Received: 23 August 2020 | Revised: 2 April 2021 | Accepted: 2 April 2021

DOI: 10.1002/spe.3000

RESEARCH ARTICLE

WILEY

Resource provisioning in collaborative fog computing for multiple delay-sensitive users

Shuaibing Lu[1] | **Jie Wu**[2] | **Ning Wang**[3] | **Yubin Duan**[2] | **Haiming Liu**[4] | **Jiayue Zhang**[1] | **Juan Fang**[1]

[1]Faculty of Information Technology, Beijing University of Technology, Beijing, China

[2]Center for Networked Computing, Temple University, Philadelphia, Pennsylvania, USA

[3]Department of Computer Science, Rowan University, Glassboro, New Jersey, USA

[4]School of Software Engineering, Beijing Jiaotong University, Beijing, China

Correspondence
Juan Fang, Faculty of Information Technology, Beijing University of Technology, No.100 Pingleyuan, Chaoyang District, Beijing 100124, P.R. China.
Email: fangjuan@bjut.edu.cn

Abstract

Fog computing is an emerging paradigm that supplies storage, computation, and networking resources between traditional cloud data centers and end devices. This article focuses on the resource provisioning problem in collaborative fog computing for multiple delay-sensitive users. Our goal is to implement a resource provisioning strategy for network operators to minimize the total monetary cost by considering the deadline and capacity constraints. Two scenarios are considered: unlimited-processor fog nodes (UPFN) and limited-processor fog nodes (LPFN). In either scenario, we prove that the resource provisioning problem is NP-hard. First, we consider the UPFN scenario that the processors of fog nodes are unlimited and users' requests can be ideally processed in parallel. Two algorithms are proposed which greedily delete fog nodes based on the local or global collaborative influences until there is no feasible provisioning to guarantee the deadline of users. Then we extend the resource provisioning problem to a more realistic and complicated scenario LPFN in which the scheduling delay cannot be ignored. Two types of tasks are considered. One is the arbitrarily divided tasks, and a near-optimal solution bounded by $\frac{8}{3}\mathrm{OPT} + \frac{\epsilon^2}{8m\alpha}$ has been found. m is the number of fog nodes, and α is the upper bound on the Lipschitz constant of the delay function. Another one is the application-driven tasks, and we propose a heuristic algorithm. Extensive experiments validate the efficiency of the proposed algorithms.

KEYWORDS

collaborative fog computing, cost-efficiency, delay-sensitive users, resource provisioning

1 | INTRODUCTION

With the unprecedented volume and variety of data generated, users' demand for high-quality services has been increasing.. Fog computing is an emerging paradigm that supplies storage, computation, and networking resources between traditional cloud data centers and end devices. In fog computing, the basic infrastructures are fog nodes, which include industrial switches, controllers, routers, video surveillance cameras, and embedded servers.[1-5] The analysis must be very rapid since the Internet of Things (IoT) devices are continuously generating data. One important problem is to find a provisioning strategy for fog nodes that can decrease the monetary cost of fog resources and reduce the transmission

latency for users. Resource provisioning for users' workload has been widely explored in most existing researches, such as offloading computational tasks to a single fog node or cloud directly. In this article, we concentrate on the resource provisioning problem by considering the collaboration of fog nodes under the capacity constraints for delay-sensitive users. The resource provisioning problem is to determine the allocation of the resources on the fog nodes to multiple users, so as to minimize the total cost while considering the constraints on the deadline of users and the computing resources of fog nodes. The collaboration means that one fog node can exploit rented fog nodes to provide service jointly. Our objective is to improve the cost-efficiency of network operators in fog computing while guaranteeing the quality of services of users.

We first use an example to illustrate the motivation of our work in the following setting scenario. Some assumptions and notations are not explicitly stated but will be explained in the later sections. We assume that there exist eight heterogeneous fog nodes ($v_{(1-8)}$), and the computing capability of each fog node provided to users ($u_{(1-4)}$) is limited. For each user, the connection scope is within a certain area as shown in the gray circle in Figure 1. In this area, users can offload tasks to the corresponding fog node. The fog nodes of different suppliers support the collaborative services, and the monetary cost of the resource provisioning strategy generates for the fog node depending on the set-up cost. Thus, the total cost is proportional to the number of fog nodes. We suppose that the connections and locations of the fog nodes have been fixed by the third-party service providers or the cloud data centers. Within a fixed deadline T, users can offload the workloads to their nearest fog nodes. We suppose that the workloads are divided into the same service entities, and a service entity is represented by one unit. Thus, the provisioning strategy for multiple users is allocating the service entities of workloads to the fog nodes. According to the scenario that we set up, there is a trade-off between cost and efficiency. We use the following example to illustrate the trade-off. As Figure 1 shows, an extreme solution is the provisioning strategy by minimizing the delay for users' set ($u_{(1-4)}$), which offloads workloads to all connected fog nodes ($v_{(1-8)}$) in the network. However, the solution has the highest total cost among all possible provisioning strategies that meet the deadline constraints. Another extreme solution is the provisioning strategy with the minimal total cost, which means that the number of fog nodes that support the set of users is the minimum, as shown in Figure 1where v_1, v_2, v_3, and v_4 are used. If the volumes of workloads become heavy, the total completion time of users in the set may exceed the deadline T. This article proposes cost-efficient resource provisioning strategies for multiple users that fall between these two extreme solutions. Two scenarios are considered in this article: unlimited-processor fog nodes (UPFN) and limited-processor fog nodes (LPFN). In the UPFN, the processors of each fog node are unlimited and users' requests can be ideally processed, that is, without scheduling delays. In the LPFN, we consider a more realistic and complicated scenario, in which the processors of each fog node are limited, that is, the scheduling delay cannot be ignored. We consider two different types of tasks: one is the tasks that can be arbitrarily divided; the other one is the application-driven task, which means that each

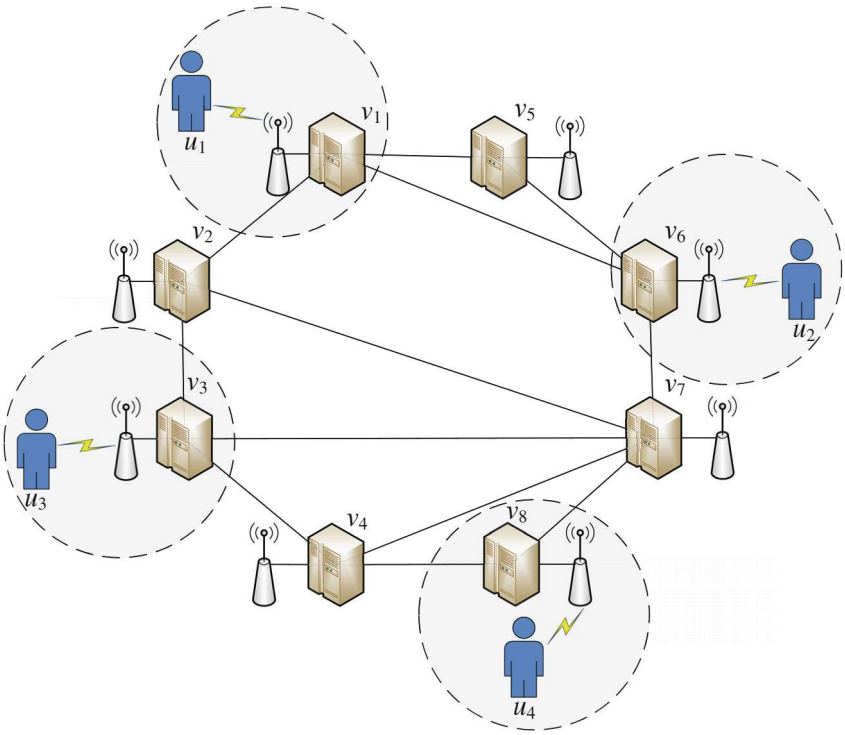


FIGURE 1 A motivation example

task contains multiple subtasks and cannot be divided arbitrarily. Our problem poses several unique challenges as follows: (i) Due to the computing capabilities of fog nodes being limited and heterogeneous, when several groups of users with different sizes of workloads arrive, finding a feasible resource provisioning strategy that can complete the workloads within deadlines for users is nontrivial. (ii) In our problem, fog nodes are allowed to connect with a part of users, and the users have to offload their workloads to the fog nodes located in their efficient regions. The resource provisioning strategy for a single user may not apply to multiple users. (iii) Even though the collaboration of fog nodes can decrease the users' latency, the communication cost may increase. To balance the trade-off between the cost and delay, it is challenging to do the resource provisioning while satisfying the requests of multiple users with minimum cost. This article focuses on the resource provisioning problem that realizing the cost-efficiency for network suppliers in collaborative fog computing of multiple sensitive users with the constraints on the capacities and delays.

The major novel contributions of this article are as follows:

- We discuss the resource provisioning problem in collaborative fog computing for multiple delay-sensitive users. We consider both the delays of users and the monetary costs, and we prove that the resource provisioning problem with minimum cost in collaborative fog computing is NP-hard.
- We discuss two different scenarios which are UPFN and LPFN. For both scenarios, we take into account the deadlines of users and the cost of suppliers as constraints to meet different requirements, and we put forward a novel idea by continuously removing and adjusting the nodes to realize the optimization under the resource provisioning in collaborative fog computing. Specifically, we first consider a simple scenario UPFN. We first propose a novel feasibility checking method and define two new collaborative influence factors that are used to minimize the total cost by iteratively removing the fog nodes. Based on that, we propose two greedy heuristic strategies and analyze their complexities. Then we extend the resource provisioning problem to a more realistic and complicated scenario LPFN and formulate it by converting an optimal provisioning finding (OPF) problem into a continuous congestion game problem. In the LPFN scenario, we consider two types of tasks: the arbitrarily divided tasks and the application-driven tasks. For the arbitrarily divided tasks, we propose a near-optimal algorithm bounded by $\frac{8}{3}\mathrm{OPT} + \frac{\epsilon^2}{8m\alpha}$. For the application-driven tasks, we propose a novel heuristic algorithm.
- We conduct various simulations on both synthetic and real datasets to compare our joint optimization methods with several state-of-the-art ones. The results are evaluated from different perspectives to provide conclusions. The simulation results show that our algorithms can reduce the resultant average costs of 12.9% and 17.8% in UPFN and LPFN, respectively. We also evaluate our algorithms on the real dataset, the simulation results show that our algorithms can reduce the average cost of 15.2% and 20.3%.

The remainder of this article is organized as follows. Section 2 surveys related works. Section 3 describes the model and then formulates the problem. Section 4 investigates the resource provisioning problem with UPFN and introduces two greedy algorithms. Section 5 discusses the resource provisioning problem with LPFN and proposes one algorithm for the arbitrarily divided tasks. Section 6 discusses the LPFN case and proposes one algorithm for application-driven tasks. Section 7 presents the experiments. Finally, Section 8concludes the article.

