Game Theoretic Resource Planning and Request Scheduling in Mobile Edge Computing Networks

Bin Xiang*, Jocelyne Elias†, Fabio Martignon‡, Elisabetta Di Nitto§ and Dusit Niyato¶
*CNRS@CREATE, Singapore, †University of Bologna, Italy, ‡University of Bergamo, Italy,
§Politecnico di Milano, Italy, ¶Nanyang Technological University, Singapore
bin.xiang@cnrsatcreate.sg, jocelyne.elias@unibo.it, fabio.martignon@unibg.it,
elisabetta.dinitto@polimi.it, dniyato@ntu.edu.sg

***Abstract*—Mobile Edge Computing (MEC) networks offer an increasing computing power through the collaboration among MEC nodes. This opens a large computing market and brings challenges for efficient resource management. In this paper, we study the joint optimization problem of planning cost-efficient edge networks, allocating link and computation resources, as well as scheduling and routing user requests in edge computing networks with arbitrary topologies and multiple ingress nodes. We formulate this problem as a Stackelberg game where the network operator, as the leader, aims at maximizing its profit, and the edge nodes, as the followers, minimize their users' costs and latency. Then, we prove the existence of the generalized Nash equilibrium for the follower subgame, and the Stackelberg equilibrium for the leader-follower game. We further propose a distributed best-response algorithm for the follower game and an alternating leader-follower optimization algorithm for the full game to compute the equilibrium and prove its convergence. A centralized optimization incorporating both profit and network latency targets is formulated and solved, which serves as benchmark for the game solution. Extensive numerical results demonstrate the effectiveness of the proposed game, achieving near-optimal planning and scheduling solutions in a very short time even for large-scale edge networks.**

***Index Terms*—Edge computing, network planning, request scheduling and routing, Stackelberg game.**

I. Introduction

Resource allocation is a well-known problem which is particularly critical in MEC networks where each individual node has limited network, memory and computational resources. Multiple approaches exist in the literature and offer a solution considering different network models and assumptions.

In this paper, we focus on addressing at the same time two different and opposite viewpoints, the one of network providers that must allocate their resources with the objective of saving unneeded costs and the one of end users that want to make sure their requests are fulfilled as soon as possible by the network, even if this implies a non-optimal use of the network providers' resources.

While in the literature some approaches that address the problem have been presented (see Section II), our novel contribution consists in addressing the problem taking also routing and network latency into account in the context of distributed edge computing networks. We experiment the usage of game theory to formulate the problem and demonstrate the existence of the equilibrium for the game and the convergence of the algorithm to such equilibrium. In this approach, we assume that different edge nodes contribute to the resource planning and user requests scheduling activities, thus resulting in a decentralized decision making process. While usually resource planning is performed in advance and is rarely modified, in this paper we assume to have the possibility to change the plan frequently, depending on the type and amount of requests observed over the time.

ISBN 978-3-903176-57-7© 2023 IFIP

To address this problem, we define a Stackelberg game where the network operator, as a market *leader*, plans and prices edge computation and communication resources and performs requests routing to maximize its profit, while edge nodes from different locations, as *followers*, compete for the shared resources and schedule requests, considering all resources in the network, to minimize the costs incurred by their end users to pay for the resource prices as well as link and processing latency. In the game, the leader moves first anticipating the followers' actions, then followers react given the leader's decisions. When the game reaches an equilibrium, if it exists, then no player can benefit by unilaterally deviating from his/her strategy. In other words, it reaches an optimal solution from a social point of view.

To summarize, this paper makes the following contributions:

- We propose a Stackelberg game for the joint optimization of planning and pricing edge network resources, scheduling and routing requests, with the objective of maximizing the profit of the network operator and minimizing costs and latency for edge users.
- We prove the existence of a Generalized Nash Equilibrium (GNE) for the follower subgame and a Stackelberg Equilibrium (SE) for the leader-follower game.
- We further propose a distributed best-response algorithm for GNE and an alternating leader-follower optimization algorithm for SE and prove its convergence.
- We perform an extensive numerical evaluation to demonstrate the performance of the proposed game and solutions for edge networks with arbitrary, large-scale topologies including a real network scenario based on the actual deployment of base stations.

The rest of the paper is organized as follows: Section II presents the related work. Section III provides an overview of the proposed approach. Section IV illustrates the proposed system model. Section V presents the game formulation for the optimization and Section VI analyzes the proposed game and proves the existence of the equilibria. Section VII develops algorithms with proven convergence to find the equilibria. Section VIII discusses numerical results in different network topologies and scenarios. Section IX concludes the paper.

II. Related Work

Resource management in MEC environments has been widely studied in the literature. Many works consider either a single edge computing node [1–3] or a set of independent nearby edge computing nodes [4–10] which might also collaborate with a remote cloud (a *cloud-edge*) [11–17]. Only few studies, however, consider an edge computing environment that includes network connections among the edge nodes in a local area due to the complexity of the problem [18, 19].

Network planning and request scheduling are very important for the edge resources utilization in MEC environments. The most relevant problems include, among the others, service placement or provisioning [6, 15, 16, 20], computation offloading [3, 5, 11, 17], and user or server selection [7, 10]. In the above works, these problems are analysed independently one from the other, while in practical edge computing networks they tend to arise simultaneously. In fact, given the distribution of requests, network planning depends on how or where the requests are scheduled, and vice versa. Among those that address the joint problem [4, 14, 18], Gu et al. [4] study microservice placement and request scheduling to maximize edge throughput. Xiang et al. [18] study the problem of jointly planning network and computation resources, offloading and routing traffic to minimize latency and costs. Both these approaches are studied from a centralized viewpoint, which has limited usage for large scale edge networks and does not capture the distributed nature of the edge and the interactions among network players, e.g., service providers and users, who focus only on their own objectives.

This behavior suggests the adoption of game theory that has been widely used in multiple areas to analyze the interactions among independent, selfish and rational players with their self-interests. The general non-cooperative game is well applied to resource allocation [2, 13]. However, network topology is not considered. As a special case, Stackelberg game distinguishes network entities into leaders and followers, where leaders move first anticipating followers' actions, then followers react given leaders' decisions. Recent studies focus on task offloading [11, 12, 17], resource allocation [8] and service caching [1]. For instance, Huang et al. [12] formulate a resource pricing and task offloading problem as a Stackelberg game among cloud and edge service providers and users. Yan et al. [1] propose a Stackelberg game of service caching and task offloading. Another well applied game is the potential game [7, 9, 19]. In all these cases network planning is not studied and the impact of network latency and topology is not considered.

To our knowledge, this paper is the first that proposes a Stackelberg game for joint optimization of planning edge networks, allocating link and computation resources, and scheduling and routing requests to maximize profit for network operator and minimize costs and latency for users. We model both link and processing latency with queueing theory and consider large-scale MEC networks of any topology.

