}{m} \right)^k. \quad (13)$$

Incorporating the first selection results in the possibility of obtaining the optimal composite service increasing by a factor of n^k , as illustrated by

$$\frac{\overline{Pr}}{Pr} = \frac{\left(\frac{1}{m} \right)^k}{\left(\frac{1}{n \cdot m} \right)^k} = n^k. \quad (14)$$

Incorporating the first selection significantly reduces complexity. For instance, assuming $k = 5$, $n = 10$, and $m = 3$, we have $Pr = (\frac{1}{30})^5$ and $\overline{Pr} = (\frac{1}{3})^5$, making $\overline{Pr}$ 100,000 times greater than Pr . This concludes the proof. $\square$

4.4 Framework Demonstration

To demonstrate the application of our proposed HSC framework, we present an example using Fig. 1, where there are five CSPs denoted as CSP_1 to CSP_5 , each offering specific cloud services with multiple settings. For

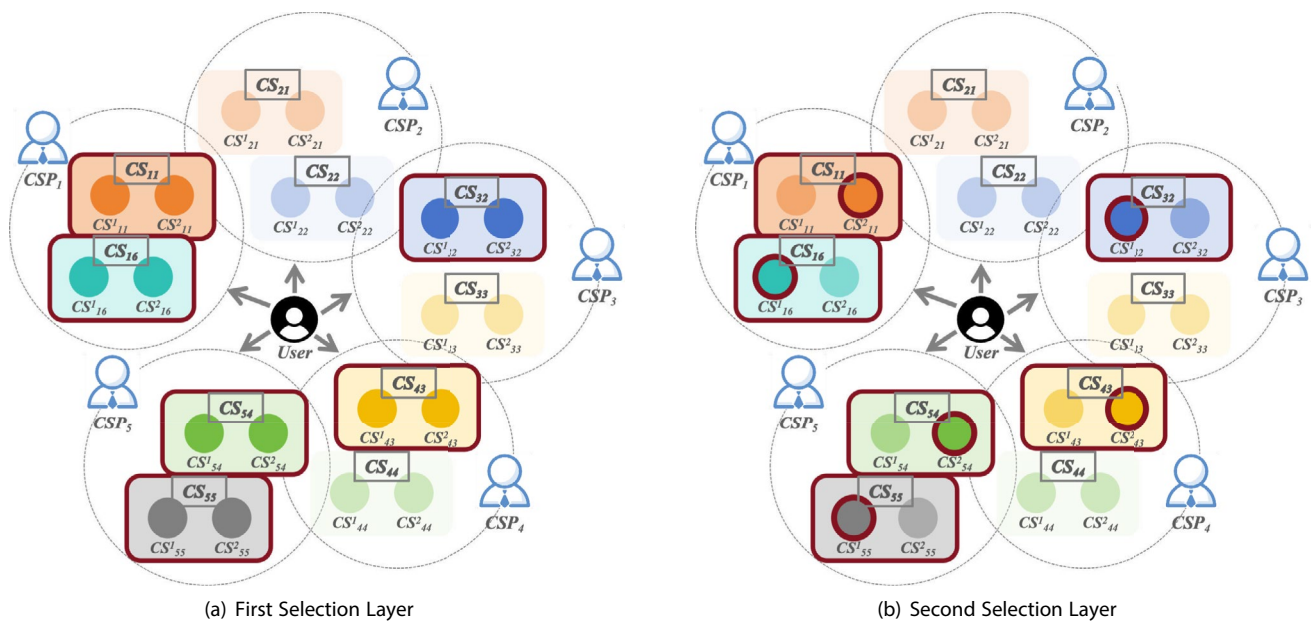


Fig. 4 Framework Demonstration. A user anticipates acquiring a composite service composed of 6 cloud services offered by 5 cloud service providers, denoted as CSP_1 , CSP_2 , CSP_3 , CSP_4 , and CSP_5

instance, CSP_1 offers cloud servers CS_{11} and CS_{16} , where CS_{11} has two settings (CS_{11}^1 and CS_{11}^2), and CS_{16} has two settings (CS_{16}^1 and CS_{16}^2).

During the initialization phase (performed only once), CSPs, ranging from CSP_1 to CSP_5 , agree on essential parameters such as the number of communication rounds, pre-defined loss function, initial gradients, convergence criterion, learning rate, and model aggregation method. Subsequently, each CSP processes and analyzes data extracted from logs to construct training datasets. These datasets incorporate information such as user access data and Quality of Service (QoS) measurements across various observed time intervals. Leveraging this accumulated information, each CSP collaborates with others to develop a global model through a standard BFL training process by using smart contract *AggSC* which is shown in Procedure 2. It employs appropriate model training techniques like Multi-Layer Perceptron (MLP) or Long Short-Term Memory (LSTM), depending on specific usage scenarios [38].