2 | RELATED WORK

Fog computing[2,3] as a new distributed paradigm[6] enables a new breed of services and applications by extending cloud computing to the edge of the networks. Unlike the cloud, there are lots of fog nodes distributed in wide-geographic locations and supplying services closer to the users and IoT devices in fog computing. Many research efforts have focused on the resource provisioning problem in fog computing.[7-9] Pham and Huh[8] study the resource scheduling problem in the cloud-fog computing architecture, where fog providers can take advantage of the collaborations between their fog nodes and the rented cloud resources. Their focus is on balancing the monetary cost of cloud resources and the makespan. Santos et al.[9] propose a network-aware scheduling approach for container-based applications in smart city deployments, and their proposal has been validated on the Kubernetes platform. Yu et al.[10] focus on the joint optimization problem on data routing and application placement to support all data streams with both delay and bandwidth guarantees. However, these resource provisioning strategies do not consider the collaboration of fog nodes. Another paradigm similar to fog computing, Wang et al.[11] study the allocation and scheduling problem of computing requests on edge nodes, while minimizing

total cost using the heterogeneity of the cloud in mobile edge computing. Wang et al.[12] and Sheng et al.[13] focus on the placement of service entities for applications of social virtual reality in edge computing.

Numerous novel studies take the collaboration of fog nodes into account. Xiao et al.[4] use the distributed alternating direction multiplier method to jointly optimize the user's QoE under a given power efficiency, which proposes that fog nodes can cooperate and jointly upload the workloads to the cloud data center. Naha et al.[14] propose resource allocation and provisioning algorithms by using ranking resources in a hybrid and hierarchical fashion to satisfy the requirement of users' deadline. Bierzynski et al.[15] provide the IoT solutions by proposing the methods used to distribute workloads among the cooperation of cloud, fog, and edge. Huang et al.[16] analyze the energy-efficient resource allocation problem in fog computing with candidate fog nodes to ensure the network loading balance under the performance constraints. Chang and Wu[17] study a collaborative system in the mobile social network which focuses on the allocation and partition of the workload. Ningning et al.[18] use the graph partitioning theory to build the load balancing algorithm based on the dynamic graph partitioning. The above studies focus on the performance guarantee or load balancing based on the collaboration of fog nodes. Although the optimization objectives are different, the idea of fog node cooperation provides inspiration for the design of our solution. Another thread of research has been focusing on reducing the energy consumption for the resource provisioning problem in fog computing. Gu et al.[19] formulate the resource provisioning problem as a one-to-many matching game and propose a new mechanism to satisfy task owners' heterogeneous delay requirements and support good scalability. Badidi[20] propose an architecture for the placement of IoT application tasks on a cluster of fog nodes in the application's data sources. Yin et al.[21] build a new task-scheduling model that ensures completion time and optimizes the number of concurrent tasks for fog nodes. Ma et al.[22] formulate the resource provisioning in a new framework as an optimization problem, and they propose algorithms by exploiting the piecewise convex property. Most of these works address the resource provisioning problem by only focusing on the latency, throughput, or energy consumption while ignoring the interdependency between the delays of users and the monetary costs.

What's more, several studies propose new architectures through horizontal and vertical expansion based on deep learning. Fatemeh et al.[23] try to find a placement of the modules using deep reinforcement learning, which can find the best destination to execute the module so as to create a compromise between the power consumption and execution time. Zhang et al.[24] propose a joint optimization framework for all fog nodes, data service operators, and data service subscribers to obtain the optimal resource allocation schemes in a distributed fashion. Li et al.[25] propose a deep learning-based classification model which can improve the computing efficiency of the manufacture inspection system with fog computing. The performance in the above studies is improved, however, the model training time and algorithm complexity are not as good as the heuristic method.

This article concentrates on the resource provisioning problem for multiple delay-sensitive users in fog computing, and our goal is to achieve a feasible heuristic provisioning strategy that minimizes the total cost under the capacity constraints by considering the collaboration of fog nodes.

3 | PROBLEM FORMULATION

The resource provisioning problem attempts to find an efficient strategy to upload data to the appropriate location for processing for multiple users in the fog computing network. In this article, we concentrate on the resource provisioning problem in collaborative fog computing for multiple delay-sensitive users. Our goal is to minimize the total cost in the resource provisioning process for multiple delay-sensitive users while meeting the constraints on the deadline of users, the communication, and computations resources in fog computing.

3.1 | Fog model

As shown in Figure 1, our fog model is constructed by a substrate distribution of nodes based on an undirected graph $\mathbf{G}=\{\mathbf{V},\mathbf{E}\}$, where $\mathbf{V}$ is a set of fog nodes, and $\mathbf{E}$ is a set of edges. Let v_j denote the jth fog node in $\mathbf{V}$, that is, $\mathbf{V}=\{v_j\}$. We suppose that the fog nodes in set $\mathbf{V}$ are heterogeneous and can be any devices with storage, computing capacity, and network connectivity. e_{ij} denotes the connection between fog nodes i and j in $\mathbf{E}$, that is, $\mathbf{E}=\{e_{ij}\}$. We suppose that the fog nodes' distributions are concentrated, and there are no communication delays. So the delays are nearly the same between fog nodes and users, which depends on the offloading delay between access nodes and users. For each v_j, it can only provide services to c_j user who is in an efficient region. It is an area where fog nodes can be connected to the

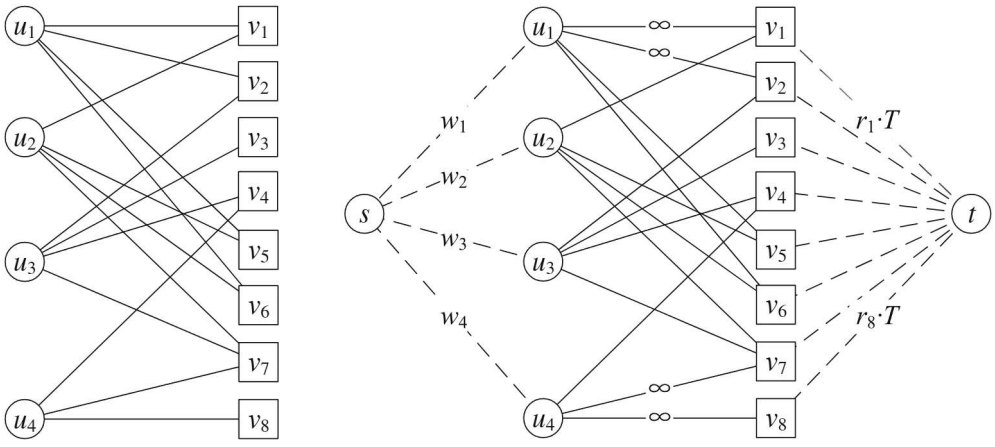
users successfully. Since the fog nodes are heterogeneous, and the form factors are different,[3] we use r_j to denote the maximum processing rate of fog node v_j. Let τ_j denote the set-up cost of v_j, which measures the efficiency of the resource provisioning strategy. We suppose that the connections and locations of fog nodes in $\mathbf{G}$ are provided by the third-party service suppliers or the cloud data center in advance. We define two different types of fog nodes in the set $\mathbf{V}$. One is **access node** that connects with the base station in the set $\mathbf{V}$, which offloads the workload from the user layer. Users can only upload workloads to their near-access fog node in an efficient region. Another one is **collaborative node** that either processes the workload by itself or from their adjacent fog nodes which can be arrived at via local communication in the set $\mathbf{V}$.

In this article, we consider two scenarios, UPFN and LPFN. Due to the difference in the capacities of processors, they have different computation delays.

- **Unlimited-processor fog nodes**: Fog nodes in set $\mathbf{V}$ have unlimited numbers of processors which can process requests parallelly, and the computation delay for the request with weight w_i on v_j is p_{ij}, that is, $p_{ij} = \frac{\lambda_{ij} \cdot w_i}{r_j}$. Let λ_{ij} denote the proportion of workload of user u_i on v_j, where $0 \leq \lambda_{ij} \leq 1$.
- **Limited-processor fog nodes**: Fog nodes in set $\mathbf{V}$ have limited numbers of processors, and there will be delays when multiple jobs process tasks on one fog node. The computation delay is defined as $p_{ij} = \alpha \cdot x_v + b$, which illustrates that users' delays have linear relationships with the numbers of fog nodes. Let b denote a constant delay of the fog node, and α is a unit rising-rate.[12]

3.2 | Users and workloads

In this subsection, we describe the model of users and workloads. Let $\mathbf{U} = \{u_i\}$ denote the delay-sensitive users that are distributed on the user layer in fog computing. We use u_i to denote the ith user and w_i to denote the weight of u_i's task. The tasks of users are fractional and continuous which is divided into two types. One is the arbitrarily divided tasks, which means that the task can be divided into several same service entities, and each service entity is represented by one unit. Another one is the application-driven tasks, which means that the task of each user contains several subtasks that cannot be divided. The access fog nodes provide service to these two types of tasks with the shortest distance. The transmission costs between access fog nodes and the corresponding connected collaborative fog nodes are ignored, and the users' total completion time in $\mathbf{U}$ cannot exceed the deadline T. The connection between fog nodes and users can be described by a connection graph in Figure 2. e_{ij} denotes the connection between u_i and v_j, which means that u_i's workload can be processed on v_j. According to the connection between the user and the fog node, there exists a specified subset of fog nodes $\mathbf{G}_i$ that only allows to process u_i's workload, where $\mathbf{G}_i = \{v_k \in \mathbf{G} | e_{ik} \in L\}$. Fog node v_j processes the specified workloads for the subset of users $\mathbf{U}_j$, where $\mathbf{U}_j = \{u_k \in \mathbf{U} | e_{kj} \in L\}$. For the convenience of reference, we summarize the main notations throughout this article in Table 1.



(A) Connectivity graph. (B) Weighted connectivity graph.

FIGURE 2 The conversion of Figure 1 for the arbitrarily divided tasks

TABLE 1 Notations

$\mathbf{G}$	Substrate topology, where $\mathbf{G}=\{\mathbf{V},\mathbf{E}\}$.
$\mathbf{V}$	Set of fog nodes in $\mathbf{G}$, where $\mathbf{V}=\{v_j\}$.
v_j	Fog node j in $\mathbf{V}$.
$\mathbf{E}$	Set of connections between fog nodes in $\mathbf{G}$.
e_{ij}	The connection between v_i and v_j in set $\mathbf{E}$.
c_j	The capacity of v_j in $\mathbf{V}$.
r_j	The maximum processing rate of v_j in set $\mathbf{V}$.
$\mathbf{U}$	Set of users, where $\mathbf{U}=\{u_i\}$.
$\mathbf{W}$	Set of weights of users' workload, where $\mathbf{W}=\{w_i\}$.
T	Deadline of the set of users $\mathbf{U}$.
$\mathbf{X}$	Provisioning strategy for the job set $\mathbf{U}$.
τ_j	The set-up cost of v_j.
d_{ij}	Offloading delay between user i on fog node j.
p_{ij}	Processing delay of user i on fog node j.
λ_{ij}	The proportion of u_i's workload on v_j.