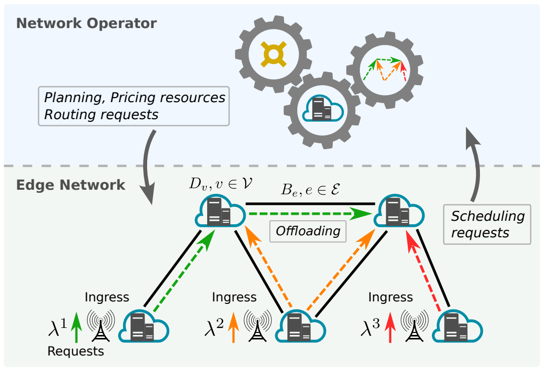


Fig. 1: Edge considered in this paper.

III. Approach Overview

The edge planning architecture proposed in this paper is illustrated in Fig. 1: it is composed of two layers, i.e., the upper one with the *network operator* and the lower one with the *edge network*. The network operator owns network resources related to edge computing and link bandwidth; its goal is to gain profit by planning an edge network and offering network resources to end users for computing their requests. The edge network (denoted by $\mathcal{G}(\mathcal{V}, \mathcal{E})$) is composed of many edge computing nodes ($\mathcal{V}$) with specific computing capacities ($D_v, v \in \mathcal{V}$), connected to each other through network links ($\mathcal{E}$) with the given bandwidth ($B_e, e \in \mathcal{E}$) in a specific topology. In practice, the edge computing nodes can be deployed at the centroids of groups/clusters of base stations for a cost-efficient edge network; a realistic edge network in a metropolitan area is adopted in our numerical evaluation (Section VIII). Naturally, we consider *aggregated requests* (e.g., web, video) at several edge nodes named *ingress* with specific arrival rates ($\lambda^k, k \in \{1, 2, 3\}$ for the example in Fig. 1) measured in *Gbps*, which already takes into account the computing density of different types of requests.

The *users* in the edge network compete for network resources for computing their requests; they can schedule requests with the help of ingress nodes, independently, controlling how to split them and selecting which subset of edge nodes to process each piece of the requests, in order to minimize costs and network latency. Anticipating the decisions of edge users, the network operator plans and prices the network resources at the edge network, and routes the user requests toward the destinations with the objective of maximizing its profit. The objectives of the network operator and edge users associated with the edge nodes are therefore different, and meeting both expectations is not straightforward. In this paper, their interaction is captured and modeled by a non-cooperative game, i.e., a Stackelberg game, where the network operator and edge users are players. Playing this game, they eventually reach an equilibrium, at which they have made the best choice given the other players' decisions.

IV. System Model

In this section we propose a mathematical model for resource planning and request scheduling in mobile edge computing networks. We first model the network planning decisions (on capacity and computation) and request routing, as well as the processing and link latency, and then formulate the *Leader-Optimal* optimization problem. For brevity, we simplify expression $\forall k \in \mathcal{K}$ as $\forall k$, and apply the same rule to other set symbols like $\mathcal{V}, \mathcal{E}$, etc., unless otherwise specified.

A. Network Planning - Capacity and Computation

We assume that, in each edge node, some processing capacity can be made available for computing requests, which will result in an operation cost associated with the amount of processing capacity. To model more closely real network scenarios, we assume that only a discrete set of capacity values are available to the network operator. This can be mapped to the different numbers of virtual machines or levels of CPU computation power for each edge node. Therefore, we adopt a piecewise-constant function D_v for the processing capacity of an edge node, in line with [16]. This is defined as:

$$D_v = \sum\nolimits_{a\in\mathcal{A}} \delta_v^a L_a,\ \forall v, \tag{1}$$

where L_a is a capacity level ($a \in \mathcal{A}$) and $\delta_v^a \in \{0,1\}$ is a binary decision variable for capacity planning, satisfying the following constraint (exactly one level of capacity is available at a node, including zero, i.e., no processing capability):

$$\sum\nolimits_{a\in\mathcal{A}} \delta_v^a = 1 - \delta_v^0,\ \forall v, \tag{2}$$

where δ_v^0 is a binary variable that indicates whether node v has currently any available computation capacity or not.

To save on operation costs, in case an edge node is not supposed to be exploited to process some requests, then no processing capacity is made available on it. We introduce binary variable b^{kv} to indicate whether any piece of request k is processed on node v. The following constraint should then be satisfied:

$$b^{kv} \leqslant 1 - \delta_v^0 \leqslant \sum\nolimits_{k'\in\mathcal{K}} b^{k'v},\ \forall k, \forall v. \tag{3}$$

We also consider a total planning budget, P, for the available computation capacity, with the following constraint:

$$\sum\nolimits_{v\in\mathcal{V}} D_v \leqslant P. \tag{4}$$

B. Request Routing

We assume that each user request can be split into multiple *pieces* only at its ingress node. Each piece can then be offloaded to another edge computing node independently of the other pieces, but it cannot be further split (we say that each piece is *unsplittable*).

In general, we consider that the user request or the virtual operator request passes through a predefined set of nodes along a given (unique) path, like a given chain of nodes providing services to the user/virtual provider.

Each link $e \in \mathcal{E}$ may carry different request pieces q^{kv}, i.e., the percentage of request k processed at node v. Then, the request flow k on e, f_e^k, can be expressed as the sum of all pieces of requests that pass through such link:

$$f_e^k = \sum_{v\in\mathcal{V}:\ e\in\mathcal{R}^{kv}} q^{kv},\ \forall k, \forall e, \tag{5}$$

where $\mathcal{R}^{kv} \subset \mathcal{E}$ denotes a routing path (set of traversed links) for the request piece $q^{kv}\lambda^k$ from ingress s^k to node v. Variable q^{kv} has to fulfill the request integrality constraint:

$$\sum\nolimits_{v\in\mathcal{V}} q^{kv} = 1,\ \forall k. \tag{6}$$

The following constraint ensures that the total requests on each link e do not exceed its capacity B_e:

$$\sum\nolimits_{k\in\mathcal{K}} f_e^k \lambda^k \leqslant B_e,\ \forall e. \tag{7}$$

The request flow conservation constraint is written as:

$$\sum_{e\in\Phi_v^-} f_e^k - \sum_{e\in\Phi_v^+} f_e^k = \begin{cases} q^{kv} - 1, & \text{if } v = s^k, \\ q^{kv}, & \text{otherwise,} \end{cases} \quad \forall k, \forall v, \tag{8}$$

where Φ_v^- and Φ_v^+ are set of incoming and outgoing links of node v, respectively. The fulfillment of this constraint guarantees *continuity* of the routing path. Moreover, the routing path $\mathcal{R}^{kv}$ should be *acyclic*.

C. Processing and Link Latency

The latency is composed of *processing latency* on some edge computing nodes and *link latency* between the nodes.

