Cost-Efficient Resource Provisioning in Delay-Sensitive Cooperative Fog Computing

Shuaibing Lu*†, Jie Wu†, Yubin Duan†, Ning Wang†, and Zhiyi Fang*
*College of Computer Science and Technology, Jilin University, Changchun, China
†Center for Networked Computing, Temple University, USA
Email: lushuaibing11@163.com, {jiewu, yubin.duan, ning.wang}@temple.edu, fangzy@jlu.edu.cn

***Abstract*—Recently, fog computing has become a highly virtualized platform that provides computation, storage, and networking services between end devices and traditional cloud data centers. In this paper, we address the resource provision (RP) problem for delay-sensitive users in cooperative fog computing. Our objective is to find a feasible provision scheme that minimizes the total monetary cost proportional to the number of fog nodes for network operators under the deadline and capacity constraints by considering the cooperation of fog nodes. We consider two cases of our RP problem: the Unlimited-Processor Fog Nodes (UPFN) case and the Limited-Processor Fog Nodes (LPFN) case. For the UPFN case, each fog node has unlimited processors. The requests on each fog node can be processed in parallel ideally, i.e. with no scheduling delay. The LPFN case corresponds to a more realistic scenario where the scheduling delay is non-eligible. In either case, our RP problem is proven to be NP-hard. For the UPFN case, we propose two greedy algorithms which iteratively remove fog nodes according to their global or local cooperative influences until there is no feasible provision that can guarantee users' deadlines. For the LPFN case, it is not trivial to check the existence of a feasible provision due to the interactive influence on the scheduling delay for requests. We find a near-optimal solution with bound $\frac{8}{3}OPT + \frac{\epsilon^2}{8m\alpha}$ using the continuous congestion game and check the feasibility, where m is the number of fog nodes and α is a constant value related to the delay function. Extensive simulations demonstrate the efficiency of our schemes.**

***Index Terms*—Fog computing, delay-sensitive, cooperation, cost efficiency.**

I. Introduction

With the extensive growth of data volumes coming from the Internet of Things (IoT), fog computing has become a highly virtualized platform that provides computation, storage, and networking services between end devices and traditional cloud data centers. Fog nodes are the basic infrastructures in fog computing, including industrial controllers, switches, routers, embedded servers, and video surveillance cameras [1–4]. Since the IoT devices generate data constantly, and the analysis must be very rapid, an important mission is to find a provision of fog nodes that can reduce the transmission latency and decrease monetary cost of fog resources. In this paper, we focus on the resource provision (RP) problem for delay-sensitive users under the capacity constraints by considering the cooperation of fog nodes, while realizing cost efficiency of network operators in fog computing.

Our RP problem arises in the following set-up scenario. We consider a set of heterogeneous fog nodes ($v_{(1-6)}$), and each fog node has the limited computing capability that can be provisioned for users ($u_{(1-3)}$). The fog node supports the cooperation service of different providers, and the cost for the resource provision scheme is generated by the set-up cost of each fog node. We assume that the locations and connections of fog nodes are already fixed by the cloud data centers or third party service providers. Each user can upload the workload to its nearest fog node with a fixed deadline T. The workload is divided into identical service entities, and each service entity is denoted by one unit. The resource provision for users is to allocate the service entities of workload to fog nodes. Based on our setting-up scenario, there exists a trade-off between efficiency and cost. We illustrate the trade-off through the following example. As shown in Fig. 1, one extreme assignment is the solution with the minimal delay for the set of users, which offloads the workload to all fog nodes that are connected in the graph, represented by $v_{(1-6)}$. However, the total cost of this solution is the maximum among all possible assignments that satisfy the deadline constraint. Another extreme assignment is to minimize the cost, i.e., by minimizing the number of fog nodes, and we use the minimal number of fog nodes to support the users. In Fig. 1, we can only use v_1, v_2 and v_3. When the sizes of workloads are large, the total delay of users in the set will go over the deadline T.

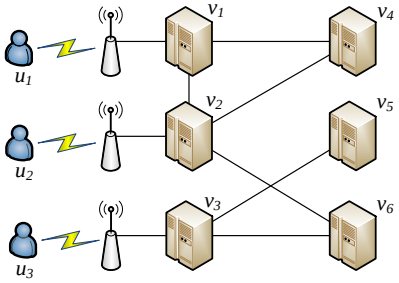
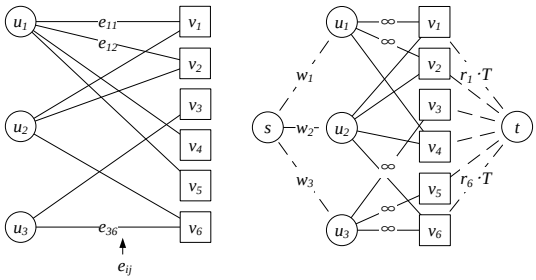


Fig. 1. A motivation example.

In this paper, we propose an efficient resource provision scheme for users that falls between the two extreme assignments. Our RP problem is how to assign the computational workloads to users on fog nodes to minimize the cost during the process for multiple users while satisfying the constraints on users' deadline and the computation resources of fog nodes. We further consider two cases for our RP problem: the Unlimited-Processor Fog Nodes (UPFN) case and the Limited-Processor Fog Nodes (LPFN) case. For the UPFN case, each fog node has unlimited processors. The requests on each fog node can be processed in parallel ideally, i.e. with no scheduling delay. For the LPFN case, we consider a



(a) Connectivity graph. (b) Weighted connectivity graph.

Fig. 2. The substructure of Fig 1.

more complicated and realistic case, in which each fog node has limited processors and has scheduling delay on processing several users' requests.

