Task Placement and Resource Allocation in Edge Computing Systems

SLAĐANA JOŠILO

Doctoral Thesis
Stockholm, Sweden 2020

KTH School of Electrical Engineering and Computer Science
TRITA-EECS-AVL-2020:21 SE-100 44 Stockholm
ISBN 978-91-7873-510-5 Sweden

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av doktorsexamen onsdag den 27 May 2020 klockan 14.00 i Kollegiesalen, KTH, Stockholm.

Tryck: Universitetsservice US AB

Abstract

The evolution of wireless and hardware technology has led to the rapid development of a variety of mobile applications. Common to these applications is that they have low latency and high computational requirements that often cannot be fulfilled by individual devices due to their insufficient computational power, memory and battery capacity. An emerging approach to meet increasing user demand for delay sensitive and computationally intensive applications is mobile edge computing. The core paradigm of mobile edge computing is to bring computing and storage resources close to the end users and by doing so to relieve devices from computationally heavy workloads while meeting delay requirements of applications. However, the overall performance of edge computing systems is determined by the efficiency of the joint allocation of wireless and computing resources. The work in this thesis proposes decentralized algorithms for allocating these two resources in edge computing infrastructures.

In the first part of the thesis, we consider the resource allocation and computational task scheduling problem in an edge computing system in which wireless devices can use cloud resources and the resources of each other with the objective to minimize their own perceived response times. We develop a game theoretical model of the problem, prove the existence of equilibrium task allocations and propose an efficient decentralized algorithm that computes an equilibrium based on average system parameters.

In the second part of the thesis, we consider the joint resource allocation and computational task assignment problem in an edge computing system that consists of an edge cloud that can be accessed by devices through multiple wireless links. We model the problem as a strategic game, in which each device aims at minimizing a combination of its response time and energy consumption. We prove the existence of equilibrium task allocations, and use game theoretical tools for designing polynomial time decentralized algorithms with a bounded approximation ratio. We then extend the analysis to a system with periodic tasks, and show that equilibrium task allocations still exist. Furthermore, we propose a polynomial complexity decentralized algorithm and characterize the structure of equilibria computed by the algorithm.

In the third part of the thesis, we consider the joint resource allocation and computational task assignment problem in an edge computing system that consists of multiple edge clouds and wireless links managed by a single network operator. We model the interaction between the operator and devices that aim at minimizing their response times as a Stackelberg game. We express the optimal resource allocation policies in closed form, prove the existence of Stackelberg equilibria and propose an efficient decentralized algorithm with a bounded approximation ratio. Finally, we consider the same edge computing system under network slicing, and based on a game theoretic treatment of the problem we develop an approximation algorithm for assigning tasks to slices and managing the resources across and within slices.

By providing constructive equilibrium existence proofs, the results in this thesis provide low complexity decentralized algorithms for allocating edge computing resources in a variety of edge computing infrastructures.

Sammanfattning

Teknologiska framsteg inom trådlös teknik och hårdvara har lett till en snabb utveckling av en rad olika mobilapplikationer. Mobilapplikationerna behöver ha låg latens och hög beräkningskraft vilket ofta inte kan uppfyllas av enskilda enheter på grund av bland annat bristande minne och batterikapacitet. En växande trend för att möta den ökade efterfrågan på applikationer med höga krav på latens och beräkningsförmåga är så kallad ”mobile edge computing”. Grunden i mobile edge computing är att placera beräknings- och lagringsresurser nära slutanvändarna. Detta befriar enheter från krävande beräkningar och uppgifter samtidigt som de uppfyller applikationernas latenskrav. Den totala prestandan för mobile edge computing bestäms dock av effektiviteten av den gemensamma tilldelningen av trådlösa resurser och beräkningsresurser. Den här avhandlingen föreslår decentraliserade algoritmer för att fördela dessa resurser i infrastruktur för mobile edge computing.

I den första delen av avhandlingen behandlar vi resursfördelning och schemaläggning av beräkningsuppgifter i ett edge computing system, där trådlösa enheter kan använda både molnresurser och varandras resurser för att minimera sina upplevda svarstider. Vi utvecklar en spelteoretisk modell av problemet, bevisar att jämviktsuppgiftstilldelningar finns och föreslår en effektiv decentraliserad algoritm som beräknar en jämvikt baserad på genomsnittliga systemparametrar.

I den andra delen av avhandlingen behandlar vi resursfördelning och tilldelningen av beräkningsuppgifter i ett edge computing system som består av ett edge cloud som kan nås av enheter via flera trådlösa länkar. Vi modellerar problemet som ett strategiskt spel, där varje enhet syftar till att minimera en kombination av dess responstid och energiförbrukning. Vi bevisar att jämviktsuppgiftstilldelningar finns och använder spelteoretiska verktyg för att designa polynomiska decentraliserade tillnärmningsalgoritmer. Vi utvidgar sedan analysen till ett system med periodiska uppgifter och visar att jämviktsuppgiftstilldelningar fortfarande finns. Vidare föreslår vi en decentraliserad algoritm med polynomisk komplexitet och karakteriserar strukturen för jämvikter som beräknas av algoritmen.

I den tredje delen av avhandlingen angriper vi problemet av resursfördelning och tilldelningen av beräkningsuppgifter i ett edge computing system som består av flera trådlösa länkar och flera edge clouds som hanteras av en enda nätoperatör. Vi modellerar interaktionen mellan operatören och enheter som syftar till att minimera deras responstider som ett Stackelberg-spel. Vi formulerar de optimala resursfördelningspolicyerna i sluten form, bevisar att Stackelberg-jämvikter finns och föreslår en effektiv decentraliserad algoritm med ett begränsat tillnärmningsförhållande. Slutligen analyserar vi samma kantberäkningssystem under nätverksskivning, och baserat på en spelteoretisk behandling av problemet utvecklar vi en tillnärmningsalgoritm för att tilldela uppgifter till olika delar och hantera resurserna både mellan och inom delar.

Genom att visa konstruktiva bevis på jämvikt ger resultaten i den här avhandlingen decentraliserade algoritmer med låg komplexitet för fördelning av edge computing resurser i en rad olika infrastrukturer för edge computing.

In loving memory of my mother, Milenija Mihajlović Jošilo,
who gave me so much love and whom I remember every day of my life.

Acknowledgments

I would like to thank my advisor György Dán for his patience, support and encouragement. I enjoyed every moment I spent working with him and I greatly appreciate the time he has taken to share his knowledge with me. His consistent guidance and valuable suggestions at every stage of the research have helped me to complete the thesis successfully. He helped me grow as a researcher and as a person in ways I cannot begin to enumerate, and I will always be grateful for that.

I am thankful to my co-advisor Viktoria Fodor, who involved me in teaching activities and helped me to sharpen my learning and presentation skills. I would also like to thank professor Gunnar Karlsson for giving me the opportunity to become a member of the NSE department.

I am happy to thank all past and present members of the NSE department for providing a friendly environment. In particular, I would like to thank Valentino for believing in me since the first day we met, Emil for being a supportive and caring friend and Peiyue for being always cheerful and fun. I would also like to thank Ezzeldin for initiating interesting conversations, Lamia for sharing her enthusiasm, Qing for her pleasant company during conference travels and Fredrik for helping me with the Swedish translation of the abstract.

I am thankful to all my friends for their love, enthusiasm and help. I especially thank Vanja, who has been my best friend for the last nineteen years and Nikola for being next to me during my good and bad times. I would also like to thank Carlos, Dina and Lars, who have helped me to maintain a balance between my work, sport activities and social life. In particular, I would like to thank Carlos for his patience, company and valuable climbing tips and Dina for being the kindest possible friend.

Finally, I am deeply thankful to my family for their love, encouragement and support. To my father Slobodan, who taught me to appreciate the small things and who inspired me to keep moving towards my goals. To my brother Nenad, who, among many other things, helped awaken my interest in science and mathematics. To my nephew Mihail, my niece Marija and my sister-in-law Dragana, who make nice memories every time I go home.