Processing Latency: We assume that each user request can be segmented and processed on different nodes, and different user requests from different ingress nodes can share the computation capacity of the same nodes. The processing of user requests on each node is described by an $M|M|1$ model. As introduced before, we indicate with q^{kv} the percentage of request k processed on node v. Let T_v denote the *processing latency* for the user requests on node v. Then, based on the computation capacity D_v with the requests $\sum_{k\in\mathcal{K}} q^{kv}\lambda^k$ to be served, $\forall v \in \mathcal{V}$, T_v is expressed as:

$$T_v = \begin{cases} \frac{1}{D_v - \sum\limits_{k'\in\mathcal{K}} q^{k'v}\lambda^{k'}}, & \text{if } \exists k : q^{kv} > 0 \,\&\, D_v > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Note that the capacity D_v is planned by the network operator, and could also be zero: when user requests are not processed on node v, v has no computation capacity, the corresponding value is in fact 0. The capacity constraint is written as follows:

$$\sum\nolimits_{k\in\mathcal{K}} q^{kv}\lambda^k \leqslant D_v,\ \forall v. \tag{10}$$

Link Latency: Let T_e denote the *link latency* for routing different user request flows through link e. Recall that $f_e^k\lambda^k$ is the flow of request k on link e. The transmission of user requests on each link is also described by an $M|M|1$ model. Specifically, $\forall e \in \mathcal{E}$, T_e is defined as:

$$T_e = \begin{cases} \frac{1}{B_e - \sum\limits_{k'\in\mathcal{K}} f_e^{k'}\lambda^{k'}}, & \text{if } \exists k : f_e^k > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Note that the *link latency* is accounted for only if a non-null request flow is routed on e, i.e., $f_e^k > 0$.

We underline that the way we model the latency or delay with the queuing system is aligned with other approaches in the literature [15, 21].

D. Optimization problem

In edge computing networks, the network operator plans the edge computing nodes and network links to serve user requests. It charges the users for these services to earn revenues and cover the operation costs for opening and maintaining edge nodes and links. Then, the profit of the network operator can be expressed as a function of the variables related to the *pricing* and the *costs* of network planning and resource allocation, and it is defined as follows:

$$\begin{aligned} U_{\mathsf{OP}}(\boldsymbol{p},\boldsymbol{\delta},\boldsymbol{\gamma},\boldsymbol{q}) &= \sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}} p^{kv}q^{kv}\lambda^k \\ &- \sum_{v\in\mathcal{V}}\sum_{a\in\mathcal{A}}\theta_v\delta_v^a L_a - \sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}}\phi_e\gamma_e^{kv}q^{kv}\lambda^k, \quad (12)\end{aligned}$$

where $\boldsymbol{p} = (p^{kv})^{\mathsf{T}}_{k\in\mathcal{K},v\in\mathcal{V}}$ is the vector of the prices set by the operator for the computation resources, $\boldsymbol{\delta} = (\delta_v^a)^{\mathsf{T}}_{a\in\mathcal{A},v\in\mathcal{V}}$ is the vector of the *network planning* decisions, γ_e^{kv} is the routing decision variable representing whether the piece of request q^{kv} is scheduled on link e, (i.e., $\gamma_e^{kv} = 1$ if $e \in \mathcal{R}^{kv}$, and 0 otherwise), while $\boldsymbol{\gamma} = (\gamma_e^{kv})^{\mathsf{T}}_{k\in\mathcal{K},v\in\mathcal{V},e\in\mathcal{E}}$ is the vector of the request *routing* decisions, $\boldsymbol{q} = (q^{kv})^{\mathsf{T}}_{k\in\mathcal{K},v\in\mathcal{V}}$ is the vector of the request *offloading* decisions, θ_v and ϕ_e are the computation and link cost factors for node v and link e, respectively.

The network operator also needs to ensure the reliability and quality of the network services for users, for which the total processing plus link latency is considered as the main metric in this model. Therefore, the centralized optimization problem can be formulated as follows:

$$\begin{aligned} \max_{\boldsymbol{p},\boldsymbol{\delta},\boldsymbol{\gamma},\boldsymbol{q}} \ & U_{\mathsf{OP}}(\boldsymbol{p},\boldsymbol{\delta},\boldsymbol{\gamma},\boldsymbol{q}) - \xi_p\sum_{v\in\mathcal{V}}T_v - \xi_l\sum_{e\in\mathcal{E}}T_e, \quad (\mathcal{LO}) \\ \text{s.t.} \ & (1)-(12), \\ & \boldsymbol{p}^{\mathsf{L}} \preceq \boldsymbol{p} \preceq \boldsymbol{p}^{\mathsf{U}}, \quad (13)\end{aligned}$$

where $\boldsymbol{p}^{\mathsf{L}}$ and $\boldsymbol{p}^{\mathsf{U}}$ are (positive) lower and upper bounds for the price, respectively; ξ_p and ξ_l are the non-negative weights for the processing and link latency, respectively. The above problem contains integer and continuous variables as well as quadratic constraints, therefore it is a mixed-integer quadratically constrained programming (MIQCP) problem.

Problem $\mathcal{LO}$ is a *Leader-Optimal* optimization, since the network operator is the market leader who targets the profit by offering the network resources to the end users while promising a certain level of quality of service. The solution of $\mathcal{LO}$ may be not beneficial to all end users. Moreover, the end users also have no control over how their own requests will be split and where the different segments will be processed.

V. Stackelberg Game Formulation

The entities in the system include the network operator and the end users associated with the edge nodes, where the operator wants to maximize its profit by providing network and computation resources, while the users want their requests to be served with the lowest possible latency and minimum costs. In practice, the operator always plans the network resources at first, anticipating the demands of the users; then, the users post their requests and compete for network resources in a distributed manner, without cooperation. This interaction forms a Stackelberg game, where the operator acts as leader and the users with the edge nodes as followers.

A. Leader Problem - Pricing, Planning and Routing

Based on the above analysis, anticipating the users' demands of requests ($\boldsymbol{q}$), the network operator performs the best action by solving the following optimization problem, leading the users towards a direction (solution) beneficial to all:

$$\begin{aligned} \max_{\boldsymbol{p},\boldsymbol{\delta},\boldsymbol{\gamma}} \ & U_{\mathsf{OP}}(\boldsymbol{p},\boldsymbol{\delta},\boldsymbol{\gamma}\,|\,\boldsymbol{q}), \quad (\mathcal{PL}) \\ \text{s.t.} \ & (1)-(5),(7),(8),(10),(12),(13).\end{aligned}$$

The leader problem ($\mathcal{PL}$) contains both integer and continuous variables, resulting in a mixed-integer linear programming (MILP) problem.

After the leader solves problem ($\mathcal{PL}$), it will communicate to the followers the set of planned resources, including the computing nodes with the specific capacities ($\boldsymbol{\delta}$) and the routing paths to the nodes ($\boldsymbol{\gamma}$) that can be used for processing the requests and the corresponding prices ($\boldsymbol{p}$).

B. Follower Game - Computation Offloading

After the pricing, resource provisioning/planning and request routing performed by the network operator (leader), users (followers) compete for the edge network resources to process their requests. Each follower decides the assignment of its request pieces to the target edge computing nodes, aiming to minimize the total costs in terms of payments as well as processing and link latency.