This problem is non-trivial mainly due to the following challenges: (i) Because fog nodes are heterogeneous, and each fog node has a limited computing capability. Given a set of users with different amounts of workload, it is non-trivial to find a feasible provision for users that can realize the workload within a deadline. (ii) Since the fog node can only be connected with limited users, each user has to offload the work to fog nodes that are located in its efficient area. The provision scheme for one user may not be suitable for multiple users. (iii) Although the cooperation between fog nodes can reduce the delay for users, it increases the cost. There is a trade-off between the delay and cost. It is difficult to balance them while still guaranteeing users' demands with the lowest cost. In this paper, we focus on the RP problem for delay-sensitive users under the capacities constraints while realizing cost efficiency of network operators in fog computing. Our contributions can be summarized as follows:

- We consider the RP problem for delay-sensitive users and show that there is a trade-off between efficiency and cost. We prove that deciding resource provision for users with cost minimization is NP-hard.
- We first discuss a simple case UPFN. We define a parameter cooperative influence to minimize the cost by removing fog nodes iteratively. Two greedy algorithms are proposed, and their complexities analysis are included.
- We extend our problem into a more complicated and realistic case LPFN. We formulate this problem by converting it into a continuous congestion game problem, and propose an algorithm which is bounded by $\frac{8}{3}OPT+\frac{\epsilon^2}{8m\alpha}$.
- We conduct various simulations to compare our joint optimization methods with several state-of-the-art ones. The results are evaluated from different perspectives to provide conclusions.

II. Related work

Fog computing extends the cloud computing paradigm to the edge of the network, thus enabling a new breed of applications and services[2, 3]. Different from the cloud, fog computing deploys a very large number of fog nodes in the wide-spread geographical distribution, and provides services closer to both the IoT devices and users. Several works have been done on the resource provision in fog computing [5, 6]. In [5], they focus on supporting the QoS-aware deployment of multicomponent IoT applications to Fog infrastructures by developing a java tool FogTorch. [6] considers task scheduling in a cloud-fog computing system, where a fog provider can exploit the collaboration between its own fog nodes and the rented cloud nodes. However, these resource provision schemes fail to consider the cooperation of fog nodes. In a case similar to the fog computing, [7] investigates the assignment and scheduling of mobile computational tasks over multiple cloudlets, while optimizing the overall cost efficiency by leveraging the heterogeneity of cloudlets in mobile edge. [8, 9] focus on the service entity placement for social virtual reality applications in edge computing. These works consider the performance guarantee for each mobile user, while ignoring the interdependent relationship between the monetary cost and user's delay.

Quite few works consider the effect of cooperative fog computing network on joint optimization of the latency and cost. [4] uses a distributed alternating direction method of multipliers to optimize the QoE of users under the given power efficiency, and they propose that multiple fog nodes can help each other to jointly offload workload to cloud data centers. [10] provides the implementation of such IoT solutions by highlighting approaches for distributing workload between cloud, fog and edge. [11] studies a new cooperative system in mobile social networks, focusing on the partition and allocation of workloads. Most of these works focus on the RP problem by only considering the latencies of users. They fail to consider the cost of fog infrastructures. In this paper, we focus on the RP problem for delay-sensitive users in fog computing. Our objective is to find a feasible provision scheme that minimizes the total cost of network operators under the capacities constraints by considering the cooperation of fog nodes.

III. Model and Problem Formulation

A. Fog Model

Given a substrate distribution of fog nodes which is modeled as a graph $\mathbf{G}$ with a node set $\mathbf{V}$ and an edge set $\mathbf{E}$, i.e., $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$. Let $\mathbf{V} = \{v_j\}$ denote the set of fog nodes, and v_j is the j^{th} fog node in $\mathbf{V}$. We consider a set of fog nodes that can be any device with computing, storage, and network connectivity. Fog nodes are heterogeneous, and each fog node has limited computing capability for provision. Let $\mathbf{E} = \{e_{ij}\}$ denote the set of connections between fog nodes, and e_{ij} is the connection between fog node i and j in $\mathbf{E}$. We assume that the distribution of fog nodes is concentrated, and there is no communication delay between fog nodes. So the delays between users and fog nodes are the same, which depend on the offloading delay between users and access nodes. Fog node v_j can only connect with c_j users who are in its efficient area. The efficient area is the region that the fog node can successfully connect with users. Since fog nodes are heterogeneous and come in different form factors [3], we let r_j be the maximum processing rate of fog node v_j. We use set-up cost to measure the efficiency of the resource provision scheme, and each fog node has a set-up cost τ_j. We assume

that the locations and connections of fog nodes in graph $\mathbf{G}$ are already fixed by the cloud data centers or third party service providers. Two types of fog nodes in $\mathbf{V}$ are defined. One type is **access nodes**, fog nodes in $\mathbf{V}$ that connect with access points, which offload workloads from the user layer. Each user can only upload its workload to a near access node in fog model. Another type is **cooperative nodes**, fog nodes in $\mathbf{V}$ that either process workload by themselves or from their multiple neighboring fog nodes which can be reached through local communication infrastructures.

We discuss two cases, Unlimited-Processor Fog Nodes (UPFN) case and Limited-Processor Fog Nodes (LPFN), which have different computation delays due to their differences in the capacities of processors. In the UPFN case, each fog node can parallelly process multiple jobs, and the computation delay for a workload with weight w_i on fog node v_j is p_{ij}, i.e., $p_{ij} = \frac{\lambda_{ij} \cdot w_i}{r_j}$. Let λ_{ij} denote the proportion of user i's workload on fog node j, where $0 \leq \lambda_{ij} \leq 1$. In the LPFN case, the computation delay is defined as $p_{ij} = \alpha \cdot x_v + b$, which indicates the delay has a linear relationship with the number of fog nodes x_v. Let α denote a unit rising rate, and b is a constant delay of each fog node [8].

B. Users and Workload

We use a set $\mathbf{U} = \{u_i\}$ to denote the delay-sensitive users which are distributed on the user layer in the system. Let u_i denote the i^{th} user, and w_i denote the weight for the workload of user u_i. The workload of each user is continuous and fractional. We assume that the workload is constructed by identical service entities, and each service entity is represented by one unit. Each user can only upload its workload to the nearest access fog node, and set $\mathbf{U}$ has its own deadline T. The total completion time of users in set $\mathbf{U}$ should not go beyond its deadline T. Although each user can only upload its workload to the nearest access fog node, in our model we ignore the transmission cost between every access fog node and corresponding connected cooperative fog node. Thus we assume that users are directly connected with access and cooperative fog nodes. The connection between users and fog nodes can be represented by the connectivity graph as shown in Fig 2. In Fig 2, u_i denotes the i^{th} user, and v_j denotes j^{th} fog nod. e_{ij} denotes the connection between u_i and v_j, which means that the workload of user i can be processed on fog node j. Based on the connections between users and fog nodes, the workload of user i is only allowed to be processed on specified subset of fog nodes $\mathbf{G}_i$, where $\mathbf{G}_i = \{v_k \in \mathbf{G} | e_{ik} \in L\}$. Fog node j processes specified workloads from subset of users $\mathbf{U}_j$, where $\mathbf{U}_j = \{u_k \in \mathbf{U} | e_{ki} \in L\}$.