Following the initialization phase, each CSP acquires a service selection model. In Fig. 4(a), during the first selection phase, CSPs utilize this model to evaluate recent data extracted from logs, generating an initial selection. In this case, the selected cloud services include CS_{11} and CS_{16} from CSP_1 , CS_{31} from CSP_3 , CS_{43} from CSP_4 , and CS_{54} and CS_{55} from CSP_5 . Subsequently, the outcomes of the first selection are communicated to each CSP for further processing.

Determining the optimal settings for the selected cloud services relies on the crucial second selection phase,

which employs smart contract *SelSC* outlined in Procedure 3. During this phase, CSPs independently execute the smart selection algorithm (SSA) and transmit the final selection outcomes to the blockchain. The resulting selection is then communicated to the user. As illustrated in Fig. 4(b), the final selection outcomes comprise CS_{11}^2 and CS_{16}^1 from CSP_1 , CS_{31}^1 from CSP_3 , CS_{43}^2 from CSP_4 , and CS_{54}^1 and CS_{55}^1 from CSP_5 .

5 Experiments and Result Analysis

In this section, we conduct extensive experiments on a real-world dataset to assess the effectiveness of our proposed framework HSC in comparison to an alternative approach.

5.1 Dataset

To assess the performance of HSC, we utilize a real-world *QWS* dataset [39] containing 2507 web services along with their associated QoS values. This dataset serves as the basis for calculating the QoS score $Q(\cdot)$ for cloud services (CSs). We focus on four QoS attributes: response time, reliability, availability, and throughput. Among these, reliability, availability, and throughput are considered positive attributes, while response time is a negative attribute. This attribute selection ensures coverage of various scenarios related to different types of QoS attributes, directly influencing users' seamless access to services. Since practical datasets for service selection are not readily available, we

Table 2 Parameter settings

		k	m	n	d
Set #1	Set #1.1	{1, 2, 3, 4, 5}	3	10	/
	Set #1.2	3	{1, 2, 3, 4, 5}	10	/
	Set #1.3	3	3	{2, 5, 10, 15, 20}	/
Set #2	Set #2.1	/	/	{2, 5, 10, 15, 20}	1000
	Set #2.2	/	/	10	{100, 500, 1000, 2000, 4000}

illustrate the training process of BFL using the KDD99 dataset, a widely-used anomaly detection dataset. This demonstration aims to showcase a typical BFL training process, drawing an analogy between service selection and anomaly detection.

5.2 Experimental Setting

To assess the effectiveness of our proposed framework, we perform a comparative analysis with the baseline genetic algorithm (GA) and the state-of-the-art shuffled frog leaping algorithm (SFLA), focusing on efficiency. To demonstrate the advantages of integrating BFL, we evaluate the framework based on the following metrics:

- *Computation Overhead*: This metric is calculated by the time taken to finish CSC. A lower computation overhead reflects more efficient processing.
- *Average Accuracy*: This is determined by averaging the accuracy of the local models across all CSPs within a fixed time interval. A higher average accuracy is indicative of better performance.

We change three parameters to simulate various scenarios, namely the number of settings within each CS (k), the number of CSs for each CSP (m), and the number of CSPs (n). In experiments related to BFL, we use d to represent the number of training data samples for each CSP. Supposing that a composite service consists of n abstract services. Each abstract service has a candidate service set that contains $k \times m$ concrete service. Table 2 provides an overview of the parameter configurations. The detailed settings are as follows: the number of local epochs is set to 5, the training round is set to 20, and the learning rate is set to 0.01. α is set to 0.25. Multi-Layer Perceptron (MLP) is employed consistently across all experiments. MLP stands as a frequently utilized method in related work, as demonstrated in references such as [40–42]. Each experiment is repeated 100 times, and the results are averaged. All experiments are conducted on a machine equipped with an Intel Core i5 CPU @ 1.8GHz and 8GB RAM, implemented using Python 3.6.

5.3 Performance Comparison

Next, we report the experimental results.

5.3.1 Computation Overhead

To evaluate the efficiency of our proposed HSC, we compare it with a baseline approach GA and a state-of-the-art approach SFLA, varying the number of settings within each CS k , the number of CSs for each CSP m , and the number of CSP n .