3.3 | Problem formulation

This article considers the resource provisioning problem for multiple users, and we measure the efficiency of the provisioning strategy by using the set-up cost for the collaborative fog computing networks. To formulate this problem, we use $\mathbf{X}$ to be the provisioning strategy for $\mathbf{U}$, which indicates the distribution of tasks to be processed on the fog nodes, that is, $\mathbf{X}=\bigcup_{j\in\mathbf{U}}\mathbf{X}_j$. $\mathbf{X}_j$ is a subset of $\mathbf{G}_j$ which denotes the provisioning strategy of each user, that is, $\mathbf{X}_j\subseteq\mathbf{G}_j$. Recall the $\mathbf{G}_j$ which is the set containing fog nodes that are available for u_j. Given a set of users, our goal is to achieve a feasible strategy for users' set $\mathbf{U}$ by minimizing the total cost and supporting all users' requests under both deadline and resource constraints. The formulation of the resource provisioning problem is shown as follow:

$$\text{minimize} \sum_{j\in\mathbf{X}} \mathbb{1}_{\left[\sum_{i\in\mathbf{U}}\lambda_{ij}w_i>0\right]}\tau_j \tag{1}$$

$$\text{subject to } D_i \le T, \quad \forall i\in\mathbf{U} \tag{2}$$

$$D_i = \max_{j\in\mathbf{X}_i}\{d_{ij}+p_{ij}\} \tag{3}$$

$$0\le\lambda_{ij}\le 1, \quad \sum_{i\in\mathbf{U},j\in\mathbf{G}}\lambda_{ij}=1 \tag{4}$$

$$\sum_{i\in\mathbf{U}}\lambda_{ij}\cdot w_i \le c_j. \tag{5}$$

Our objective is to minimize the total provisioning cost for the set of users $\mathbf{U}$ in Equation (1). $\mathbb{1}_{[\cdot]}$ is an indicator function, when $\sum_{i\in\mathbf{U}}\lambda_{ij}w_i>0$ the value of this function will be 1, otherwise it will 0. Equations (2)–(5) are the constraints. Equation (2) states the deadline constraint, which means that the completion time of the last user cannot exceed their deadline T. Equation (3) states the constraint on the user's delay. D_i denotes the maximum delay of u_i which includes the computation delay p_{ij} and the offloading delay d_{ij}. Since the delays between users and their access nodes have no relation to the deployment decision,[12] we set them to be the same. Equations (4) and (5) state the constraint on the computing resource, which indicates that the offloading workloads on the fog nodes cannot exceed their capacities.

Theorem 1. *The resource provisioning problem with minimum cost in the collaborative fog computing is NP-hard.*

Proof. Here, we prove our theorem 1 and reduce the resource provisioning with minimum cost in the collaborative fog computing to the set-covering problem in a polynomial-time,[26] which is NP-complete. For a set-covering problem, there exist an instance $(X, \mathcal{F})$ where X is a finite set and $\mathcal{F}$ is a family including the subsets of X. In this way, each element of X belongs to at least a subset of $\mathcal{F}$: $X = \bigcup_{S\in\mathcal{F}} S$. One set covers its elements $S \in \mathcal{F}$. The objective of a set-covering problem is to find a subset $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all elements of X with minimum size, that is, $X = \bigcup_{S\in\mathcal{C}} S$. We map $\mathbf{U}$ as set X, a substrate distributions fog nodes $\mathbf{X} = \bigcup_{j\in\mathbf{U}}\{P_j\}$ as $\mathcal{F}$, which are the subsets of $\mathbf{U}$, that is, $P_j \subseteq \mathbf{G}_j$. The fog nodes are reduced to the subsets $\mathbf{G}_j \in \mathbf{U}$, which contain users that they can provide service. According to the reduction, we have a set covering problem that can be reduced to the resource provisioning problem. The resource provisioning problem in collaborative fog computing with minimum cost is to find a subset $\mathcal{P} \subseteq \mathbf{X}$ whose fog nodes provide service for all elements in set $\mathbf{U}$ with minimum size, that is, $\mathbf{U} = \bigcup_{j\in\mathcal{P}} \mathbf{X}_j$. Since the set covering problem is NP-complete, the resource provisioning problem with minimum cost in the collaborative fog computing is NP-hard. ∎

4 | RESOURCE PROVISIONING WITH UPFN

In this section, we discuss the problem under the UPFN scenario, and two greedy algorithms are proposed as solutions. Both of them use a feasibility checking method which is first introduced in this section. The difference between these two algorithms is that they are based on distinct collaborative influences, the global and local collaborative influence, which are defined after the checking method. In the last part of this section, we show the details of these two algorithms.

4.1 | Feasibility checking

This subsection proposes an algorithm for checking the feasibility by converting the original scenario into the maximum flow problem in the weighted connectivity graph. Let $\mathbf{U}$ denote the set of users. The feasibility checking of this problem is to check whether there is a provisioning strategy that can satisfy their requirements for users under the constraint of the deadline T. Based on the information and connection between the fog node and the user, we construct a graph. On the basis of the connectivity graph, two virtual nodes s and t are added, and the workloads' sizes are indicated by the links' weights between s and users. The maximum processing volumes of the fog nodes are indicated by the links' weights between fog nodes and the destination t. Based on the connections in Figure 1, we have the relationship between fog nodes and users. Suppose that the communication between the fog node and the user has no limitation, and the weight between the fog node and the user is ∞. Due to the deadline of users in $\mathbf{U}$ is T, the maximum processing volume of the fog node will be $r_j \cdot T$, which is the weight of the link between fog node v_j and t. The constructed graph is shown in Figure 2. The specific description is represented in Algorithm 1, the inputs include $\mathbf{G}$ and $\mathbf{U}$. The output is the result of the feasibility for users in $\mathbf{U}$ on $\mathbf{G}$, that is, FC($\mathbf{G}$, $\mathbf{U}$). We first construct an auxiliary graph $\mathbf{G}'$ in regard to the connectivities between fog nodes and users in line 1. The feasible provisioning strategy is based on the maximum flow that goes through the edges between the destination t and the fog nodes, and we obtain the maximum flow Φ by using the Edmonds-Karp algorithm[26] in line 2. Lines 3–7 include the process of the feasibility checking; if the total users' demands $\sum_{i\in\mathbf{U}} w_i$ can be covered by the maximum flow, the set of users $\mathbf{U}$ is feasible with the resource provisioning of $\mathbf{G}$, otherwise, the demands of $\mathbf{U}$ will be rejected.

4.2 | Collaborative influences

In this subsection, we define two different collaborative influences according to the mechanisms of fog nodes: the global and local collaborative influence. Based on that, we further propose two greedy algorithms.

4.2.1 | Global collaborative influences

We introduce a definition of the global collaborative influence. We use p_{ij} to represent the completion time of u_i on v_j. Thus, the total completion time of v_j is $p_j = \sum_{i\in\mathbf{U}_j} p_{ij}$, and the average completion time in $\mathbf{G}$ is $\overline{R} = \frac{\sum_{v_j\in\mathbf{G}} p_j}{|\mathbf{G}|}$, where $|\mathbf{G}|$

Algorithm 1. Feasibility checking (FC)

Require: Users' set $\mathbf{U}$, topology $\mathbf{G}$;
Ensure: Feasibility on $\mathbf{G}$ of $\mathbf{U}$;
Construct $\mathbf{G}'$ in regard to the connections between fog nodes and users;
Obtain Φ based on $\mathbf{G}'$ using Edmonds-Karp algorithm;
if $\Phi \geq \sum_{i \in \mathbf{U}} w_i$ **then**
It is feasible to $\mathbf{U}$ on $\mathbf{G}$;
return Provisioning strategy $\mathbf{X}$ of $\mathbf{U}$;
else
It is not feasible to $\mathbf{U}$ on $\mathbf{G}$;
return False;
end if

denotes the fog nodes' number in $\mathbf{G}$. For the fog nodes, the global influences are to describe the increments of the average completion time for all the remaining ones after removing themselves. Here is the specific definition.

Definition 1 (Global collaborative influence). Let ψ_j indicate the global collaborative influence of v_j and $\psi_j = |\overline{R}_{\mathbf{G}/v_j} - \overline{R}_{\mathbf{G}}|$, where $\mathbf{G}/v_j$ is the set of fog nodes after removing v_j from $\mathbf{G}$.

4.2.2 | Local collaborative influences

This subsection further introduces the local collaborative influence. Based on the above definition, we have that the global collaborative influence calculates the increment on the average completion time of all fog nodes except v_j. However, when the fog nodes connected by users are relatively sparse, the completion time of partial ones may not be affected by removing the fog node v_j, and there may be have an underestimation problem. Therefore, we define the local collaborative influence which ignores the irrelevant fog nodes when calculating the local collaborative influence. The completion time of the fog nodes increases after removing v_j which is represented by $\mathbf{G}^*/v_j$, where $\mathbf{G}^*$ includes the fog node connected with the same set of users $\mathbf{U}_j$, that is, $\mathbf{G}^* = \bigcup_{i \in \mathbf{U}_j} \mathbf{G}_i$. Specifically, let $M_{\mathbf{G}^*}$ be the completion time of the last finished workloads in $\mathbf{G}^*$, and $M_{\mathbf{G}^*} = \max_{j \in \mathbf{G}^*}\{p_j\}$. For each fog node, the local collaborative influence is to describe the increment of the maximum completion time for the fog nodes in $\mathbf{G}^*$. Here is the specific definition.

Definition 2 (Local collaborative influence). Let ϕ_j indicate the local collaborative influence of v_j and $\phi_j = |\overline{M}_{\mathbf{G}^*/v_j} - \overline{M}_{\mathbf{G}^*}|$, where $\mathbf{G}^*/v_j$ is the set of fog nodes after removing v_j from $\mathbf{G}^*$.