C. Problem Formulation

In this subsection, we formulate the RP problem, and we use the set-up cost to measure the efficiency of resource provision in the cooperative fog computing networks. $\mathbf{X}$ is the set of provision schemes for users in set $\mathbf{U}$, which denotes the amount of workloads be processed on each fog node, i.e., $\mathbf{X} = \bigcup_{j \in \mathbf{U}} \mathbf{X}_j$. For each user, the provision scheme $\mathbf{X}_j$ is a subset of $\mathbf{G}_j$, where $\mathbf{X}_j \subseteq \mathbf{G}_j$. Recall the $\mathbf{G}_j$ which is the set of available fog nodes for u_j. Our objective is to find an appropriate scheme for the set of users $\mathbf{U}$ with minimum cost, and support all users' demands in both resource and deadline constraints.

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

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

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

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

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

Eq. 1 shows the objective of minimizing the total provision cost for users. $\mathbb{1}_{[\cdot]}$ is an indicator function, the value of this function is 1 when $\sum_{i \in \mathbf{U}} \lambda_{ij} w_i > 0$, otherwise it is 0. Eq. 2 is the constraint on the deadline of users, which means the computation time of the latest finished user cannot exceed the deadline T. Eq. 3 shows the maximum delay of each user. D_i is constructed by two parts: the offloading delay between user and access fog node d_{ij} and the computation delay on the fog node p_{ij}. Since the delay between a user and its access node is irrelevant to the placement decision [8], we set the delay between users and access node to be the same. Eqs. 4 and 5 are the constraints on the computation resource of the fog node, which means the offloaded workload on each fog node cannot exceed its capacity c_j.

Theorem 1. *The RP problem in cooperative fog computing with minimum cost is NP-hard.*

Proof: We conduct the proof via a polynomial-time reduction from the set covering problem, which is known to be NP-hard [12]. An instance $(X, \mathcal{F})$ of the set-covering problem consists of a finite set X and a family $\mathcal{F}$ of subsets of X, such that every element of X belongs to at least one subset in $\mathcal{F}$: $X = \bigcup_{S \in \mathcal{F}} S$. A set $S \in \mathcal{F}$ covers its elements. A set covering problem is to find a minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all elements of X, i.e., $X = \bigcup_{S \in \mathcal{C}} S$. The reduction from the set covering problem to our RP problem can be built by treating users in set $\mathbf{U}$ as a finite set X, and then we reduce a family of substrate distributions of fog nodes $\mathbf{X} = \bigcup_{j \in \mathbf{U}} \{P_j\}$ as $\mathcal{F}$, which are the subsets of $\mathbf{U}$, i.e., $P_j \subseteq \mathbf{G}_j$. Each fog node is transferred to a subset $\mathbf{G}_j \in \mathbf{U}$ which contains users it can serve. P_j as the set S. Resource provisioning with the minimum cost is to find a minimum-size subset $\mathcal{P} \subseteq \mathbf{X}$, whose fog nodes serve all elements of $\mathbf{U}$, i.e., $\mathbf{U} = \bigcup_{j \in \mathcal{P}} \mathbf{X}_j$. Since the set covering problem is NP-hard, the resource provision with the minimum cost is NP-hard. ■

IV. RP with Unlimited-Processor Fog Node (UPFN)

A. Feasible Provision

In this subsection, we propose a feasibility checking method by finding the maximum flow problem in our weighted connectivity graph. Given a user set $\mathbf{U}$, we first discuss the feasibility of solving this problem, which is whether there exists a provision for users that can support their demands within the constraint of its deadline T. We construct a graph based on the information and connections of users and fog nodes. Based on that, we add two virtual nodes s and t, and

Algorithm 1 Feasibility Checking (FC)

Input: Topology $\mathbf{G}$, set of users $\mathbf{U}$;
Output: Feasibility of $\mathbf{U}$ on $\mathbf{G}$;

1: Construct an auxiliary graph $\mathbf{G}'$ with respect to the connections between users and fog nodes;
2: Obtain the maximum flow Φ on graph $\mathbf{G}'$ using Edmonds-Karp algorithm;
3: **if** $\Phi \geq \sum_{i \in \mathbf{U}} w_i$ **then**
4: Set $\mathbf{U}$ is feasible on $\mathbf{G}$;
5: **return** Provision Scheme $\mathbf{X}$ of set $\mathbf{U}$;
6: **else**
7: Set $\mathbf{U}$ is unfeasible on $\mathbf{G}$;
8: **return** False;

Algorithm 2 Global Influence Greedy (GIG) Algorithm

Input: Topology $\mathbf{G}$, set of users $\mathbf{U}$;
Output: Provision Scheme $\mathbf{X}$ of $\mathbf{U}$;

1: Find a feasible solution using Algorithm 1;
2: Calculate the average computation time of fog nodes $\bar{R}$;
3: **for** fog node $j = 1$ to $j = n$ in $\mathbf{V}$ **do**
4: Calculate ψ_i for each fog node;
5: Rebuild the set $\mathbf{V}$ with fog nodes by an increasing order $j = \arg\min\{\psi_j/\tau_j\}$;
6: **while** $\mathbf{V} \neq \Phi \wedge FC(\mathbf{G}, \mathbf{U}) \neq false$ **do**
7: Check the feasibility of $\mathbf{G}$ using Algorithm 1;
8: Remove v_j from set $\mathbf{V}$;
9: Update the topology $\mathbf{G}$;
10: **return** Provision Scheme $\mathbf{X}$ of $\mathbf{U}$;