-Effect of k in Set #1.1. Figure 5 illustrates the impact of different approaches as the number of settings, denoted by k , for each cloud service, increasing from 1 to 5. A logical correlation is expected, where a larger k implies the involvement of more cloud services. As depicted in Fig. 5, HSC consistently outperforms GA and SFLA in terms of efficiency across all instances, showcasing a significant performance advantage. This difference can be attributed to the fact that HSC initially employs a model to select the most suitable cloud services, resulting in the use of a smaller set of cloud services for the final decision. In contrast, GA and SFLA need to explore the entire solution space to find the optimal solution, inevitably incurring more computation overhead.

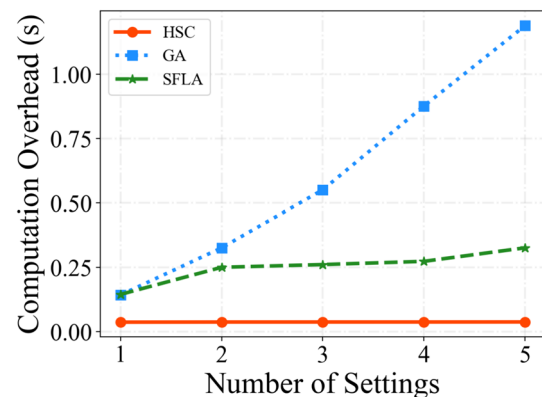


Fig. 5 Effect of k (Set #1.1). HSC achieves the best performance in computation overhead when varying k

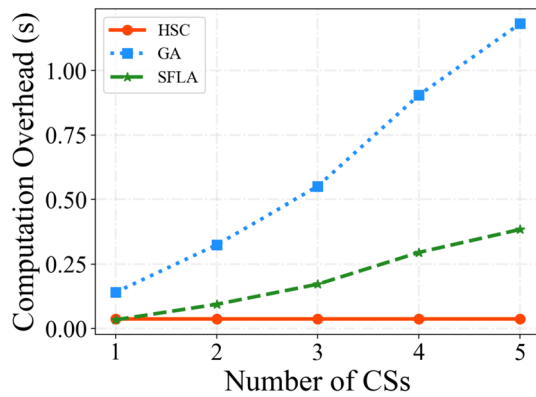


Fig. 6 Effect of m (Set #1.2). The parameter m only exerts an impact on GA and SFLA. HSC achieves the best computation performance

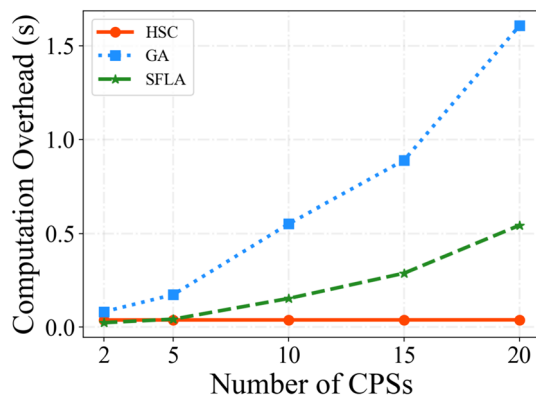


Fig. 7 Effect of n (Set #1.3). HSC is more efficient, gradually expanding computation increment difference with n increasing

Moreover, with fewer cloud services to select, the final decision-making process in HSC requires less time. The varying outcomes with different values of k underscore the practicality of HSC, confirming its potential to significantly reduce computation overhead for CSC. This outcome unequivocally confirms the effectiveness of our proposed framework. On average, HSC outperforms GA and SFLA by factors of 15.2 and 5.6 across all instances, respectively. Importantly, the scalability of HSC is evident, as its performance remains consistent even with higher values of k . Consequently, HSC emerges as a suitable solution for large-scale CSC scenarios.

-Effect of m in Set #1.2. Figure 6 presents the experimental results for varying the parameter m from 1 to 5 in Set #1.2. In all cases, HSC consistently demonstrates the lowest overall computation overhead. It is worth highlighting that the parameter m affects only GA and SFLA, as HSC employs a model to pre-select the most suitable CSs. On average across all cases, HSC outperforms GA and SFLA by a factor of 15.2 and 4.1, respectively.