4.3 | Global influence greedy

This subsection describes a greedy strategy for the resource provisioning problem, which iteratively removes the fog node by its global collaboration influence. In Algorithm 2, the inputs are the users' set $\mathbf{U}$ and the topology of fog nodes $\mathbf{G}$. The resource provisioning strategy $\mathbf{X}$ of the set of users $\mathbf{U}$ is the output. In line 1, Algorithm 2 first finds a feasible provisioning strategy using Algorithm 1. Then it calculates the average completion time $\overline{R}$ of each fog node in set $\mathbf{V}$ in line 2. Lines 3–6 calculate the global influences for the fog nodes in $\mathbf{V}$ and reorder them with increasing values of $\psi_j \cdot \tau_j$, that is, $j = \arg\min\{\psi_j/\tau_j\}$. After that, Algorithm 2 iteratively removes the fog node according to its global collaboration influence until there is no feasible provisioning strategy that can satisfy the deadlines of users in lines 7–11. In line 8, it checks the feasibility of the strategy using Algorithm 1. The iteration terminates once finding a feasible solution or the set of fog nodes is empty. Finally, the resource provisioning strategy $\mathbf{X}$ of the set of users $\mathbf{U}$ returns in line 12.

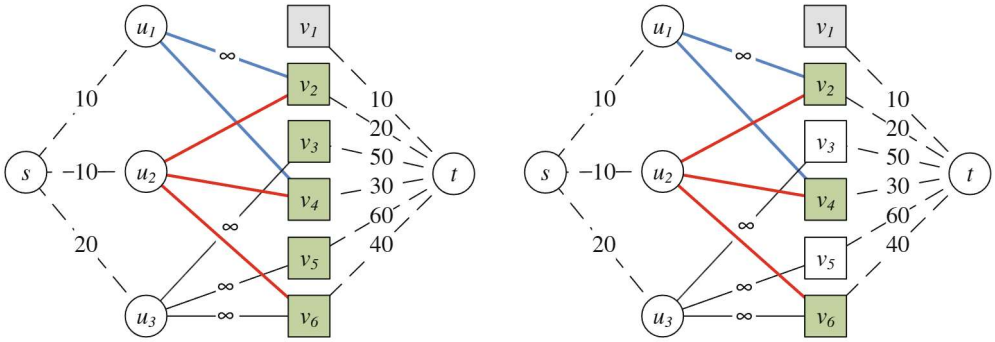
We extend the global influence greedy (GIG) algorithm into the scenario that the delays between fog nodes and users are different, which means when we choose one fog node to be removed, we need to consider both the set-up cost τ_j and the communication delay d_{ij}. Therefore, we make a slight change in lines 3–5 of the GIG algorithm. We calculate the global influence for each fog node in $\mathbf{V}$ and reorder them with increasing values, that is, $j = \arg\min\{\psi_j/\{\tau_j \cdot d_{ij}\}\}$.

Algorithm 2. Global influence greedy

Input: Users' set **U**, topology **G**;
Output: Provisioning strategy **X** of **U**;

1: Get a feasible provisioning strategy using Algorithm 1;
2: Calculate the average completion time of $\bar{R}$;
3: **for** $j = 1$ to $j = n$ in **V do**
4: Calculate ψ_j for v_j;
5: **end for**
6: Reconstruct **V** by reordering $j = \arg\min\{\psi_j/\tau_j\}$ for the fog nodes;
7: **while** $\mathbf{V} \neq \Phi \wedge \mathrm{FC}(\mathbf{G}, \mathbf{U}) \neq \mathit{false}$ **do**
8: Feasibility checking of **G** using Algorithm 1;
9: Remove v_j from set **V**;
10: Update topology **G**;
11: **end while**
12: **return** Provisioning strategy **X** of **U**;

FIGURE 3 Example of collaborative influences



(A) Global Collaborative Influence. (B) Local Collaborative Influence.

4.4 | Local influence greedy

The insight of the local influence greedy (LIG) strategy is similar to the GIG, except that the iterative removal of the fog node according to its local collaboration influence has subtle changes. The specific description is shown in Algorithm 3. Line 1 first finds a feasible strategy using Algorithm 1, and then line 2 calculates the latest finished completion time of $M_{\mathbf{G}^*}$ in $\mathbf{G}^*$. After that, it calculates the local collaboration influences of fog nodes and removes them by an increasing order in lines 3–5. Line 6 shows the difference between Algorithm 3 (LIG) and Algorithm 2 (GIG) and rebuilds **G** by reordering $j = \arg\min\{\phi_j/\tau_j\}$. Algorithm 3 does the same feasibility checking as Algorithm 2 in lines 7–11. Finally, it returns the resource provisioning strategy **X** for the set of users **U**.

The time complexity of LIG and GIG is $O((|\mathbf{V}|+|\mathbf{U}|)\cdot|\mathbf{E}|^2)$, where $|\mathbf{V}|$ and $|\mathbf{U}|$ are the numbers of fog nodes and users, respectively. $|\mathbf{V}|+|\mathbf{U}|+2$ results from the calculation of the total vertices' number in graph $\mathbf{G}'$. $|\mathbf{E}|$ denote links' number in graph $\mathbf{G}'$ that fog nodes and users are not fully connected. The time complexity of construct $\mathbf{G}'$ is $O(|\mathbf{V}|+|\mathbf{U}|+|\mathbf{E}|)$. In addition, each pair of fog nodes in graph $\mathbf{G}'$ has at most one link, thus $|\mathbf{E}|=O(|\mathbf{V}|^2)$. The time complexity of finding the maximum flow is $O((|\mathbf{V}|+|\mathbf{U}|)\cdot|\mathbf{V}|^4)$. Consequently, the complexity of LIG and GIG is $O(|\mathbf{V}|^5\cdot(|\mathbf{V}|+|\mathbf{U}|))$.

To better understand these two algorithms, we show an example as follows. Figure 3(A) is based on the algorithm GIG. We assume that there are three users, and the sizes of the workloads are 10, 20, and 10, respectively. There are in total six fog nodes, and the processing speeds of all nodes are 1, 2, 5, 3, 6, and 4, respectively. Let the deadline of users be $T = 10$, and we construct the substructure by adding one source node s and one destination node t. The maximum processing volume of the fog node is the weight between t and v_j in **V**, which is $r_j \cdot T$. Thus, the links' weights are 10, 20, 50, 30, 60, and 40, respectively. We first find one feasible provision using the Edmonds-Karp algorithm, then we remove each fog node with the basis of its global influence. For example, assuming that v_1 is removed, we refind the feasible provision

Algorithm 3. Local influence greedy

Input: Users' set **U**, topology **G**;
Output: Provisioning strategy **X** of **U**;

1: Get a feasible provisioning strategy using Algorithm 1;
2: Calculate the latest finished completion time of $M_{\mathbf{G}^*}$ in $\mathbf{G}^*$;
3: **for** $j = 1$ to $j = n$ in **G do**
4: Update φ_j for v_j;
5: **end for**
6: Construct **G** by reordering $j = \arg\min\{\varphi_j/\tau_j\}$ for the fog nodes;
7: Same as Algorithm 2 in lines 7–11;
8: **return** Provisioning strategy **X** of **U**;

using the Edmonds-Karp algorithm and update the average delay of all remaining fog nodes (v_2 to v_6). The increment of the average delay is the global influence of v_1. We do the calculation of the global influences for the fog nodes and sort them by increasing order. As shown by the example in Figure 3(A), we have the removing order $v_3 \to v_2 \to v_4 \to v_5 \to v_6$. After each iteration, we check the feasibility and choose v_6 and v_1 as the provision. As shown in Figure 3(B), we remove each fog node and check its local influence, respectively. We take v_1 as an example. If we remove v_1, we refind the feasible provision using the Edmonds-Karp algorithm, and update the maximum delay of fog nodes which are connected with the same users (v_2, v_4, and v_6) by removing v_1, the increment of the maximum delay is the local influence of v_1. We sort fog nodes based on the local influences, then we have $v_2 \to v_3 \to v_4 \to v_5 \to v_6$. After each iteration, we check the feasibility and choose v_5 and v_6 as the provision.

5 | RESOURCE PROVISIONING WITH LPFN

This section extends the naive scenario into a more realistic and complicated one, which considers the limitation of processing capacities for fog nodes. This scenario is referred to the resource provisioning problem with LPFN. In LPFN, the fog nodes have limitations on the processors, thus there will be scheduling delays when processing workloads for multiple users. The extension in execution time is considered to be the performance degradation of using the planning strategy to process the requests.[12] A general function $d_v(x_v)$ is used to indicate the performance degradation, where x_v is the number of users processing on the fog node. We suppose that $d_v(x_v) = \alpha \cdot x_v + b$, where α is a unit rising rate, and b is the constant delay of the fog node. Since all fog nodes are at least connected with one user, we consider the worst case in which one fog node is connected with all users, and the users' requests to be served at the same time. The solution follows the greedy method of the UPFN scenario, and the challenge is provisioning the workloads between fog nodes in each iteration. Under this scenario, since the time consumption of processing a unit workload is not constant, we cannot use the maximum flow to check the feasibility like Algorithm 1.

Therefore, we find the optimal resource provisioning strategy and compare the total completion time with the deadline of users to check the feasibility. We start with a definition as follows.

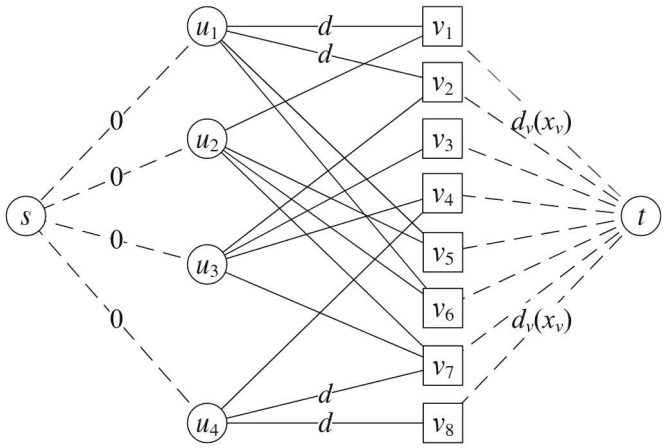
Definition 3 (Optimal provisioning finding problem). Given **U**, **G**, and $d_v(x_v)$, an OPF problem is how to find a provisioning to minimize the delay of users **U** in **G**.

5.1 | Problem conversion

On the basis of Definition 3, we convert the OPF problem into a continuous symmetric network congestion game (CSNCG) problem for each iteration in this subsection.