the sizes of workloads are represented by the weights of the links between s and user's node u_i. The maximum processing volumes of fog nodes are represented by the weights of the links between t and fog node v_j. We take Fig. 1 as an example, where based on the connections of fog nodes, we can have the relationship between users and fog nodes. We assume that there is no limitation on communications between users and fogs, and the weights between users and fog nodes are ∞. Since the deadline of job set $\mathbf{U}$ is T, the maximum processing volumes of fog nodes will be $v_j \cdot T$, which are the weights of the links between t and fog node v_j. The constructed graph is shown in Fig. 2. As shown in Algorithm 1, we use the topology of fog nodes $\mathbf{G}$, and the set of users $\mathbf{U}$ as our inputs. The feasibility of set $\mathbf{U}$ on fog nodes $\mathbf{G}$ is our output, i.e., $FC(\mathbf{G}, \mathbf{U})$. In line 1, we construct an auxiliary graph $\mathbf{G}'$ with respect to the connections between users and fog nodes. We use the Edmonds-Karp algorithm [12] to obtain the maximum flow Φ in line 2, and the feasible provision is based on the maximum flow that passes through the links between fog nodes and the destination t. The feasibility checking is shown in lines 3 to 7, if the maximum flow can cover the total demand of users $\sum_{i \in \mathbf{U}} w_i$, set $\mathbf{U}$ is feasible under the fog nodes set $\mathbf{G}$, otherwise, the request of set $\mathbf{U}$ will be rejected.

B. Cooperative Influences

1) Global Cooperative Influences: We first propose a definition of the global cooperative influence. Let p_{ij} denote the computation time of user u_i on fog node v_j, and the total computation time of fog node v_j is $p_j = \sum_{i \in \mathbf{U}_j} p_{ij}$. The average completion time of fog nodes in set $\mathbf{G}$ is $\bar{R} = \frac{\sum_{v_j \in \mathbf{G}} p_j}{|\mathbf{G}|}$, where $|\mathbf{G}|$ represents the number of fog nodes in $\mathbf{G}$. Each fog node has a global influence, which describes the increment of the average completion time of all rest fog nodes after removing itself. Formal definition is given as follows.

Definition 1 (Global cooperative influence)**.** *Let ψ_j denotes the global cooperative influence of fog node v_j and $\psi_j = \left|\bar{R}_{\boldsymbol{G}/v_j} - \bar{R}_{\boldsymbol{G}}\right|$, where $\boldsymbol{G}/v_j$ denotes the set of fog nodes after removing v_j in $\boldsymbol{G}$.*

2) Local Cooperative Influences: We further define the local cooperative influence. The global cooperative influence computes the increment of the average computation time among all fog nodes except v_j. However, the removing of v_j does not affect the completion time of some fog nodes. These unaffected fog nodes are ignored when we calculate the global cooperative influence. The fog node whose completion time is increased after removing v_j is denoted by $\mathbf{G}^*/v_j$, where $\mathbf{G}^*$ is the set of fog nodes that are connecting with the same set of users $\mathbf{U}_j$, i.e., $\mathbf{G}^* = \bigcup_{i \in \mathbf{U}_j} \mathbf{G}_i$. Specifically, let $M_{\mathbf{G}^*}$ be the completion time of the latest finished fog node in $\mathbf{G}^*$, and $M_{\mathbf{G}^*} = \max_{j \in \mathbf{G}^*}\{p_j\}$. Each fog node has a local influence, which describes the increment of the maximum completion time of the other fog nodes in $\mathbf{G}^*$. The formal definition is given as follows.

Definition 2 (Local cooperative influence)**.** *Let φ_j denote the local cooperative influence of fog node v_j and $\varphi_j = \left|\bar{M}_{\boldsymbol{G}^*/v_j} - \bar{M}_{\boldsymbol{G}^*}\right|$, where $\boldsymbol{G}^*/v_j$ denotes the set of fog nodes after v_j is removed from $\boldsymbol{G}^*$.*

C. Global Influence Greedy (GIG) Algorithm

In this subsection, we propose a greedy algorithm for the cost efficient resource provision, which removes fog nodes iteratively according to their global cooperation influence. As shown in Algorithm 2, we use the topology of fog nodes $\mathbf{G}$ and the set of users $\mathbf{U}$ as the input. The provision scheme $\mathbf{X}$ is the output. We first find a feasible solution using Algorithm 1 in line 1. Then we calculate the average computation time $\bar{R}$ of fog nodes $\mathbf{V}$ in $\mathbf{G}$ in line 2. In lines 3 to 5, we calculate the global influences for all fog nodes in $\mathbf{V}$ and rebuild the set $\mathbf{V}$ by reordering fog nodes with increasing values of $\psi_j \cdot \tau_j$, i.e., $j = \arg\min\{\psi_j/\tau_j\}$. In lines 6 to 9, we start to remove fog nodes iteratively according to their global cooperation influence until there is no feasible provision that can guarantee users' deadlines. We check the feasibility by recalling Algorithm 1 in line 7. The provision scheme $\mathbf{X}$ of users $\mathbf{U}$ is returned in line 10.

D. Local Influence Greedy (LIG) Algorithm

The main idea of Local Influence Greedy (LIG) Algorithm is the same as GIG, except the subtle change in the iterative removal of fog nodes based on their local cooperation influence. In the algorithm LIG, we calculate the computation time of the latest finished fog node $M_{\mathbf{G}^*}$ in $\mathbf{G}^*$. We calculate the

increment of the local cooperator influence of each node, and start to remove them by an increasing order. The difference between LIG and GIG is on line 5 in Algorithm 2, we build the set $\mathbf{G}$ with fog nodes by an increasing order $j = \arg\min\{\varphi_j/\tau_j\}$. Then, we check the feasibility which is the same as Algorithm 2 in lines 6-9. Finally, we return the provision scheme $\mathbf{X}$ for $\mathbf{U}$.