-Effect of n in Set #1.3. Figure 7 illustrates the computation overhead of each approach as the number of CSPs n

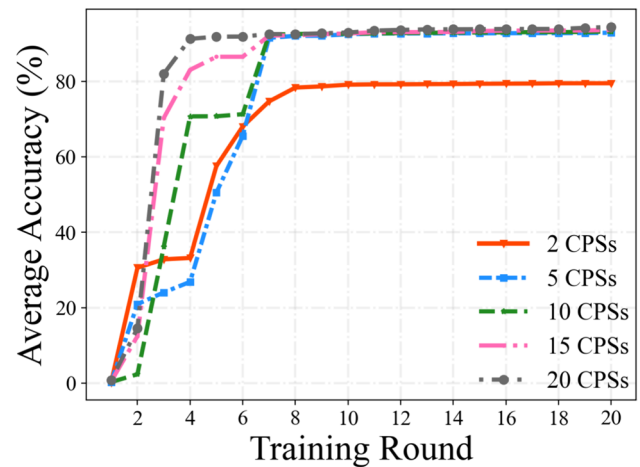


Fig. 8 Accuracy vs. n (Set #2.1). Higher model accuracy is achieved with n increasing

increases from 2 to 20. The results underscore GA's and SFLA's sensitivity to larger values of n compared to HSC. Across all cases of n , HSC consistently outperforms GA and SFLA. For example, for n values of 2, 5, 10, 15, and 20, HSC shows efficiency improvements of 0.044s, 0.135s, 0.513s, 0.851s, and 1.569s over GA. This outcome is attributed to HSC's intelligent selection of a subset of cloud services for further consideration, contrasting with GA and SFLA, which seek the optimal composition result from the entire set. GA and SFLA lack effective preprocessing mechanisms, limiting flexibility in forming the combined service and resulting in higher computation overhead. As n increases, HSC consistently benefits from enhanced efficiency, as n significantly influences overall efficiency. For instance, as n increases from 2 to 20, HSC's overall computation overhead rises only slightly from 0.0378 to 0.0386s, while GA's increases from 0.0820 to 1.607s and SFLA's increases from 0.0235s to 0.5436s. In summary, HSC demonstrates superior performance with respect to n .

5.3.2 Average Accuracy

Ensuring the accuracy of the selection process is crucial for the viability of HSC. Therefore, we compare the model accuracy and convergence speed in Set #2 to demonstrate the benefits of incorporating BFL in the first layer of HSC.

-Effect of n in Set #2.1. We analyze the accuracy of the global model by varying the number of CSPs (n) from 2 to 20. The experimental results are illustrated in Fig. 8. Intuitively, as the number of CSPs (n) increases, the model accuracy and convergence speed rise for all datasets. We further show the average accuracy in Fig. 9. Overall, the average accuracy demonstrates an upward trend with an increase in n . For instance, it improves from 66.34 to 84.08% with n

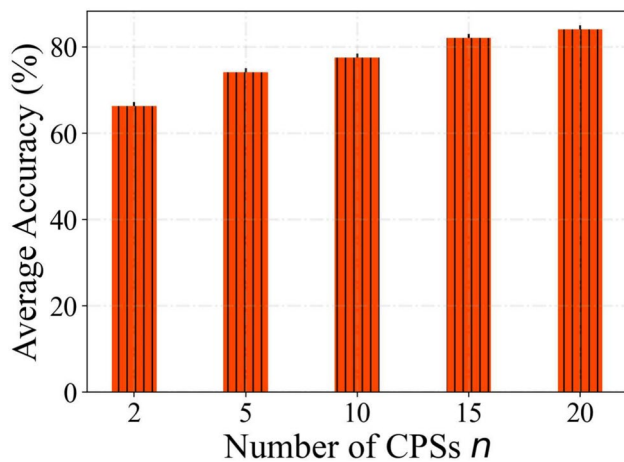


Fig. 9 Average accuracy vs. n (Set #2.1). More CSPs result in a better global model

growing from 2 to 20. This observation underscores that the involvement of more CSPs in the training process contributes additional learnable features to the global model, resulting in a more robust and accurate global model. Another intriguing finding is that the improvement in model accuracy is greater when the number of CSPs is increased from 2 to 5 than when it is increased from 5 to 10 on all datasets. This implies that the MLP model has a limited ability in feature expression, and adding more CSPs will not bring a huge improvement once the model accuracy is high enough. Moreover, the growth in the number of CSPs is associated with an accelerated convergence speed in FL. More fresh features from additional CSPs lead to quicker convergence of the global model. In conclusion, these experimental results validate the scalability of using FL in our proposed framework, particularly its suitability for scenarios involving multiple CSPs.