Contents

Contents **vii**

1 Introduction **1**
1.1 Background 1
1.2 Challenges 2
1.3 Thesis Structure 2

2 Edge Computing Infrastructure and Resources **3**
2.1 Communication Resources 4
2.2 Computing Resources 6
2.3 Network Slicing 7

3 Computational Tasks in Edge Computing Systems **9**
3.1 Computational Task Modeling 9
3.2 Performance Metrics 9
3.3 Cost Model 12
3.4 Joint Task Placement and Resource Allocation Problem 12

4 Task Placement and Allocation of Edge Resources **15**
4.1 Allocation of Communication Resources 17
4.2 Allocation of Computing Resources 20
4.3 Allocation of Communication and Computing Resources 27

5 Summary of Original Work **37**

6 Conclusion and Future Work **43**

Bibliography **45**

Paper A: Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems **55**

Paper B: Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks **89**

Paper C: Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing **123**

Paper D: Joint Management of Wireless and Computing Resources for Computation Offloading in Mobile Edge Clouds **163**

Paper E: Joint Wireless and Edge Computing Resource Management with Dynamic Network Slice Selection **201**

Chapter 1

Introduction

1.1 Background

The number of wireless devices including smartphones, tablets, sensors, and other portable devices, has been rapidly increasing over the past years. According to a recent estimate by Cisco, the number of mobile devices and connections will grow to 13.1 billion by 2023 at a compound annual growth rate (CAGR) of 8% between 2018 and 2023 [Cis20]. At the same time, this widespread availability of affordable wireless devices has given rise to a variety of Internet of Things (IoT) applications such as video surveillance, tracking, healthcare monitoring and autonomous vehicles [Cis20; Cis17].

Despite steady improvement in the capabilities of the hardware components (e.g., computing units (CPU, GPU, NPU), battery and memory), the individual wireless devices are still not capable of fully supporting the requirements of the emerging computationally intensive and delay sensitive applications [Kum+13; Sat15]. This is due to several factors, including the following two main constraints. The first constraint is related to the fact that battery technology has still not been able to meet energy consumption requirements without limiting the clock speed of processors; doubling the clock speed approximately octuples the energy consumption [KL10b]. The second constraint is related to the users' requirements for light and small devices, which puts additional limitations on the achievable capacity of wireless devices' hardware components [SLS03].

A promising approach to closing the gap between the limited computing capabilities of wireless devices and high computing requirements of the emerging applications is computation offloading [Cue+10; WZL12]. Computation offloading was first introduced as a promising paradigm for augmenting the computing capabilities of devices by allowing them to use remote cloud servers for performing their computational tasks. Computation offloading to remote cloud servers may indeed accelerate the execution of computational tasks, but due to high communication delays it may not be able to meet the requirements of latency sensitive applica-

tions. In order to meet the extremely low latency requirements of emerging delay sensitive applications, novel paradigms of computation offloading propose bringing computing and storage resources closer to the end users, that is, to the network edge. The key idea of these novel paradigms is to build edge computing systems by equipping the existing mobile network components (e.g., base stations and wireless access points) with computing resources and by allowing the end users' devices to perform the computational tasks of each other in a collaborative way [ETS; Hu+15; CZ16; Bon+14].

1.2 Challenges

Edge computing systems would likely consist of heterogeneous communication and computing resources and heterogeneous wireless devices (e.g., devices that differ in terms of computing capabilities, battery states and types of computational tasks). In this context, there are three major problems that need to be addressed before deploying computation offloading paradigms. First, there is a need for efficient task placement algorithms that will take into consideration diverse characteristics of computational tasks and heterogeneity of communication and computing resources. Second, there is a need for efficient management of heterogeneous communication resources that will be used for data exchange between the end users' devices and the external computing resources. Finally, there is a need for efficient management of heterogeneous computing resources that will be used for executing the computational tasks offloaded by diverse wireless devices. These three problems need to be addressed jointly, which is an inherently challenging task in highly heterogeneous edge computing systems.

1.3 Thesis Structure

The structure of this thesis is as follows. In Chapter 2, we present a general model of an edge computing infrastructure and we discuss the characteristics of the constituent communication and computing resources. In Chapter 3, we discuss different models of computational tasks, introduce performance metrics for evaluating edge computing systems and provide a general formulation of the joint task placement and resource allocation problem. In Chapter 4, we categorize works on task placement and resource allocation in edge computing systems, and we discuss their main contributions. In Chapter 5, we provide a summary of the papers included in this thesis, and in Chapter 6 we conclude the work and discuss potential directions for future research.

Chapter 2

Edge Computing Infrastructure and Resources

The paradigm of bringing computing and storage resources close to the end users is known under multiple names and its precise definition is a subject of ongoing debate. The three most widespread names are *cloudlet-based edge computing*, *fog computing* and *multiple-access edge computing* (MEC). *Cloudlet-based edge computing* was introduced in 2009 and defined as a computing system with "a 3-tier hierarchy: mobile or IoT device-cloudlet-cloud", where "a cloudlet can be viewed as a data center in a box whose goal is to bring the cloud closer" [Sat+09]. *Fog computing* was introduced by Cisco in 2012 and defined as "a highly virtualized platform, which provides storage, computation, and network services between smart devices and cloud servers, typically but not exclusively deployed at the edge of the network" [Bon+12]. Finally, MEC was introduced by European Telecommunications Standards Institute (ETSI) in late 2014 and defined as "a technology that provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers" [Hu+15]. These definitions, however, do not specify the geographic span of an edge computing system.

A general model of an edge computing infrastructure compliant with the above three definitions consists of three layers, in the following referred to as fog, edge and remote layers.The fog layer is geographically closest to the end users and it consists of the end users' devices referred to as fog nodes. The edge layer consists of multiple heterogeneous edge clouds installed in base stations and small cell access points and it is considered as the second closest layer to the end users. Finally, the remote layer consists of computationally rich remote clouds that are geographically distant from the end users.

Fog, edge and remote computing components are connected with each other through variant types of communication resources and they together form a 3-layer edge computing infrastructure that allows fog nodes to perform their computational

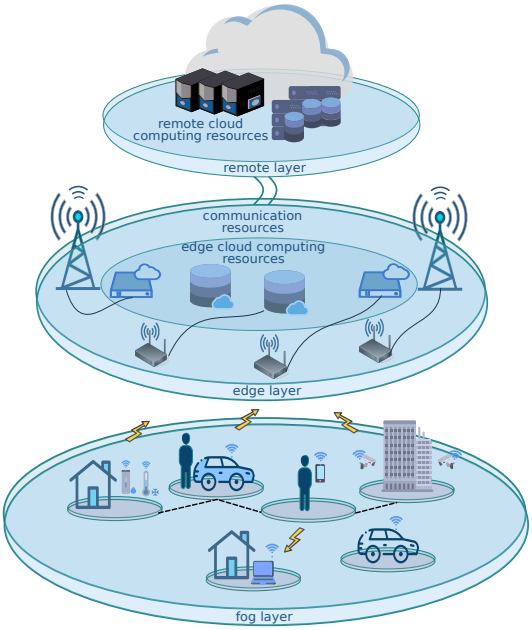

Figure 2.1: An example of a 3-layer edge computing infrastructure.

tasks locally or to offload them to each other, to edge clouds or to remote clouds. In Figure 2.1 we illustrate this model of 3-layer edge computing infrastructure, which is conceptually most similar to the model introduced in [Sat+09]. However, different from the concept proposed in [Sat+09], this general model considers that devices in the fog layer can offload the computation to each other and it also considers that the edge layer consists of multiple edge clouds with heterogeneous computing resources. In this chapter, we discuss the main characteristics of communication and computing resources from the point of view of an edge computing system.

2.1 Communication Resources

The computing components such as fog nodes, edge and remote clouds can be connected by wireline or wireless communication resources.

Wireline communication resources: In a traditional mobile network, the components such as base stations and access points are typically connected by high speed wireline links (e.g., optical fiber), and it is customary to model these links as having infinite bandwidth, and locally no latency, unless in switches. Since edge clouds are planned to be installed in the existing mobile network components, it is reasonable to assume that the edge clouds will also be connected through wireline communication resources. This, however, may become impractical and expensive

in the case of densely deployed edge clouds [Ryu+19; Pha+19a], and thus there is a need for an alternative way of connecting the computing components in edge computing systems.

Wireless communication resources: Wireless communication has been considered as an additional way of connecting mobile network components, and thus it can be considered as an alternative way of connecting edge clouds [Sio+18; Sid+15]. Furthermore, wireless communication is the main technology for connecting the end users' wireless devices with each other and with the mobile network components. Unlike wireline communication models, wireless communication models usually assume that the communication may suffer from high latency and scarce bandwidth problems.