Definition 4 (Continuous symmetric network congestion game problem). A CSNCG problem is defined as a tuple $(\mathbf{U}, \mathbf{G}, s, t, c_e(\cdot))$, where **U** is a set of players, **G** is a directed graph that represents the communication network, $c_e(\cdot)$ is a nondecreasing and nonnegative cost function for each edge $e \in E$. s and t are the common initial vertex and the common

FIGURE 4 The converted graph of Figure 1



target vertex, respectively. A strategy $\mathbf{X}_i$ for the symmetric network congestion game is a route from s to t of user i.[12] The overall cost of one user is:

$$C_{u_i}(s) = \sum_{j \in \mathbf{X}_i} c_e(x_e). \tag{6}$$

According to the definition in Reference 27, we confirmed the strong isomorphism from OPF to CSNCG. Substantially, these two problems have the same user set $\mathbf{U}$. First, since the initial and target vertices are the same, these two problems are symmetric. Second, the congestion occurs on the link of the CSNCG problem. In an OPF problem, the connections between fog nodes and users construct the graph, and users' congestion also exists on the links (links between the target and the fog nodes). We suppose that the delay between fog nodes and users is fixed by d, and $d_v(x_v)$ is the delay function. When the number of users allocated to the fog node increases, the OPF problem will consider the limited number of processors, and the processing delay on the fog node will increase accordingly. So, the delay function $d_v(x_v)$ is nonnegative and nondecreasing. Finally, one route r from s to t represents one strategy in CSNCG, which is the same as the OPF problem. Let $\mathbf{X}$ indicate the provisioning strategy of $\mathbf{U}$. Each resource provisioning plan is a route from s to t. The total completion time of the user is $D = \frac{1}{n}(d + d_v(x_v))$, which is also the sum of values on all links equal to the definition in Reference 27. The OPF problem can be redefined as $(\mathbf{U}, \mathbf{G}, s, t, d_v(x_v))$.

Furthermore, the OPF problem is converted into the CSNCG problem. The premise for users to share the fog node is that all users can complete their workloads before the deadline. We build a graph according to the connections and information of fog nodes and users, and we add s and t to denote the virtual source and destination. Users in $\mathbf{U}$ distribute workloads starting from s, after that, they go through the fog nodes. The resource provisioning process is terminated when the users arrive at t. An example is shown in Figure 4, we have a relationship between fog nodes and users based on the connections in Figure 1. Since there is no congestion between the virtual source s and the user, the weight of the link is 0. In addition, in the middle part, since the delays between fog nodes and users are d, the links' weights between these two sets of nodes are d. In the right half part, the links' weights between the destination t and the internal nodes are the delays processing on fog nodes, which are related to the size of workloads and users. The converted graph of Figure 1 is presented in Figure 4.

5.2 | Algorithms

This subsection outlines two resource provisioning strategies for the LPFN problem. The key insight of the solution is removing fog nodes according to the local and global collaborative influences which are described in Algorithms 4 and 5. The inputs are $\mathbf{U}$, $\mathbf{G}$, and $d_v(x_v)$. We take the resource provisioning strategy $\mathbf{X}$ for $\mathbf{U}$ as the output. Algorithm 4 first constructs $\mathbf{G}''$ according to the connections between $\mathbf{U}$ and $\mathbf{G}$ in line 1, based on that it adds two virtual nodes s and t as the source and destination. Intuitively, all users in set $\mathbf{U}$ need to find a route from s to t with the constraint on deadline T. As we have discussed in the last subsection, the OPF problem can be converted into a CSNCG problem, Algorithm 4 replaces each link with n parallel ones with weight $d_v(1)$, $d_v(2)$, … , $d_v(n)$ between each node in $\mathbf{G}''$. Algorithm 4 finds a provisioning strategy with minimum delay using the min-cost flow and calculates the collaborative influence ψ_j for

Algorithm 4. Global influence greedy under LPFN

Input: Users' set **U**, topology **G**, delay function $d_v(x_v)$;
Output: Provisioning strategy **X** of **U**;

1: Construct $\mathbf{G}''$ in regard to the connections between **G** and **U**;
2: Replace the link with n parallel ones with weight $d_v(1), d_v(2), \dots, d_v(n)$ between each node in $\mathbf{G}''$;
3: **for** $j = 1$ to $j = n$ in **V do**
4: Get a provisioning with min-cost flow of $\mathbf{G}/v_j$;
5: Update ψ_j for v_j;
6: **end for**
7: Reconstruct the set **G** by reordering $j = \arg\min\{\psi_j/\tau_j\}$;
8: **while** $\mathbf{G} \neq \Phi \wedge D \leq T$ **do**
9: Remove v_j from **G**;
10: **end while**
11: **return** Provisioning strategy **X** of **U**;

Algorithm 5. Local influence greedy under LPFN

Input: Users' set **U**, topology **G**, delay function $d_v(x_v)$;
Output: Provisioning strategy **X** of **U**;

1: Same as Algorithm 4 in lines 1–2;
2: **for** $j = 1$ to $j = n$ in **V do**
3: Get a provisioning with min-cost flow of $\mathbf{G}/v_j$;
4: Update φ_j for v_j;
5: **end for**
6: Reconstruct **G** by reordering $j = \arg\min\{\varphi_j/\tau_j\}$;
7: Same as Algorithm 4 in lines 8–11;

the fog node in each iteration in lines 3–6. In line 7, Algorithm 4 rebuilds the **G** by an increasing order of the fog node with the global collaborative influence $j = \arg\min\{\psi_j/\tau_j\}$. Lines 8–11 are starting to remove the fog nodes based on the global collaborative influences, and the process terminates when either the delay for **U** exceeds the deadline T or the set **G** is empty.

Based on the above method, we introduce a local influence greedy LPFN (LIG-LPFN) in Algorithm 5 by removing the fog nodes according to the local collaborative influences. Similar to Algorithm 4 in lines 1–2, Algorithm 5 constructs a new graph $\mathbf{G}''$ and uses the same variables to be the input. However, the difference is that during each iteration, Algorithm 5 calculates the local collaborative influence ϕ_j for the fog node in line 4. Lines 6–7 reconstruct **G** by reordering $j = \arg\min\{\phi_j/\tau_j\}$ of fog nodes and remove them according to their local collaborative influences. The process of removing fog nodes is the same as Algorithm 4 (GIG-LPFN).

5.3 | Properties

The provisioning strategy is bounded by $\frac{8}{3}\mathrm{OPT} + \frac{\varepsilon^2}{8m\alpha}$ which is proved in Theorem 5. Before we prove that, we first introduce two prerequisites in Theorems 2 and 3.

Theorem 2. *Every OPF problem has at least one pure Nash equilibrium (NE) in LPFN.*

Proof. The work in References 28,29 has the proof that every potential game at least has one pure NE, which is represented as S that minimizes $\Phi(S)$. They demonstrated that any congestion game is a potential game,[30] and we make a conversion from the OPF problem to a CSNCG problem in subsection 5.1. The theorem holds. ∎

Since CSNCG is a continuous problem that cannot be solved in polynomial time, we discretize and transfer it to the discrete congestion game problem. Before the discretization, we have the following theorem, we first proved that the delay function satisfies the Lipschitz assumption. As a result, we have the following theorem

Theorem 3. *The delay function $d_v(x_v)$ satisfies the Lipschitz assumption.*

Proof. The delay function $d_v(x_v) = \alpha \cdot x_v + b$ is linear, it has a constant α, so that for the edge e and all $0 \le x < y \le 1$, $|d_e(y) - d_e(x)| \le \alpha|y - x|$. Therefore, we can prove that the delay function $d_v(x_v)$ satisfies the Lipschitz assumption. ∎

Theorem 4. *The bound of GIG-LPFN and LIG-LPFN is* $\frac{8}{3}\mathrm{OPT} + \frac{\varepsilon^2}{8m\alpha}$.

Proof. We suppose that $\delta = \frac{\varepsilon}{4m\alpha}$ is a minimum unit, where α is the upper bound on the Lipschitz constant of the delay function $d_v(x_v)$, and m is the number of fog nodes in use. ε is the ε-approximate NE [*]. We use $\varphi(f)$ to denote the potential function for the CSNCG problem, where $\varphi(f) = \sum_e \varphi(f)(x_v)$, $\varphi_e(x_v) = \int_0^{x_v}(\alpha \cdot x_v + b)\mathrm{d}t$. Then we have $\varphi_e(x_v) = \frac{1}{2}\alpha x_v^2 + bx_v$, and the delay function $\hat{C}(x_v) = x_v \cdot d_v(x_v) = ax_v^2 + bx_v$. We discuss the relationship between $\varphi_e(x_v)$ and $\hat{C}(x_v)$(continuous potential function) with $\frac{\varphi_e(x_v)}{\hat{C}(x_v)} = \frac{\frac{1}{2}ax_v + b}{ax_v + b}$. Accordingly, there are existing two extreme cases: none of the users offload their requests to this fog node, or all users' requests offload to this fog node. Therefore, $\frac{1}{2} \le \frac{\varphi_e(x_v)}{\hat{C}(x_v)} \le 1$. We have

$$\varphi_e(x_v) \le \hat{C}(x_v). \tag{7}$$

We use $C(f)$ to denote the function of total delay for the discrete condition, where $C(f) = \sum_e C_e(x_v)$. In Reference 27, the authors prove that $C(f)$ approximates the continuous potential function $\varphi(f)$ in the additive error of $\frac{\varepsilon^2}{16m\alpha}$. So, we get $|\varphi(f) - P(f)| \le \frac{\varepsilon^2}{16m\alpha}$, $P(f) - \varphi(f) \le \frac{\varepsilon^2}{16m\alpha}$. Based on that, we have $P(f) \le \varphi(f) + \frac{\varepsilon^2}{16m\alpha}$. Combining with Equation (8), we have

$$P(f) \le \hat{C}(x_v) + \frac{\varepsilon^2}{16m\alpha}. \tag{8}$$

Consequently, we analyze the relationship between the delay function and the discrete potential function. We discretize the problem by dividing it into δ units, the delay function for one user is $d(x_v) = \int_{m-\frac{1}{2}}^{m+\frac{1}{2}} f(t)\mathrm{d}t = \int_{m-\frac{1}{2}}^{m+\frac{1}{2}}(\alpha t + b)\mathrm{d}t = am + b$. Thus, the delay function on the fog node is $c(x_v) = md(x_v) = am^2 + bm$. The potential function in the discrete case is $p(x_v) = \sum_1^m d(x_v) = a(1 + 2 + \ldots + m) + bm = \frac{m(m+1)}{2}a + bm$. $\frac{p(x_v)}{c(x_v)} = \frac{\frac{m(m+1)}{2}a + bm}{am^2 + bm} = \frac{\frac{1}{2}m + \frac{a}{2} + b}{am + b}$, where $\frac{1}{2} \le \frac{p(x_v)}{c(x_v)} \le 1$. Since the total delay function and the potential function are the sum of $c(x_v)$ and $p(x_v)$ of all fog nodes, the relationship between $P(x_v)$ and $C(x_v)$ is also $\frac{1}{2} \le \frac{P(x_v)}{C(x_v)} \le 1$. Then we have $P(x_v) \ge \frac{1}{2}C(x_v)$. Equation (8) will be transformed into

$$\frac{1}{2}C(x_v) \le \hat{C}(x_v) + \frac{\varepsilon^2}{16m\alpha}. \tag{9}$$