-Effect of d in Set #2.2. Considering different numbers of training samples d , the experimental results of accuracy and average accuracy are presented in Figs. 10 and 11, respectively. Similar to the findings in Figs. 8 and 9, more data samples d result in better accuracy with faster convergence speed. For example, when d is equal to 100, 500, 1000, 2000, and 4000, the average accuracy is 51.74%, 71.20%, 78.29%, 81.96%, and 86.35%, respectively. This is because (1) a larger dataset allows the model to learn more diverse patterns and variations present in the data, which helps the model generalize better to unseen examples; (2) with more data, the model is less likely to memorize specific instances in the training set (i.e., overfitting) and can focus on learning the underlying patterns that are more likely to apply to new data, helping the model become more robust to noisy or irrelevant features present in the data; (3) more data enables the model to see a wider range of examples, capturing the variability in the input space. This makes the model more

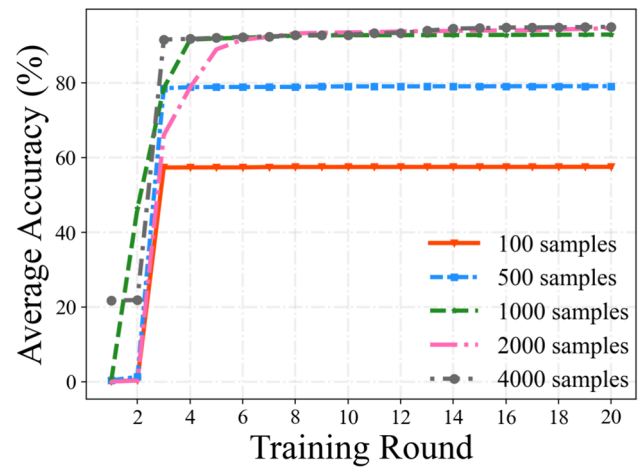


Fig. 10 Accuracy vs. d (Set #2.2). As d increases, the model accuracy improves

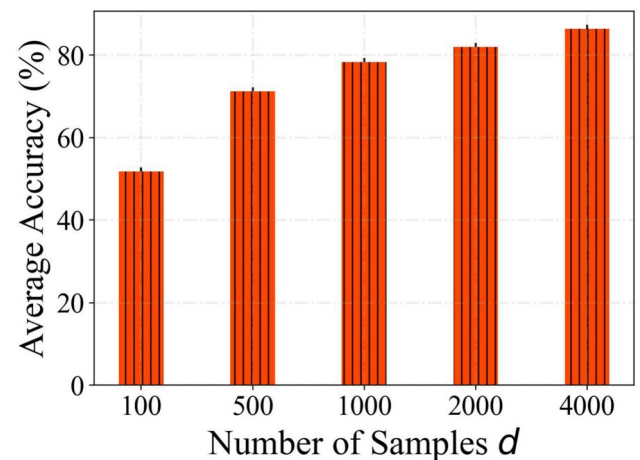


Fig. 11 Average accuracy vs. d (Set #2.2). A larger d leads to an enhanced global model

versatile and capable of handling diverse inputs. The experimental results clearly validate the superiority and practicality of adopting BFL in HSC.

5.3.3 Ablation Study

To further demonstrate the superiority of our design, we conduct a series of ablation experiments. First, we compare the model accuracy of federated learning (FL) with blockchain-enabled federated learning (BFL) to validate the benefits of integrating blockchain into FL. Additionally, we assess the efficiency of using two selection phases versus only the second selection phase to confirm the validity of using both phases.

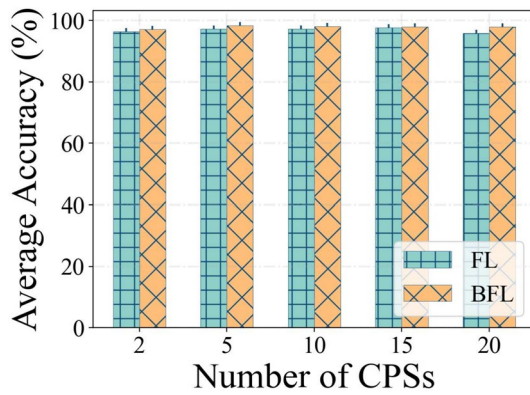


Fig. 12 Average Accuracy vs. n (Set #2.1). BFL consistently achieves slightly higher average accuracy than FL

