Toward a Virtual Edge Service Provider: Actor-Critic Learning to Incentivize the Computation Nodes

Mohammad Cheraghinia, Seyed Hamed Rastegar, Vahid Shah-Mansouri, *Member, IEEE*, Hamed Kebriaei, *Senior Member, IEEE*, Kun Zhu, *Member, IEEE*, and Dusit Niyato, *Fellow, IEEE*

***Abstract*—The growing development of computation-intensive applications has considerably increased the demand for computing resources in the network. Many of these applications require computations with low delays. Edge computing, which offers computing resources at the edge of the network, is a prominent solution. However, an edge service provider might not have sufficient resources at the edge to satisfy the demands of its users. Computation offloading can fill the gap by letting the service provider transfer the users' computational tasks to other computation nodes (CNs). Nevertheless, due to the imposed costs, CNs are not always interested in sharing their computation capacity. This article considers the concept of a virtual edge service provider, and studies an incentive mechanism to motivate the CNs to share their resources. We formulate the problem as a Stackelberg game and present a complete information analysis. We prove the existence and uniqueness of the game equilibriums. However, the complete information scenario is not accessible in practice due to privacy reasons. Therefore, we propose a soft actor-critic algorithm as an incomplete information method using deep reinforcement learning. Finally, through extensive numerical evaluations, we show that the incomplete information method converges to the same equilibrium that the complete information analysis proved.**

***Index Terms*—Computation offloading, deep reinforcement learning (DRL), incentive mechanism, soft actor-critic (SAC), stackelberg game, virtual edge service provider.**

I. Introduction

THE evolution of computation-intensive applications such as smart city, online gaming, e-health, and virtual/augmented reality (VR/AR) with low delay requirements [1], [2], [3] has created significant challenges for users and IoT devices with limited computation resources. A potential solution for this problem is using computing services offered by different entities. Compared to traditional cloud computing services, the emerging edge computing paradigm makes low-delay computations also possible. Therefore, edge computing has become a popular and feasible solution for executing the computational tasks of these applications.

An edge service provider is an entity that offers centralized cloud-based computing services and delivers low-delay computations at the edge. However, providing computational resources at different locations of a large network to provide reliable and high-quality computations at the edge is too complex and actually impossible. A practical and interesting approach to tackle the issue is to employ the computational resources of other computation nodes (CNs) available in the network and thus move toward the realization of a virtual edge service provider. A CN might be any entity with unused computational resources that, depending on its location, can serve tasks of some users with their target delays. Therefore, an edge service provider can offload the computational tasks of its users to CNs by making economic contracts and incentivizing them. Incentive mechanisms are attractive procedures to motivate the CNs at the edge of the network to increase their cooperation and overcome the challenges mentioned above [4], [5].

The computation offloading process includes nodes with diverse internal factors that result in different utility functions. Each node in the network is a rational decision-maker who takes action based on its unique utility function and interdependence caused by the other nodes. The game theory, which is a widely-used mathematical tool for the analysis of interactive decision-making problems, has been recently applied to computation offloading [6], [7]. However, these methods have difficulties handling the solutions in large-scale environments and finding solutions based on incomplete information. On the other hand, the network size increases, given the growth of IoT applications, and information sharing decreases based on privacy constraints. To overcome these challenges, the authors in [8] employ the Bayesian Stackelberg game, and [9], [10] have

Manuscript received 25 June 2022; revised 7 April 2023; accepted 27 May 2023. Date of publication 26 June 2023; date of current version 8 January 2024. This work was supported in part by the Institute for Research in Fundamental Sciences (IPM) under Grant CS 1402-4-208, in part by the National Natural Science Foundation of China under Grant 62071230, in part by MOE Tier 1 under Grant RG87/22, in part by the National Research Foundation, in part by the Singapore and Infocomm Media Development Authority, in part by Future Communications Research Development Programme, in part by DSO National Laboratories through AI Singapore Programme under Grant AISG2-RP-2020-019, in part by the Energy Research Test-Bed and Industry Partnership Funding Initiative, part of the Energy Grid (EG) 2.0 programme, and in part by the DesCartes and the Campus for Research Excellence and Technological Enterprise (CREATE) programme. Recommended for acceptance by Dr. Chunxiao Jiang. *(Corresponding author: Vahid Shah-Mansouri*

Mohammad Cheraghinia and Vahid Shah-Mansouri are with the School of Electrical and Computer Engineering, University of Tehran, Tehran 1417935840, Iran (e-mail: m.cheraghinia@ut.ac.ir; vmansouri@ut.ac.ir).

Seyed Hamed Rastegar is with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran 19395-5746, Iran (e-mail: hrastegar@ipm.ir).

Hamed Kebriaei is with the School of Electrical and Computer Engineering, University of Tehran, Tehran 1417935840, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran 19395-5746, Iran (e-mail: kebriaei@ut.ac.ir).

Kun Zhu is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China (e-mail: zhukun@nuaa.edu.cn).

Dusit Niyato is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: dniyato@ntu.edu.sg).

Digital Object Identifier 10.1109/TNSE.2023.3283410

2327-4697 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

designed two diverse solutions for incomplete information using deep reinforcement learning named proximal policy optimization (PPO) [11].

Model-free deep reinforcement learning is a combination of reinforcement learning (RL) and function approximations, such as neural networks, applied to a wide range of decision-making tasks. However, they suffer from critical issues: the high sample complexity and sensitivity to the hyperparameters. One of the significant reasons for the lower sample efficiency is on-policy learning methods, i.e., PPO [11], trust region policy optimization (TRPO) [12], and synchronous advantage actor-critic (A2C) [13]. On-policy employs the newly collected samples for each gradient step; nonetheless, off-policy reuses past experiences and indicates higher sample efficiency. Off-policy methods are difficult to use due to hyperparameters sensitivity, i.e., deep deterministic policy gradient (DDPG) [14]. In [15], an off-policy method is proposed, which is called soft actor-critic (SAC). It maximizes the entropy besides the expected reward and outperforms prior methods. This structure is very stable and reduces sensitivity to hyperparameters. In [16], the authors use SAC to learn the optimal bidding strategy under incomplete information assumptions. However, SAC has not been applied for motivation processes in computation offloading to the best of our knowledge.

This article designs a procedure based on game theory and deep reinforcement learning to overcome the mentioned challenges by moving toward the realization of a virtual edge service provider. We present an offloading structure, including a service provider, a set of users, and various computation nodes, where the service provider incentivizes the CNs to share their resources. Users are the task owners who subscribe to the service provider to get computation resources. We use the Stackelberg game to formulate the problem that matches the two-layer structure of the model. We analyze the uniqueness and existence of the Nash equilibrium for the CNs and the Stackelberg equilibrium for the entire game, considering complete information. Due to incomplete information constraints, we formulate this system model as a Markov decision process (MDP). We then propose a decentralized non-cooperative multi-agent SAC-based training algorithm for the CNs and the service provider to overcome continuous action spaces and stability. To summarize, the main contributions of this article can be listed below:

- We present a distributed computing system that includes a multi-objective service provider capable of considering both internal profit maximization and total energy consumption minimization, making it a green computation-aware entity in the network. This entity considers the users' task deadline as a reliable service provider.
- To increase the network's total computation capacity under the service provider's management, we introduce a bonus based on the maximum CPU frequency of the CNs to increase the contribution of the larger resources.
- To make the problem tractable, we propose a two-layer incentive mechanism for computation offloading led by a service provider paying a specified amount of money to all the CNs to manage their cooperation.
- We formulate the incentive mechanism as a Stackelberg game. Furthermore, provide detailed proof of the existence and uniqueness of the Nash and Stackelberg equilibriums analytically under the complete information scenario to provide a completely accurate comparison with the incomplete information scenario solution.
- We propose an algorithm based on a deep reinforcement learning method named SAC to solve the problem under an incomplete information scenario. This algorithm shows better stability and lower dependence on hyperparameters than PPO, DDPG, and A2C and outperforms them in continuous action spaces.

The organization of the rest of the article is as follows. Section II explains the related works. In Section III, we present our system model. Then, in Section IV, we formulate our system model. In the first subsection of Section V, we discuss the Stackelberg equilibrium with information sharing assumption. In the second subsection, we propose a DRL-based training algorithm for an incomplete information environment. Section VI provides the results of the simulations for the proposed algorithms.

II. Related Works

This section aims to include the related papers on computation offloading using incentive mechanisms and reinforcement learning. In the end, we have some documents using SAC to solve different problems with the highest capable accuracies to have a big picture of the previous related works in the methods contained in this article.

A. Incentive Mechanism in Computation Offloading

Data offloading incentive mechanisms have received attention recently [17], [18], [19], [20], [21], [22]. In [17], authors propose a three-stage Stackelberg game and aim to maximize the mobile network operator's profit and present an iterative algorithm for incomplete information to obtain the equilibrium. [18] proposes a congestion-aware scheduling scheme for cellular offloading, then offers a congestion-aware network selection algorithm and devises a contract-based incentive mechanism to maximize operator profit. In [19], authors propose an incentive mechanism to motivate users to leverage their delay tolerance for cellular offloading, and they investigate the trade off between the amount of traffic being offloaded and the users' satisfaction. [20] investigates the mobile data offloading problem through a third-party WiFi access point for a cellular mobile system and formulates the mobile data offloading problem as a utility maximization problem. In [21], authors consider the privacy and competition among the providers and propose a reinforcement learning technique to design an incentive mechanism for multiple providers and multiple IoT devices. In [22], an incentive system is used to encourage selfish users to use the higher-level fog computing layer based on their delay tolerance and available computing power. A multi-dimensional contract theory model is adopted to address the diverse needs of users with various applications and unique characteristics. The edge server determines rewards for users' efforts under this model.

B. Reinforcement Learning in Computation Offloading

A large number of previous works used deep reinforcement learning methods in solving task offloading problems [23], [24], [25], [26]. In [23], authors investigate a significant computation

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

offloading scheduling problem in a typical vehicular edge computing scenario and design a PPO-based DRL algorithm to solve the problem. [24] proposes a multi-agent deep reinforcement learning framework to achieve long-term performance for cooperative computation offloading, and agents explore the environment collaboratively for fast convergence and robustness. [25] considers multi-user edge-assisted video analytic task offloading problem, where users have video analytic tasks with various accuracy requirements, and proposes an A2C-based algorithm for the problem. In [26], a speed-aware and customized scheme are proposed to minimize total service latency for mobile users, and A2C is used to choose the task execution computation node dynamically. We note that computation offloading generally faces challenges in terms of communication latency, resource constraints, and privacy and security risks [27], [28]. However, the dynamic nature of edge environments and device heterogeneity further complicate the implementation of reinforcement learning models on these devices [29]. These challenges must be addressed to achieve optimal performance and reliability in edge computing.

C. SAC in Prior Works

Many papers used SAC proposed in [15] in diverse areas. [30] uses SAC to solve a novel live video transcoding and streaming scheme that maximizes the video bitrate and decreases time delays and bitrate variations in vehicle fog computing-enabled internet of vehicles. In [31], they propose a task offloading scheme in the context of vehicular fog computing, where vehicles are incentivized to share their idle computing resource by dynamic pricing, and propose a SAC-based algorithm to solve the problem. [32] investigates the issues of cooperative computation offloading for MEC in the 6G era. The proposed MEC system enables edge-cloud and edge-edge cooperation to address the limitations of single edge servers and the nonuniform distribution of computation task arrivals among multiple ESs. They propose centralized and decentralized SAC algorithms for intelligent computation offloading.