The scarcity of bandwidth resources is one of the most important problems facing wireless communication and one can identify two different approaches to cope with this problem. The first approach is to reuse existing bandwidth resources by using device-to-device (D2D) communication [Mar09; Shi+12a; Jan+17] and the second approach is to manage the bandwidth resources used for device-to-infrastructure (D2I) communication in a more efficient way. From the perspective of an edge computing system, the first approach allows fog nodes to communicate directly without the involvement of wireless network infrastructure [Bon+14] and the second approach is especially relevant for the communication between fog nodes and edge clouds [Fry17].

The potential for implementing wireless bandwidth management depends on the wireless medium access protocol [Giu+18; Kek+18; Rez+18]. In general, one can consider two different types of wireless bandwidth management mechanisms. According to the first one, the bandwidth of an access point $a \in \mathcal{A}$ is shared equally among a set $\mathcal{N}_a$ of devices connected to the access point. In this case, the uplink rate $\omega_{i,a}$ of device $i \in \mathcal{N}_a$ depends on the specific set $\mathcal{N}_a$ of devices sharing the access point and it can be expressed as

$$\omega_{i,a} = f_a(\mathcal{N}_a), \forall a \in \mathcal{A}, \forall i \in \mathcal{N}_a. \tag{2.1}$$

Wireless bandwidth management described by (2.1) can be implemented in the CSMA/CA based protocols. For example, (2.1) can be used for modelling distributed coordination function (DCF) medium access mechanism [Bia00; Heu+03].

According to the second type of wireless bandwidth management mechanism, the uplink rate $\omega_{i,a}$ of device $i \in \mathcal{N}_a$ does not depend on the specific set $\mathcal{N}_a$ of devices sharing the access point, but it may depend on the total number $|\mathcal{N}_a|$ of devices sharing the access point. In this case, the uplink rate $\omega_{i,a}$ of device $i \in \mathcal{N}_a$ can be device specific, and it can be expressed as

$$\omega_{i,a} = f_{i,a}(|\mathcal{N}_a|), \forall a \in \mathcal{A}, \forall i \in \mathcal{N}_a. \tag{2.2}$$

Wireless bandwidth management described by (2.2) can be implemented in the TDMA and OFDM based medium access protocols [Jos+08; Bia+09]. For example,

(2.2) can be used for modelling proportional-fair scheduling (PFS) algorithm that has been used in 3G networks [LZL11].

2.2 Computing Resources

The sources of computing resources in an edge computing system may be diverse, from the fog nodes to the edge and remote clouds (c.f. Figure 2.1).

Fog nodes: Common to the fog nodes is that they may have at disposal limited computing resources when considered alone, but by pooling their resources through collaboration they can form a computationally rich distributed computing platform that makes use of the D2D communication [Mao+17; Wan+17; OEY11; Shi+12b; Mti+13]. Two big challenges facing the fog computing layer are how to integrate heterogeneous devices into a common computing platform, and how to efficiently distribute the computational tasks among numerous devices [BD14].

Edge clouds: By providing significant amount of heterogeneous computing resources close to the end users, edge clouds have the potential to provide low communication delays [ETS; RVM16]. Yet, the computing resources provided by the edge clouds may not be scalable with the number of wireless devices offloading their tasks, and thus the edge clouds may suffer from high execution times. Therefore, one of the biggest challenges facing the edge computing layer is how to efficiently manage limited computing resources [Sab+19a; Sab+19b].

Remote clouds: By providing abundant heterogeneous computing resources that scale with the number of devices, remote clouds ensure low execution times of computational tasks [Gro17; De16; Loe11]. However, remote clouds are typically located far away from the end users, and thus the computation offloading to remote clouds may suffer from high communication delays [Ost+09; He+10; Jac+10; Ios+11]. For this reason, remote clouds are usually considered as optional components in edge computing systems that through providing additional storage, memory and computing power have the potential to support the computation offloading in cases when the fog nodes and the edge clouds are overly congested.

As discussed above, computing components such as fog nodes, edge clouds and remote clouds may have different computing capabilities and they also may have different computing architectures. Consequently, the computing components in an edge computing system may differ from the perspective how fast they can execute the same computational task. One way of capturing the heterogeneity of computing resources in a system that consists of a set $\mathcal{C}$ of computing components is to introduce the notation for the clock frequency F^c for every computing component $c \in \mathcal{C}$ and the notation for the coefficient $k_i^c \in [0, 1]$ that indicates how suited the computing architecture of component c is for executing task t_i. Along with the heterogeneity of computing components, it is also important to model the congestion that may occur when a set $\mathcal{K}_c$ of computationally intensive tasks is assigned for the execution on a resource-poor computing component c. Using the introduced notation, the actual frequency F_i^c at which component c can execute task $t_i \in \mathcal{K}_c$

can be expressed as

$$F_i^c = f_{i,c}(F^c, k_i^c, \mathcal{K}_c), \forall c \in \mathcal{C}, \forall t_i \in \mathcal{K}_c. \tag{2.3}$$

Equation (2.3) allows for flexibility in modeling diverse computing resource management mechanisms. For example, (2.3) can be used for modeling scheduling algorithms that in practice approximate an ideal generalized processor sharing (GPS) scheduler [LBH09].

2.3 Network Slicing

The main driver of network slicing is the need for creating a flexible infrastructure that is capable of meeting the requirements of numerous types of IoT applications in many different domains such as autonomous vehicles, smart homes, healthcare, automated industry and surveillance [All15; Ord+17; Red+19]. Network slicing is about creating multiple isolated and customized logical networks with tailored capabilities for specific requirements of a variety of IoT applications. These logical networks, referred to as slices, can be vertical or horizontal [All15; Mav17; Ban+19]. Vertical slices are dedicated to serving a specific industry (e.g., health, automotive, home or energy) and horizontal slices are dedicated to serving specific classes of applications (e.g., streaming visual analytics, real-time control or media delivery). Vertical and horizontal slicing together appear as an attractive solution for providing services in edge computing systems in an isolated manner [Mav17].

Chapter 3

Computational Tasks in Edge Computing Systems

In this chapter we focus on the system performance of an edge computing system from the point of view of the end users. In this regard, we discuss different models of computational tasks, introduce the main performance metrics that can be used to measure the overall efficiency of an edge computing system, and we provide a general formulation of the joint task placement and resource allocation problem.

3.1 Computational Task Modeling

Computational tasks can be partitioned into subtasks at different granularity levels [AGH16]. In this regard, they can be divided into atomic tasks and tasks divisible into multiple subtasks that can be executed in parallel or in series or in a mixture. Some of these subtasks must be performed locally and some of them can be both performed locally and offloaded to the external computing resources. In Figure 3.1 we illustrate this classification of computational tasks.

In general, one can characterize a task by two parameters. The first parameter is the size D_i of the input data and the second parameter is the complexity L_i that is defined as the number of CPU cycles required to perform the computation. The relation between the size D_i of the input data and the task complexity L_i can be expressed as $L_i = D_i X_i$ [MN10], where X_i is the number of CPU cycles per data bit, which can be estimated from measurements by applying the methods described in [LS01; YN06; Cue+10; Net+18; Chu+11].

3.2 Performance Metrics

The two main metrics for assessing the performance of edge computing systems are the task completion time and the energy consumption. The factors that affect

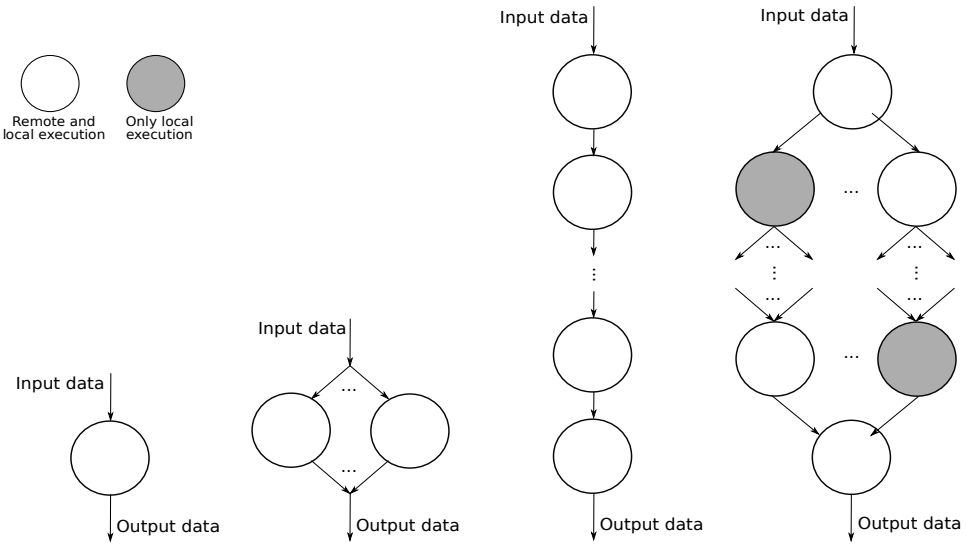

Figure 3.1: Typical examples of computational task modeling.

these performance metrics depend on which computing resources are used to execute a task. In the following we define the task completion time and the energy consumption both in the case of local computing and in the case of computation offloading.