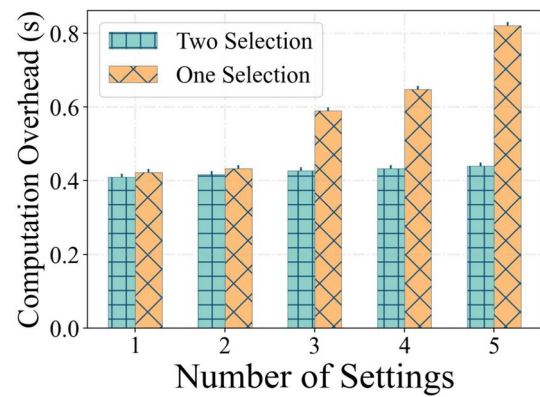


Fig. 14 Effect of k (Set #1.1)

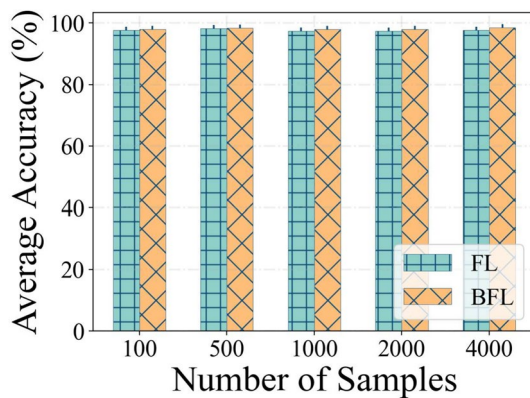


Fig. 13 Average accuracy vs. d (Set #2.2). BFL maintains a superior average accuracy across d compared to FL

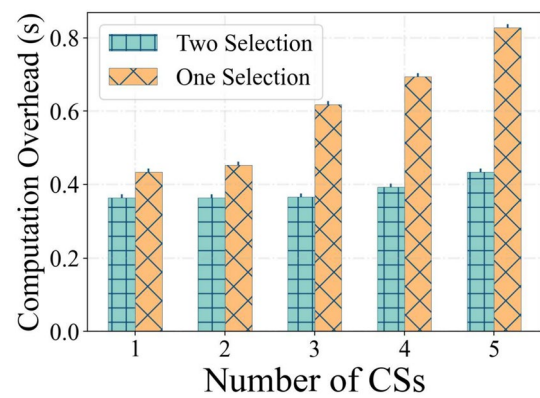


Fig. 15 Effect of m (Set #1.2)

FL vs. BFL. We compare the model accuracy (measured in %) of using federated learning (FL) and blockchain-enabled federated learning (BFL) by varying the number of CSPs n in Set #2.1 and the number of training data samples d in Set #2.2. As shown in Fig. 12, BFL consistently achieves comparable or better average accuracy than FL across different values of n , demonstrating the robustness and scalability of BFL in various operational settings. Similarly, Fig. 13 reveals that BFL maintains a competitive or superior average accuracy across all sample sizes compared to FL. This observation highlights the effectiveness of integrating blockchain into federated learning frameworks for enhanced model performance.

Two Selection vs. One Selection. We compare the computation efficiency (measured in s) of employing two-selection phases versus one-selection phase to demonstrate the effectiveness of our design by varying the number of settings k in Set #1.1, the number of CSs m in Set #1.2, and the number of CSPs n in Set #1.3. As illustrated in Fig. 14, the results show that although both methods exhibit an increase in overhead with a larger k , the increase is less steep for

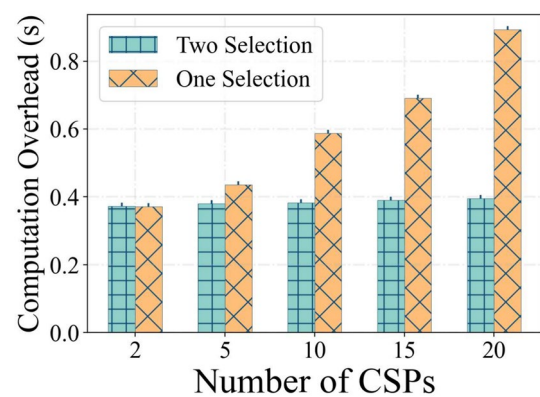


Fig. 16 Effect of n (Set #1.3)

the two selection phases. This indicates the superior efficiency and scalability of the two selection phases method in diverse operational settings. Similar observations are made in Figs. 15 and 16. In summary, the two-selection method consistently outperforms the one-selection method, with

the disparity becoming more pronounced in more complex scenarios.