Considering the overviewed works, we propose our article as a combination of the solutions employed in these papers. We propose a computation offloading system model for incentivizing the servers and a mathematical model that uses game theory methods. We prove the existence and uniqueness of the Nash equilibriums. To make the problem applicable to real problems, we present an incomplete information method using SAC to overcome the difficulties because of the lack of pieces of information.

III. System Model

We consider a scenario in which $|\mathcal{M}|$ users denoted by $\mathcal{M} = \{1, 2, \ldots, |\mathcal{M}|\}$ submit their computational requests to a service provider. The service provider, which has no computational resources on its own, makes a contract with an available set of computation nodes (CNs) stated as $\mathcal{S} = \{1, 2, \ldots, |\mathcal{S}|\}$. As depicted in Fig. 1, the CNs include servers, fog nodes, edge nodes, and any other computation-capable devices with unused computational resources and tend to share them to earn money. The computational request of the mth user is composed of its task size, i.e., the number of required CPU clocks for the computation

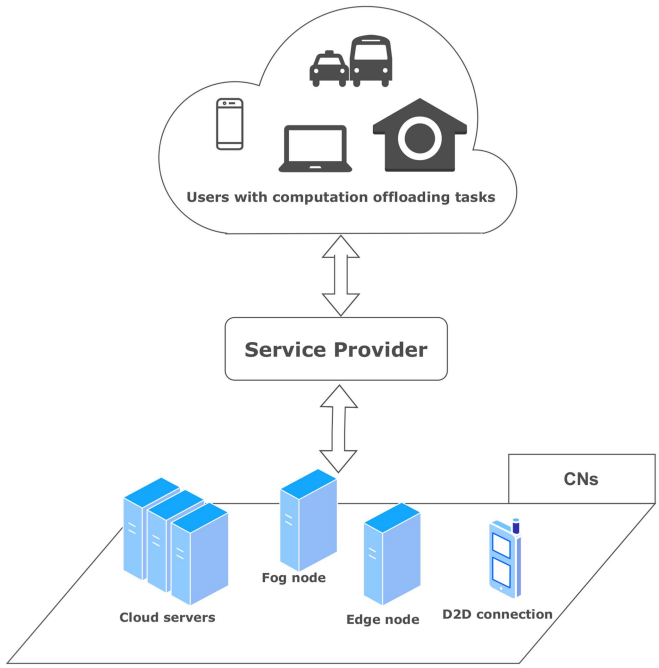

Fig. 1. Follower Leader based structure for Incentive Mechanism.

and deadline in seconds denoted by β_m and τ_m, respectively. We assume that the service provider pays a total of ρ units of money to the CNs based on their contributions to tasks' computations. Each CN would like to maximize its revenue which is a portion of ρ. This proportion is paid by the service provider to each CN, defined as the bonus based on their full contribution capability.

The shared processing power of CNs for computing the offloaded tasks is equal to f_s in clocks per second. We note that clock speed refers to the number of pulses per second generated by an oscillator that sets the tempo for the processor. A higher number of pulses per second results in faster processing. The cost of computation with frequency f_s is denoted by ρ_s^{cmp} per clock per second. This cost includes the energy consumption price, maintenance, and other computation costs for the sth CN. Their bonus is calculated based on their maximum contribution capacity defined as $f_s^{\max}$. Consequently, they maximize the utility and would not contribute if they get insufficient money to have positive utility. Ultimately, the key notations used in system model are summarized in Table I.

We suppose that a user's task can be divided into several smaller disjoint tasks to be computed by different CNs [40], [41]. The service provider is assumed to be aware of the users' task sizes and deadlines and is responsible for incentivizing the computation nodes. The offloading process is also assumed to be done directly between the computation nodes and the users without involving the leader. This design ensures that the leader does not have access to the offloaded data and, therefore, cannot compromise the users' privacy. In the considered scenario, the service provider aims to maximize its profit while minimizing the CNs' total energy consumption. The goal of internal profit maximization is to enable the service provider to generate enough revenue to compensate for its internal costs and achieve a satisfactory level of profitability, as the CNs intend to make the maximum possible revenue for computing the offloaded tasks

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Description
$\mathcal{M} = \{1, 2, ..., \lvert\mathcal{M}\rvert\}$	Set of the users
$\mathcal{S} = \{1, 2, ..., \lvert\mathcal{S}\rvert\}$	Set of the CNs
ρ	Total money paid to all of the CNs
β_m	Task size of the user m in clocks
τ_m	Deadline of the user m in seconds
ρ_s^{cmp}	Computation price per clock/second
f_s	Computation of CN in clocks/second
f_s^{max}	Max computation capacity in clocks/second
η	Scaling factor
κ	Energy consumption capacitance coefficient
μ	The profit proportion of the leader

to them. Additionally, minimizing total energy consumption is very crucial for decreasing the system's environmental impact and lowering operating costs. Considering the above objective, the task requirements of the users, including computations and deadlines, should be met. The following section presents a detailed mathematical formulation for this scenario.

We can consider two scenarios for the system model, including complete and incomplete information. Under complete information assumptions, the CNs are aware of all the parameters of each other, and the leader is mindful of their decisions; on the other hand, in incomplete information conditions that are realistic compared to the previous one, the CNs and leader are not aware of the internal parameters of other CNs. The incomplete information assumption exists because of the servers' privacy and competition with other network nodes.

IV. PROBLEM FORMULATION

This section presents a mathematical formulation for the system model. The problem has two stages. The first stage includes the service provider's payment to the CNs, and the second stage is the CNs' turn to determine their contribution value. We formulate such a problem as a Stackelberg game where the service provider and CNs correspond to the leader and followers, respectively. CNs always take action following the action of the service provider.

A Stackelberg game is a two-stage game consisting of two types of players named leaders and followers. The main property of this game is that some of the players (i.e., the leaders) first perform their actions, followed by the action of the remaining players (i.e., followers). This Stackelberg game is to maximize the benefit of the leader, given that the followers also maximize their benefits [33]. Any interaction with the above property between several entities can be modeled and studied as a Stackelberg game. In the following, we adopt the backward induction method to introduce the Stackelberg game formulation.

A. The Followers Problem

Followers are the computation-capable devices in the network, such as edge nodes, fog nodes, and D2D devices denoted by set $\mathcal{S}$. They can share their computation capacity with other users for computation offloading. However, there are many costs involved, such as energy consumption and maintenance costs, that we can motivate them to participate by paying them all the costs plus some additional incentive. CNs share a specific amount of their computation capacity f_s and receive two different proportions of ρ based on the shared value. The first part of the earned money is based on the contribution value of the CN compared to all CNs' contributions, and the second part is the bonus that depends on two factors, f_s compared to $\sum_{x\in\mathcal{S}} f_x$, and $f_s^{\max}$ compared to $\sum_{x\in\mathcal{S}} f_x^{\max}$. Dependence on these factors ensures that the bonus will not be paid to the users with a zero contribution and increases their bonus based on their maximum contribution capability compared to other CNs' maximum contribution capacity. The bonus motivates the CNs to increase their shareable computation capacity and improves the leader's reliability. The leader is responsible for assuring the QoS, and a higher computation capacity increases its reliability. Because of sharing f_s, CNs will be charged a computation fee as the cost of their computation process. Consequently, there is a cost based on the shared CPU frequency, reduced from the money gained. We formulate a utility function for each of the followers as below:

$$U_s(f_s, f_{-s}) = (1 + v_s)G(f_s, f_{-s}, \rho) - \rho_s^{\text{cmp}} f_s, \tag{1}$$

where $G(\cdot,\cdot,\cdot)$ denotes the contribution evaluation function, which is a function of f_s, f_{-s}, and ρ. A CN earns the total amount of contribution evaluation value and a proportion of that as a bonus. We define the bonus factor as v_s that equals $\frac{f_s^{\max}}{\sum_{x\in\mathcal{S}} f_x^{\max}}$ based on the CNs' maximum contribution capacity. The idea behind employing the bonus is that CNs with more available computation capacity is given more rewards to motivate them to contribute their resources to the system. This approach is based on the belief that CNs with greater resources will be able to contribute more to the system and should, therefore, be rewarded more. G assesses the contribution level of each CN compared to all other CNs and then assigns a certain amount of money to the CN based on the evaluation. Due to the effect of diminishing return, it is considered to be a continuous and quasi-concave function that satisfies the following constraints,

$$G(f_s = 0, f_{-s}, \rho) = 0, \tag{2}$$

$$\frac{\partial G(f_s, f_{-s}, \rho)}{\partial f_s} > 0, \tag{3}$$

$$\frac{\partial^2 G(f_s, f_{-s}, \rho)}{\partial f_s^2} < 0. \tag{4}$$

Without loss of generality, the contribution evaluation function is defined as:

$$G(f_s, f_{-s}, \rho) = \frac{\rho f_s}{\sum_{x\in\mathcal{S}} f_x}. \tag{5}$$

Considering (5), the contribution evaluation function for each CN is a ratio in which f_s is divided by the sum of other CNs' shared processing power, which is positive and smaller than one; therefore, its multiplication with ρ results in a proportion of total money for that CN. Furthermore, the followers aim to maximize their utility as below:

$$\max_{f_s} \quad U_s(f_s, f_{-s}) \tag{6a}$$

$$\text{s.t.} : 0 \le f_s \le f_s^{\max}. \tag{6b}$$

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

In this formulation, (6b) defines the shared computation power bound for each CN, which should be positive and smaller than the maximum capacity of each CN.

B. The Leader Problem

The leader aims to jointly maximize the utility and minimize the CNs' total energy consumption. The utility is the subtraction of the money paid to CNs denoted by $W\rho$ and money achieved from the users indicated by $W\mu\rho$. Variable μ is a constant value defined as a characteristic of the leader from the users' point of view that defines the profit percentage of the leader for leading the process and ensuring QoS quality to the users. The utility is defined as

$$U_l(\rho) = \rho W(\mu - 1) - \kappa\eta \left(\sum_{x\in\mathcal{M}} \beta_x\right)\left(\sum_{y\in\mathcal{S}} f_y\right)^2. \tag{7}$$

in which W is expressed as follows:

$$W = 1 + \frac{(\sum_{x\in\mathcal{S}} f_x f_x^{\max})}{(\sum_{x\in\mathcal{S}} f_x)(\sum_{x\in\mathcal{S}} f_x^{\max})}. \tag{8}$$

Also in (7), η is employed to have the same scale of the numbers in both parts of the utility function, and κ is a constant related to the hardware architecture [34]. We note that from an economical perspective, it is a widespread and valid assumption to consider the revenue of an agent to be proportional to its obtained money or payment [35]. Accordingly, we define the utility of the leader to be proportional to $W\rho(\mu - 1)$. Considering the total energy consumption of CNs makes this multi-objective utility function energy aware and, besides personal economic profit, considers the social effect of incentivization. Therefore, the leader aims to solve the following optimization problem:

$$\max_{\rho} \quad U_l(\rho) \tag{9a}$$

$$\text{s.t.:} \sum_{x\in\mathcal{S}} f_x \geq \sum_{y\in\mathcal{M}} \frac{\beta_y}{\tau_y} \tag{9b}$$

$$\rho \geq 0. \tag{9c}$$

The constraint in (9b) is used to ensure that the amount of computation capacity (CPU frequencies) is more than the required amount. In this constraint, we assume that the task can be divided into subtasks.