Given the leader decisions $\boldsymbol{p},\boldsymbol{\delta}$ and $\boldsymbol{\gamma}$, the following sets are defined for clarity: $\mathcal{V}' = \{v \in \mathcal{V} : D_v > 0\}$, $\mathcal{V}^k = \{v \in \mathcal{V} : b^{kv} > 0\}$, $\mathcal{E}' = \bigcup_{k\in\mathcal{K},v\in\mathcal{V}'}\mathcal{R}^{kv}$ and $\mathcal{E}^k = \bigcup_{v\in\mathcal{V}'}\mathcal{R}^{kv}$. Then, the disutility (or total costs) of each follower k can be defined as:

$$U^k(\boldsymbol{q}^k\,|\,\boldsymbol{q}^{-k},\boldsymbol{p},\boldsymbol{\delta},\boldsymbol{\gamma}) = \sum_{v\in\mathcal{V}'} p^{kv}q^{kv}\lambda^k + w_pT_P^k + w_lT_L^k,\ \forall k, \quad (14)$$

where $\boldsymbol{q}^{-k}$ denotes the actions of the other followers (excluding k). w_p and w_l are the weights for the processing and link latency costs, respectively. T_P^k and T_L^k are the total processing and link latency for each follower k, fusing the leader decisions information. Based on their definitions (9) and (11), they can be expressed as:

$$T_P^k = \sum_{v\in\mathcal{V}^k}\frac{1}{D_v - \sum_{k'\in\mathcal{K}:\mathcal{V}^{k'}\ni v} q^{k'v}\lambda^{k'}},\ \forall k, \quad (15)$$

$$T_L^k = \sum_{e\in\mathcal{E}^k}\frac{1}{B_e - \sum_{k'\in\mathcal{K}} f_e^{k'}\lambda^{k'}},\ \forall k. \quad (16)$$

Then, in the followers' game, $\forall k$, the optimization problem can be formulated as:

$$\begin{aligned} \min_{\boldsymbol{q}^k} \ & U^k(\boldsymbol{q}^k\,|\,\boldsymbol{q}^{-k},\boldsymbol{p},\boldsymbol{\delta},\boldsymbol{\gamma}), \quad (\mathcal{PF}) \\ \text{s.t.} \ & (5)-(7),(10),(14)-(16), \\ & \boldsymbol{0} \leqslant \boldsymbol{q}^k \leqslant \boldsymbol{1}. \quad (17)\end{aligned}$$

Each optimization in the game $(\mathcal{PF})$ is a nonlinear programming problem due to the processing and link latency expressions (15) and (16). Furthermore, both the utility functions and the decisions of the followers are coupled due to the latency and the corresponding capacity constraints (7) and (10).

VI. Game Analysis

A. The Follower Game

The follower game $(\mathcal{PF})$ is a Generalized Nash Equilibrium Problem (GNEP), since each player's payoff function depends on both his/her own variables and that of other players due to the latency, and the strategy of each player also depends on the rival players' strategies due to the coupled constraints (7) and (10).

Lemma 1. *The follower game $(\mathcal{PF})$ is a potential game.*

Proof. Refer to report [22]. ∎

Remark. *The potential function of the follower game can be defined as:*

$$P(\boldsymbol{q}) = \sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}'} p^{kv} q^{kv}\lambda^k + w_p \sum_{v\in\mathcal{V}'} \frac{1}{D_v - \sum_{k\in\mathcal{K}:\mathcal{V}^k\ni v} q^{kv}\lambda^k} + w_l \sum_{e\in\mathcal{E}'} \frac{1}{B_e - \sum_{k\in\mathcal{K}} f_e^k \lambda^k}, \tag{18}$$

which satisfies $\frac{\partial P(\boldsymbol{q})}{\partial \boldsymbol{q}^k} = \frac{\partial U^k(\boldsymbol{q})}{\partial \boldsymbol{q}^k}$, $\forall k$.

Corollary 1.1. *The follower game possesses a pure-strategy Equilibrium.*

Lemma 2. *The follower game $(\mathcal{PF})$ is a jointly convex GNEP.*

Proof. Refer to report [22]. ∎

Based on [23, Theorem 3.9], the jointly convex GNEP can be reduced to a Variational Inequality (VI) problem $\mathbf{VI}(\mathbf{Q}, \mathbf{F}(\boldsymbol{q}))$[1], where $\boldsymbol{q} \in \mathbf{Q}$ and $\mathbf{F}(\boldsymbol{q}) = (\nabla_{\boldsymbol{q}^k} U^k(\boldsymbol{q}))_{k\in\mathcal{K}}$. Specifically, every solution of the $\mathbf{VI}(\mathbf{Q}, \mathbf{F}(\boldsymbol{q}))$ is also a solution of the jointly convex GNEP, but the opposite is not necessarily true. Such a solution is called a *Variational Equilibrium* (VE), and the variational equilibria are more "socially stable" than other equilibria of the GNEP [23].

Lemma 3. *The follower game admits a social optimal Variational Equilibrium.*

Proof Sketch. Based on the derivation in Lemma 1, the Jacobian matrix of the mapping $\mathbf{F}(\boldsymbol{q})$ can be expressed as:

$$\mathbf{JF} = \left[\mathbf{T}^{k\bar{k}}\right]_{k\bar{k}\in\mathcal{K}^2} = \left[\left[\frac{\partial^2 U^k}{\partial q^{kv}\partial q^{\bar{k}\bar{v}}}\right]_{v\bar{v}\in\mathcal{V}'^2}\right]_{k\bar{k}\in\mathcal{K}^2}. \tag{19}$$

Then, $\mathbf{JF}$ is positive-definite, implying that $\mathbf{F}$ is strictly monotone[2]. Thus, the follower game has a global unique VE solution [23]. The next is to prove that the VE solution is also the global minimizer of the potential function $P(\boldsymbol{q})$ (see Eq. (18)). The full proof is detailed in report [22]. ∎

[1]The variational inequality problem $\mathbf{VI}(\mathbf{Q}, \mathbf{F}(\boldsymbol{q}))$ consists of finding a vector $\boldsymbol{q}^\star \in \mathbf{Q}$ such that $(\boldsymbol{q} - \boldsymbol{q}^\star)^\mathsf{T}\mathbf{F}(\boldsymbol{q}^\star) \geqslant 0$, $\forall \boldsymbol{q} \in \mathbf{Q}$.

[2]A mapping $\mathbf{F} : \mathbf{X} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is strictly monotone with respect to $\mathbf{X}$ if $(\mathbf{F}(\boldsymbol{x}) - \mathbf{F}(\boldsymbol{y}))^\mathsf{T}(\boldsymbol{x} - \boldsymbol{y}) > 0, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{X}$ and $\boldsymbol{x} \neq \boldsymbol{y}$.

B. Stackelberg Game

This Stackelberg game is a bilevel optimization problem in which the constraints are partially defined by another *parametric optimization* problem. In general, the bilevel optimization may carry some ambiguities coming from the possible non-uniqueness of the solution when solving the lower-level problem. Hence, to tackle these ambiguities we can consider the *optimistic* or the *pessimistic* formulation of the original problem [24]. For this game, due to the unique followers' equilibrium (see Lemma 3), the optimistic and pessimistic bilevel optimizations coincide with the following formulation:

$$\begin{aligned} \max_{\boldsymbol{p},\boldsymbol{\delta},\boldsymbol{\gamma},\boldsymbol{q}} \quad & U_{\mathsf{OP}}(\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma}, \boldsymbol{q}), \\ \text{s.t.} \quad & (\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma}) \in \mathbf{P}, \\ & \boldsymbol{q} \in \mathit{GNEP}(\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma}), \end{aligned} \tag{$\mathcal{BP}$}$$

where $\mathbf{P}$ represents the domain of leader decisions $\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma}$ and $\mathit{GNEP}(\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma})$ represents the generalized Nash equilibria of the non cooperative game among the followers defined above. Note that even a linear bilevel programming problem was proved to be strongly $\mathcal{NP}$-hard [25].