6 Summary and Future Work

In this paper, we introduce a novel hierarchical service composition framework entitled HSC, which employs blockchain and federated learning to optimize composition efficiency by minimizing computational complexity. The framework comprises an initialization phase and two selection layers. During the initialization phase, we establish two smart contracts and employ blockchain-enabled federated learning to train a service selection model. This approach facilitates machine learning model training with decentralized data, ensuring practicality and fairness in the training process. In the first selection layer, each cloud service provider utilizes the pre-trained service selection model for efficient service selection, eliminating the need for complex optimization algorithms. The second selection layer employs a blockchain-based QoS-aware selection mechanism to attain the final composition result without involving third parties, thereby addressing potential privacy concerns. The incorporation of these layers significantly diminishes overall computational overhead, resulting in a substantial enhancement of service composition efficiency.

For future work, we plan to enhance the generality of HSC by evaluating its performance on a diverse set of datasets within real platforms. Additionally, we aim to explore and incorporate additional QoS metrics to create a more comprehensive and robust solution.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Li Huang, Lu Zhao, and Yao Zhao. The first draft of the manuscript was written by Li Huang and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript. The incremental experiments in the revised manuscript were conducted by Yansong Liu. The revised manuscript and response letter were written by Li Huang and all authors commented on previous versions of the revised manuscript.

Funding This work is supported in part by Natural Science Foundation of Jiangsu Province under Grant No. BK20230351 and Natural Science Foundation of the Higher Education Institutions of Jiangsu Province under Grant No. 23KJB520023.

Declarations

Conflict of interest The authors have no Conflict of interest to declare that are relevant to the content of this article.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long

as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Tan W, Zhao Y, Hu X, Xu L, Tang A, Wang T (2019) A method towards web service combination for cross-organisational business process using qos and cluster. *Enterprise Information Systems* 13(5):631–649
2. Zhao Y, Tan W, Jin T (2017) “Qos-aware web service composition considering the constraints between services,” in *Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, pp. 229–232
3. Ibrahim GJ, Rashid TA, Akinsolu MO (2020) An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment. *J Parallel and distributed computing* 143:77–87
4. Hosseinzadeh M, Hama HK, Ghafour MY, Masdari M, Ahmed OH, Khezri H (2020) Service selection using multi-criteria decision making: a comprehensive overview. *J Netw Syst Manage* 28:1639–1693
5. Dastjerdi AV, Buyya R (2014) Compatibility-aware cloud service composition under fuzzy preferences of users. *IEEE Transactions on Cloud Computing* 2(1):1–13
6. Asghari S, Navimipour NJ (2016) Review and comparison of meta-heuristic algorithms for service composition in cloud computing. *Majlesi Journal of Multimedia Processing* 4(4):28–34
7. Vakili A, Navimipour NJ (2017) Comprehensive and systematic review of the service composition mechanisms in the cloud environments. *J Netw Comput Appl* 81:24–36
8. Zhao Y, Qu Y, Chen F, Xiang Y, Gao L (2023) “Data integrity verification in mobile edge computing with multi-vendor and multi-server,” *IEEE Transactions on Mobile Computing*
9. Tang M, Ai L (2010) A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. In: *IEEE congress on evolutionary computation*. IEEE, pp. 1–8
10. Gao A, Yang D, Tang S, Zhang M (2006) “Qos-driven web service composition with inter service conflicts. In: *Frontiers of WWW Research and Development-APWeb 2006: 8th Asia-Pacific Web Conference*, Harbin, China, January 16–18, 2006. *Proceedings* 8. Springer 121–132 (2006)
11. Lu Y, Xu X (2017) A semantic web-based framework for service composition in a cloud manufacturing environment. *J Manuf Syst* 42:69–81
12. Zhou J, Yao X (2017) Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing. *Appl Soft Comput* 56:379–397
13. Li Y, Wen T (2009) “Quality and relation driven service selection for web services composition. In: *2009 International Conference on New Trends in Information and Service Science*. IEEE, pp. 152–155
14. Ai L, Tang M (2008) A penalty-based genetic algorithm for qos-aware web service composition with inter-service dependencies and conflicts. In: *2008 international conference on computational intelligence for modelling control & automation*. IEEE 738–743