V. Stackelberg Equilibrium Analysis

Our problem consists of a service provider as a leader and CNs as followers. The CNs play a non-cooperative game given the strategy of the service provider, in which the outcome of this game is the Nash equilibrium. The CNs, as the followers, also play a Stackelberg game with the service provider as the leader, in which the outcome of this game is the Stackelberg equilibrium. Nash and Stackelberg equilibriums are both concepts that describe the behavior of multiple players in a game or decision-making scenario. In a Nash equilibrium, all players make their decisions simultaneously. In contrast, in a Stackelberg equilibrium, one player makes their decision first, and the other players observe this decision and make their own decisions based on it. Considering the two-layer structure of our problem and sequential decision-making setting, we formulate our system model as a hierarchical Nash-Stackelberg game. We know that the CNs and the leader are not thoroughly aware of all parameters of other players, and the incomplete information scenario must be considered due to communications and privacy reasons. In this regard, we first analyze the complete information case. Then, to consider the practical limitations, we formulate and analyze an incomplete information scenario. Finally, by employing the uniqueness of the Nash equilibrium, we show that our proposed incomplete information framework has the same performance as the complete information case. We summarize our Stackelberg game below:

$$\mathcal{G} = \begin{cases} \textbf{Players}: \text{ CNs as the followers } \mathcal{S} = \{1, 2, \ldots, |\mathcal{S}|\}, \\ \quad \text{and service provider as the leader} \\ \textbf{Strategies}: f_s \text{ is the strategy of the CNs and } \rho \text{ is} \\ \quad \text{the strategy of the leader} \\ \textbf{Utilities}: U_s(f_s, f_{-s}) \text{ is the utility of the followers} \\ \quad \text{and } U_l(\rho) \text{ is the utility of the leader} \end{cases}$$

A. Complete Information

This section analyzes the Stackelberg game with complete information in which the leader and followers are aware of the utility functions and the individual parameters of each other for determining the optimal strategies.

1) Existence Analysis: We first consider the analysis of the existence of the equilibrium.

Definition 1 (Nash Equilibrium): An s-tuple of action variables $f^\star = \{f_1^\star, f_2^\star, \ldots, f_s^\star\}$ constitutes Nash equilibrium (or, non-cooperative equilibrium) (NE) if, for all $s \in \mathcal{S}$ we have:

$$U_s(f_s^\star, f_{-s}^\star) \geq U_s(f_s, f_{-s}^\star), \tag{10}$$

for any $f_s > 0$.

We prove that there exists a unique Nash equilibrium for the CNs with complete information. To this aim, we first consider the follower's utility and calculate the first derivative of their utility:

$$\frac{\partial U_s}{\partial f_s} = \frac{\rho(1+\upsilon_s)}{\sum_{x\in\mathcal{S}} f_x} - \frac{\rho(1+\upsilon_s) f_s}{(\sum_{x\in\mathcal{S}} f_x)^2} - \rho_s^{\text{cmp}}, \tag{11}$$

and the second derivative is:

$$\frac{\partial^2 U_s}{\partial f_s^2} = \frac{-2\rho(1+\upsilon_s)\sum_{x\neq s} f_x}{(\sum_{x\in\mathcal{S}} f_x)^3} < 0. \tag{12}$$

It can be shown that the second-order derivative is negative, and the utility of each one of the followers is a concave function.

Lemma 1: There exists a Nash Equilibrium for the followers.

Proof: The players have a finite strategy set, and their strategies are bounded, closed, and concave. On the other hand, their utilities are continuous and quasi-concave in that strategy set. So, we can calculate the Nash equilibrium for the followers. Based on these, theorem 1 in [36] proves that there exists an equilibrium point for every concave n-person game. □

The results show that the utility is a concave function, and there exists an optimum for any of the agents, which is:

$$f_s^\star = \sqrt{\frac{\rho(1+\upsilon_s)\sum_{x\neq s} f_x}{\rho_s^{\text{cmp}}}} - \sum_{x\neq s} f_x. \tag{13}$$

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

Based on (13) and (6b), we can say that different ρ results in a different outcome of $f_s^\star$. Therefore we can have:

$$f_s^\star = \begin{cases} 0, & \rho \le \frac{\rho_s^{\text{cmp}} \sum_{x\neq s} f_x}{(1+v_s)} \\ f_s^{\max}, & \rho \ge \frac{\rho_s^{\text{cmp}} (\sum_{x\neq s} f_x + f_s^{\max})^2}{(1+v_s)\sum_{x\neq s} f_x} \\ \sqrt{\frac{\rho(1+v_s)\sum_{x\neq s} f_x}{\rho_s^{\text{cmp}}}} - \sum_{x\neq s} f_x, & otherwise \end{cases} \tag{14}$$

The following corollary gives closed-form expressions for optimal CPU frequency of computing nodes.

Corollary 1: We denote the subset of CNs that contributes to the computation by the set $\mathcal{G} \subset \mathcal{S}$. The optimal CPU frequency of any CN $g \in \mathcal{G}$ could be obtained as:

$$f_g^\star = \frac{(g-1)\rho}{\sum_{y\in\mathcal{G}} \frac{\rho_y^{\text{cmp}}}{1+v_y}} \left(1 - \frac{(g-1)\frac{\rho_g^{\text{cmp}}}{1+v_g}}{\sum_{y\in\mathcal{G}} \frac{\rho_y^{\text{cmp}}}{1+v_y}}\right). \tag{15}$$

Proof: (15) shows that the CNs who share can be a set named $\mathcal{G} = \{1, 2, \ldots, |\mathcal{G}|\}$ and $\mathcal{G} \subset \mathcal{S}$. For this set, we can find the Nash Equilibrium following the previous results. First we can change (13) as below:

$$\sum_{x\in\mathcal{G}} f_x = \sqrt{\frac{\rho(1+v_g)\sum_{x\neq g} f_x}{\rho_g^{\text{cmp}}}}. \tag{16}$$

We assume the change of the variable $\sum_{x\in\mathcal{G}} f_g = \Phi$. The result is:

$$\frac{\Phi^2 \rho_g^{\text{cmp}}}{\rho(1+v_g)} = \sum_{x\neq g} f_x. \tag{17}$$

If we sum both sides of the equation with all of $f_1, f_2, \ldots, f_g$, we will have:

$$\begin{cases} f_1 = \Phi - \frac{\Phi^2 \rho_1^{\text{cmp}}}{\rho(1+v_1)} \\ f_2 = \Phi - \frac{\Phi^2 \rho_2^{\text{cmp}}}{\rho(1+v_2)} \\ \ldots \\ f_g = \Phi - \frac{\Phi^2 \rho_g^{\text{cmp}}}{\rho(1+v_g)} \end{cases} \tag{18}$$

Summing all the equations of (18) will result in:

$$\Phi = g\Phi - \frac{\Phi^2}{\rho} \left(\sum_{y\in\mathcal{G}} \frac{\rho_y^{\text{cmp}}}{1+v_y}\right). \tag{19}$$

Solving the (19) to find Φ will have the below outcome:

$$\Phi = \frac{(g-1)\rho}{\sum_{y\in\mathcal{G}} \frac{\rho_y^{\text{cmp}}}{1+v_y}}. \tag{20}$$

Substituting (20) in one of the expressions in (18) gives us the Nash Equilibrium:

$$f_g^\star = \frac{(g-1)\rho}{\sum_{y\in\mathcal{G}} \frac{\rho_y^{\text{cmp}}}{1+v_y}} \left(1 - \frac{(g-1)\frac{\rho_g^{\text{cmp}}}{1+v_g}}{\sum_{y\in\mathcal{G}} \frac{\rho_y^{\text{cmp}}}{1+v_y}}\right). \tag{21}$$

□

The $f_g^\star$ is independent of other followers' strategy and only depends on their basic information, including the sum of ρ_g^{cmp} and $f_g^{\max}$.

Remark 1: The Total required money from the leader to incentivize the followers is bounded.

Proof: From (1), by summing the total earned money for all the agents we have:

$$\sum_{y\in\mathcal{S}} \frac{\rho f_y + \rho f_y \frac{f_y^{\max}}{\sum_{x\in\mathcal{S}} f_x^{\max}}}{\sum_{x\in\mathcal{S}} f_x} = \rho + \rho \frac{(\sum_{x\in\mathcal{S}} f_x f_x^{\max})}{(\sum_{x\in\mathcal{S}} f_x)(\sum_{x\in\mathcal{S}} f_x^{\max})}. \tag{22}$$

We can derive that:

$$(\sum_{x\in\mathcal{S}} f_x)(\sum_{x\in\mathcal{S}} f_x^{\max}) = \sum_{k\in\mathcal{S}} f_k f_k^{\max} + \sum_{i\in\mathcal{S}} \sum_{j\neq i} f_i f_j^{\max}. \tag{23}$$

Therefore:

$$\frac{(\sum_{x\in\mathcal{S}} f_x f_x^{\max})}{(\sum_{x\in\mathcal{S}} f_x)(\sum_{x\in\mathcal{S}} f_x^{\max})} \le 1, \tag{24}$$

and we can have:

$$U_s(f_s, f_{-s}) + \rho_s^{\text{cmp}} f_s \le 2\rho. \tag{25}$$

□

Based on this proof, we can clarify that the leader can pay lower than 2ρ to all the CNs and ensure that this money is enough to incentivize the CNs and will never tend to infinity or large numbers.

2) Uniqueness Analysis: We next consider the analysis of the uniqueness of the equilibrium.

Definition 2 (Diagonally strictly concave function [36]): A weighted non-negative sum of the functions $U_s(f_s, f_{-s})$ for each non-negative vectors r and f that $r_s \ge 0$ and $f_s \ge 0$, is defined as $\sigma(f, r) = \sum_{x\in\mathcal{S}} r_x f_x$. $\sigma(f, r)$ is diagonally strictly concave function if for every $m, n \in f$ we have:

$$(m-n)^T g(n, r) + (n-m)^T g(m, r) > 0. \tag{26}$$

In this equation, $g(f, r)$ is defined as the pseudogradient of f, which is:

$$g(f, r) = \left[r_1 \frac{\partial U_1(f)}{\partial f_1}, r_2 \frac{\partial U_2(f)}{\partial f_2}, \ldots, r_s \frac{\partial U_s(f)}{\partial f_s}\right]^T. \tag{27}$$

Theorem 1: Followers have a unique Nash equilibrium.

Proof: Theorem 2 in [36] states that if $\sigma(f, r)$ is diagonally strictly concave for an S-player game, then the Nash equilibrium is unique. Based on the definition of $g(f, r)$ and followers' utility functions, we have:

$$g(f, r) = \begin{bmatrix} r_1 \frac{\rho(1+v_1)\sum_{x\neq 1} f_x}{(\sum_{x\in\mathcal{S}} f_x)^2} - r_1 \rho_1^{\text{cmp}} \\ r_2 \frac{\rho(1+v_2)\sum_{x\neq 2} f_x}{(\sum_{x\in\mathcal{S}} f_x)^2} - r_2 \rho_2^{\text{cmp}} \\ \ldots \\ r_s \frac{\rho(1+v_s)\sum_{x\neq s} f_x}{(\sum_{x\in\mathcal{S}} f_x)^2} - r_s \rho_s^{\text{cmp}} \end{bmatrix} \tag{28}$$

Theorem 6 in [36] states that a sufficient condition that $\sigma(f, r)$ is diagonally strictly concave for f and fixed $r > 0$ is that the symmetric matrix $[G(f, r) + G^T(f, r)]$ be negative definite for f. $G(f, r)$ is the Jacobian matrix of the $g(f, r)$. Based on this

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

theorem, we will have:

$$\big[G(f,r)+G^T(f,r)\big]_{ii} = -4\rho(1+v_i)r_i\frac{\sum_{x\neq i} f_x}{\big(\sum_{x\in\mathcal{S}} f_x\big)^3}, \tag{29}$$

$$\big[G(f,r)+G^T(f,r)\big]_{ij} = -2\rho(1+v_i)r_i\frac{\sum_{x\neq i,j} f_x}{\big(\sum_{x\in\mathcal{S}} f_x\big)^3}. \tag{30}$$