Lemma 4. *The full game admits at least one global Stackelberg Equilibrium.*

Proof. Refer to report [22]. ∎

A typical approach to solve the problem $\mathcal{BP}$ is to reformulate it as a Mathematical Program with Equilibrium Constraints (MPEC) problem, in which the lower-level (follower) GNEP problem is replaced by the VI constraints, i.e., $\boldsymbol{q} \in \boldsymbol{\mathcal{S}}(\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma})$, where $\boldsymbol{\mathcal{S}}(\cdot)$ denotes the solution for $\mathbf{VI}(\mathbf{Q}, \mathbf{F}(\boldsymbol{q} \,|\, \boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma}))$. However, since the upper-level (leader) problem is a MILP (see Section V-A), the problem becomes a Mixed-Integer MPEC (MI-MPEC) involving both continuous and discrete variables, which is extremely hard to solve, both theoretically and computationally.

VII. Heuristics

Since solving the original optimization problem is computationally cumbersome in realistic network scenarios, in this section, we propose an approximate solution approach for the bilevel optimization problem, and then we design an iterative algorithm to implement it.

A. Followers' Best Response Dynamics

From Lemmas 1 and 3, the follower game is a potential game having a social optimal equilibrium, which then can be solved with *best response dynamics*. Given the leader's decisions $\boldsymbol{p}, \boldsymbol{\delta}$ and $\boldsymbol{\gamma}$, the followers perform their best responses *in parallel* in an iterative way from a feasible starting point $\boldsymbol{q}_0$. Specifically, each follower $k \in \mathcal{K}$ separately optimizes its objective, given the reactions in the previous iteration of the other followers $\boldsymbol{q}_i^{-k}$. The iteration process is written as $\boldsymbol{q}_{i+1}^k = \operatorname{argmin}_{\boldsymbol{q}^k\in\mathbf{Q}^k} U^k(\boldsymbol{q}^k \,|\, \boldsymbol{q}_i^{-k}, \boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma})$, $\forall k$. The iteration repeats until the Generalized Nash Equilibrium is reached by checking if the followers' objective function values or

Algorithm 1 *Initial Request Scheduling Estimation*

```
1: For each request $k \in \mathcal{K}$, estimate the minimum number of computing nodes needed to process it ($n^k = \lceil \frac{\lambda^k}{\max\{L_a\}} \rceil$) and select the best $n^k$ candidate nodes from the neighbors of $k$'s ingress node $s^k$ ($b^{kv}$);
2: Assign all requests fairly ($b^{kv}$, $q^{kv}$) to their corresponding candidate nodes based on $\lambda^k$ and $\max\{L_a\}$ values, considering the sharing status of nodes;
3: If any candidate node's capacity constraints are violated, find more neighbor nodes to completely process all requests and then go to step 2.
```

the decisions have converged to a fixed value. Note that the starting point $\boldsymbol{q}_0$ will be estimated by Algorithm 1.

B. Computation of the Solution of the Stackelberg Game

In the followers' game all requests are enforced to be served, hence the leader can first set the price $\boldsymbol{p}$ in a dominant strategy to the upper bound $\boldsymbol{p}^{\mathsf{U}}$. Then, the leader makes decisions on the planning of computation resources ($\boldsymbol{\delta}$) and the request routing ($\boldsymbol{\gamma}$), anticipating the followers' best reactions. However, the routing variables $\boldsymbol{\gamma}$ and request allocation variables $\boldsymbol{q}$ are "intertwined": to find the optimal routing, the fraction of each request processed at each node v should be known, and at the same time, to solve the optimal resource allocation for a request, the routing path should be known.

To tackle the above reciprocal dependence, the leader could first consider the resource allocation/scheduling subproblem estimating *where to process the requests*, and that is indicated by the binary variable b^{kv}. This can be useful to preliminarily limit the range of the destination nodes and estimate the initial placement of the requests. To this end, we propose a local neighborhood search heuristic considering the rates of requests in different ingress nodes and the maximum capacity of computing nodes. The detailed procedure is illustrated in Algorithm 1, which estimates the initial request scheduling solution in a greedy manner. It searches the neighbor computing nodes to accommodate the requests considering fair sharing of computing capacity by different requests on any computing node. It finally returns the initial solution of $\boldsymbol{q}$.

Given the initial solution of $\boldsymbol{q}$, the leader can optimize its own strategy to get the initial solutions for the computing resources planning $\boldsymbol{\delta}$ and the request routing $\boldsymbol{\gamma}$. Based on the above solutions, the followers play in a non-cooperative game taking their best responses. The follower game will converge to a social optimal equilibrium, refining the requests placement. Furthermore, based on the reactions of the followers, the leader optimizes its own strategy making the followers replay a new game. This results in an iterative process/procedure of reactions of both the leader and the followers until their strategies converge to a Stackelberg Nash Equilibrium (SNE) (see Algorithm 2). Note that the estimated initial placement of the requests (see Algorithm 1) may be redundant to ensure feasibility, which is eventually refined during the iterations.

Lemma 5. *With normal (i.e., sequential/non-distributed) best response dynamics of followers, Algorithm 2 converges to a local Stackelberg Equilibrium.*

Algorithm 2 *Computation of the SG's Solution*

```
1: Initialize $\boldsymbol{q}_0 \in \mathbf{Q}$ (Algorithm 1), $j = 0$;
2: while SNE is not reached do
3:     Leader: $(\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma})_{j+1} = \underset{(\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma}) \in \mathbf{P}}{\operatorname{argmin}}\ U_{\mathsf{OP}}(\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma} \,|\, \boldsymbol{q}_j)$;
4:     Followers: $\boldsymbol{q}_{j+1} = BestResponse(\boldsymbol{q}_j, (\boldsymbol{p}, \boldsymbol{\delta}, \boldsymbol{\gamma})_{j+1})$;
```

Proof. Refer to report [22]. ■

C. Complexity Analysis

Here we provide a coarse-grained analysis of the complexity of the algorithms. Algorithm 1 aims at producing an initial estimation of request scheduling. In the worst case, assuming all requests $\mathcal{K}$ fairly share the computing capacities of the same nodes, the pre-scheduling will be recomputed each time a new overbooking happens at any computing node until all requests are well fitted. Then, the complexity is $O(\frac{|\mathcal{K}| \sum \lambda^k}{\max\{L_a\}})$. Algorithm 2 aims at computing the equilibrium. Following the proofs of Lemma 5, the algorithm converges along with the decreasing of D_v value for all nodes until a certain accuracy ϵ is reached, then the worst-case complexity is $O(\frac{1}{\epsilon}(O(\mathrm{OPT}) + O(\mathrm{BR})))$, assuming $O(\mathrm{OPT})$ and $O(\mathrm{BR})$ are the complexity of the leader optimization and the followers best response, respectively.