The time complexities of GIG and LIG are $O((|\mathbf{V}|+|\mathbf{U}|)\cdot|\mathbf{E}|^2)$, where $|\mathbf{V}|$ is the number of fog nodes in topology $\mathbf{G}$, and $|\mathbf{U}|$ is the number of users in set $\mathbf{U}$. The total number of vertices in graph $\mathbf{G}'$ is $|\mathbf{V}|+|\mathbf{U}|+2$. Users and fog nodes are not fully connected, and we use $|\mathbf{E}|$ to denote the number of links in graph $\mathbf{G}'$. In graph $\mathbf{G}'$ there is at most one link between each pair of nodes, we have $|\mathbf{E}| = O(|\mathbf{V}|^2)$. Since the time complexity of finding the maximum flow is $O((|\mathbf{V}|+|\mathbf{U}|)\cdot|\mathbf{V}|^4)$, we have that the complexities of GIG and LIG are $O(|\mathbf{V}|^5\cdot(|\mathbf{V}|+|\mathbf{U}|))$.

V. RP WITH LIMITED-PROCESSOR FOG NODE (LPFN)

In this section, we extend our work into a more complicated and realistic scenario, which considers the processing ability limitation of each fog nodes. We refer to the problem as the resource provision with Limited-Processor Fog Node (LPFN) problem. In LPFN, each fog node has limited processors, and there is scheduling delay on processing several users' workloads. The stretch on execution time is known as performance degradation using scheduling policy to process those requests [8]. We use a general function $d_v(x_v)$ to denote the delay for the performance degradation on each fog node, where x_v is the number of users on the fog node. Let $d_v(x_v) = \alpha\cdot x_v + b$, where α is a unit rising rate, and b is the constant delay of each fog node. We follow the same greedy idea of the simple version as the LPFN case. However the challenge here is to provision the workload among fog nodes in each greedy iteration. Under this condition, we cannot use the maximum flow to check the feasibility as Algorithm 1, since the time consumption of processing one unit workload is not a constant. To check the feasibility, we intend to find the optimal provision and compare the delay with users' deadline. We introduce a new definition.

Definition 3 (Optimal Provision Finding (OPF) Problem)**.** *Given the set of users **U**, the topology of fog nodes **G**, and the delay function $d_v(x_v)$, an Optimal Provision Finding (OPF) Problem is how to find a provision in **G** to minimize the delay of users **U**.*

A. Conversion

In this subsection, we convert the OPF problem into a Continuous Symmetric Network Congestion Game (CSNCG) problem for each iteration. We confirm the strong isomorphism from a OPF to a CSNCG according to its definition in the reference [13]. Basically, the two games have the same user set $\mathbf{U}$. First, these two problems are symmetric by having the same initial and target vertices. Second, in the CSNCG problem, the congestion is on the edge. In a OPF problem, the graph is constructed based on the connections of users and fog nodes, and the congestion of users is also on links

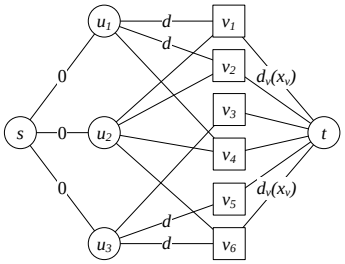


Fig. 3. The converted graph of Fig 1.

(links between fog nodes and the target node). We assume that the delays between users and fog nodes are fixed at d, and we use $d_v(x_v)$ as our delay function on each fog node. Since the OPF problem considers the condition with limited processors, when the number of users assigned to the fog node is increasing, the processing delay on this fog node will be increasing correspondingly. So, $d_v(x_v)$ is a non-decreasing and non-negative delay function. Thirdly, each strategy in CSNCG is a route r from the common source s to the common target t, which is the same as in our OPF problem. We use $\mathbf{X}$ to denote the provision scheme for the set of users. Each provision is a route from source s to the destination t. The total delay of one user is $D = \frac{1}{n}(d + d_v(x_v))$, which is also the sum of delays on all edges equal to the definition in [13]. Our OPF problem can be redefined as $(\mathbf{U}, \mathbf{G}, s, t, d_v(x_v))$.

We convert the OPF problem to a CSNCG problem. In our problem, users share fog nodes after all users can finish processing their workload before the deadline. We construct a graph based on the information and connections of users and fog nodes. From that, we add two virtual node s and t as the virtual source and destination. All users start to assign their workload from s, and when they arrive at t, the provision process is terminated. We take Fig. 3 as an example, based on the connections of fog nodes, we can have the relationship between users and fog nodes. Since there is no congestion between s and users, the weights of links are 0. In the second part, the delays between users and fog nodes are d, the weights of links are d. The weights of links between fog nodes and destination t are the delays on fog nodes, which are related to the number of users and the size of workloads. The constructed graph is shown in Fig. 3. The converted graph helps us develop the congestion game for the OPF problem.

B. Solutions

In this subsection, we outline two greedy algorithms for computing a solution to the LPFN problem according to the global and local cooperative influences. As shown in Algorithm 3, we consider removing fog nodes based on the global cooperative influence, which is known Global Influence Greedy on LPFN (GIG-LPFN). We take the topology of fog nodes $\mathbf{G}$, the set of users $\mathbf{U}$, and the delay function $d_v(x_v)$ as our inputs. The output is the provision scheme $\mathbf{X}$ for $\mathbf{U}$. In line 1, we first construct a new graph $\mathbf{G}''$ based on the connections between $\mathbf{G}$ and $\mathbf{U}$, and we add two virtual nodes s and t as the starting and terminating nodes. All users need to find a path from s to t with the deadline constraint T. Since we have already transferred OPF into a CSNCG problem, we

Algorithm 3 Global Influence Greedy on LPFN (GIG-LPFN)

Input: Topology $\mathbf{G}$, set of users $\mathbf{U}$, delay function $d_v(x_v)$;
Output: Provision Scheme $\mathbf{X}$ of $\mathbf{U}$;

1: Construct $\mathbf{G}''$ based on the connections of $\mathbf{G}$ and $\mathbf{U}$;
2: Replace in $\mathbf{G}''$ each edge with n parallel edges between each node, with weight $d_v(1)$, $d_v(2)$, ... , $d_v(n)$;
3: **for** fog node $j = 1$ to $j = n$ in $\mathbf{V}$ **do**
4: Find a minimum delay provision with min-cost flow of $\mathbf{G}/v_j$;
5: Calculate ψ_j for each fog node;
6: Rebuild the set $\mathbf{G}$ with fog nodes by an increasing order $j = \arg\min\{\psi_j/\tau_j\}$;
7: **while** $\mathbf{G} \neq \Phi \wedge D \leq T$ **do**
8: Remove v_j from set $\mathbf{G}$;
9: **return** Provision Scheme $\mathbf{X}$ of $\mathbf{U}$;

replace in $\mathbf{G}''$ each edge with n parallel edges between each nodes, with weight $d_v(1)$, $d_v(2)$, ... , $d_v(n)$. In lines 3-5, we find a minimum delay provision using the min-cost flow in each iteration, and calculate the cooperator influence ψ_j for each fog node. In line 6, we rebuild the set $\mathbf{G}$ with fog nodes by an increasing order of the global cooperative influence $j = \arg\min\{\psi_j/\tau_j\}$. In lines 7-9, we start to remove the fog node based on the global cooperative influence, and the removing process stops when either the set $\mathbf{G}$ is empty or the delay for the set of users goes beyond the deadline T.