These values clearly show that all the elements of the matrix are negative. We assume that $r_i = \frac{1}{1+v_i}$ and simplify the notation as below, $Q = \sum_{x\in\mathcal{S}} f_x$, $Q_i = \sum_{x\neq i} f_x$, $Q_{i,j} = \sum_{x\neq i,j} f_x$. Therefore, we have:

$$G(f,r)+G^T(f,r) = \frac{-2\rho}{Q^3}\begin{bmatrix} 2Q_1 & Q_{1,2} & \cdots & Q_{1,s} \\ Q_{2,1} & 2Q_2 & \cdots & Q_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{s,1} & Q_{s,2} & \cdots & 2Q_s \end{bmatrix} \tag{31}$$

Since the matrix elements are positive, our criteria change to prove that the matrix part is positive definite. In addition, we will split the matrix into different matrices and use the Minkowski determinant theorem [37] as if we have three positive semidefinite matrices of A, B, and C, and thus we can write:

$$det(A+B+C) \geq det(A)+det(B)+det(C). \tag{32}$$

In the remaining proof, we adapt the approach employed in the proof of Theorem 1 in [38]. To simplify our matrix splitting notation, we define $Q_i^k = \sum_{x=1,x\neq i}^{k} f_x, Q_{i,j}^k = \sum_{x=1,x\neq i,j}^{k} f_x$, and $\hat{Q}^k = \sum_{x=k+1}^{s} f_x$

$$det\Bigg(\underbrace{\begin{bmatrix} Q_1^k & Q_{1,2}^k & \cdots & Q_{1,k}^k \\ Q_{2,1}^k & Q_2^k & \cdots & Q_{2,k}^k \\ \vdots & \vdots & \ddots & \vdots \\ Q_{k,1}^k & Q_{k,2}^k & \cdots & Q_k^k \end{bmatrix}}_{A^k} + \underbrace{\begin{bmatrix} \hat{Q}^k & \hat{Q}^k & \cdots & \hat{Q}^k \\ \hat{Q}^k & \hat{Q}^k & \cdots & \hat{Q}^k \\ \vdots & \vdots & \ddots & \vdots \\ \hat{Q}^k & \hat{Q}^k & \cdots & \hat{Q}^k \end{bmatrix}}_{B^k}$$
$$+ \underbrace{\begin{bmatrix} Q_1 & 0 & \cdots & 0 \\ 0 & Q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_k \end{bmatrix}}_{C^k}\Bigg) \tag{33}$$

As we have defined, Q_i is positive for all $i \in \mathcal{S}$. Hence, C^k is a positive definite and has a strictly positive determinant. We can show that the B^k can be written as below:

$$B^k = {M^k}^T M^k, M^k = \sqrt{\hat{Q}^k}\vec{1}_{1\times k}. \tag{34}$$

Therefore, we can say that B^k is a positive semidefinite. These results and (32) show that if we can prove that the A^k is positive semidefinite, we will prove the theorem. We will use the leading principal minors of the A^k and prove that the determinant of the nth leading principal minor of A^k is non-negative. We define:

$$A_n^k \triangleq \begin{bmatrix} Q_1^k & Q_{1,2}^k & \cdots & Q_{1,n}^k \\ Q_{2,1}^k & Q_2^k & \cdots & Q_{2,n}^k \\ \vdots & \vdots & \ddots & \vdots \\ Q_{n,1}^k & Q_{n,2}^k & \cdots & Q_n^k \end{bmatrix} \tag{35}$$

Again we use matrix splitting and define $\hat{Q}_{n+1}^k = \sum_{x=n+1}^{k} f_x$, we will have:

$$|A_n^k| = det\Bigg(\underbrace{\begin{bmatrix} Q_1^n & Q_{1,2}^n & \cdots & Q_{1,n}^n \\ Q_{2,1}^n & Q_2^n & \cdots & Q_{2,n}^n \\ \vdots & \vdots & \ddots & \vdots \\ Q_{n,1}^n & Q_{n,2}^n & \cdots & Q_n^n \end{bmatrix}}_{D^n}$$
$$+ \underbrace{\begin{bmatrix} \hat{Q}_{n+1}^k & \hat{Q}_{n+1}^k & \cdots & \hat{Q}_{n+1}^k \\ \hat{Q}_{n+1}^k & \hat{Q}_{n+1}^k & \cdots & \hat{Q}_{n+1}^k \\ \vdots & \vdots & \ddots & \vdots \\ \hat{Q}_{n+1}^k & \hat{Q}_{n+1}^k & \cdots & \hat{Q}_{n+1}^k \end{bmatrix}}_{E_n^k}\Bigg) \tag{36}$$

As discussed for B_k, E_n^k is positive definiteness. Similarly, using the Minkowski determinant theorem for positive definiteness of A_n^k results in positive definiteness of D^n. If we compare D^n with A^k, we will find out the structural similarity between them. If we assume that the theorem holds for the $A^1 = 0$, the theorem is proved by induction. □

Definition 3 (Stackelberg Equilibrium): Let $f^\star$ and $\rho^\star$ define the best responses of the leader and the followers, respectively. We have the point $(f^\star, \rho^\star)$ as the Stackelberg equilibrium(SE) for the Stackelrberg game if any (f,ρ) with $f \geq 0$ and $\rho \geq 0$, results in $U_l(f^\star,\rho^\star) \geq U_l(f^\star,\rho)$ and $U_s(f,\rho^\star) \geq U_s(f^\star,\rho^\star)$.

Theorem 2: There exists a unique optimal point in the leader that satisfies the (9b) and results in unique Stackelberg Nash Equilibrium $(f^\star,\rho^\star)$.

Proof: We will use (20) in the leader Nash calculation. If we replace Φ instead of $\sum_y f_y$ in (7) we will have:

$$U_{\text{leader}}(\rho) = \rho W(\mu-1) - \kappa\eta\left(\sum_{x\in\mathcal{M}}\beta_x\right)\left(\frac{(g-1)\rho}{\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y}}\right)^2. \tag{37}$$

The first order derivative of the leader utility is:

$$\frac{\partial U_{\text{leader}}}{\partial\rho} = W(\mu-1) - 2\rho\kappa\eta\left(\sum_{x\in\mathcal{M}}\beta_x\right)\left(\frac{(g-1)}{\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y}}\right)^2. \tag{38}$$

And the second order derivative is:

$$\frac{\partial^2 U_{\text{leader}}}{\partial\rho^2} = -2\kappa\eta\left(\sum_{x=1}^{m}\beta_x\right)\left(\frac{(g-1)}{\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y}}\right)^2 < 0. \tag{39}$$

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

Therefore the optimal value will be:

$$\rho^{\star} = \frac{W(\mu-1)}{2\kappa\eta\left(\sum_{x=1}^{m}\beta_x\right)\left(\frac{(g-1)}{\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y}}\right)^2}. \tag{40}$$

We have (9b) that might change the optimal point for the leader problem. If we substitute (20) in (9b), we would have:

$$\rho < \frac{\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y}\sum_{y\in\mathcal{M}}\frac{\beta_y}{\tau_y}}{(g-1)}. \tag{41}$$

This condition should meet that the (40) be the optimal point. We can define the optimal point as below:

$$\rho^{\star} = \begin{cases} \frac{\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y}\sum_{y\in\mathcal{M}}\frac{\beta_y}{\tau_y}}{(g-1)}, & \rho < \frac{\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y}\sum_{y\in\mathcal{M}}\frac{\beta_y}{\tau_y}}{(g-1)} \\ \frac{W(\mu-1)}{2\kappa\eta(\sum_{x=1}^{m}\beta_x)(\frac{(g-1)}{\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y}})^2}, & otherwise \end{cases} \tag{42}$$

□

3) Discussion: The existence of only one decision variable for the followers can decrease the leader's flexibility in diverse situations of the problem. We want to show that the CNs need another variable to increase their diversity. More decision variables might increase the complexity of the incomplete information solution; however, the numerical results will depict that the complexity increase does not affect the results.

Corollary 2: Besides having ρ_g^{cmp} as a decision variable of each CN, $f_g^{\max}$ can positively affect the CNs strategy.

Proof: If we calculate the derivative of (15) with respect to $f_g^{\max}$ we will have:

$$\begin{aligned}\frac{\partial f_g}{\partial f_g^{\max}} &= \frac{\partial f_g}{\partial v_g}\frac{\partial v_g}{\partial f_g^{\max}} \\ &= \frac{\rho(g-1)\rho_g^{\text{cmp}}[g(1+v_g)\sum_{x\neq g}\frac{\rho_x^{\text{cmp}}}{1+v_x} - (g-2)\rho_g^{\text{cmp}}]}{(\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y})^3} \\ &\times \frac{\sum_{x\neq g} f_x^{\max}}{(\sum_{x\in\mathcal{G}} f_x^{\max})^2}.\end{aligned} \tag{43}$$

From (15) we can derive that for positive strategy we should have $\sum_{y\in\mathcal{G}}\frac{\rho_y^{\text{cmp}}}{1+v_y} \geq (g-1)\frac{\rho_g^{\text{cmp}}}{1+v_g}$, and if we subtract $\frac{\rho_g^{\text{cmp}}}{1+v_g}$ from both sides will have:

$$\sum_{y\neq g}\frac{\rho_y^{\text{cmp}}}{1+v_y} \geq (g-2)\frac{\rho_g^{\text{cmp}}}{1+v_g}. \tag{44}$$

The (44) easily proves that $\frac{\partial f_g}{\partial f_g^{\max}} \geq 0$. □

Based on the Corollary, if we assume that the CNs change their $f_s^{\max}$ in different time steps, this property can dispense the task on the CNs and does not have pressure on only one of them just because of lower computation cost.

B. Incomplete Information

Due to privacy-preserving reasons and the economic competition between the CNs, we propose a method using SAC [15] to formulate our problem as an MDP problem, which is defined as $< S, A, P, R >$ where S and A are the state and action spaces. P denotes the transition probability between states, and R represents the Reward function.

While gradient-based methods may be effective in some scenarios, the use of model-free methods such as the SAC algorithm is more appropriate for our problem due to the ambiguity of the utility function and the need for a robust and adaptive solution in the case of incomplete information.

1) State and Action Spaces: We have s follower agents and one agent known as the leader. The followers have different utility and strategies compared to the leader, So we have to define the action and state spaces specifically for the followers and the leader.

Leader State Space: The leader should have information about its previous strategies and the followers' resulting strategies based on the leader's action for each time step. To ensure enough information for the leader, we assume previous D strategies and the interactions. So the state space for a leader is:

$$S_t^{\text{leader}} = \{\mathbf{f}_{t-1}, \rho_{t-1}, \mathbf{f}_{t-2}, \rho_{t-2}, \ldots, \mathbf{f}_{t-D}, \rho_{t-D}\}, \tag{45}$$

and $\mathbf{f}_t = \sum_{c=1}^{s} f_t^c$. By considering the state space to be the sum of the followers' actions, the leader may not need to exchange as much information with the followers. This is because the leader would only need to consider the followers' combined actions rather than each action. This change could simplify the decision-making process for the leader and improve the system's overall efficiency.

Follower State Space: Any follower agent is assumed to have the strategy of all other agents for previous D time steps at time step l. Besides all the strategies of the other agents, they have the total paid money in time step t as well.

$$S_l^{t,s} = \{\mathbf{f}_{-s}^{l-1}, \mathbf{f}_{-s}^{l-2}, \ldots, \mathbf{f}_{-s}^{l-D}, \rho^t\}. \tag{46}$$

Leader action Space: The action space of the leader is continuous and bounds in $\rho \in (\rho_{\min}, \infty)$ where $\rho_{\min}$ is a result of (9b) that ensures the total required computation capability. Using a policy-gradient method to solve the problem allows us to have no upper bound on the value of ρ without negatively affecting the ability to find an optimal solution.