For each linear delay function, it has proved that there is an $\frac{4}{3}$-approximation ratio in the NE of any CSNCG problem,[31] we can obtain that $\hat{C}(x_v) \le \frac{4}{3}\mathrm{OPT}$. Then we have

$$C(x_v) \le \frac{8}{3}\mathrm{OPT} + \frac{\varepsilon^2}{8m\alpha}. \tag{10}$$

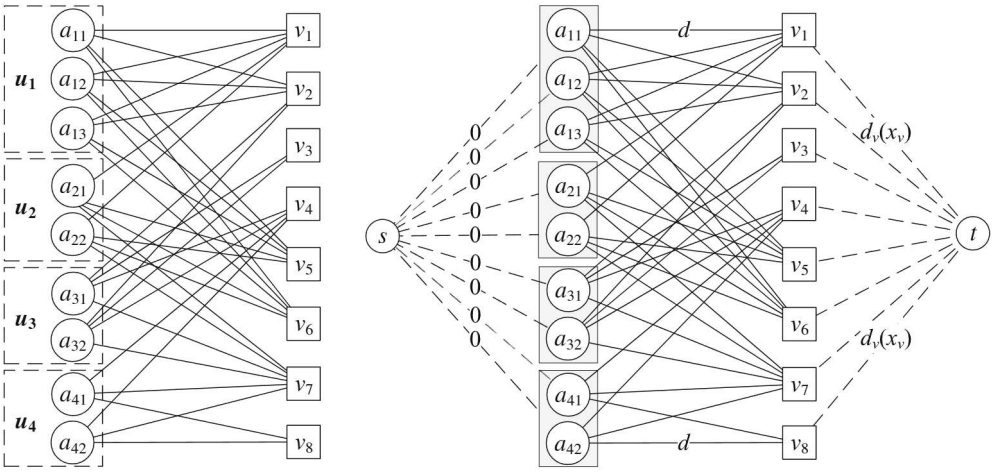
∎

The time complexity of LIG-LPFN and GIG-LPFN is $O(|\mathbf{V}|\cdot(|\mathbf{E}|)^3)$. $O(|\mathbf{V}| + |\mathbf{U}| + |\mathbf{E}|)$ results from the construction of graph $\mathbf{G}''$. Since the number of edges $|\mathbf{E}|$ in $\mathbf{G}''$ is much larger than the numbers of vertexes $|\mathbf{V}|$ and $|\mathbf{U}|$, we simplify that $O(|\mathbf{V}| + |\mathbf{U}| + |\mathbf{E}|) = O(|\mathbf{E}|)$. We adopt a provisioning strategy by traversing each fog node with minimum delay, and the min-cost flow algorithm complexity is $O((|\mathbf{V}| + |\mathbf{E}|)^2 \cdot |\mathbf{E}|) = O(|\mathbf{E}|^3)$. Consequently, the complexity of LIG-LPFN and GIG-LPFN is $O(|\mathbf{V}|\cdot(|\mathbf{E}|)^3)$.

[*] ε-approximate NE means that for every user i, every flow path carrying at least ε units of flow, and the cost on every path is larger than $c_e(x_v) + \varepsilon$.[27]

6 | RESOURCE PROVISIONING WITH APPLICATION-DRIVEN TASKS

In this section, we consider another scenario where users' workloads are application-driven. It means that each workload contains multiple subtasks and cannot be divided arbitrarily. For each user, multiple subtasks of one workload can be assigned to different fog nodes, and each subtask can only be executed on one node at the same time. In Figure 5(A), we assume that the workload of each user contains several subtasks, u_1 contains three subtasks a_{11}, a_{12}, and a_{13}, while u_2, u_3, and u_4 contain two subtasks, respectively. According to the motivation example in Figure 1, we have the connections between fog nodes and users. From this, we can artificially generate the connectivity graph as shown in Figure 5(B). For the UPFN case, the resource provisioning problem can be transferred into an assignment problem which can be solved by using the Hungarian method. For the LPFN case, the problem will be more complicated. In order to solve this problem, we propose a new heuristic algorithm.



(A) Connectivity graph.

(B) Weighted connectivity graph.

FIGURE 5 The structure of Figure 1 for the application-driven tasks

Algorithm 6. LPFN for application-driven tasks

Input: Users' set **U**, topology **G**, delay function $d_v(x_v)$;
Output: Provisioning strategy **X** of **U**;

1: Initialize chosen set of fog nodes $\mathbf{I} \leftarrow \Phi$, $D = \infty$;
2: $\mathbf{U}' \leftarrow \mathbf{U}$;
3: **while** $\mathbf{U}' \neq \Phi$ **do**
4: Select a set $P_j \in \mathcal{P}$ that maximizes $|P_j \bigcap \mathbf{U}'|$;
5: $\mathbf{U}' \leftarrow \mathbf{U}' - P_j$;
6: $\mathbf{X} \leftarrow \mathbf{X} \bigcup \{P_j\}$;
7: **end while**
8: $\mathbf{I} \leftarrow \mathcal{P} - \mathbf{X}$;
9: **while** $D > T$ **do**
10: Construct $\mathbf{G}''$ in regard to the connections of **X** and **U**;
11: Same as Algorithm 4 (GIG-LPFN) in lines 2–6;
12: **while** $\mathbf{I} \neq \Phi \wedge \mathbf{U} \neq \Phi$ **do**
13: Select a set $P_j \in \mathbf{I}$ with minimum ς;
14: $\mathbf{X} \leftarrow P_j$;
15: Reprovisioning users in set P_j with fog node v_j;
16: **end while**
17: **end while**
18: **return** Provisioning strategy **X** of **U**;

In this subsection, we introduce a new heuristic algorithm. We take the users' set $\mathbf{U}$, the fog nodes' topology $\mathbf{G}$, and the delay function $d_v(x_v)$ as the inputs. The provisioning strategy $\mathbf{X}$ for $\mathbf{U}$ is the output. First, we initialize the chosen set of fog nodes $\mathbf{I}$ as an empty set, and the value of delay for the set of users $\mathbf{U}$ is set to be infinite in line 1. Then we start to find the minimum set of fog nodes that can serve all users, which is shown in lines 2–6. In line 2, we construct a temporary set $\mathbf{U}'$ with the same elements of $\mathbf{U}$. Let P_j denote the set of users that fog node v_j can serve, and $\mathcal{P}= \bigcup_{j\in\mathbf{V}}\{P_j\}$. In lines 3–7, we start to find the set that covers all users with minimum cost, where $\mathbf{U}= \bigcup_{P_j\in\mathcal{P}}\{P_j\}$. In line 4, we select a set $P_j \in \mathcal{P}$ that maximizes $|P_j \cap \mathbf{U}'|$. Then we remove the corresponding from the temporary set $\mathbf{U}'$, and add the chosen set P_j into $\mathbf{X}$. This process stops when the temporary user set $\mathbf{U}'$ is empty. In line 8, we put the fog nodes that have not been assigned in set $\mathcal{P}$ into set $\mathbf{I}$. In lines 9–16, we do the feasibility checking for the current provisioning strategy $\mathbf{X}$, and we start to reprovision by adding new fog nodes. This process will stop when the delay beyond the deadline T. Line 10 constructs $\mathbf{G}''$ according to the connections of the new set of fog nodes $\mathbf{X}$ and the set of users $\mathbf{U}$. Based on that, we do the same operations as lines 2–6 of GIG-LPFN that to find a provisioning strategy with minimum delay using the min-cost-flow. In lines 13–15, we select a set $P_j \in \mathbf{I}$ with minimum factor ς, where $\varsigma = \frac{|P_j|}{r_j}$. $|P_j|$ is the number of users that are covered by fog node v_j, and r_j is the maximum processing rate of fog node v_j. In line 14, we update the provisioning strategy $\mathbf{X}$ by adding set P_j. Then we reprovision users in set P_j with fog node v_j. Either the set $\mathbf{I}$ or the user set $\mathbf{U}$ is empty, the adding process will be terminated. Finally, the provisioning strategy $\mathbf{X}$ for $\mathbf{U}$ is returned in line 18. The time complexity of LPFN-AT is $O(|U| \cdot |\mathcal{P}|)$, where $|U|$ is the number of users and $\mathcal{P}$ is the number of fog nodes.

7 | EVALUATIONS

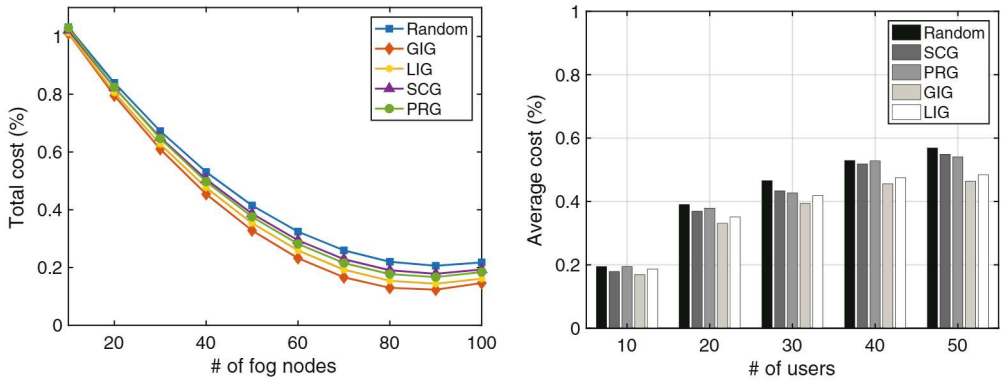
In this section, we will conduct experiments on both synthetic and real datasets to study the cost-efficiency of resource provisioning in delay-sensitive collaborative fog computing networks.

7.1 | Basic setting of the synthetic dataset

In this subsection, we develop a prototype of our algorithms using MATLAB, which consists of the construction of fog computing network and the users' requests. We construct the fog computing network by generating a graph model function *construct_graph* and evaluate the performance of our algorithms by building a synthetic dataset. In the synthetic scenario, the number of fog nodes ranges from 0 to 100, and we record the users' average cost by dividing them into 10 groups. We use unit weight to measure users' workloads, and each unit is denoted by 1GB workload.[32] The workloads' sizes are generated randomly from 0 to 50, which follows the uniform random distribution. The deadline of each set of users is generated randomly from 0 to 10 ms. We suppose that the processing capabilities of the fog nodes are different, and they are connected as an undirected graph. Assuming that the processing capability of the fog node is denoted by the number of unit weight processed per millisecond, which follows the uniform random distribution between 1 and 10. For each fog node, the set-up costs are generated randomly from 0.01 to 0.1 million dollars according to different capacities. In our experiments, we design three benchmark algorithms referring to the partial idea of References 11,17. Since the constraints and the main idea of solving the problem have many differences. Such as the main idea of our solution is to iteratively pre remove the fog nodes, however, in References 11,17, they consider the resource allocation in fog computing which is an adding or allocating process. We do not direct reproduce the algorithms in these references and only use some ideas by designing our benchmark algorithms. Thus, three benchmark algorithms (Random, SCG, and PRG) are used for comparisons as follows: (i) Random removes the fog node by a random order iteratively. (ii) Set-up cost greedy algorithm (SCG) removes the fog node by an increasing order with the set-up cost iteratively. (iii) Processing rate greedy algorithm (PRG) removes the fog node in increasing order of the maximum processing rate iteratively.