VIII. Numerical results

In this section we evaluate the performance of the proposed game model and leader-optimal approach in terms of the profit of the operator (leader), costs of the users (followers) and the computing time for the solution. We first illustrate the network topologies considered in the numerical evaluation campaign and the experimental setup, then discuss the results obtained in different network scenarios.

Network Topologies: Multiple network topologies are considered in our evaluation, consisting of several random graphs as well as a topology built on a real network scenario. Table I shows the structural information for all network topologies.

1) Random graphs: We first consider Erdös-Rényi random graphs [26], generating several topologies with different numbers of nodes and edges, starting from simple to large and dense networks. They can be considered as representatives of various edge networks where edge nodes are distributed and connected with each other in different ways over the territory.

2) A real network scenario: We further consider a topology based on a real network scenario with the actual deployment of Base Stations (BSs), where the information of BSs is collected from an open database, OpenCellID [27]. This topology was first introduced in [18], but we use it in a different context, solving the game theoretic resource allocation problem.

TABLE I: Structural information of the considered topologies

Topology	#Nodes	#Edges	#Ingress	Degree (Min, Max, Avg)	Diameter
5N5E	5	5	3	(1.0, 3.0, 2.0)	3
10N20E	10	20	4	(3.0, 5.0, 4.0)	3
CittàStudi	30	35	6	(1.0, 6.0, 2.3)	10
50N50E	50	50	10	(1.0, 4.0, 2.0)	15
100N150E	100	150	20	(1.0, 7.0, 3.0)	9

Experimental Setup: Our model and approaches are implemented based on an open-source solver SCIP (Solving Constraint Integer Programs) [28]. The parameters of SCIP are set to the default in our experiments. All numerical results presented in this section are obtained on a server equipped with an Intel(R) Xeon(R) E5-2640 v4 CPU @ 2.40GHz and 126 Gbytes of RAM. The results illustrated in the following figures are obtained by averaging up to 30 instances, with 97% confidence intervals.

For each network, we select the source (ingress) nodes uniformly at random. The number of ingress nodes varies from 3 to 20 according to the size of the network topologies.

In Table II, we summarize the reference values defined for the main parameters, which represent a scenario with a high load of requests compared to the computation capacity of nodes. The request rates result from the aggregation of requests generated by the users at a given ingress node. In case of smart metropolitan areas, the network capabilities and requirements of MEC hosts are investigated in a survey [29], based on which, we set the parameters for our demands. We generate random request rates on the ingress nodes of the network topologies according to a Gaussian distribution $N(\lambda^k, \sigma^2)$, where λ^k is uniformly selected varying from 30 to 55 Gbps and $\sigma = 0.5$. The link bandwidth is set in the 50 to 100 Gbps range. We set three possible levels for computation capacity: 30, 40 and 50 Gbps, as it happens in typical cloud IaaS, where users see a predefined computation service offer. The computation budget is set in the range 300 to 4000 Gbps according to the size of network and the corresponding ingress nodes. The cost factors of the bandwidth and computing resources are set to 0.25 and 0.5 respectively. The latency weights for both user and operator are set to different values for different priorities.

In our experiments, the request rates at ingress nodes are scaled up to 3 times to cover extreme cases, for which almost all the requests cannot be served with only the computing resources of their respective ingress nodes, and therefore must be offloaded to other edge computing nodes. Finally, note that our proposed model and heuristics are general, and can be applied to optimize resource allocation in all network scenarios with any parameter settings.

Discussion of Results: In the following, we first compare the results obtained from both the Stackelberg game (SG) and leader-optimal (LO) approaches for the relatively small network scenarios including the *5N5E* network with 3 requests and the *10N20E* with 4 requests. Note that the centralized LO approach can obtain results in a reasonable computing time only for such small networks, since this latter increases exponentially with the network scale. Even for the *10N20E* network, the average computing time is already larger than 10^4 seconds. Subsequently, we analyze the solution mainly obtained by SG for 3 large network scenarios, including one real network (*CittàStudi* scenario, with 6 requests) and two random network topologies: *50N50E* with 10 requests and *100N150E* with 20 requests respectively.

SG is a *semi-distributed* approach, where the leader coordinates with the fully distributed followers, achieving a Stackelberg equilibrium beneficial to all entities, while LO is a *centralized* optimization approach standing for the interests of the leader. Besides, the LO approach can be tuned to consider different settings of latency weights ξ (see the objective function in model ($\mathcal{LO}$), where here the subscripts p, l are omitted for simplicity), which represent to what extent the leader cares about the processing and link latencies as a part of the followers' objectives. Higher values ($\xi = 1$ or 10 in the following figures) correspond to giving more weight to the latencies experienced by the followers, lower values ($\xi = 0.01$ or 0.1) give instead more weight to the profit U_{OP} of the network operator. The default values are set in Table II.

Fig. 2 illustrates the variations of the leader profit and user disutility against the request rate λ^k, which is scaled ($\forall k$) w.r.t. the initial value reported in Table II of a factor between 0.5 and 1.75. The solutions obtained by SG and LO are compared for the *5N5E* network. From Fig. 2, it is clear that SG, as a semi-distributed approach, achieves indeed a high leader profit, which is close to the best profit obtained by the centralized LO approach with an average ratio 0.94, and at the same time, causes low disutility (costs) for the followers (in average 0.67 times that obtained with LO).

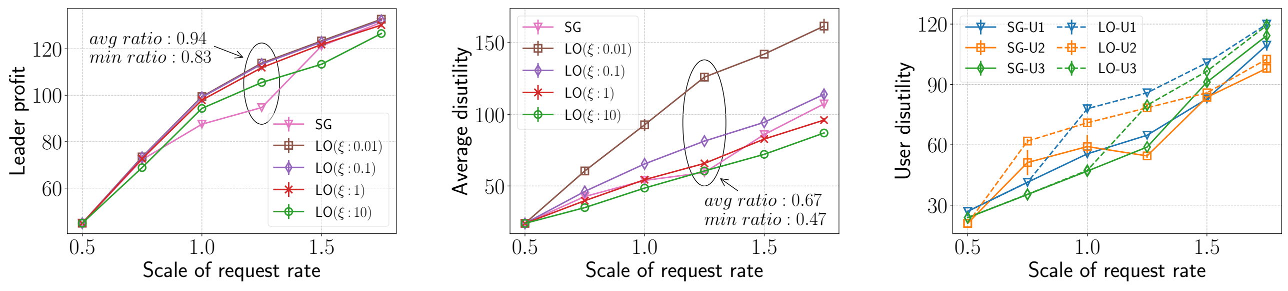
In Figs. 2(a) and 2(b), for the LO approach, a lower latency weight ξ can lead to a higher profit for the leader/network operator, but it also causes higher disutility (costs) for the followers/users. For SG, in the worst case (at scale 1.25), compared with LO (with $\xi = 0.01$), it obtains 0.84 times the profit for the leader, but only causes less than half (0.47) times the disutility for users. Fig. 2(c) compares the disutility of each user for SG and LO with the default weight ($\xi = 0.1$), where SG obtains lower disutility for each user than LO.