Follower action Space: Each one of the followers has to find the best strategy, $f_s \in (0, f_s^{\max})$. All the agents of the followers are assumed to have continuous action space.

2) Reward Functions: After applying any action in each state, the environment will return a specific value to reward that state and action. In our problem, both the followers and the leader have their utility as the reward function. Hence, we will have:

$$R_{\text{leader}} = U_l(\rho). \tag{47}$$

$$R_{\text{follower}} = U_s(f_s, f_{-s}). \tag{48}$$

3) Training Process: In this method, we divide the total training period as depicted in Fig. 2. The leader trains based on the information of CNs and the users, then attempts to maximize the utility in T time steps for any request summation of the users, defined as $\sum_{x\in\mathcal{M}}\beta_m$ and $\sum_{x\in\mathcal{M}}\beta_m/\tau_m$. To clarify, Firstly, the leader gets β_m and τ_m from all m users and starts a training process for these parameters. Then, gather the CNs' contribution, f_s, from all s CNs in any time step. Secondly, finds

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Training algorithm based on **SAC**.

Initialize $\sum_{x\in\mathcal{M}}\beta_x$, $\sum_{x\in\mathcal{M}}\beta_x/\tau_x$, η, κ, μ, θ_{leader}, ϕ^1_{leader}, ϕ^2_{leader}, empty replay buffer
Set target parameters equal to main parameters, $\phi^{1,\text{target}}_{\text{leader}} \leftarrow \phi^1_{\text{leader}}$, $\phi^{2,\text{target}}_{\text{leader}} \leftarrow \phi^2_{\text{leader}}$
for $t = 0, 1, 2, \ldots, T$ **do**
  for CN $s \in \mathcal{S}$ **do**
    Initialize θ^s_t, $f^{\max}_s$, ρ^{cmp}_s, ϕ^1_s, ϕ^2_s
    Set target parameters equal to main parameters, $\phi^{1,\text{target}}_s \leftarrow \phi^1_s$, $\phi^{2,\text{target}}_s \leftarrow \phi^2_s$
  end for
  for $l = 0, 1, 2, \ldots, L$ **do**
    for CN $s \in \mathcal{S}$ **do**
      Collect set of partial trajectories and rewards on policy $\pi(f^s_l|S^{t,s}_l, \theta^s_l)$
      Store trajectories and rewards in the replay buffer
      if $l \mod 20 == 0$ **then**
        for j in range of updates per step **do**
          randomly sample batch of transitions from replay buffer
          Compute targets for the Q-functions (59)
          Update Q-function by one step of gradient descent $\nabla_{\phi_i} L(\phi_i, \mathcal{B})$; $for\ i = 1, 2$
          Update policy by one step of gradient descent on objective of (62)
          Update target networks $\phi^i, \text{target}_s \leftarrow \tau\phi^{i,\text{target}}_s + (1-\tau)\phi^i_s$; $for\ i = 1, 2$
        end for
      end if
    end for
  end for
  if $t \mod 20 == 0$ **then**
    for j in range of updates per step **do**
      randomly sample batch of transitions from replay buffer
      Compute targets for the Q-functions (59)
      Update Q-function by one step of gradient descent $\nabla_{\phi_i} L(\phi_i, \mathcal{B})$; $for\ \ i = 1, 2$
      Update policy by one step of gradient descent on objective of (62)
      Update target networks $\phi^i, \text{target}_{\text{leader}} \leftarrow \tau\phi^i, \text{target}_{\text{leader}} + (1-\tau)\phi^i_{\text{leader}}$; $for\ i = 1, 2$
    end for
  end if
end for

out the state S^{leader}_t, and based on that, takes action ρ_t and gets the reward R_{leader}.

On the other hand, the followers are agents who maximize their utility in the L time steps. They will face off each other, get the total money ρ_t from the leader, and the actions of the other CNs in previous D time steps, then form the state $S^{t,s}_l$. Based on the current state, they take action f_s and get the reward R_{follower} until they converge to the Nash Equilibrium. At first, all the agents generate the states with random values. So we can see in Fig. 2 that each training period of the CNs is a time step for the

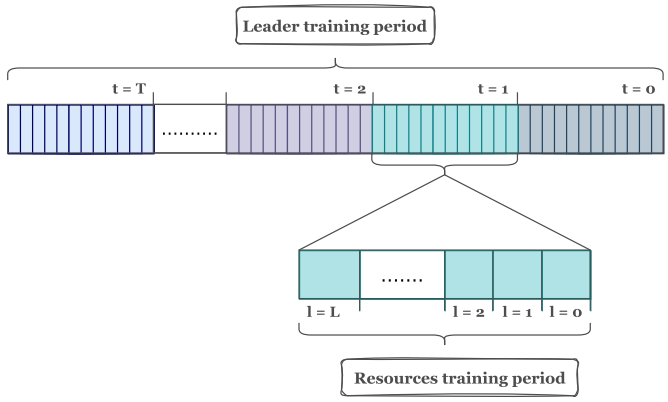

Fig. 2. Incomplete information training process.

leader. We can define the policy of the followers and the leader as below:

$$\begin{cases} \text{leader policy} \Rightarrow \pi(\rho_t|S^{\text{leader}}_t) \\ \text{followers policy} \Rightarrow \pi(f^s_l|S^{t,s}_l) \end{cases} \tag{49}$$

In our study, each agent trained with the SAC algorithm, a model-free and off-policy technique that is very stable and achieves very similar performance across different random seeds. This method has better results in continuous action and state spaces than the other methods and forms a bridge between stochastic policy optimization and DDPG-style methods. Entropy is a measure of randomness in the policy, and SAC tries to maximize a trade-off between entropy and expected return. It is often advantageous for an agent to try out different actions and states in order to acquire more information and enhance its understanding of the environment. This can be especially useful in situations where the agent's understanding of the environment is incomplete or uncertain. By exploring different actions and states, the agent can gain insights that can help it to make better decisions and improve its performance in the long run. We can define the entropy for the random variable x and density function P as below:

$$\mathcal{H}(P) = \underset{x\sim P}{E}\left[-\log P(x)\right]. \tag{50}$$

Hence, the optimization objective of the SAC will be as below:

$$\pi^\star = arg\max_{\pi} \underset{\tau\sim\pi}{E}\left[\sum_{t=0}^{\infty}\gamma^t\Big(R(S_t, A_t, S_{t+1}) + \alpha\mathcal{H}(\pi(.|S_t))\Big)\right], \tag{51}$$

$\alpha > 0$ is the temperature used to balance the trade-off between entropy and expected return. This value can be constant or automatically tuned, and the differences between both conditions will be depicted in the next section. This method uses the bonuses of the entropy in the definition of the value and Q-functions as below:

$$V^\pi(S) = \underset{\tau\sim\pi}{E}\left[\sum_{t=0}^{\infty}\gamma^t\Big(R(S_t, A_t, S_{t+1}) + \alpha\mathcal{H}(\pi(.|S_t))\Big)\,\middle|\, S_0 = S\right]. \tag{52}$$

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

$$Q^{\pi}(S,A)=\underset{\tau\sim\pi}{E}\Bigg[\sum_{t=0}^{\infty}\gamma^{t}R(S_t,A_t,S_{t+1})+\alpha\sum_{t=1}^{\infty}\gamma^{t}\mathcal{H}(\pi(.|S_t))\Bigg|S_0=S,A_0=A\Bigg]. \tag{53}$$

As we can see, the Q-function includes the entropy bonus except for the first timestep. Based on the definition of the Q-function and the value function, we will have:

$$V^{\pi}(S)=\underset{A\sim\pi}{E}\left[Q^{\pi}(S,A)+\alpha\mathcal{H}(\pi(.|S))\right]. \tag{54}$$

$$Q^{\pi}(S,A)=\underset{S'\sim P}{E}\left[R(S,A,S')+\gamma V^{\pi}(S')\right]. \tag{55}$$

By using the definition of entropy, our Bellman equation will be:

$$Q^{\pi}(S,A)=\underset{\substack{S'\sim P\\ A\sim\pi}}{E}\left[R(S,A,S')+\gamma(Q^{\pi}(S',A')-\alpha\log\pi(A'|s'))\right]. \tag{56}$$

Now we can estimate the $Q^{\pi}(S,A)$ using the replay buffer data. But inside the replay buffer, we do not have information about the next actions; hence, we will use fresh samples of the policy and name them as $\hat{A}'$. In the below formulation, R and S' are the replay buffer samples. we have:

$$Q^{\pi}(S,A)\approx R+\gamma(Q^{\pi}(S',\hat{A}')-\alpha\log\pi(\hat{A}'|s')). \tag{57}$$

SAC uses the clipped double-Q trick and chooses the minimum of one of the calculated Q functions. Thus, the loss function of the Q-networks is:

$$L(\phi_i,\mathcal{B})=\underset{(S,A,R,S',d)\sim\mathcal{B}}{E}\left[\left(Q_{\phi_i}(S,A)-y(R,S',d)\right)^2\right]. \tag{58}$$

And the target is given by:

$$y(R,S',d)=R+\gamma(1-d)\Big(\min_{j=1,2}Q_{\phi_{\text{target},j}}(S',\hat{A}')-\alpha\log\pi(\hat{A}'|s')\Big). \tag{59}$$

On the other hand, to optimize the policy, the reparameterization trick is used as below:

$$\hat{a}_{\theta}(S,\xi)=\tanh(\mu_{\theta}(S)+\sigma_{\theta}(S)\odot\xi),\quad \xi=\mathcal{N}(0,1). \tag{60}$$

This method uses squashed Gaussian policy, in which the samples are drawn from $\pi_{\theta}(.|S)$ using the state, policy parameters, and independent noise defined as ξ. The resulting equation allows us to rewrite the (54) over actions and noise without dependency on policy parameters. We will have:

$$\underset{\xi\sim\mathcal{N}}{E}\left[Q^{\pi_{\theta}}(S,\tilde{a}_{\theta}(S,\xi))-\alpha\log\pi_{\theta}(\tilde{a}_{\theta}(S,\xi)|S)\right]. \tag{61}$$

In the end, we will use one of the approximations of the Q-functions, and the policy is going to be optimized as below:

$$\max_{\theta}\underset{\substack{\xi\sim\mathcal{N}(0,1)\\ S\sim\mathcal{D}}}{E}\left[\min_{j=1,2}Q_{\phi_j}(S,\tilde{a}_{\theta}(S,\xi))-\alpha\log\pi_{\theta}(\tilde{a}_{\theta}(S,\xi)|S)\right]. \tag{62}$$

This objective is almost the same as the DDPG and TD3, with differences in the minimum factor for the Q-functions, the stochasticity, and the entropy term. We will use Algorithm.1 to train the followers and the leader based on the training process and the equations. It is proven in [15] that SAC converges to the optimal policy, i.e., the Nash and Stackelberg equilibriums of our problem. Also, empirical studies in [15] show the improved performance of SAC compared to both off-policy and on-policy prior methods.

SAC uses actor-critic as depicted in Fig. 3. This method is a combination of value-based and policy-based approaches. In this structure, the actor is responsible for the policy gradient approach, and the critic is learned through the value-based process. As Fig. 3 clearly shows, each one of the followers and the leader uses the same structure (SAC) for the learning process. Based on the incomplete information assumption, we could not consider the shared critic used in the MADDPG [39]. Hence, we considered that each agent, follower or leader, has a unique environment, replay buffer, and actor-critic networks as used for the PPO agents in [10] with the difference of using SAC agents. Thus, our designed SAC-based training process uses a unique structure for each agent and trains them based on Algorithm.1.In this algorithm, we first initialize the leader training parameters and start the training process for the leader. Inside each training process of the leader, we initiate the parameters of the followers the same as the leader and start the followers' training process since they converge to the Nash equilibrium; then, based on the follower's contribution, we update the leader for one timestep. Then continue this process until the leader converges.