Task completion time

When a task $<D_i, L_i>$ is executed locally on the device it was generated at, the time T_i^l needed to complete the task is the time needed for its execution using local computing resources,

$$T_i^l = \frac{L_i}{F_i^l}, \tag{3.1}$$

where F_i^l is the actual frequency at which device i can execute a task and it can be defined by (2.3).

On the contrary, when a device offloads its task $<D_i, L_i>$ to external computing node, the task completion time consists of three parts. The first part is the time $T_{i,a}^t$ needed to transmit the amount D_i of input data through an access point $a \in \mathcal{A}$, which can be expressed as

$$T_{i,a}^t = \frac{D_i}{\omega_{i,a}}, \tag{3.2}$$

where $\omega_{i,a}$ is the uplink rate of device i at access point a, which depends on the bandwidth sharing mechanism (e.g., defined by (2.1) or (2.2)).

The second part is the time $T_{i,e}^{exe}$ needed to execute the task using external computational resources,

$$T_{i,c}^{exe} = \frac{L_i}{F_i^c}, \tag{3.3}$$

where F_i^c is the actual frequency at which the external computing node c can execute a task generated by wireless device i and it can be defined by (2.3).

The third part is the time needed to transmit the result of the computation from the external computing node to the wireless device. For many applications (e.g., applications that require object, face and speech recognition), the size of the result is much smaller than the size D_i of the input data, and thus the third part can be neglected [HWN12; KL10a]. Therefore, in the case of computation offloading, a simple linear model can be used to model the task completion time,

$$T_{i,a}^c = T_{i,a}^t + T_{i,c}^{exe}. \tag{3.4}$$

Energy consumption

When a task $<D_i, L_i>$ is executed locally, the energy consumption E_i^l of wireless device i is the energy needed to execute the task using local computing resources at frequency F_i^l. According to the measurements reported in [WZL12; MN10], the energy consumption per CPU cycle is linearly proportional to the square of the frequency, and it can be expressed as

$$E_i^l = c(F_i^l)^2 L_i, \tag{3.5}$$

where the constant $c \sim 10^{-11}$ according to reported measurements.

On the contrary, when a wireless device i offloads its task $<D_i, L_i>$ to external computational resources through an access point a, the energy consumption $E_{i,a}$ is the energy spent to upload the amount D_i of the input data. According to measurements reported in [Ba09], the energy spent to upload the data over the cellular network consists of three parts. The first part is the energy spent to scan available wireless connections, the second part is the energy spent to transmit data, and the third part is the energy spent to keep the interface up during the transmission period.

When a task is characterized by a large size D_i of the input data, it is reasonable to consider that the energy spent to transmit data dominates the energy spent to scan available wireless connections and the energy spent to keep the interface up during the transmission period. Consequently, given that wireless device i transmits the data through an access point a at rate $\omega_{i,a}$, the energy consumption $E_{i,a}$ can be expressed as

$$E_{i,a} = \frac{D_i P_{i,a}}{\omega_{i,a}}, \tag{3.6}$$

where the transmit power $P_{i,a}$ that wireless device i uses to transmit the data through an access point a can be determined using an algorithm as the ones proposed in [SMG02; XSC03].

3.3 Cost Model

Wireless devices are heterogeneous in terms of computational capabilities, battery states, types of computational tasks and the rate at which they generate the tasks. This multi-level heterogeneity of wireless devices makes it reasonable to consider that different devices may have different preferences over introduced performance metrics (i.e., task completion time and energy consumption). This heterogeneity can be captured by introducing two parameters, $0 \leq \gamma_i^T \leq 1$ and $0 \leq \gamma_i^E \leq 1$, which characterize the preferences of device i over the completion time and the energy consumption, respectively. Given these parameters, the cost of wireless device i can be formulated as a function of the completion time and the energy consumption,

$$\textit{local execution:} \quad C_i^l = f(\gamma_i^T T_i^l, \gamma_i^E E_i^l), \tag{3.7}$$

$$\textit{offloading:} \quad C_{i,a}^c = f(\gamma_i^T T_{i,a}^c, \gamma_i^E E_{i,a}), \tag{3.8}$$

where T_i^l, E_i^l, $T_{i,a}^c$ and $E_{i,a}$ are given by (3.1), (3.5), (3.4) and (3.6), respectively.

Equations (3.7) and (3.8) can be used for expressing different cost functions, such as completion time, energy consumption, throughput or a combination of these. Furthermore, they allow for a model in which each device can dynamically adjust its objective to the specific task requirements, and to its current battery state by changing the values of the parameters γ_i^T and γ_i^E.

3.4 Joint Task Placement and Resource Allocation Problem

A general formulation of the joint task placement and resource allocation problem can be provided by considering a general model of an edge computing system that consists of a set $\mathcal{N}$ of devices, $|\mathcal{N}| = N$, a set $\mathcal{A}$ of communication resources, $|\mathcal{A}| = A$, and a set $\mathcal{C}$ of computing resources, $|\mathcal{C}| = C$. Task placement matrices for communication and computing resources are then given by $\mathbf{X} \in \{0,1\}^{N \times A}$ and $\mathbf{Y} \in \{0,1\}^{N \times C}$, respectively. Furthermore, the policies according to which communication and computing resources are managed can be defined as $\mathcal{P}_{\mathbf{X}} : \mathbf{X} \rightarrow \mathbb{R}_{[0,1]}^{N \times A}$ and $\mathcal{P}_{\mathbf{Y}} : \mathbf{Y} \rightarrow \mathbb{R}_{[0,1]}^{N \times C}$, respectively.

Since the edge resources are in general shared among many devices, it is reasonable to consider that the system cost $C(\mathbf{X}, \mathbf{Y}, \mathcal{P}_{\mathbf{X}}, \mathcal{P}_{\mathbf{Y}})$ is a function of task placement matrices $\mathbf{X}$ and $\mathbf{Y}$, and the resource allocation policies $\mathcal{P}_{\mathbf{X}}$ and $\mathcal{P}_{\mathbf{Y}}$. For example, in an end-user-oriented model of an edge computing system, the cost $C(\mathbf{X}, \mathbf{Y}, \mathcal{P}_{\mathbf{X}}, \mathcal{P}_{\mathbf{Y}})$ can be defined as a function of the end users' costs (3.7) and (3.8), which may depend on the congestion on the shared resources. The introduced notation allows us to define the joint task placement and resource allocation problem as the following mixed-integer optimization program,

$$\min_{\mathbf{X},\mathbf{Y},\mathcal{P}_\mathbf{X},\mathcal{P}_\mathbf{Y}} C(\mathbf{X},\mathbf{Y},\mathcal{P}_\mathbf{X},\mathcal{P}_\mathbf{Y}) \tag{3.9}$$

$$\text{s.t.} \quad g_i(\mathbf{X},\mathbf{Y},\mathcal{P}_\mathbf{X},\mathcal{P}_\mathbf{Y}) \leq \theta_i, \forall i \in \mathcal{N}, \tag{3.10}$$

$$h(\mathbf{X},\mathcal{P}_\mathbf{X}) \leq \theta_a, \forall a \in \mathcal{A}, \tag{3.11}$$

$$q(\mathbf{Y},\mathcal{P}_\mathbf{Y}) \leq \theta_c, \forall c \in \mathcal{C}, \tag{3.12}$$

$$\sum_{a\in\mathcal{A}} \sum_{c\in\mathcal{C}\setminus\{i\}} x_{i,a}y_{i,c} + y_{i,i} = 1, \forall i \in \mathcal{N}, \tag{3.13}$$

$$\mathbf{X} \in \{0,1\}^{N\times A}, \tag{3.14}$$

$$\mathbf{Y} \in \{0,1\}^{N\times C}, \tag{3.15}$$

$$\mathcal{P}_\mathbf{X} : \mathbf{X} \rightarrow \mathbb{R}_{[0,1]}^{N\times A}, \tag{3.16}$$

$$\mathcal{P}_\mathbf{Y} : \mathbf{Y} \rightarrow \mathbb{R}_{[0,1]}^{N\times C}. \tag{3.17}$$