In summary, in the considered scenarios, users are better off under the SG than the LO approach, while obtaining at the same time a profit for the network operator which is comparable or close to that obtained with the centralized LO approach. This indeed justifies our choice to solve the resource planning and assignment problems using SG in network environments where it is intrinsically unpractical or difficult to devise a completely centralized solution.

In fact, when the request rate is low, SG can easily obtain the best solution. As the rate increases, the solution space becomes too large to be explored efficiently, and SG will find suboptimal solution in terms of leader profit, but it still achieves a very good balance between leader profit and user

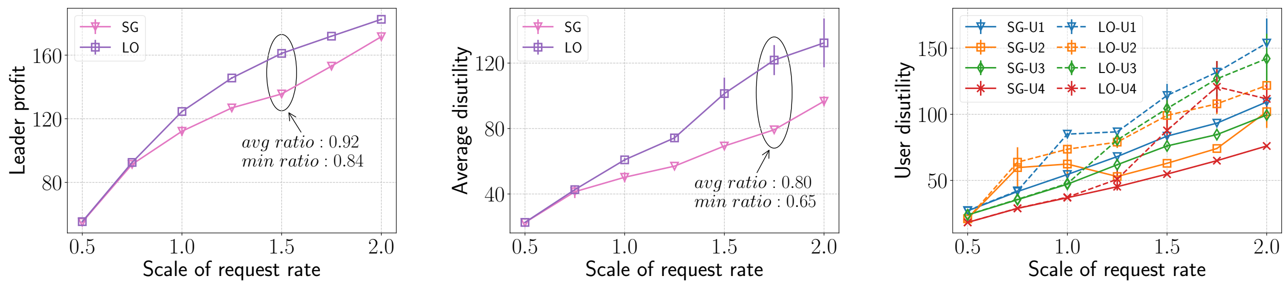
TABLE II: Parameters setting - initial (reference) data (for the case of high incoming request load).

Parameter		Initial value	
Request rate λ^k	(Gbps)	$30 \sim 55$	$(k \in \mathcal{K})$
Link bandwidth B_e	(Gbps)	$50 \sim 100$	$(e \in \mathcal{E})$
Computation level L_a	(Gbps)	30, 40, 50	$(a \in \mathcal{A})$
Computation budget P	(Gbps)	$300 \sim 4000$	
Cost factors θ_v, ϕ_e		0.25, 0.5	$(v \in \mathcal{V},\ e \in \mathcal{E})$
Price bounds $\boldsymbol{p}^{\mathsf{L}}$, $\boldsymbol{p}^{\mathsf{U}}$		0.1, 1.0	
Latency weights (user) w_p, w_l		10	
Latency weights (operator) ξ_p, ξ_l		0.1	



(a) Leader profit (b) Average disutility (c) User disutility ($\xi = 0.1$)

Fig. 2: Comparison of Stackelberg Game (SG) and Leader Optimal (LO) solutions, scaling the request rate $\lambda^k, \forall k$ in the 5N5E scenario with 4 players including 3 followers and 1 leader (the network operator).



(a) Leader profit (b) Average disutility (c) User disutility

Fig. 3: Comparison of SG and LO ($\xi = 0.1$), scaling $\lambda^k, \forall k$ in the 10N20E scenario with 5 players (4 followers and 1 leader).

disutility due to the Stackelberg equilibrium that is reached. When the request rate reaches a higher level, e.g., at scale 1.75 for the *5N5E* network, close to the saturation, the solution space shrinks, and SG can obtain the best solution.

Fig. 3 compares the results obtained in network *10N20E*. Considering the LO approach takes a long time and large computing resources to obtain the solution, we only illustrate the case where LO uses the default latency weight ($\xi = 0.1$). Figs. 3(a) and 3(b) show similar trends to the ones in Fig. 2; SG achieves lower leader profit with a lowest ratio 0.84 and average ratio 0.92 to LO, but better disutility than LO with lowest ratio 0.65 and average ratio 0.80. As shown in Fig. 3(c), SG achieves a better disutility, (which increases somehow monotonically with the requests' rates) than LO for each user.

Table III compares SG and LO w.r.t. leader profit, average disutility and computing time in three large network scenarios, i.e., *CittàStudi*, *50N50E* and *100N150E* (for LO, due to its high computing time, only 3 points are shown). For *CittàStudi* and *50N50E*, the leader profit and average disutility achieved by SG overlap with that of LO at scale point 0.5. For points 1.0 and 2.0, the leader profit obtained by SG is lower than LO, while the average disutility is better than LO. The minimum and average ratios for the leader profit are 0.86 and 0.93, for the average disutility, 0.57 and 0.74, respectively. For *100N150E*, LO cannot find a solution in an acceptable time. Indeed, comparing computing time, SG is extremely fast for all network scenarios.

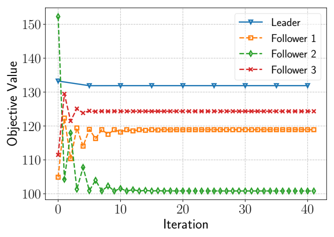
The average number of iterations and computing time for the convergence of SG are reported in Table IV in different network scenarios. SG can converge to a local Stackelberg equilibrium within few iterations and few seconds: even for the largest considered topology (*100N150E*), SG needs only 23 iterations and 9 seconds in average. For *CittàStudi*, the topology is close to a tree shape, resulting in straightforward routing strategies. Therefore, SG needs less iterations for *CittàStudi* than *10N20E*, but a longer computing time due to the larger problem scale. Similar properties apply to *50N50E*

TABLE III: Comparison of solutions in different scenarios.

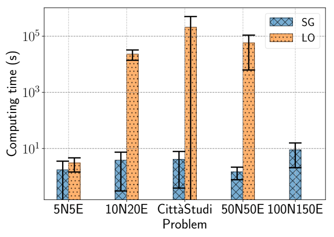
Scaling requests $\lambda^k, \forall k$		Leader Profit		Avg. Disutility		Comput. Time (s)	
		SG	LO	SG	LO	SG	LO
Città Studi	0.5	**82.28**	**82.28**	**22.82**	**22.82**	**0.10**	560.61
	1.0	153.30	**177.25**	**44.20**	77.13	**1.16**	3518.60
	1.5	205.24	-	67.66	-	1.84	-
	2.0	248.58	**265.44**	**95.46**	148.58	**5.80**	621755.30
	2.5	265.63	-	115.57	-	8.51	-
	3.0	317.33	-	138.58	-	9.52	-
50N50E	0.5	**129.61**	**129.61**	**21.66**	**21.66**	**0.13**	17383.41
	1.0	266.83	**298.65**	**46.31**	63.32	**2.10**	25255.48
	1.5	338.11	-	65.39	-	1.96	-
	2.0	397.30	**441.44**	**95.27**	141.23	**2.07**	129818.94
	2.5	472.62	-	117.52	-	1.94	-
	3.0	498.70	-	132.80	-	1.73	-
100N150E	0.5	238.14	-	20.56	-	**0.20**	-
	1.0	527.50	-	44.45	-	**1.35**	-
	1.5	627.31	-	61.78	-	**8.12**	-
	2.0	773.92	-	84.46	-	**9.77**	-
	2.5	909.33	-	113.08	-	**14.44**	-
	3.0	981.64	-	127.31	-	**22.74**	-

TABLE IV: Average iterations and computing time (sec) for the convergence of SG in different scenarios.