VI. Numerical Results

A. Simulation Setup

This section conducts simulations to evaluate our strategies using PyTorch 1.4.0 and OpenAI Gym to simulate the reinforcement learning environment. We used a simulated reinforcement learning environment, including the leader and the followers. The followers start the training using the randomly generated state space. After the convergence, the leader trains one episode using the randomly generated state space and updates that in every episode based on the followers' contribution value. In the end, the leader converges, the task parameters change, and the process starts for those parameters; hence the learning process continues to the wholly trained point. The parameters of the DRL agent are selected through fine-tuning. The actor and the critic networks in the SAC algorithm have two fully-connected layers. The first layer has 250 nodes, and similarly, the second layer has 250 nodes too. We have $D=10$ for the previous time step information in our leader and follower state spaces and assumed three CNs as the followers and one leader in our simulations. The assigned parameters are in Table II, and any simulation result with non-clarified values is using this table's assigned values.

As stated in [15], we use the hyperparameters the same as the SAC(hard target update) and the gradient steps used for the humanoid environment for our followers' SAC agents. Hence, the learning rate is 3×10^{-4}, and gamma equals 0.99, the replay buffer size is 10^6, the target smoothing coefficient is equal to 1, the target updating interval equals 1000, and the gradient steps is 1. In all the simulations except the entropy tuning comparison,

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

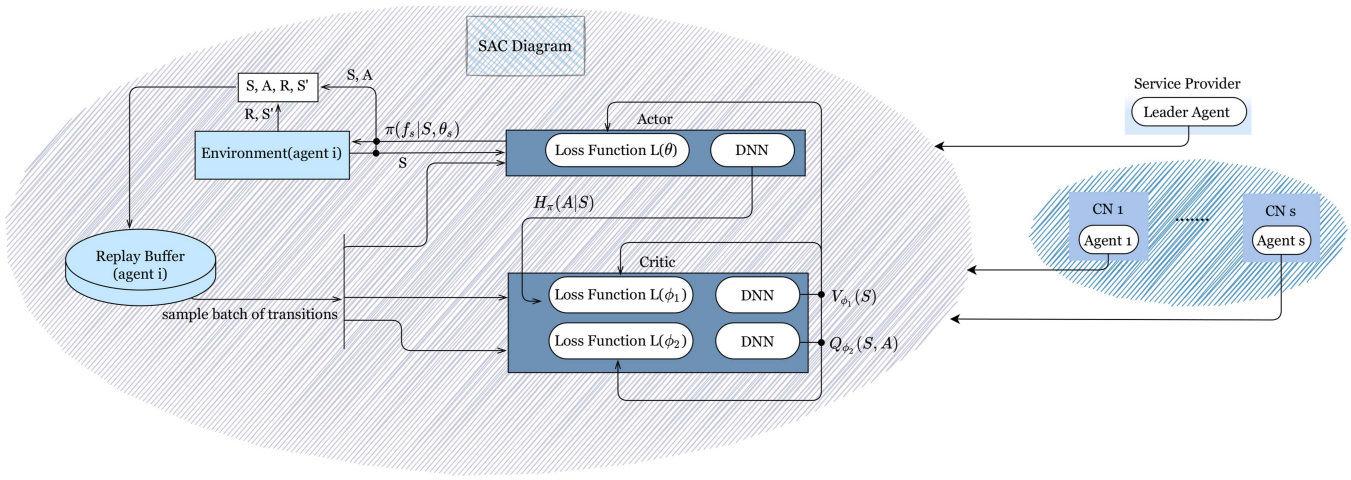

Fig. 3. SAC agent structure.

TABLE II
NUMERICAL VALUES

parameter	value	parameter	value
ρ_1^{cmp}	2×10^{-6}	ρ	10
ρ_2^{cmp}	1.8×10^{-6}	μ	1.1
ρ_3^{cmp}	2×10^{-6}	κ	1×10^{-27}
$f_1^{\max}$	3.5×10^6	η	0.7×10^7
$f_2^{\max}$	4×10^6	$f_3^{\max}$	3×10^6

we use the entropy tuning property of the SAC. On the other hand, the leader has a SAC agent with differences in learning rate that equals 1.2×10^{-4}, and the target smoothing coefficient that equals 1×10^{-2}.

B. Simulation Results

First, we show the results of the convergences to the Nash equilibriums. To depict the convergence of the followers, we used three different scenarios in Figs. 4, 5, and 6, and all of them clearly show that the followers are converging to the Nash equilibrium. In our simulations, we applied three agents to represent the computation nodes (CNs) in the system. However, our proposed solution can be easily extended to a larger number of CNs without significant modifications. The use of three agents in our simulations is primarily for presentational purposes, as it allowed us to clearly illustrate the behavior of the CNs and the interaction between them. In Fig. 4, we assumed that the followers have identical maximum CPU frequency, and we set $f_s^{\max} = 4 \times 10^6$ for all followers and made the difference by setting $\rho_1^{\text{cmp}} = 1.6 \times 10^{-6}$, $\rho_2^{\text{cmp}} = 1.8 \times 10^{-6}$, and $\rho_3^{\text{cmp}} = 2 \times 10^{-6}$. It clearly shows that the lower computation cost incentives the CNs to have more contributions. For Fig. 5, we assumed all the followers have constant computation cost and set the maximum CPU frequency as $f_0^{\max} = 3 \times 10^6$, $f_1^{\max} = 3.5 \times 10^6$, and $f_2^{\max} = 4 \times 10^6$. Based on the paid bonus, a

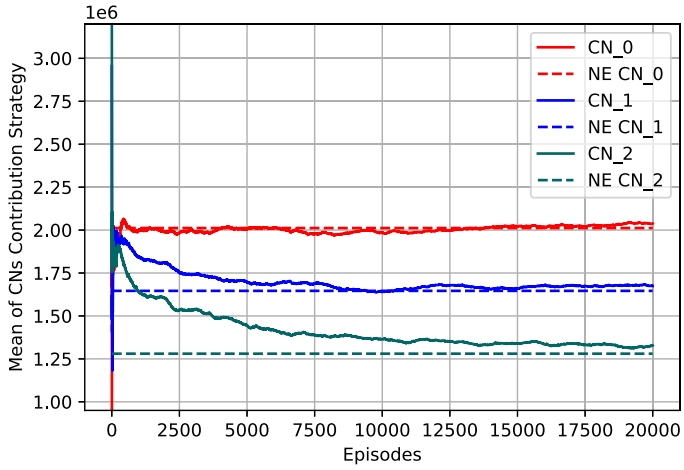

Fig. 4. Mean of followers strategy with identical maximum computation capacities and distinct computation costs.

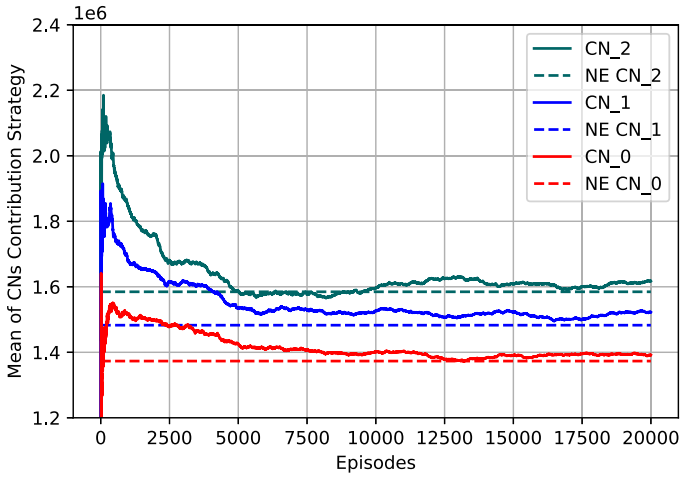

Fig. 5. Mean of followers strategy with distinct maximum computation capacities and identical computation costs.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

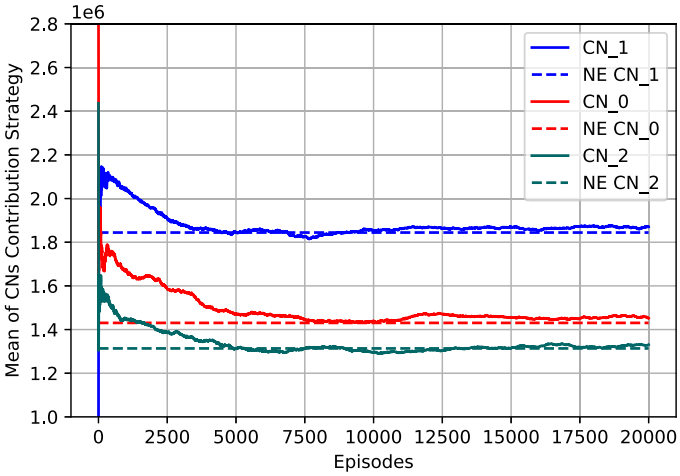

Fig. 6. Mean of followers strategy based on Table II values and entropy tuning.

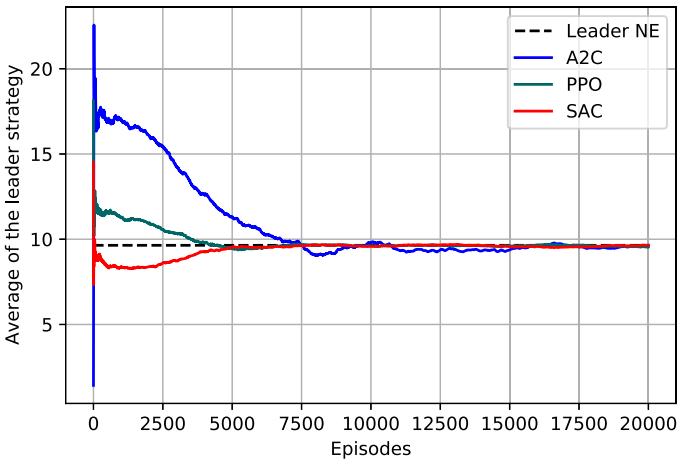

Fig. 7. Mean of the Leader strategy using SAC, PPO, and A2C.

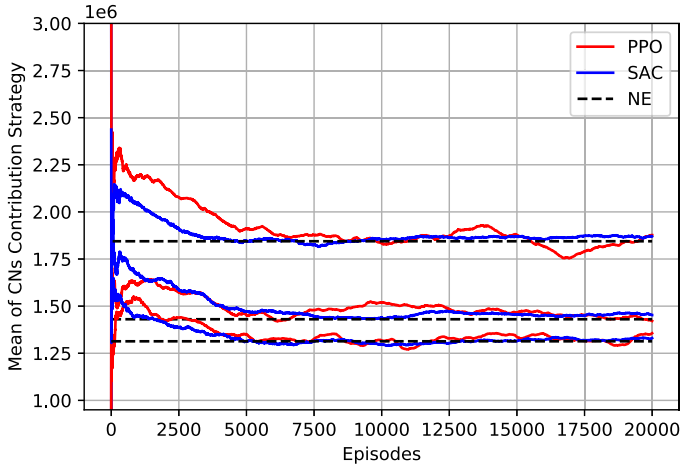

Fig. 8. Followers strategy comparison using PPO and SAC.