7.2 | Simulation results for UPFN

This subsection discusses the average cost ratio of multiple users under the UPFN scenario which presents in Figure 6. Three benchmark approaches (Random, SCG, PRG) are used to compare with our algorithms. We choose five groups of users, ranging from 10 to 50. In order to facilitate the analysis of the results, we calculate the cost ratio which is the quotient



(A) The total cost of 30 users.

(B) Average cost of users.

FIGURE 6 Performance comparison under the unlimited-processor fog nodes scenario

value of the actual cost and the cost of all fog nodes. According to the results, we obtain the following observations: (i). The impact of algorithms on the total costs is related to the number of fog nodes. We first analyze the relationship between costs and fog nodes by choosing the median group (30 users). Figure 6(A) shows the evaluation result that the total costs of users are fluctuating with a different number of fog nodes. Compared with the other three benchmark algorithms, the decreasing speeds of GIG and LIG are faster. What's more, the connections of fog nodes change according to scale, and the total cost fluctuates. It indicates that the total cost is related to the connections and locations of the fog nodes. For a small scale of fog nodes, the performance gap among these five algorithms is not obvious. However, when the scale of fog nodes is increasing, the evaluation results of PRG and SCG fluctuate between GIG and Random. Since GIG considers overall fog nodes, the cost of fog nodes becomes lower than that of LIG when the scale of the fog nodes increases. (ii). The performance gap in the total costs of algorithms is also related to the distribution of users' workloads. We evaluate the impact of the distributions of users' workloads, in terms of the relationship between the average costs and different groups of users by ranging from 10 to 50. Figure 6(B) shows the evaluation result that the average costs are fluctuating with different groups of users. When the distributions of users' workloads are the same, a larger number of users in one group will result in a higher average cost. We can see the evaluation result on the average cost shows that 10 users is much lower than that of 50. The performance gap is not obvious for the small-scale fog nodes as the weights distribute randomly. However, for the large-scale fog nodes, algorithms GIG and LIG are more efficient. The evaluation results of GIG and LIG are similar, and the slight performance differences are caused by the topology of fog nodes. Moreover, compared with the three benchmark algorithms, the average costs of LIG and GIG are lower which reach 11.1% and 14.8%, respectively.

7.3 | Simulation results for LPFN

This subsection discusses the average cost ratios of multiple users under the LPFN case which presents in Figure 7. The settings are the same as the UPFN scenario. Figure 7(B) shows the preconfiguration on the values of α ranging from $[1, 5]$ and calculates the average total cost. We do the same operation as in the previous subsection and first select the group with 30 users to analyze the trend. Then, we use three delay functions different α values, where $\alpha = 1$, $\alpha = 3$, and $\alpha = 5$. Based on that, we calculate the average cost ratios range from 10 to 50 ($[10, 50]$) for five groups of users. In addition, we obtain the following observations according to the results: (i). When the sizes of workloads are scaling up, the resource provisioning strategy has a greater impact on the performance of the cost-efficiency. What's more, the users' costs using fog nodes are related to the connectivities and localities of the topology. Figure 7(A) shows the comparison of evaluation results with 10 groups of fog nodes. The average cost increases along with the scaling of users' numbers, which means that more workloads require more resources. (ii). For the users processing on fog nodes in the same set, the total costs in the scenario of LPFN are larger than those in the scenario of UPFN. In the scenario of LPFN, the processing capacities limitation of fog nodes is considered, which indicates that the workload processing on each fog node of users will be increased. Compared the evaluation results between Figures 6(B) and 7(B), LPFN has larger average costs than UPFN of these five algorithms with the same group of users. (iii). As the total workload of users for the fog nodes increases, the algorithm has a greater impact on the average cost. In addition, we can observe that when the numbers of users are 30 and 40, the average performances of algorithms LIG-LPFN and GIG-LPFN are almost the same. Moreover, the average costs of LIG-LPFN and GIG-LPFN are lower than comparison algorithms, which achieve 15.2% and 20.3%, respectively.

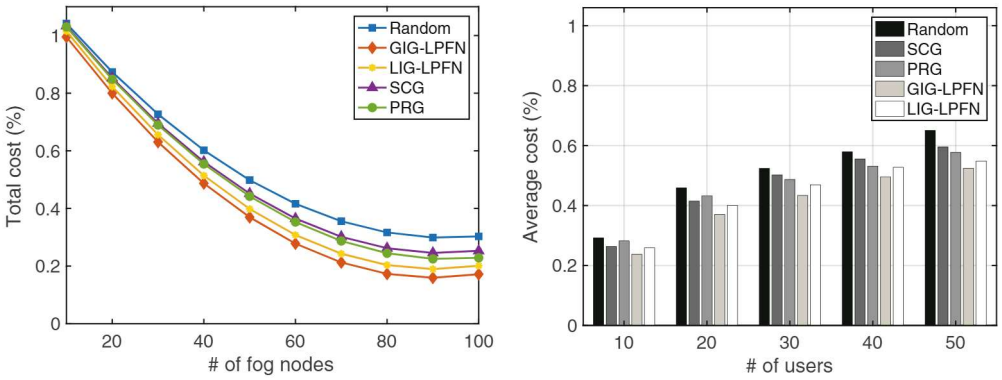


FIGURE 7 Performance comparison under the limited-processor fog nodes scenario

(A) The total cost of 30 users. (B) Average cost of users.

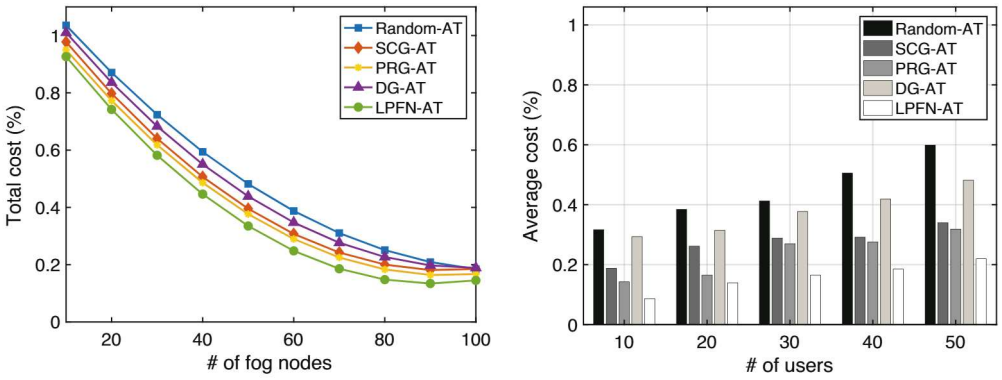


FIGURE 8 Performance comparison under the LPFN-AT scenario

(A) The total cost of 30 users. (B) Average cost of users.

7.4 | Simulation results for LPFN-AT

This subsection discusses the average cost ratios of multiple users for the resource provisioning problem with the application-driven tasks under the LPFN case. We use the same basic setting except for adjustments to the comparison algorithms. In this experiment, four benchmark approaches (Random-AT, SCG-AT, PRG-AT, and DG-AT) are used to compare with our algorithms: (i) Random-AT iteratively adds fog nodes in random order. (ii) SCG iteratively adds fog nodes in increasing order of the set-up costs. (iii) Processing rate greedy algorithm (PRG-AT) adds fog nodes iteratively in increasing order of the maximum processing rate. (iv) Distance greedy algorithm (DG-AT) iteratively adds fog nodes with the shortest distance.[33]

We first select 10 to 50 users to calculate their average cost ratios, and then we choose the median group 30 of the users' set to analyze the trend. According to the results, we obtain the following observations: (i). For the same set of users, since the expansion of their selection range, the scaling of fog nodes will reduce the cost, as shown in Figure 8(A). When the operator provides fewer fog nodes, the gap between different algorithms is not obvious, and the total costs of SCG-AT and PRG-AT algorithms are close. However, the total cost of DG-AT is always higher than the other two benchmark algorithms, which means fog nodes that are close to the users may have problems such as limited processing capacities and large costs. LPFN-AT has lower average costs which can reach 18.6% under the 30 users case than the comparison algorithms.

Compared with the other four benchmark algorithms, (ii). When the number of users in one group increases, the average cost has the same trend. Figure 8(B) shows the comparison of evaluation results with five groups of users. When the number of users is small, the gap between the cost of PRG-AT and that of LPFN-AT is not large. However, when the number of users increases, the total cost of LPFN-AT decreases significantly. In addition, the total costs of PRG-AT and DG-AT are higher than LPFN-AT. It means that although the cost of serving users related to the localities and properties of the fog nodes topology, a single consideration of one constraint cannot get a lower cost. In summary, LPFN-AT has a lower average cost than the comparison algorithms which can reach 14.3%.

7.5 | Real dataset

7.5.1 | Dataset setting

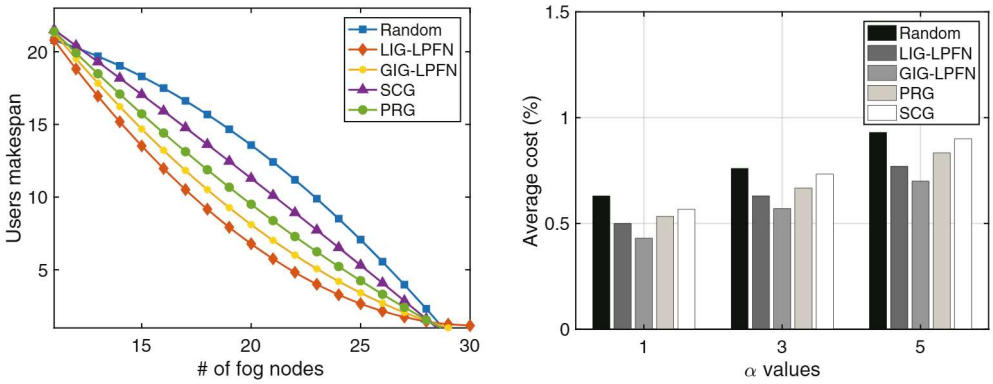
The real dataset we used is published by the New York City (NYC) open data website. In order to do the same setting as the general scenario, we combine three different and related datasets, which are the NYC entrances of the subway stations, transit subway entrance data, and the Wi-Fi hotspot locations.[34-36] We choose a dataset with a 4-month-long trip from December 23, 2017 to April 23, 2018, which includes the information on station entrances of subway from 3 : 00am to 23 : 00pm of users. These records are constructed by several attributes which include the station entrances' names, time, date, and the number of passengers. We take the daily data as a unit and calculate the average values over 4 months for users. We first point out the locations of Wi-Fi hotspots in NYC (as shown by the red points in Figure 9A), then we mark out the entrances of the subway stations covered by these Wi-Fi hotspots (as shown by the blue points in Figure 9A). For the locations of the envisioned fog nodes, we use the locations of Wi-Fi hotspots to represent fog nodes that serve a decent coverage for users in the subway stations. We assume that each user only uploads one unit of workload, so the number of users that passes through one entrance can represent the total workload uploaded to that subway station (as shown by the yellows areas in Figure 9B).