The function $g_i(\mathbf{X},\mathbf{Y},\mathcal{P}_\mathbf{X},\mathcal{P}_\mathbf{Y})$ takes into account sharing both communication and computing resources, and thus constraint (3.10) can be used to ensure that the task completion time or the energy consumption of each device $i \in \mathcal{N}$ is lower than the threshold specified by θ_i. Functions $h(\mathbf{X},\mathcal{P}_\mathbf{X})$ and $q(\mathbf{Y},\mathcal{P}_\mathbf{Y})$ take into account sharing only one type of resources, that is, the communication and computing resources, respectively. Hence, constraints (3.11) and (3.12) can be used thus to enforce a limitation on the amount of communication and computing resources that can be provided to the devices, respectively. Constraint (3.13) enforces that each device $i \in \mathcal{N}$ either performs the computation locally ($x_{i,a} = 0, y_{i,i} = 1, y_{i,c} = 0, \forall a \in \mathcal{A}, \forall c \in \mathcal{C} \setminus \{i\}$) or it offloads the task to computing resource c using communication resource a ($x_{i,a} = 1, y_{i,c} = 1, x_{i,a'} = 0, y_{i,c'} = 0, \forall a' \in \mathcal{A}\setminus\{a\}, \forall c' \in \mathcal{C}\setminus\{c\}$). Constraints (3.14) and (3.15) specify that the task placement decisions are integer variables, and constraints (3.16) and (3.17) describe policies for allocating communication and computing resources, respectively. Finally, solving the problem (3.9) − (3.17) may be impractical in realistic edge computing systems, because it involves searching a large solution space.

It is important to note that (3.9)-(3.17) can be easily used for formulating different problems that may be of interest in diverse edge computing systems. For example, (3.9)-(3.17) can be easily reduced to the completion time minimization problem by defining cost $C(\mathbf{X},\mathbf{Y},\mathcal{P}_\mathbf{X},\mathcal{P}_\mathbf{Y})$ as a sum of costs (3.7) and (3.8) and by setting the completion time parameter $\gamma_i^T = 1$, and the energy consumption parameter $\gamma_i^E = 0$, for all wireless devices. With this in mind, in the following we consider different versions of problem (3.9) − (3.17), and we discuss the most important results from the literature.

Chapter 4

Task Placement and Allocation of Edge Resources

Works on task placement and resource allocation in edge computing systems can be categorized according to four criteria:

1) *Design objective:* The common design objectives are minimization of the completion time, of the energy consumption, joint minimization of the two and the maximization of the throughput. The objective design choice is mostly determined by the characteristics of computational tasks. For example, completion time minimization is a suitable design objective in the case of delay sensitive tasks, while energy consumption minimization is a suitable design objective in the case of computationally intensive tasks. Finally, multi-objective optimization, such as joint minimization of the completion time and the energy consumption, is a suitable design objective for both delay sensitive and computationally intensive tasks.

2) *Information availability:* The second criterion is related to the amount of information available for optimization. In the case of offline optimization, the assumption is that there is complete knowledge of system resources and the tasks to be allocated. In the case of online optimization, information is incomplete, e.g., due to stochastic task arrivals, and the objective is to optimize the long-term system performance (e.g., the system throughput, the long-term average task completion time and the long-term average energy consumption).

3) *Edge computing system components:* The main edge computing components are end users' devices, edge clouds and remote clouds. Depending on which of these components are available for performing computational tasks, we can distinguish between multiple edge computing infrastructures. The simplest edge computing infrastructure consists of one or multiple edge clouds. More complex edge computing infrastructures support computation offloading not only to edge clouds, but also to remote clouds and to nearby end users' devices.

Table 4.1: Classification of works according to the types of shared resources.

Paper	Communication resources	Computing resources
[Zha+16]	✓	
[Mes+17]	✓	
[XHS19]	✓	
[Che15]	✓	
[Che+16]	✓	
[Guo+16]	✓	
[Hua+18]	✓	
[Zhe+18]	✓	
[CC17]	✓	
[Yan+13]	✓	
[XSM18]	✓	
[Zhu+18]	✓	
[Zho+18]	✓	
[BZ18]	✓	
[CH18]		✓
[Liu+18]		✓
[SL18]		✓
[Yan+15]		✓
[CL17]		✓
[CL19]		✓
[Zen+16]		✓

Paper	Communication resources	Computing resources
[XCZ18]		✓
Paper A [JD19a]		✓
[Ge+12]		✓
[XLL15]		✓
[Den+16]		✓
[Cha+17]		✓
[CZ17]		✓
[SW18]		✓
[Rah+15]		✓
[WLP13]		✓
[Car+16]		✓
[XLX13]		✓
[Hu+19]		✓
[Yan+17]	✓	✓
[Gao+19]	✓	✓
[Ren+19]	✓	✓
[Zha+18]	✓	✓
[Guo+19]	✓	✓
[EDF17]	✓	✓
Paper D [JD19b]	✓	✓
Paper E [JD20b]	✓	✓

Paper	Communication resources	Computing resources
[You+16]	✓	✓
[TL17]	✓	✓
[Bar+16]	✓	✓
[SSB15]	✓	✓
[AlS+17]	✓	✓
[ZF19]	✓	✓
[Yan+19]	✓	✓
[Pha+19b]	✓	✓
[TP18]	✓	✓
[Lyu+16]	✓	✓
[Zha+17]	✓	✓
[CLD18]	✓	✓
[Du+18]	✓	✓
[EL19]	✓	✓
Paper B [JD19c]	✓	✓
Paper C [JD20a]	✓	✓
[Tär+17]	✓	✓
[Yan+18]	✓	✓
[LOD18]	✓	✓
[Lyu+17]	✓	✓
[Aya+19]	✓	✓

4) *Management architecture:* Task placement decisions and resource allocation policies can be implemented in centralized, decentralized and distributed manners. Centralized management architecture is considered as a suitable choice in systems with an entity that has full information about the system. Decentralized management architecture is considered as a suitable choice in systems with a centralized entity that based on full or partial knowledge about the system coordinates multiple other hierarchically lower entities in making task placement decisions or resource allocation policies. Finally, a distributed management architecture is considered as a suitable choice in systems in which the task placement decisions and the resource allocation policies can be made by multiple hierarchically equal entities with or without information exchange.

Table 4.1 shows a summary of the research papers that addressed the task placement and the resource allocation problems in edge computing systems with shared communication, computing, and both communication and computing resources, respectively. In what follows we discuss the most important results from these papers.

Table 4.2: Classification of works that model the congestion on communication resources.

Paper	Design objective			Information availability		System components			Management architecture		
	Time	Energy	Throughput	Offline	Online	D2D	Edge cloud	Remote cloud	Centralized	Decentralized	Distributed
[Zha+16]		✓		✓			✓		✓		
[Mes+17]		✓		✓			✓			✓	
[XHS19]	✓	✓		✓			✓	✓	✓		
[Che15]	✓	✓		✓			✓			✓	
[Che+16]	✓	✓		✓			✓			✓	
[Guo+16]	✓	✓		✓			✓			✓	
[Hua+18]	✓	✓		✓			✓				✓
[Zhe+18]	✓	✓			✓		✓				✓
[CC17]	✓	✓			✓		✓				✓
[Yan+13]			✓		✓		✓		✓		
[XSM18]			✓	✓			✓		✓		
[Zhu+18]			✓	✓			✓			✓	
[Zho+18]			✓	✓			✓		✓		
[BZ18]			✓	✓			✓		✓		

4.1 Allocation of Communication Resources

Works that modeled the congestion on communication resources and assumed the elasticity of computing resources are relevant in the case of computationally rich edge computing infrastructures and the tasks that are computationally light, but require the transmission of a large amount of data. Only a few works from this category addressed the problem of minimizing the energy consumption [Zha+16; Mes+17]. On the contrary, majority of the works addressed the problem of joint minimization of the task completion time and the energy consumption [XHS19; Che15; Che+16; Guo+16; Hua+18; Zhe+18; CC17]. Finally, some of the works focused on the problem of maximizing the system throughput [Yan+13; XSM18; Zhu+18; Zho+18; BZ18]. A summary of works is shown in Table 4.2.