15. Ai L, Tang M (2008) Qos-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm. In: 2008 IEEE Fourth International Conference on eScience. IEEE, pp. 119–126
16. Chen F, Dou R, Li M, Wu H (2016) A flexible QOS-aware web service composition method by multi-objective optimization in cloud manufacturing. *Comput Ind Eng* 99:423–431
17. Wu Y, Wang X (2011) Applying multi-objective genetic algorithms to qos-aware web service global selection. *Adv Inf Sci Serv Sci* 3(11):134–144
18. Nakamoto S (2008) “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized business review*
19. Aghamohammadzadeh E, Fatahi Valilai O (2020) A novel cloud manufacturing service composition platform enabled by blockchain technology. *Int J Prod Res* 58(17):5280–5298
20. Yu C, Zhang L, Zhao W, Zhang S (2020) A blockchain-based service composition architecture in cloud manufacturing. *Int J Comput Integr Manuf* 33(7):701–715
21. Haouari M, Mhiri M, El-Masri M, Al-Yafi K (2022) A novel proof of useful work for a blockchain storing transportation transactions. *Inf Process Manag* 59(1):102749
22. Gu X, Akoglu L, Rinaldo A (2019) Statistical analysis of nearest neighbor methods for anomaly detection. *Advances in Neural Information Processing Systems*, vol. 32
23. Saligrama V, Chen Z (2012) “Video anomaly detection based on local statistical aggregates. In: 2012 IEEE Conference on computer vision and pattern recognition. IEEE, pp. 2112–2119
24. Soule A, Salamatian K, Taft N (2005) Combining filtering and statistical methods for anomaly detection. In: in Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, pp. 31–31
25. Syarif I, Prugel-Bennett A, Wills G (2012) Unsupervised clustering approach for network anomaly detection. In: International conference on networked digital technologies. Springer, pp. 135–145
26. Li J, Izakian H, Pedrycz W, Jamal I (2021) Clustering-based anomaly detection in multivariate time series data. *Appl Soft Comput* 100:106919
27. Markovitz A, Sharir G, Friedman I, Zelnik-Manor L, Avidan S (2020) Graph embedded pose clustering for anomaly detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10 539–10 547
28. Gonzalez F, Dasgupta D, Kozma R (2002) Combining negative selection and classification techniques for anomaly detection. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No. 02TH8600), vol. 1. IEEE, pp. 705–710
29. Garg S, Kaur K, Kumar N, Batra S, Obaidat MS (2018) Hyclass: Hybrid classification model for anomaly detection in cloud environment. In: 2018 IEEE International Conference on Communications (ICC). IEEE, pp. 1–7
30. George A, Vidyapeetham A (2012) Anomaly detection based on machine learning: dimensionality reduction using PCA and classification using SVM. *Int J Comput Appl* 47(21):5–8
31. Aloqaily M, Al Ridhawi I, Kanhere S (2023) Reinforcing industry 4.0 with digital twins and blockchain-assisted federated learning. *IEEE Journal on Selected Areas in Communications*
32. Liu Y, Qu Y, Xu C, Hao Z, Gu B (2021) Blockchain-enabled asynchronous federated learning in edge computing. *Sensors* 21(10):3335
33. Zhao Y, Qu Y, Xiang Y, Chen F, Gao L (2024) “Context-aware consensus algorithm for blockchain-empowered federated learning,” *IEEE Transactions on Cloud Computing*
34. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: Challenges, methods, and future directions. *IEEE Signal Process Mag* 37(3):50–60
35. Wang Z, Hu Q (2021) “Blockchain-based federated learning: A comprehensive survey,” *arXiv preprint arXiv:2110.02182*
36. Liu Y, Garg S, Nie J, Zhang Y, Xiong Z, Kang J, Hossain MS (2020) Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet Things J* 8(8):6348–6358
37. Wang B, Wu F, Long Y, Rimanic L, Zhang C, Li B (2021) “DataLens: Scalable privacy preserving training via gradient compression and aggregation,” in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 2146–2168
38. Qu Y, Uddin MP, Gan C, Xiang Y, Gao L, Yearwood J (2022) Blockchain-enabled federated learning: A survey. *ACM Comput Surv* 55(4):1–35
39. Zhao Y, Qu Y, Xiang Y, Shi C, Chen F, Gao L (2023) “Long-term over one-off: Heterogeneity-oriented dynamic verification assignment for edge data integrity,” *IEEE Transactions on Mobile Computing*
40. Nilsson A, Smith S, Ulm G, Gustavsson E, Jirstrand M “A performance evaluation of federated learning algorithms,” in Proceedings of the second workshop on distributed infrastructures for deep learning, pp. 1–8
41. Chen Z, Yi W, Shin H, Nallanathan A, Li GY (2024) Efficient wireless federated learning with partial model aggregation. *IEEE Transactions on Communications*
42. Rey V, Sánchez PMS, Celdrán AH, Bovet G (2022) Federated learning for malware detection in iot devices. *Comput Netw* 204:108693