	5N5E	10N20E	CittàStudi	50N50E	100N150E
Avg. Iteration	11	16	14	5	23
Avg. Comput. Time	2.07	3.89	4.16	1.45	9.06



(a) Convergence of SG.



(b) Computing time.

Fig. 4: Convergence of SG (a) and computing time (b).

which has a simple semi-tree topology.

Fig. 4(a) illustrates the convergence process of SG. For clarity, we select the *5N5E* network with request rate λ^k scaled to 1.8 $\forall k$, which almost saturates the network. The oscillating *waves* of the followers and the slightly decreasing curve of the leader reflect the convergence process. At first, the leader estimates the required resources and the initial assignment of requests, and optimizes its strategy. Then, given these decisions, the followers compete for the resources targeting at their own objectives in a distributed way, reaching a subgame equilibrium (in about 5 iterations); the leader reacts again given the followers' decisions, and this process repeats until convergence to a local Stackelberg equilibrium. Fig. 4(b) shows the *computing time* of SG and LO for different networks. For LO, it increases exponentially as the problem scale increases, while for SG, it is almost around 10 seconds. Due to the tree-shape topologies of *CittàStudi* and *50N50E*, the computing time is not that higher than *10N20E*. Finally, LO is unable to obtain results in a reasonable time for *100N150E*.

IX. Conclusion

In this paper we investigated the joint optimization of network planning and request scheduling in edge computing networks and formulated it as a Stackelberg game, where the network operator, as market leader, plans and prices edge computation and communication resources, and provides routing paths for end user requests to maximize its profit, while edge users, as followers, compete for the network resources and schedule their requests to be served to minimize costs as well as link and processing latency. We proved that the follower subgame admits generalized Nash equilibrium, and the full leader-follower game admits Stackelberg equilibrium. We proposed an algorithm to find the local Stackelberg equilibrium that is guaranteed to converge, which runs extremely fast even for large-scale networks, and achieves high leader profit, close to the leader-optimal (LO) solution, while at the same time causing much lower user disutility than LO.

Acknowledgment

This research was supported by the H2020-MSCA-ITN-2016 SPOTLIGHT No. 722788 and the H2020-ICT-2020-1 PIACERE No. 101000162.

References

[1] J. Yan, S. Bi, L. Duan, and Y.-J. A. Zhang, "Pricing-driven service caching and task offloading in mobile edge computing," *IEEE Tran. Wirel. Commun.*, vol. 20, no. 7, pp. 4495–4512, 2021.

[2] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Trans. Mob. Comput.*, vol. 19, no. 1, pp. 29–43, 2019.

[3] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 85–97, 2018.

[4] L. Gu, D. Zeng, J. Hu, B. Li, and H. Jin, "Layer aware microservice placement and request scheduling at the edge," in *IEEE INFOCOM*, 2021, pp. 1–9.

[5] J. Zhou, D. Tian, Z. Sheng, X. Duan, and X. Shen, "Distributed task offloading optimization with queueing dynamics in multiagent mobile-edge computing networks," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 311–12 328, 2021.

[6] S. Deng *et al.*, "Optimal application deployment in resource constrained distributed edges," *IEEE Trans. Mob. Comput.*, vol. 20, no. 5, pp. 1907–1923, 2020.

[7] P. Lai *et al.*, "Quality of experience-aware user allocation in edge computing systems: A potential game," in *IEEE ICDCS*, 2020, pp. 223–233.

[8] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 273–287, 2020.

[9] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, 2019.

[10] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for joint cooperation and competition of edge resource allocation," in *IEEE ICDCS*, 2019, pp. 503–513.

[11] H. Zhou, Z. Wang, N. Cheng, D. Zeng, and P. Fan, "Stackelberg game-based computation offloading method in cloud-edge computing networks," *IEEE Internet of Things Journal*, 2022.

[12] S. Huang, H. Huang, G. Gao, Y.-E. Sun, Y. Du, and J. Wu, "Edge resource pricing and scheduling for blockchain: A stackelberg game approach," *IEEE Trans. Serv. Comput.*, 2022.

[13] S. Long, W. Long, Z. Li, K. Li, Y. Xia, and Z. Tang, "A game-based approach for cost-aware task assignment with QoS constraint in collaborative edge and cloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1629–1640, 2020.

[14] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *IEEE INFOCOM*, 2019, pp. 10–18.

[15] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, 2019.

[16] A. Santoyo-González and C. Cervelló-Pastor, "Latency-aware cost optimization of the service infrastructure placement in 5G networks," *J. Netw. Comput. Appl*, vol. 114, pp. 29–37, 2018.

[17] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A stackelberg game approach," *Computer Networks*, vol. 129, pp. 399–409, 2017.

[18] B. Xiang, J. Elias, F. Martignon, and E. Di Nitto, "Joint planning of network slicing and mobile edge computing: Models and algorithms," *IEEE Trans. Cloud Comput.*, 2021.

[19] B. Wu, J. Zeng, S. Shao, W. Ni, and Y. Tang, "New game-theoretic approach to decentralized path selection and sleep scheduling for mobile edge computing," *IEEE Tran. Wirel. Commun.*, 2022.

[20] K. Velasquez, D. P. Abreu, L. Paquete, M. Curado, and E. Monteiro, "A rank-based mechanism for service placement in the fog," in *IFIP Networking*, IEEE, 2020, pp. 64–72.

[21] Y. Niu, B. Luo, F. Liu, J. Liu, and B. Li, "When hybrid cloud meets flash crowd: Towards cost-effective service provisioning," in *IEEE INFOCOM*, 2015, pp. 1044–1052.

[22] *Technical Report*, https://www.dropbox.com/s/m1hsi65s1rrn9as/proofs.pdf, Created: 2023-02-11.

[23] F. Facchinei and C. Kanzow, "Generalized nash equilibrium problems," *4OR*, vol. 5, no. 3, pp. 173–210, 2007.

[24] S. Dempe, *Foundations of bilevel programming*. Springer Science & Business Media, 2002.

[25] P. Hansen, B. Jaumard, and G. Savard, "New branch-and-bound rules for linear bilevel programming," *SIAM Journal on scientific and Statistical Computing*, vol. 13, no. 5, pp. 1194–1217, 1992.

[26] P. Erdős and A. Rényi, "On Random Graphs I," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.

[27] *OpenCellID*, https://www.opencellid.org, Accessed: 2023-01-25.

[28] *SCIP*, https://www.scipopt.org, Accessed: 2023-01-25.

[29] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Comm. Surveys & Tutorials*, vol. 23, no. 1, pp. 596–630, 2020.