Then, we consider removing the fog node based on the local cooperative influence, which is known as Local Influence Greedy LPFN (LIG-LPFN). Similarly to GIG-LPFN in lines 1 to 4, we use the same input and construct a new graph $\mathbf{G}''$. However, during each iteration, we calculate the local cooperative influence φ_j for each fog node. We rebuild the set $\mathbf{G}$ with fog nodes by an increasing order $j = \arg\min\{\varphi_j/\tau_j\}$. We remove the fog node based on the local cooperative influence, and the removing process is same as GIG-LPFN.

C. Properties

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

Proof: The works in [14, 15] have proved that every potential game has at least one pure NE, namely the strategy S that minimizes $\Phi(S)$. [16] proved that any congestion game is a potential game, and in subsection A of Section V, we convert the OPF problem to a CSNCG problem, so we can have Theorem 3. ■

Since the CSNCG is a continuous problem, and this kind of problem can not be solved in a polynomial time, we do the discretization and transfer it to a discrete congestion game problem. Before that, we prove that our delay function satisfies Lipschitz assumption in Theorem 4.

Theorem 3. *Function $d_v(x_v)$ satisfies Lipschitz assumption.*

Proof: Since we have that $d_v(x_v) = \alpha \cdot x_v + b$, which is a linear function. There exists a constant α such that for edges e and all $0 \leq x < y \leq 1$, $|d_e(y) - d_e(x)| \leq \alpha\,|y - x|$. So, we have that our delay function $d_v(x_v)$ satisfies Lipschitz assumption. ■

Theorem 4. *GIG-LPFN and LIG-LPFN are bounded by $\frac{8}{3}OPT + \frac{\epsilon^2}{8m\alpha}$.*

Proof: We set a minimum unit $\delta = \frac{\epsilon}{4m\alpha}$, where α is the upper bound on the Lipschitz constants of our delay function above, and m is the number of fog nodes in use, and ϵ denotes the ϵ-approximate Nash equilibrium [1]. We introduce $\phi(f)$ as the potential function for the CSNCG problem, where $\phi(f) = \sum_e \phi(f)(x_v)$, $\phi_e(x_v) = \int_0^{x_v}(\alpha \cdot x_v + b)\mathrm{d}t$. Then we have $\phi_e(x_v) = \frac{1}{2}\alpha x_v^2 + bx_v$, 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 $\phi_e(x_v)$ and $\hat{C}(x_v)$(continuous potential function) with $\frac{\phi_e(x_v)}{\hat{C}(x_v)} = \frac{\frac{1}{2}ax_v + b}{ax_v + b}$. There are two extreme cases. One case is that there is no user request to this fog node. Then $\frac{1}{2} \leq \frac{\phi_e(x_v)}{\hat{C}(x_v)} \leq 1$. We have

$$\phi_e(x_v) \leq \hat{C}(x_v) \tag{6}$$

Let $C(f)$ be the total delay function for the discrete condition, where $C(f) = \sum_e C_e(x_v)$. In [13], the authors prove that the discrete delay function $C(f)$ approximates the continuous potential function $\phi(f)$ within an additive error of $\frac{\epsilon^2}{16m\alpha}$. Thus, we have $|\phi(f) - P(f)| \leq \frac{\epsilon^2}{16m\alpha}$, $P(f) - \phi(f) \leq \frac{\epsilon^2}{16m\alpha}$, we can obtain that $P(f) \leq \phi(f) + \frac{\epsilon^2}{16m\alpha}$. According to Eq. 7, we can obtain

$$P(f) \leq \hat{C}(x_v) + \frac{\epsilon^2}{16m\alpha} \tag{7}$$

Then we discuss the relationship between discrete the potential function and the delay function. Since we do the discretization by dividing our problem into δ units, our delay function for one user is $d(x_v) = \int_{m-\frac{1}{2}}^{m+\frac{1}{2}} f(t)\mathrm{d}t = \int_{m-\frac{1}{2}}^{m+\frac{1}{2}}(\alpha t + b)\mathrm{d}t = am + b$. Then, for each fog node, the delay function is $c(x_v) = md(x_v) = am^2 + bm$. The potential function under the discrete case is $p(x_v) = \sum_1^m d(x_v) = a(1+2+...+m) + bm = \frac{m(m+1)}{2}a + bm$. $\frac{p(x_v)}{c(x_v)} = \frac{\frac{m(m+1)}{2}a + bm}{am^2 + bm} = \frac{\frac{1}{2}m + \frac{a}{2} + b}{am + b}$, where $\frac{1}{2} \leq \frac{p(x_v)}{c(x_v)} \leq 1$. Since the total delay function and 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 $\frac{1}{2} \leq \frac{P(x_v)}{C(x_v)} \leq 1$. Then we have $P(x_v) \geq \frac{1}{2}C(x_v)$. The Eq. 7 will be transformed into