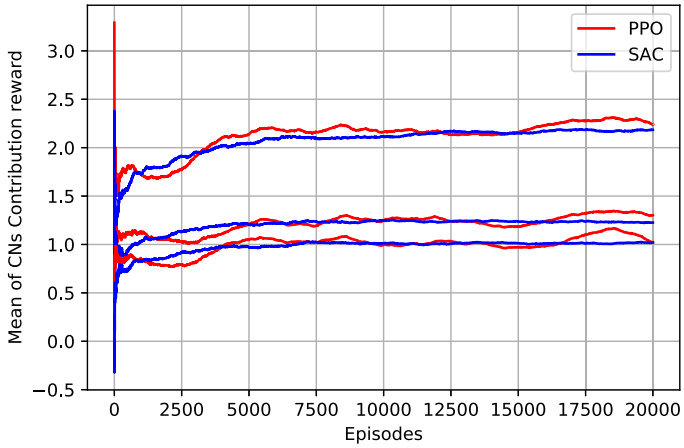

Fig. 9. Followers reward comparison using PPO and SAC.

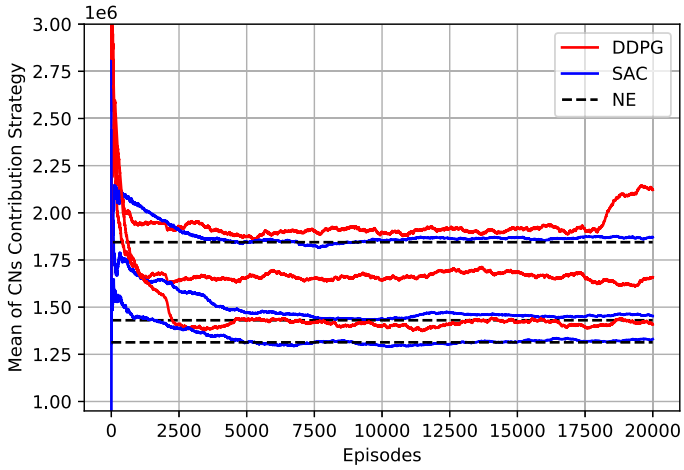

Fig. 10. Followers strategy comparison using DDPG and SAC.

higher maximum CPU frequency leads to a higher contribution, which is clearly depicted. As mentioned in the previous section, Table II parameters are used in Fig. 6 simulation.

The Fig. 7 delineates the leader strategy convergence and compares the SAC method with two other methods named PPO [11] and A2C [13]. As we can see, A2C does not have good convergence, and the variance of the actions after the convergence is more considerable than the SAC and PPO. On the other hand, SAC has a highly non-comparable performance contrasted to the PPO and converges after about 5000 episodes. Still, PPO converges after about 6000 episodes, and their action variances are incredibly close. It means that SAC does not significantly show better performance, but our comparison based on the followers' strategy in Fig. 8 depicts that both PPO and SAC have somehow near convergence performance, but SAC shows better performance based on the lower action variance and is more stable compared to PPO. This property shows that the SAC is more reliable and stable than the A2C and PPO. This comparison continued using the followers' reward in the Fig. 9. It is crystal clear that same, as the previous comparisons, SAC shows better results. In Figs. 10 and 11, we compared SAC with DDPG [14] as two different off-policy methods. Fig. 10 depicts that SAC performs better in convergence and stability and accurately converges to the Nash equilibrium, and Fig. 11 shows that SAC reaches higher rewards than the DDPG, which makes it more effective in incentivizing the followers.

In the Figs. 12 and 13, we attempt to show the effect of two hyperparameters. Fig. 12 compares the CN-1's contribution strategy based on three learning rates. Clearly shown that

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

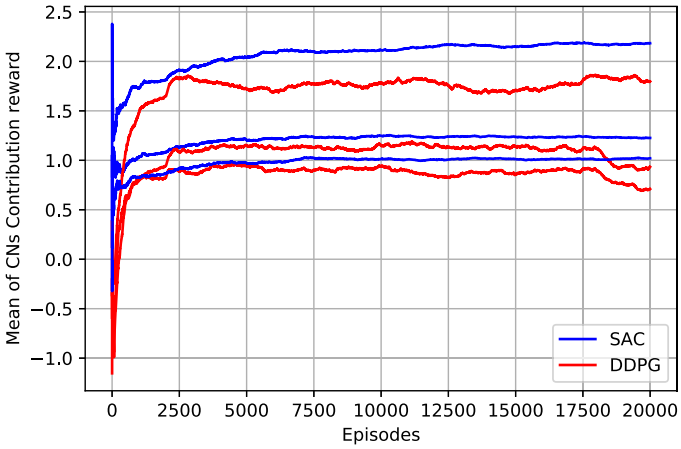

Fig. 11. Followers reward comparison using DDPG and SAC.

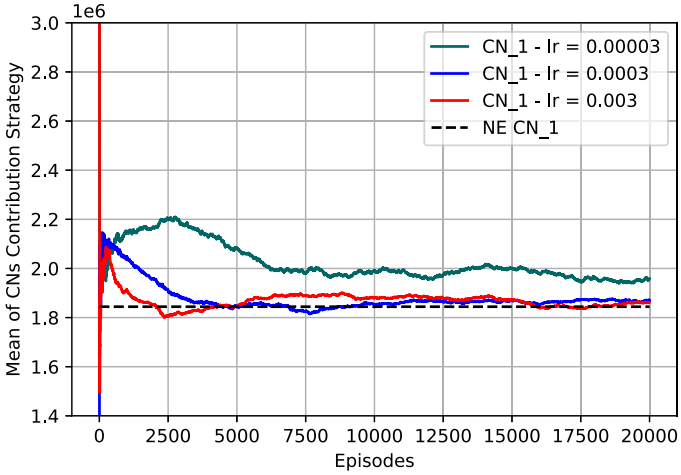

Fig. 12. Mean of CN1 strategy based on three different learning rates.

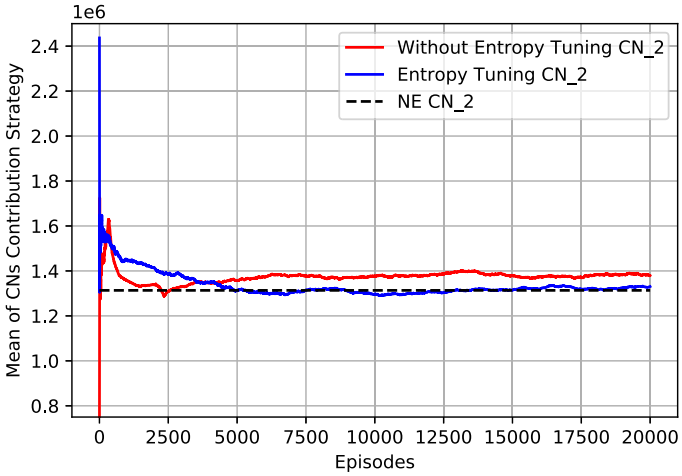

Fig. 13. Mean of CN2 strategy based on different entropy tuning strategies.

3×10^{-4} has better performance than the one then times more significant and ten times more minor. Based on this comparison, we chose the learning rate. Fig. 13 shows the difference of the entropy tuning and constant α used in (50). The entropy tuning has a better performance than the constant α. Thus, we used

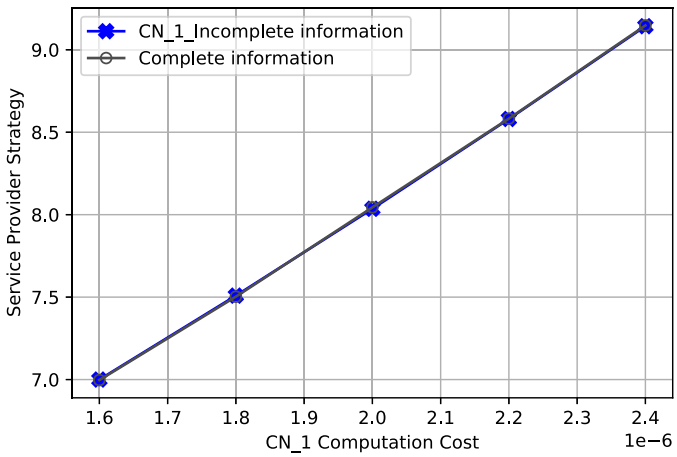

Fig. 14. Service provider strategy based on the CN1 computation cost alteration.

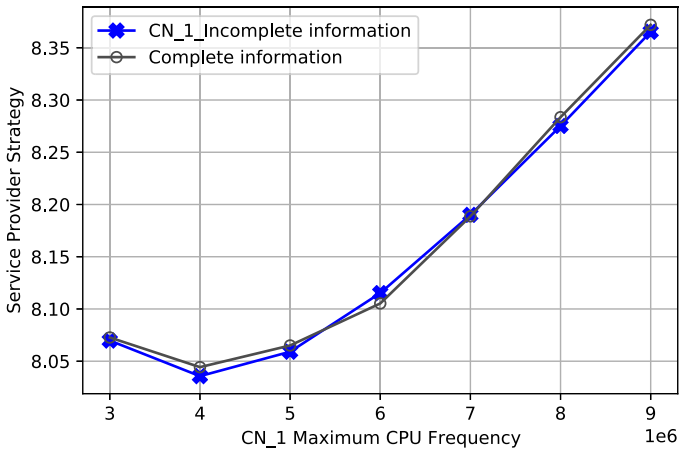

Fig. 15. Service provider strategy based on the CN1 maximum CPU frequency alteration.

entropy tuning in all of our simulations, but as it is clear from Fig. 13, using entropy information ($\alpha = 0.2$) outperforms solely expected return ($\alpha = 0$) settings.

Based on the previous comparisons, we have shown that the leader and the followers can converge to the Nash equilibrium. In addition, the impact of one of the follower's parameters on the leader's Nash equilibrium is depicted in Figs. 14 and 15. Fig. 14 shows the impact of a follower's computation cost on the leader's strategy and shows that higher computation cost results in larger payment from the service provider; this is because of the total sufficient CPU frequency that the service provider should obtain for the users. On the other hand, Fig. 15 shows the impact of a follower's maximum CPU frequency on the leader's strategy if the follower's maximum CPU frequency increases; as a result, the total paid bonus and the total payment will increase. In the end, if we compare both Figs. 14 and 15, we will see that the impact of the computation cost on the leader payment is more than the maximum CPU frequency. This can be an exact difference between the bonus and the cost imposed due to processing that should be more than the bonus.

Now we will check the impact of the followers' parameters on each other's strategies. So we have Fig. 16 that depicts the

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

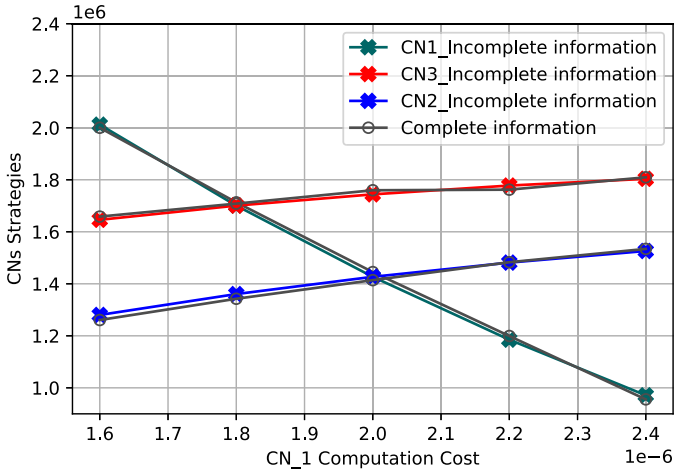

Fig. 16. Followers strategy based on the CN1 computation cost alteration.