Energy consumption minimization

A simple model of an edge computing system consists of a single edge cloud and multiple devices, each of them with a computational task, which can be performed locally or can be offloaded to the edge cloud through one or multiple wireless links [Zha+16; Mes+17]. In [Zha+16] the authors assumed that devices can access the edge cloud either through a small or a macro base station, both of them operating in the same frequency band divided into a set of identical channels. They

formulated the problem as offline minimization of the sum of devices' energy consumptions under constraints on the completion times of the tasks and the number of available wireless channels. The authors then proposed a centralized heuristic for assigning the tasks and allocating radio resources to offloading devices. In [Mes+17] the authors considered that devices can offload the computation through a single wireless link and they used game theoretical tools to address the offline computation offloading problem. The authors formulated the problem as a congestion game in which the devices aim at minimizing their own energy consumption while meeting constraints on the completion times of their tasks. The authors then showed that the game has a pure Nash equilibrium and proposed a decentralized algorithm for computing an equilibrium of offloading decisions.

Completion time and energy consumption minimization

A widely adopted model of the cost is a linear combination of completion time and energy consumption both in the case of local execution and in the case of computation offloading [XHS19; Che15; Che+16; Guo+16; Hua+18; Zhe+18; CC17]. In [XHS19] the authors considered offline optimization of the joint task placement and bandwidth allocation problem in a system that consists of an edge and a remote cloud. Taking into account constraints on the total available bandwidth, the authors proposed a centralized algorithm for computing the optimal task assignment and bandwidth allocation vectors. They showed that the best and the worst case complexity of the proposed algorithm is quadratic and exponential in the number of devices, respectively.

Several works focused on developing decentralized solutions to offline computation offloading problems [Che15; Che+16; Guo+16]. The works presented in [Che15; Che+16] used game theoretical tools to model and analyze the interaction among multiple devices that aim at minimizing their own costs in a system with a single edge cloud. They modeled the congestion on communication resources as a strategic game in which the devices may offload their tasks to the edge cloud through a single and multiple wireless links, respectively. In [Che15] the authors provided an algorithm for computing a pure strategy Nash equilibrium of offloading decisions and established a bound on the price of anarchy of the game. In [Che+16] they showed that the same results hold in the case of multiple identical wireless links. The algorithms proposed in [Che15; Che+16] can be implemented in a decentralized manner by letting devices compute their offloading decisions asynchronously based on the information provided by a central coordinator located in the edge cloud. The authors in [Guo+16] considered a system that consists of a single edge cloud that can be reached through a base station or a small access point. They assumed that a task of each device can be modeled as a directed acyclic graph with the same number of subtasks and they formulated the problem of minimizing the sum of costs of all devices under constraints on the task completion times and the subtask dependencies. The authors then decomposed the original problem into two problems that can be solved in a decentralized manner. The first problem is

solved by each wireless device based on the information provided by a network operator; wireless devices decide where to execute their subtasks, at which CPU clock frequency to perform the local executions and at which transmission rate to offload their subtasks. The second problem is solved by the operator that uses the subgradient method to compute the Lagrangian multipliers associated with constraints on the task completion times and the subtask dependencies.

Finally, a few works used machine learning techniques to develop distributed solutions to the task placement and radio resource allocation problems that may exist in the systems in which devices can either perform their tasks locally or offload them to a single edge cloud [Hua+18; Zhe+18; CC17]. In [Hua+18] the authors considered that the cloud can be accessed through a single wireless link. They formulated the problem as an offline minimization of the system cost under the constraint on the total available bandwidth. The authors then proposed a deep learning-based solution that deploys multiple parallel deep neural networks. Each neural network has the function of an offloading actor that computes the candidate task placement and resource allocation vectors, exchanges the computed vectors with the other actors, and based on the exchanged information selects the best candidate solution for updating the input to its own deep neural network. The authors in [Zhe+18] considered a system with stochastic task arrivals and a single wireless link. They formulated the online task placement problem as a stochastic game and proved the existence of an equilibrium task allocation. The authors proposed a stochastic learning algorithm that can compute an equilibrium of offloading decisions in a distributed manner without any information exchange. In [CC17] the authors considered a time-slotted communication system with multiple wireless links and CSMA based protocol according to which devices experience random transmission rates over time slots. They modeled the interaction among the devices as a noncooperative game in which each device aims at maximizing its own reward function and proved that the game has a pure strategy Nash equilibrium. Based on stochastic learning automata, the authors proposed a fully distributed algorithm that devices can use for computing their equilibrium offloading decisions without exchanging the information with each other.

Throughput Maximization

The simplest model of an edge computing system consists of a single edge cloud, a single wireless link and multiple wireless devices that can perform their computational tasks locally or offload them to the edge cloud [Yan+13; XSM18; Zhu+18; Zho+18; BZ18]. The authors in [Yan+13] modeled a computational task as a directed acyclic graph with multiple subtasks and formulated the load scheduling problem as an online bin packing problem. They proposed a genetic based centralized algorithm for assigning the subtasks for the execution while taking into consideration constraints on the amount of available bandwidth. Another centralized solution was proposed in [XSM18], where the authors assumed that offloading decisions of devices are known. Under this assumption, they formulated the prob-

lem as offline maximization of the total system throughput under constraints on the allowed transmission powers and acceptable interference levels. The authors derived a closed-form expression for the optimal transmit power allocation policy. In [Zhu+18] the authors used game theoretical tools to address the offline problem of allocating communication resources. They first formulated the problem as a bargaining game in which the objective is to optimize the assignment of transmission powers and the allocation of wireless subchannels to wireless devices. The authors then relaxed the original problem to consider the optimization of time-shares of the bandwidth resources and by solving the relaxed version of the problem they proposed a heuristic for solving the original problem. The proposed algorithm can be implemented in a decentralized way, by letting devices compute their transmission powers and the central coordinator to allocate bandwidth resources.

A few works considered offline optimization of computation offloading in edge computing systems in which the wireless devices are equipped with energy harvesting capabilities [Zho+18; BZ18]. The authors in [Zho+18] considered a system with an unmanned aerial vehicle (UAV) that transmits energy to the devices and provides computing services. The authors formulated the problem of maximizing the amount of computation that can be offloaded to the UAV under constraints on the computing and battery capacities of devices, constraints on the amount of available bandwidth and constraints on the feasible UAV trajectories. They considered partial and binary computation offloading problems in which the computational tasks can and cannot be partitioned, respectively. In both cases, they decoupled the problem of optimizing the trajectory of the UAV from the problem of optimizing devices' CPU frequencies, offloading times and transmit powers. For a given trajectory of the UAV, the authors derived closed-form expressions for the latter optimization problem and proposed two-stage and three-stage iterative algorithms for solving partial and binary computation offloading problems in a centralized manner, respectively. The authors in [BZ18] considered a system with a single access point that can transmit the energy to the devices and it can also execute computational tasks offloaded by the devices. The authors assumed that the wireless power transmission and the task offloading are performed in the same frequency band. Under this assumption and constraints on computing capabilities and battery capacities of the devices, the authors formulated the offline computation rate maximization problem. They decoupled the original problem into the task assignment problem and the problem of optimizing the devices' CPU frequencies, transmission powers and bandwidth allocations. Based on a coordinate decent method they proposed a heuristic for solving the original problem in a centralized manner.

4.2 Allocation of Computing Resources

There is a significant body of works that considered the congestion on computing resources without taking into account the congestion on communication resources. These works are relevant for edge computing systems with abundant bandwidth

Table 4.3: Classification of works that model the congestion on computing resources.

Paper	**Design objective**			**Information availability**		**System components**			**Management architecture**		
	Time	Energy	Throughput	Offline	Online	D2D	Edge cloud	Remote cloud	Centralized	Decentralized	Distributed
[CH18]	✓			✓			✓		✓		
[Liu+18]	✓			✓			✓	✓	✓		
[SL18]	✓			✓		✓	✓	✓	✓		
[Yan+15]	✓			✓	✓		✓		✓		
[CL17]	✓			✓	✓		✓	✓	✓		
[CL19]	✓			✓			✓	✓	✓		
[Zen+16]	✓				✓		✓		✓		
[XCZ18]	✓				✓		✓	✓			✓
Paper A [JD19a]	✓				✓	✓	✓			✓	
[Ge+12]		✓		✓			✓			✓	
[XLL15]		✓		✓		✓					✓
[Den+16]		✓			✓		✓	✓	✓		
[Cha+17]		✓			✓		✓			✓	
[CZ17]	✓	✓		✓		✓	✓		✓		
[SW18]	✓	✓		✓			✓	✓		✓	
[Rah+15]	✓	✓		✓			✓	✓	✓		
[WLP13]	✓	✓			✓		✓			✓	
[Car+16]			✓		✓		✓	✓		✓	
[XLX13]			✓		✓		✓	✓	✓		
[Hu+19]			✓	✓			✓		✓	✓	

resources and computationally heavy tasks that require the transmission of a small amount of data. Majority of the works from this category considered the problem of minimizing the completion time of computational tasks [CH18; Liu+18; SL18; Yan+15; CL17; CL19; Zen+16; XCZ18]. Minimization of the energy consumption [Ge+12; XLL15; Den+16; Cha+17] and the joint minimization of the task completion time and the energy consumption [CZ17; SW18; Rah+15; WLP13] received considerable attention in the literature. Finally, few works from this category addressed the problem of maximizing the system throughput [Car+16; XLX13; Hu+19]. A summary of works is shown in Table 4.3.