7.5.2 | Simulation results under the real dataset

This subsection focuses on performance comparison under the LPFN-AT scenario according to the real dataset. There are 171 fog nodes in the real dataset, which are constructed by 141 collaborative fog nodes and 30 access nodes. The beginning time and ending time are 3 : 00 a.m. and 23 : 00 p.m., respectively, we set 23 : 00 p.m. to be users' deadline. We set the delay functions by $\alpha = 1$, $\alpha = 3$, and $\alpha = 5$. Figure 10 shows the results with 10 group of fog nodes, and we obtain the following observations. According to the experiment results shown in Figure 10(A), we find that with the decreasing number of fog

(A) New York City (B) Users Distribution

FIGURE 9 Subway locations with users distribution in New York City



(A) Users makespan. (B) Average cost of users.

FIGURE 10 Average cost with the real dataset

nodes, the makespan of users increasing. The performances in the four benchmark algorithms are compared, LIG-LPFN and GIG-LPFN decrease the fastest. After that, we evaluate users' average cost under $\alpha = 1, \alpha = 3$, and $\alpha = 5$, and the result is shown in Figure 10(B). The result shows that users' average cost becomes larger under a delay function with a higher α value. What's more, with the increasing scale of the fog node, LIG-LPFN and GIG-LPFN have a better performance than the other four benchmark approaches. According to the comparison results, algorithms of Random, SCG, and PRG have higher costs on the synthetic dataset than LIG-LPFN and GIG-LPFN, which can average achieve 10.8% and 14.9%, respectively. In addition, they can obtain lower costs on the real dataset than the comparison algorithms, which can on average achieve 12.8% and 11.4%, respectively.

8 | CONCLUSION

This article has concentrated on the resource provisioning problem in collaborative fog computing with capacity constraints in order to minimize the total cost. First, we prove the NP-hardness of the resource provisioning problem with minimum cost for multiple users. Based on that, we discuss two different scenarios. One is UPFN that two greedy strategies are proposed which iteratively remove fog nodes based on the global and local collaborative influences. The other one is LPFN, in which the fog nodes have delays in processing workloads of multiple users. We consider two different types of tasks. For the arbitrarily divided tasks, we propose a near-optimal strategy with bound $\frac{8}{3}\mathrm{OPT} + \frac{\epsilon^2}{8m\alpha}$ which mainly based on the continuous congestion game, where α is the upper bound on the Lipschitz constant of the delay function and m is the number of fog nodes. For the application-driven tasks, we propose a heuristic algorithm. Extensive simulations validate that our algorithms can reduce the resultant average costs of 12.9% and 17.8% in UPFN and LPFN, respectively. Moreover, the evaluation results on the real dataset validate that the algorithms we propose can reduce users' latency and save the average cost.

ACKNOWLEDGMENTS

This research was supported in part by National Natural Science Foundation of China 61202076, Beijing Natural Science Foundation 4192007, Beijing Municipal Postdoc Science Foundation Q6007011202001, Chaoyang District Postdoc Science Foundation, and NSF grants CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and IIP 1439672.

REFERENCES

1. Shanhe Y, Cheng L, Qun L. A survey of fog computing: concepts, applications and issues. Paper presented at: Proceedings of the 2015 Workshop on Mobile Big Data. Association for Computing Machinery. Hangzhou, China; 2015:37-42.
2. Cisco The Internet of Things: extend the cloud to where the things are 2015. http://www. cisco.com/c/dam/en_us/solutions/trends/iot/docs/computingoverview. pdf. Accessed 2015.
3. Bonomi F, Milito R, Zhu J, Addepalli S. fog computing and its role in the Internet of Things. Paper presented at: Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing. Association for Computing Machinery. Helsinki, Finland; 2012:13-16.
4. Xiao Y, Krunz M. QoE and power efficiency tradeoff for fog computing networks with fog node cooperation. Paper presented at: Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. Atlanta, GA; 2017:1-9; IEEE.
5. Shuaibing L, Jie W, Yubin D, Ning W, Zhiyi F. Cost-efficient resource provisioning in delay-sensitive cooperative fog computing. Paper presented at: Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). Singapore; 2018:706-713.
6. Wu J, Wang Y. *Opportunistic Mobile Social Networks*. Boca Raton, FL: CRC Press; 2014.
7. Brogi A, Forti S. QoS-aware deployment of IoT applications through the fog. *IEEE IoT J*. 2017;4(5):1185-1192.
8. Xuan-Qui P, Eui-Nam H. Towards task scheduling in a cloud-fog computing system. Paper presented at: Proceedings of the 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS). Kanazawa, Japan; 2016:1-4.
9. Santos J, Wauters T, Volckaert B, De Turck F. Towards network-aware resource provisioning in kubernetes for fog computing applications. Paper presented at: Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft). Paris, France; 2019:351-359.
10. Yu R, Xue G, Zhang X. Application provisioning in FOG computing-enabled Internet-of-Things: a network perspective. Paper presented at: Proceedings of the IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. Honolulu, HI; 2018:783-791.
11. Lin W , Jiao L, Kliazovich D, Bouvry P. Reconciling task assignment and scheduling in mobile edge clouds. Paper presented at: Proceedings of the 2016 IEEE 24th International Conference on Network Protocols (ICNP). Singapore; 2016:1-6.
12. Wang L, Jiao L, He T, Li J, Mühlhäuser M. Service entity placement for social virtual reality applications in edge computing. Paper presented at: Proceedings of the IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. Honolulu, HI; 2018:468-476.

13. Sheng Z, Mahapatra C, Leung VCM, Chen M, Sahu PK. Energy efficient cooperative computing in mobile wireless sensor networks. *IEEE Trans Cloud Comput*. 2018;6(1):114-126.
14. Ranesh Kumar N, Saurabh G, Andrew C, Sudheer KB. Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. *Futur Gener Comput Syst*. 2020;104:131-141.
15. Bierzynski K, Escobar A, Eberl M. Cloud, fog and edge: cooperation for the future? Paper presented at: Proceedings of the 2017 2nd International Conference on Fog and Mobile Edge Computing (FMEC). Valencia, Spain; 2017:62-67.
16. Huang X, Fan W, Chen Q, Zhang J. Energy-efficient resource allocation in fog computing network with the candidate mechanism. *IEEE IoT J*. 2020;7(9):8502-8512.
17. Chang W, Wu J. Progressive or conservative: rationally allocate cooperative work in mobile social networks. *IEEE Trans Parall Distrib Syst*. 2015;26(7):2020-2035.
18. Ningning S, Chao G, Xingshuo A, Qiang Z. Fog computing dynamic load balancing mechanism based on graph repartitioning. *China Commun*. 2016;13(3):156-164.
19. Gu B, Yapeng C, Haijun L, Zhenyu Z, Di Z. A distributed and context-aware task assignment mechanism for collaborative mobile edge computing. *Sensors*. 2018;18(8):2423.
20. Elarbi B. Qos-aware placement of tasks on a fog clusters in an edge computing environment. *J Ubiquitous Syst Pervas Netw*. 2020;13(1):11-19.
21. Yin L, Lou J, Luo H. Task scheduling and resource allocation in fog computing based on containers for smart manufacturing. *IEEE Trans Ind Inform*. 2018;14(10):4712-4721.
22. Ma X, Wang S, Zhang S, Yang P, Lin C, Shen XS. Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing. *IEEE Transactions on Cloud Computing (Early Access)*. 2019;1–1.
23. Fatemeh J, Shahidinejad A, Mostafa G. Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach. *Journal of Ambient Intelligence and Humanized Computing*. 2020;3:1–20.
24. Zhang H, Bu S, Niyato D, Yu FR, Han Z. Computing resource allocation in three-tier IoT fog networks: a joint optimization approach combining Stackelberg game and matching. *IEEE IoT J*. 2017;4(5):1204-1215.
25. Li L, Ota K, Dong M. Deep learning for smart industry: efficient manufacture inspection system with fog computing. *IEEE Trans Ind Inform*. 2018;14 (10):4665–4673.
26. Cormen TH. *Introduction to Algorithms*. Cambridge, MA: MIT Press; 2009.
27. Fabrikant A, Papadimitriou C, Talwar K. The complexity of pure Nash equilibria. Paper presented at: Proceedings of the 36th Annual ACM Symposium on Theory of Computing. Valencia, Spain; 2004:604-612.
28. Nisan N, Roughgarden T, Tardos E, Vazirani VV. *Algorithmic Game Theory*. Cambridge, MA: Cambridge University Press; 2007.
29. Von Neumann J, Morgenstern O. *Theory of Games and Economic Behavior*. Commemorative ed. Princeton, NJ: Princeton University Press; 2007.
30. Rosenthal RW. A class of games possessing pure-strategy Nash equilibria. *Int J Game Theory*. 1973;2(1):65-67.
31. Roughgarden T, Tardos É. How bad is selfish routing? *J ACM*. 2002;49(2):236-259.
32. Chuang H, Bao W, Wang D. IoT communication sharing: scenarios, algorithms and implementation. Paper presented at: Proceedings of the IEEE INFOCOM 2018 - IEEE Conference on Computer Communications; 2018:114-126.
33. Swati A, Shashank Y, Arun KY. An efficient architecture and algorithm for resource provisioning in fog computing. *International Journal of Information Engineering and Electronic Business*. 2016;8(1):48.
34. MTA information, average weekday subway ridership; 2018. http://web.mta.info/nyct/facts/ridership/ridership_sub.htm. Accessed 2018.
35. New York city open data, entrances of NYC subway stations; 2018. https://data.cityofnewyork.us/Transportation/Subway-Entrances/drex-xx56. Accessed 2018.
36. New York city open data, NYC Wi-Fi hotspot locations; 2018. https://data.cityofnewyork.us/Social-Services/NYC-Wi-Fi-Hotspot-Locations/a9we-mtpn. Accessed: 2018.

How to cite this article: Lu S, Wu J, Wang N, et al. Resource provisioning in collaborative fog computing for multiple delay-sensitive users. *Softw Pract Exper*. 2023;53:243–262. https://doi.org/10.1002/spe.3000