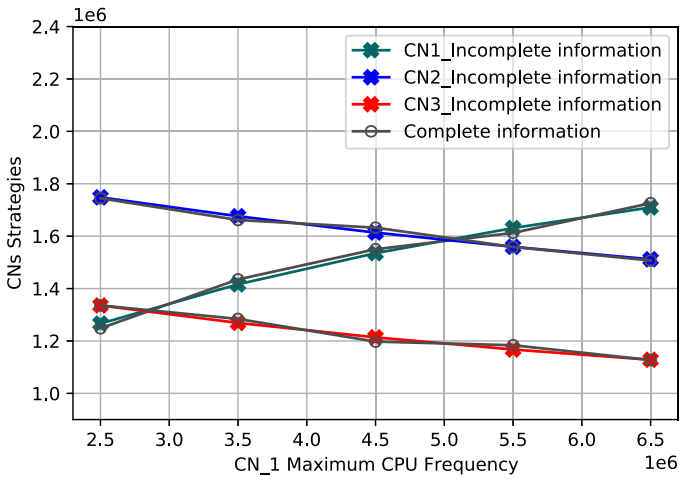

Fig. 17. Followers strategy based on the CN1 maximum CPU frequency alteration.

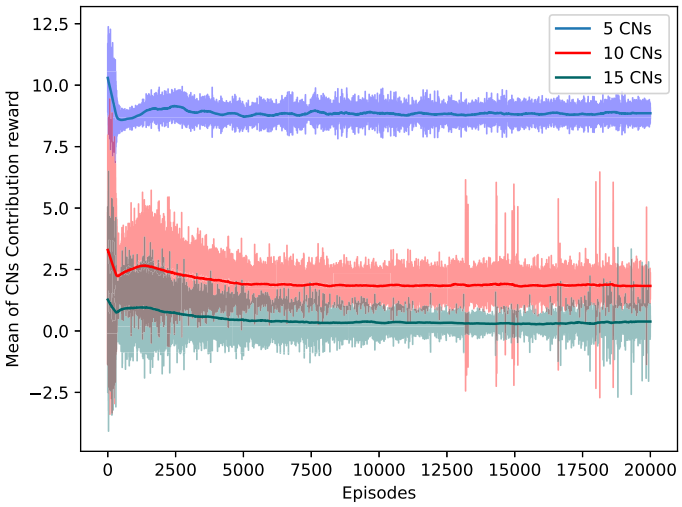

Fig. 18. Convergence of the proposed solution to the equilibrium with different numbers of the CNs.

impact of the CN-1's computation cost on all the followers' strategies; as we can see, if we increase the CN-1's computation cost, the CN-1's strategy will decrease, and the others will increase their strategy. In this simulation, we set $\rho_2^{\text{cmp}} = 2 \times 10^{-6}$, $\rho_3^{\text{cmp}} = 1.8 \times 10^{-6}$, and $f_s^{\max} = 4 \times 10^6$ for all the followers. On the other hand, Fig. 17 depicts the CN-1's maximum CPU frequency impact on all the followers' strategies. A larger maximum CPU frequency will result in more bonuses, and the CN-1's strategy will increase, which will decrease the other followers' bonus, and their strategy will decrease. In this simulation, we set $\rho_2^{\text{cmp}} = 2 \times 10^{-6}$, $\rho_3^{\text{cmp}} = 2.1 \times 10^{-6}$, $f_2^{\max} = 5 \times 10^6$, and $f_3^{\max} = 3.5 \times 10^6$.

We should note that our game model discussed in this article lays in the category of aggregative games [46], [47]. Therefore, the performance of the solution is not violated by increasing the number of players (and thus the CNs in our problem). Fig. 18 illustrates the convergence of our method for the number of CNs equal to 5, 10, and 15, while all the other parameters for the simulations are the same. The plots in this figure demonstrate the effectiveness of our approach by increasing the number of CNs.

VII. Conclusion and Future Work

This article studied an incentive mechanism for computation offloading, considering the CNs' computation costs and motivating them by paying a contribution assessment based on their shared CPU frequency and a bonus based on their maximum contribution capability. We modeled this incentive mechanism as a two-layer game considering a leader responsible for QoS guarantee and green computing awareness. This problem was analyzed and formulated using the Stackelberg game for complete information assumptions. We investigated the existence and uniqueness of the Nash Equilibrium and Stackelberg Equilibrium and provided proof for them. Due to privacy conditions, we proposed an incomplete information method based on deep reinforcement learning; this method could converge to the same Nash Equilibriums proven under complete information assumptions. Finally, the numerical results showed that the proposed SAC-based method performed better than previous algorithms and converged to the Nash Equilibriums under incomplete information assumptions. We note that an interesting issue is to deploy a reinforcement learning model, trained for incentivization of computation nodes in an edge computing environment, to another edge computing environment, and it would work well in some scenarios. However, for improving and guaranteeing the model's performance in the new environment, it might be necessary to employ transfer learning or model-based RL approaches [42], [43], [44], [45].

References

[1] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge computing enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10200–10232, Oct. 2020.

[2] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Netw. Appl.*, vol. 25, pp. 1–24, 2020.

[3] A. Maimone, A. Georgiou, and J. S. Kollin, "Holographic near-eye displays for virtual and augmented reality," *ACM Trans. Graph.*, vol. 36, no. 4, 2017, Art. no. 85.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.

[4] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit maximization incentive mechanism for resource providers in mobile edge computing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 138–149, Jan./Feb. 2022.

[5] Q. Wang, S. Guo, Y. Wang, and Y. Yang, "Incentive mechanism for edge cloud profit maximization in mobile edge computing," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[6] S. M. A. Kazmi et al., "A novel contract theory-based incentive mechanism for cooperative task-offloading in electrical vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8380–8395, Jul. 2022.

[7] X. Kang and S. Sun, "Incentive mechanism design for mobile data offloading in heterogeneous networks," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 7731–7736.

[8] X. Lin, J. Wu, J. Li, X. Zheng, and G. Li, "Friend-as-learner: Socially driven trustworthy and efficient wireless federated edge learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 269–283, Jan. 2023.

[9] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.

[10] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jun. 2020.

[11] J. Schulman et al., "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[12] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P Abbeel, "Trust region policy optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[13] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[14] T. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Repesentations*, 2016.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018.

[16] A. Taghizadeh, M. Montazeri, and H. Kebriaei, "Deep reinforcement learning-aided bidding strategies for transactive energy market," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4445–4453, Sep. 2022.

[17] H. Shah-Mansouri, V. W. S. Wong, and J. Huang, "An incentive framework for mobile data offloading market under price competition," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 2983–2999, Nov. 2017.

[18] Y. Li, B. Shen, J. Zhang, X. Gan, J. Wang, and X. Wang, "Offloading in HCNs: Congestion-aware network selection and user incentive design," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6479–6492, Oct. 2017.

[19] X. Zhuo, W. Gao, G. Cao, and S. Hua, "An incentive framework for cellular traffic offloading," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 541–555, Mar. 2014.

[20] X. Kang, Y. -K. Chia, and S. Sun, "Mobile data offloading through a third-party WiFi access point: An operator's perspective," in *Proc. IEEE Globecom Workshops*, 2013, pp. 696–701.

[21] H. Xu, X. Qiu, W. Zhang, K. Liu, S. Liu, and W. Chen, "Privacy-preserving incentive mechanism for multi-leader multi-follower IoT-edge computing market: A reinforcement learning approach," *J. Syst. Archit*, vol. 114, 2020, Art. no. 101932.

[22] M. Diamanti, P. Charatsaris, E. E. Tsiropoulou, and S. Papavassiliou, "Incentive mechanism and resource allocation for edge-fog networks driven by multi-dimensional contract and game theories," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 435–452, 2022.

[23] W. Zhan et al., "Deep-reinforcement-Learning-Based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.

[24] Z. Li, M. Xu, J. Nie, J. Kang, W. Chen, and S. Xie, "NOMA-Enabled cooperative computation offloading for blockchain-empowered Internet of Things: A. learning approach," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2364–2378, Feb. 2021.

[25] Y. Chen, S. Zhang, M. Xiao, Z. Qian, J. Wu, and S. Lu, "Multi-user edge-assisted video analytics task offloading game based on deep reinforcement learning," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst.*, 2020, pp. 266–273.

[26] D. Zhu et al., "Speed-aware and customized task offloading and resource allocation in mobile edge computing," *IEEE Commun. Lett.*, vol. 25, no. 8, pp. 2683–2687, Aug. 2021.

[27] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019, doi: 10.1109/JPROC.2019.2922285.

[28] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A. survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019, doi: 10.1109/ACCESS.2019.2938660.

[29] W. Chen, X. Qiu, T. Cai, H. -N. Dai, Z. Zheng, and Y. Zhang, "Deep reinforcement learning for Internet of Things: A. comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1659–1692, thirdquarter 2021, doi: 10.1109/COMST.2021.3073036.

[30] F. Fu, Y. Kang, Z. Zhang, F. R. Yu, and T. Wu, "Soft actor–critic DRL for live transcoding and streaming in vehicular fog-computing-enabled IoV," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1308–1321, Feb. 2021.

[31] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.

[32] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. M. Leung, "Cooperative computation offloading for multi-access edge computing in 6G mobile networks via soft actor critic," *IEEE Trans. Netw. Sci. Eng.*, early access, Apr. 30, 2021, doi: 10.1109/TNSE.2021.3076795.

[33] 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, no. 7, pp. 273–287, 2020.

[34] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook, 2017, *arXiv:1701.01090*.

[35] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspectivep," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Oct.–Dec. 2017, doi: 10.1109/COMST.2017.2745201.

[36] J. B. Rosen, "Existence and uniqueness of equilibrium points for concave n-person games," *Econometrica: J. Econometric Soc.*, vol. 33, pp. 520–534, 1965.

[37] M. Marcus and H. Minc, *A. Survey of Matrix Theory and Matrix Inequalities*, vol. 14. Chelmsford, MA, USA: Courier Corporation, 1992.

[38] A. Taghizadeh, H. Kebriaei, and D. Niyato, "Mean field game for equilibrium analysis of mining computational power in blockchains," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7625–7635, Aug. 2020.

[39] R. Lowe, Y. Wu, A. Tamar, J. Harb, O.P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[40] C. Shu, Z. Zhao, Y. Han, and G. Min, "Dependency-aware and latency-optimal computation offloading for multi-user edge computing networks," in *Proc. IEEE 16th Annu. Int. Conf. Sens., Commun., Netw.*, 2019, pp. 1–9.

[41] Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami, and M. Shamim Hossain, "Intelligent task prediction and computation offloading based on mobile-edge cloud computing," *Future Gener. Comput. Syst.*, vol. 102, pp. 925–931, 2020.

[42] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.

[43] Z. Xu et al., "An actor-critic-based transfer learning framework for experience-driven networking," in *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 360–371, Feb. 2021, doi: 10.1109/TNET.2020.3037231.

[44] I. Higgins et al., "DARLA: Improving zero-shot transfer in reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1480–1490.

[45] C.R. Dance, J. Perez, and T.. Cachet, "Demonstration-conditioned reinforcement learning for few-shot imitation," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 2376–2387.

[46] F. Fabiani, M. A. Tajeddini, H. Kebriaei, and S. Grammatico, "Local stackelberg equilibrium seeking in generalized aggregative games," *IEEE Trans. Autom. Control*, vol. 67, no. 2, pp. 965–970, Feb. 2022, doi: 10.1109/TAC.2021.3077874.

[47] M. Shokri and H. Kebriaei, "Leader–Follower network aggregative game with stochastic agents' communication and activeness," *IEEE Trans. Autom. Control*, vol. 65, no. 12, pp. 5496–5502, Dec. 2020, doi: 10.1109/TAC.2020.2973807.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 24,2024 at 15:11:35 UTC from IEEE Xplore. Restrictions apply.