Completion time minimization

Solving the completion time minimization problems in a centralized manner received significant attention for systems with limited computing resources [CH18; Liu+18; SL18; Yan+15; CL17; CL19; Zen+16]. The authors in [CH18] considered

a system that consists of a single macro and multiple micro base stations equipped with computing power. They formulated the offline problem of joint task assignment and computing resource allocation with the objective to minimize the sum of completion times of all devices under their energy consumption constraints. The authors then provided a closed form solution for the optimal computing resource allocation policy and solved a relaxed version of the task assignment problem. In [Liu+18] the authors considered an edge computing system that consists of multiple edge clouds and a single remote cloud. They considered a set of augmented reality users and formulated the offline problem of minimizing the overall service latency and maximizing the total object recognition accuracy. The authors then considered a relaxed version of the problem and proposed a centralized algorithm according to which the network orchestrator assigns users to the clouds and decides about the frame resolutions. In [SL18] the authors considered a system with a single remote cloud and a finite number of computationally limited local processors that may be located in edge clouds or peer wireless devices. They considered that the task of each device consists of multiple subtasks, whose dependency can be modeled as a directed acyclic graph, and they defined a cost of performing a subtask as a linear combination of the processing time and the transmission price (if any). The authors proposed a heuristic for solving the offline task placement problem in which the objective is to minimize the sum of costs of devices under constraints on the dependency among subtasks, the task completion time deadlines and constraints on the amount of available computing resources.

The authors in [Yan+15] considered a set of tasks generated during a fixed time period and modeled a task as a sequence of subtasks that can be either processed locally or can be offloaded to a cloud that has a limited number of processors. The authors formulated the average task completion time minimization problem under constraints on the dependency among subtasks and constraints on the amount of available computing resources. They considered a system with known and unknown release times of the future tasks and proposed two heuristics for addressing the corresponding offline and online versions of the problem, respectively. The authors in [CL17] considered a system that consists of a set of clouds (edge and remote) with known and unknown processing times, respectively. They formulated the problem of minimizing the completion time of the last task, that is, the makespan of a given set of computational tasks. For the case when the processing time is unknown for only one of the clouds, the authors proposed a constant-factor semi-online approximation algorithm for scheduling the tasks. They extended the analysis to the case of multiple clouds with unknown processing times, and in this case they proposed an online heuristic algorithm. In [CL19] the authors considered a system that consists of a set of identical edge clouds and a single remote cloud with unknown processing times. They assumed that the offloading costs are known a priori for each task and they formulated a semi-online task scheduling problem with the objective to minimize a weighted sum of the makespan and the offloading cost. Similarly to their work in [CL17], the authors assumed that the task execution can be restarted multiple times and they proposed a constant-factor approximation

algorithm for scheduling the tasks.

There are several works that considered online computation offloading problems in edge computing systems with Poisson task arrivals [Zen+16; XCZ18]. In [Zen+16] the authors considered the joint optimization of task allocation and task image placement in a system that consists of a set of storage servers and a set of computation servers. The authors formulated the problem as a mixed-integer linear program in which the objective is to minimize the maximum average task completion time under constraints on the system stability and the computing capacities of servers. They decoupled the original problem into task placement and task scheduling problems and proposed a heuristic for solving the original problem. The authors in [XCZ18] considered an edge computing system that consists of a remote cloud and multiple base stations equipped with computing resources. They assumed that each base station offers specific computing services and decides what portion of the requests to serve locally and what portion of the requests to offload to the remote cloud. The authors addressed the problem of minimizing the average long-term completion times of tasks under long-term constraints on the storage capacity and the energy consumption and short-term constraints on the task completion time deadlines and energy consumption. Based on the Lyapunov optimization technique, the authors proposed an online service caching and task offloading algorithm that iteratively optimizes the vectors of offloading and caching decisions of each base station. The algorithm is designed for a distributed implementation that requires information exchange between neighbouring base stations.

Our work presented in Paper A [JD19a] falls into the class of works that consider online completion time minimization problems. We addressed the task placement problem in an edge computing system that consists of a single edge cloud and multiple heterogeneous devices, which may process the tasks of each other. We considered that devices use dedicated bandwidth resources to communicate with each other and with the cloud. We modeled the task arrival process of each device as a Poisson process and used queuing theory to capture the contention on dedicated communication resources and to model sharing of computing resources. We denoted by $\overline{T^c}_{i,j}(p_{i,j})$ the mean time needed to complete the task generated by device $i \in \mathcal{N}$ using the computing resources of node $j \in \mathcal{N} \cup \{0\}$, where 0 corresponds to the edge cloud. Finally, we defined the system cost $\bar{C}(\mathbf{P})$ as the average system delay

$$\bar{C}(\mathbf{P}) = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N} \cup \{0\}} p_{i,j} \overline{T^c}_{i,j}(p_{i,j}), \tag{4.1}$$

where $\mathbf{P} \in [0,1]^{N \times (N+1)}$ is a task assignment matrix and $p_{i,i}$, $p_{i,0}$ and $p_{i,j}$ indicate the probability that device i executes its task locally, offloads the task to the cloud, and offloads the task to device $j \in \mathcal{N} \setminus \{i\}$, respectively. Given the set $\mathcal{P}$ of all task assignment matrices that ensure the stability of the queuing system, our task placement problem can be formulated as the following online optimization problem

$$\min_{\mathbf{P}\in\mathcal{P}} \bar{C}(\mathbf{P}) \tag{4.2}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{N}\cup\{0\}} p_{i,j} = 1, \forall i \in \mathcal{N}. \tag{4.3}$$

Since devices in edge computing systems are expected to be autonomous [VR14], in Paper A instead of solving (4.2) − (4.3) we defined the problem as a strategic game, in which each device plays a mixed strategy and aims at minimizing its own cost. We used game theoretical tools to prove the existence of an equilibrium task allocation in static mixed strategies. We proposed a decentralized algorithm that allocates tasks according to the computed mixed strategy profile, and thus it relies on average system parameters only. By performing simulations, we compared the performance of the proposed algorithm with the performance of an algorithm that allocates tasks according to the optimal static mixed strategy (i.e., to the solution of (4.2) − (4.3)), and with the performance of a greedy algorithm that relies on global knowledge of the system state. We showed that the proposed algorithm achieves good system performance close to that of the performance of the optimal static mixed strategy and even close to the performance of a greedy algorithm.

Energy consumption minimization

One way of addressing the task placement problem with the objective of minimizing the energy consumption is to model and analyze the interaction among multiple devices using game theoretical tools [Ge+12; XLL15]. In [Ge+12] the authors considered a system with multiple clouds that serve multiple devices, each of them with a single computational task to be performed. They modeled the offline task placement problem as a congestion game in which each device partitions its task into a part for offloading and a part for local computing and it also decides to which cloud server to offload each part of its computation. The authors defined the objective function of each wireless device as a weighted sum of the energies consumed for executing the two portions of its task. They proved the existence of a task placement equilibrium and proposed a decentralized algorithm for computing an equilibrium based on the information provided by a central coordinator. In [XLL15] the authors considered a system in which the objective is to minimize the total energy consumption of multiple wireless devices that can execute each others' tasks in a collaborative way. They assumed that the full knowledge about the system state is available and formulated the offline energy minimization problem as a coalition game. The authors then proposed a distributed coalition formation algorithm that requires information exchange between neighbouring devices.