$$\frac{1}{2}C(x_v) \leq \hat{C}(x_v) + \frac{\epsilon^2}{16m\alpha} \tag{8}$$

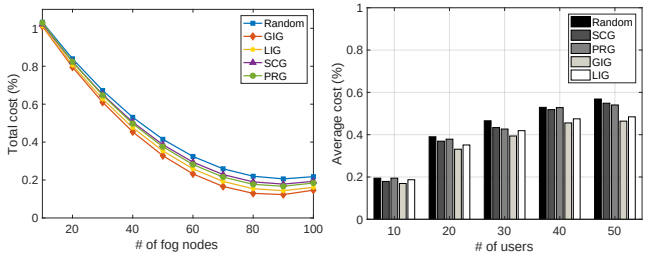
For each linear delay function, [17] proves that there exists an $\frac{4}{3}$-approximation ratio in Nash equilibrium of any CSNCG problem, so we have $\hat{C}(x_v) \leq \frac{4}{3}OPT$. Then we have

$$C(x_v) \leq \frac{8}{3}OPT + \frac{\epsilon^2}{8m\alpha} \tag{9}$$

■

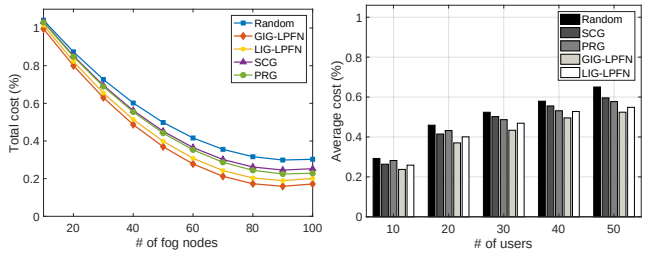
The time complexities of GIG-LPFN and LIG-LPFN are $O(|\mathbf{V}| \cdot (|\mathbf{E}|)^3)$. Since graph $\mathbf{G}''$ is constructed in $O(|\mathbf{V}| + |\mathbf{U}| + |\mathbf{E}|)$, and the number of links $|\mathbf{E}|$ in the graph is much larger than the number of nodes $|\mathbf{V}|$ and $|\mathbf{U}|$, we have that $O(|\mathbf{V}| + |\mathbf{U}| + |\mathbf{E}|) = O(|\mathbf{E}|)$. We find a minimum delay provision by going through each fog node, and the time complexity

[1] ϵ-approximate Nash equilibrium 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 $c_e(x_v) + \epsilon$ [13].



(a) The fluctuation of 30 users. (b) Average cost of users.

Fig. 4. The cost under the UPFN case.



(a) The fluctuation of 30 users. (b) Average cost of users.

Fig. 5. The cost under the LPFN case.

of the min-cost flow algorithm is $O((|\mathbf{V}| + |\mathbf{E}|)^2 \cdot |\mathbf{E}|) = O(|\mathbf{E}|^3)$. Therefore, the complexities of both GIG-LPFN and LIG-LPFN are $O(|\mathbf{V}| \cdot (|\mathbf{E}|)^3)$.

VI. Evaluations

A. Basic Setting-Synthetic Dataset

We use randomly generated topologies and applications for performance evaluation. We measure users' workloads using unit weight, and each unit denotes $1GB$ workload [18]. Let the sizes of workloads follow a uniform randomly distribution between 0 and 50. We calculate the average cost of users within 10 groups, and the scale of fog nodes ranges from 0 to 100. We simulate the topology of fog nodes as an undirected graph, and each fog node has its own processing capability. Let the number of unit weights processed per millisecond denote the processing capability, which is uniformly randomly distributed between 1 and 10 unit weights per millisecond. We compare the proposed algorithms with three baseline approaches which are Random (remove fog nodes iteratively by random order), Set-up Cost Greedy Algorithm (SCG) (greedy remove fog nodes iteratively by an increasing order of the set-up cost), and Processing Rate Greedy Algorithm (PRG) (greedy remove fog nodes iteratively by an increasing order of the maximum processing rate).

B. Experiment Results-UPFN

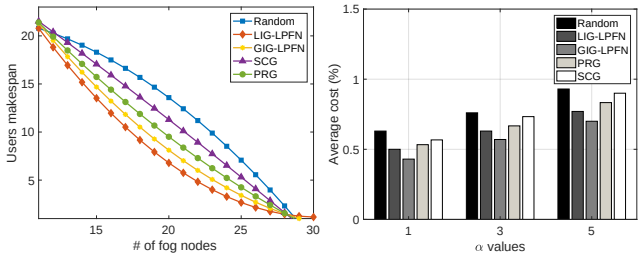
Fig. 4 presents average cost ratio for the users under the UPFN case. We compare our algorithms with the three baseline approaches, and we calculate the average cost ratios for five groups of users that range from 10 to 50. Additionally, we have the following observations: (i). With the increasing number of fog nodes, the impact of algorithms on the total costs is greater. We choose one group, which is a median one (30 users) to analyse the relationship between workloads and fog nodes. As shown in Fig. 4 (a), the costs of users fluctuate with different amount of fog nodes, which means that the cost depends on the locations and connections of fog nodes. Since the connection of fog nodes is changing with the scaling, the total cost of users is fluctuating. In Fig. 4(a), we can find that, GIG and LIG have the faster decreasing speeds, and the total cost is lower than other three baseline algorithms. When the number of fog nodes is small, the performances of these five algorithms are nearly the same. With the increasing number of fog nodes, the total costs of SCG and PRG are fluctuating between Random and GIG. Since all fog nodes are considered in GIG, it leads to a lower cost than LIG when the number of fog nodes gets larger. (ii). For the same distribution of users' workloads, a larger size of the user set will lead to a higher cost. As shown in Fig. 4(b), we can see that the average cost under the 10 users is much lower than that under 50 users. Since the distribution of the weights is random, for small-scale fog nodes, the difference is not obvious, as shown in the first two groups in Fig. 4(b). In summary, compared with the three baseline algorithms, our two algorithms have better performances on cost efficiency.

C. Experiment Results-LPFN

In this subsection, we discuss the average cost ratio for the users in the LPFN problem, and we use five algorithms (Random, SCG, PRG, GIG-LPFN, LIG-LPFN) on each group of datasets and calculate the average cost ratios for users. As shown in Fig 5(b), we calculate the average total cost with values of α from $[1, 5]$. Similar to the previous subsection, we first choose one group of users 30 to analyse the trend. Then we calculate the average cost ratios for five groups of users ($[10, 50]$) with three different delay functions $\alpha = 1$, $\alpha = 3$, and $\alpha = 5$. Additionally, we have the following observations: (i). When the scale of the total amount of workload is larger, the impacts of algorithms on the cost efficiency are bigger. The cost of fog nodes for users depends on the localities and connectivities of the topology. Compared with the three groups in Fig. 5(a), the average cost has the same trend with the increasing number of users, which means more workloads need more resource. (ii). For the same set of users under the same amount of fog nodes, the total costs under LPFN case are larger than the UPFN one. In LPFN case, we consider the processing ability limitation of each fog node, which means that the total amount of workload of users is increased. As the comparison between Fig. 4(b) and Fig. 5(b), the average cost of these five algorithms under LPFN case is larger than the UPFN one with the same group of users. (iii). With a bigger amount of the total workloads of users for fog node, the impact of the algorithms on the average costs is greater. In summary, compared with the three baseline algorithms, GIG-LPFN and LIG-LPFN have lower average costs which can reach 20.25% and 15.15%, respectively.

D. Real Dataset

1) Dataset Setting: We use the published data of New York City (NYC) open data website to construct our real dataset. We combine three datasets, which are the NYC Wi-Fi hotspot locations, entrances of the subway stations, and the transit subway entrance data [19]. We use a set of four-month-long history trip data from $12/23/2017$ to $4/23/2018$, which contains the users' information on subway station entrances



(a) The average makespan of users. (b) Average cost of users.

Fig. 6. The average cost of users.

from 3 : 00am to 23 : 00pm. The records include the names of the station entrances, data, time, and the total number of users. We use one-day data as one unit, and calculate the average number of users over four months.

2) Experiment Results-Real Dataset: We run the five algorithms (Random, SCG, PRG, GIG-LPFN, LIG-LPFN) based on the real dataset. The total amount of fog nodes in our data is 171, which includes 30 access nodes and 141 cooperative fog nodes. Since the start time and finish time are 3 : 00am and 23 : 00pm, we use 23 : 00pm as the deadline of users, and the delay functions are $\alpha = 1$, $\alpha = 3$, and $\alpha = 5$. The experiment results are shown in Fig 6, we have the following observations. As shown in Fig 6(a), we find that the users' makespan increases with the decreasing number of fog nodes, and the GIG-LPFN and LIG-LPFN have the fastest decreasing speed. As shown in Fig 6(b), we calculate the average cost of users under $\alpha = 1$, $\alpha = 3$, and $\alpha = 5$. We find that the average cost of users becomes higher under a larger delay function, and with the scaling of fog nodes, GIG-LPFN can obtain a better performance than LIG-LPFN and other baseline algorithms under the ranges. In summary, GIG-LPFN and LIG-LPFN have better performances on cost efficiency under both synthetic and real datasets. Compared with the Random, SCG, and PRG algorithms, GIG-LPFN and LIG-LPFN have lower average costs under the synthetic dataset, which can reach 14.9% and 10.8% on average, respectively. Additionally, they have lower average costs under the real dataset, which can reach 11.4% and 12.8% on average, respectively.

VII. Conclusion

In this paper, we focus on the RP problem for delay-sensitive users in fog computing. Our objective is to find a feasible provision scheme that minimizes the total cost for network operators, under the capacities constraints by considering the cooperation of fog nodes. We first prove that the RP problem for users with minimum cost is NP-hard. We consider two cases which are the UPFN and LPFN. For the UPFN case, we propose two greedy algorithms which iteratively remove fog nodes according to their global or local cooperative influences until there is no feasible provision that can guarantee users' deadline. We check the feasibility using the Edmonds-Karp max-flow algorithm of the converted graph based on the information and connections of users and fog nodes. We analyze its complexity. Then we extend the UPFN into a more complicated and realistic case LPFN, in which each fog node has scheduling delay on processing several users requests. We check the feasibility of a near-optimal solution with bound $\frac{8}{3}OPT + \frac{\epsilon^2}{8m\alpha}$ found using the continuous congestion game, where m is the number of fog nodes and α is the upper bound on the Lipschitz constants of our delay function. Extensive simulations demonstrate that our schemes outperform the existing method in terms of both efficiency and effectiveness.

ACKNOWLEDGEMENT

This work of the first author was done during her stay as a visitor scholar at Temple University. This research was supported in part by NSF grants CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, IIP 1439672, and CSC 20163100.

References

[1] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Workshop on Mobile Big Data*. ACM, 2015, pp. 37–42.

[2] Cisco. the internet of things: Extend the cloud to where the things are. [Online]. Available: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computingoverview.pdf

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[4] Y. Xiao and M. Krunz, "Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation," in *INFOCOM*. IEEE, 2017, pp. 1–9.

[5] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.

[6] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *APNOMS*. IEEE, 2016, pp. 1–4.

[7] L. Wang, L. Jiao, D. Kliazovich, and P. Bouvry, "Reconciling task assignment and scheduling in mobile edge clouds," in *ICNP*. IEEE, 2016, pp. 1–6.

[8] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *INFOCOM*, 2018.

[9] Z. Sheng, C. Mahapatra, V. Leung, M. Chen, and P. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," *IEEE Trans. on Cloud Computing*, 2015.

[10] K. Bierzynski, A. Escobar, and M. Eberl, "Cloud, fog and edge: Cooperation for the future?" in *FMEC*. IEEE, 2017, pp. 62–67.

[11] W. Chang and J. Wu, "Progressive or conservative: Rationally allocate cooperative work in mobile social networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 26, no. 7, pp. 2020–2035, 2015.

[12] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[13] A. Fabrikant, C. Papadimitriou, and K. Talwar, "The complexity of pure nash equilibria," in *STOC*. ACM, 2004, pp. 604–612.

[14] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge university press, 2007.

[15] J. Von Neumann and O. Morgenstern, *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[16] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," *International Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.

[17] T. Roughgarden and É. Tardos, "How bad is selfish routing?" *Journal of the ACM (JACM)*, vol. 49, no. 2, pp. 236–259, 2002.

[18] C. Hu, W. Bao, and D. Wang, "Iot communication sharing: Scenarios, algorithms and implementation."

[19] MTA Information, Average Weekday Subway Ridership. [Online]. Available: http://web.mta.info/nyct/facts/ridership/ridership_sub.htm