A couple of works considered systems with stochastic task arrivals and focused on developing solutions to the online computation offloading problems [Den+16; Cha+17]. In [Den+16] the authors addressed the task assignment problem in a

system that consists of a set of cloud servers and a set of fog devices that are located close to the end users. The authors formulated the problem of minimizing the energy consumption of the system while meeting the end users' constraints on the average task completion times. They proposed an approximate centralized solution based on the decomposition of the original problem into three subproblems: optimization of the amount of locally executed workload for each fog device, optimization of the traffic rate for each device-to-cloud pair and joint optimization of the amount of workload assigned to each cloud, the number of active processors in each cloud and CPU frequencies of the active processors. In [Cha+17] the authors considered an edge computing system in which devices can perform their tasks locally or they can offload them to an edge cloud through a single wireless link. The authors assumed Poisson task arrivals and formulated the problem of finding the optimal transmission powers and offloading rates of the devices that aim at minimizing their average energy consumptions under constraints on the average task completion times. They proposed an iterative algorithm that sequentially updates local and global decisions made by the devices and a central cloud coordinator, respectively. The algorithm can be implemented in a decentralized way by letting devices make their local decisions and the cloud coordinator to make global decisions while taking into consideration devices' preferences specified by their local decisions.

Completion time and energy consumption minimization

The common multi-objective optimization function is defined as a linear combination of the task completion time and the energy consumption [CZ17; SW18]. In [CZ17] the authors assumed that each device can execute the computation locally, offload its task to a neighbouring device or to an edge cloud. They assumed that the edge cloud has a limited number of virtual machines, each of them capable of hosting a single task per time. Under this assumption, the authors developed a centralized algorithm for solving the offline task placement problem, which they defined as the minimum weight matching problem over the three-layer graph. The authors in [SW18] considered an edge computing system in which each device can perform its task locally or offload it to an edge or to a remote cloud. The authors modeled the offline task placement problem as a strategic game for which they proved the existence of a pure strategy Nash equilibrium. They proposed an algorithm that can compute an equilibrium of offloading decisions in a decentralized manner and used the price of anarchy to establish a bound on the cost approximation ratio of the proposed algorithm.

Multi-objective optimization of the computation offloading can be defined with respect the task completion time, the energy consumption and the monetary cost [Rah+15; WLP13]. In [Rah+15] the authors considered a system in which a set of computing services is provided by multiple edge and remote clouds. The authors described the mobility patterns of devices by space-time trajectories and modeled a computational task as a space-time workflow. They presented workflows as directed

acyclic graphs in which each node is associated with a certain type of computing service. The authors defined a single device utility as the minimum of price, delay and energy consumption needed for performing a task. Finally, they proposed a centralized heuristic that aims at minimizing the sum of devices' utilities through mapping their space-time workflows with the available cloud computing services. Different from the above works, the authors in [WLP13] used game theoretical tools to capture the interaction between multiple wireless devices that generate computational tasks according to a Poisson process. Each device can decide whether to process its tasks locally or to offload them to a cloud that consists of multiple homogeneous servers. The authors provided a two stage game-based formulation of the problem in which the players are the devices and the cloud controller. In the first stage of the game, each device determines the size of the task portion that it offloads with the objective to minimize a linear combination of its average task completion time and the corresponding energy consumption. In the second stage of the game, the cloud controller implements the optimal task dispatching and the resource allocation policies that maximize its profit. The authors proved the existence of a subgame perfect equilibrium and proposed a decentralized algorithm according to which devices make their decisions based on the information provided by the cloud controller.

Throughput Maximization

A general model of an edge computing system consists of multiple wireless devices, an edge cloud and a remote cloud [Car+16; XLX13]. The authors in [Car+16] assumed that the task arrival process at each device can be modeled as a Poisson process and that each task can be offloaded or performed locally. They modeled the cost of a single device as the expected number of its unfinished tasks in the system and assumed that each device aims at minimizing its own cost under energy consumption constraints. The authors addressed the online task placement problem using game theoretical tools and showed the existence of a mixed strategy equilibrium task allocation. They developed a decentralized algorithm for computing equilibrium offloading strategies under the assumption that the information about the system load is provided to the devices by the cloud coordinator. In [XLX13] the authors considered a system with stochastic task arrivals and assumed that the tasks are either processed by an edge cloud or forwarded to a remote cloud. They addressed the online request admission problem in a centralized way with the objective to maximize the system throughput under constraints on computing capabilities of the edge cloud. According to the proposed admission control algorithm, the edge cloud decides which computational requests to accept and which requests to send to the remote cloud.

Different from the above works, the authors in [Hu+19] considered offline optimization of the joint task placement and computing resource allocation in a system in which each device can either perform its task locally or offload the task to one of

multiple edge clouds. Assuming the underlying correlation between task execution times under different computing architectures, the authors first proposed a low-rank learning algorithm for estimating task execution times. Given predicted task execution times, the authors formulated the problem of maximizing the number of completed tasks under constraints on the limited edge cloud resources (CPU, memory and hardware) and constraints on the maximum allowed task completion times. They showed that the problem is NP-hard and proposed an algorithm with a bounded approximation ratio. The authors showed that the worst case complexity of the proposed algorithm is quadratic in the number of tasks and they discussed its centralized and decentralized implementations.

4.3 Allocation of Communication and Computing Resources

Most of the recent works considered edge computing systems in which multiple wireless devices share both communication and computing resources when offloading their tasks. These works provide general models of sharing edge computing resources and thus the results obtained in these works can be easily extended to simpler models in which only one type of the resources is shared. Minimization of the task completion time [Yan+17; Gao+19; Ren+19; Zha+18; Guo+19; EDF17] and minimization of the energy consumption [You+16; TL17; Bar+16; SSB15; AlS+17; ZF19] received significant attention in the literature. Majority of the works from this category considered the joint minimization of the task completion time and the energy consumption [Yan+19; Pha+19b; TP18; Lyu+16; Zha+17; CLD18; Du+18; EL19]. Finally, the problem of maximizing the system throughput received considerable attention [Tär+17; Yan+18; LOD18; Lyu+17; Aya+19]. A summary of works is shown in Table 4.4.

Completion time minimization

There is a significant body of works that proposed centralized solutions to the offline [Yan+17; Gao+19; Ren+19] and the online [Zha+18; Guo+19; EDF17] completion time minimization problems, respectively. In [Yan+17] the authors extended the model from [Yan+15] to consider not only the allocation of computing, but also the allocation of communication resources and by following the approach proposed in [Yan+15] they provided a heuristic for the task placement problem. The authors in [Gao+19] considered an edge computing system that consists of multiple edge clouds. They formulated the joint offline network selection and service placement optimization problem with the objective to minimize the long-term average task completion times under constraints on the available communication and computing resources. The authors decomposed the problem into a sequence of one-shot problems, for which they provided the NP-hardness proof. They proposed an iteration-based algorithm for solving the sequence of one-shot problems

Table 4.4: Classification of works that model the congestion on communication and computing resources.

Paper	**Design objective**			**Information availability**		**System components**			**Management architecture**		
	Time	Energy	Throughput	Offline	Online	D2D	Edge cloud	Remote cloud	Centralized	Decentralized	Distributed
[Yan+17]	✓			✓			✓		✓		
[Gao+19]	✓			✓			✓		✓		
[Ren+19]	✓			✓			✓	✓	✓		
[Zha+18]	✓				✓		✓		✓		
[Guo+19]	✓				✓		✓	✓	✓		
[EDF17]	✓				✓		✓	✓	✓	✓	✓
Paper D [JD19b]	✓			✓			✓			✓	
Paper E [JD20b]	✓			✓			✓			✓	
[You+16]		✓		✓			✓		✓		
[TL17]		✓		✓		✓	✓		✓		
[Bar+16]		✓		✓		✓	✓	✓	✓		
[SSB15]		✓		✓			✓			✓	
[AlS+17]		✓		✓			✓			✓	
[ZF19]		✓		✓			✓				✓
[Yan+19]	✓	✓		✓			✓		✓		
[Pha+19b]	✓	✓		✓			✓		✓		
[TP18]	✓	✓		✓			✓		✓		
[Lyu+16]	✓	✓		✓			✓			✓	
[Zha+17]	✓	✓		✓			✓			✓	
[CLD18]	✓	✓		✓			✓	✓	✓		
[Du+18]	✓	✓		✓			✓	✓	✓		
[EL19]	✓	✓			✓		✓	✓	✓		
Paper B [JD19c]	✓	✓		✓			✓			✓	
Paper C [JD20a]	✓	✓		✓			✓			✓	
[Tär+17]			✓	✓			✓		✓		
[Yan+18]			✓	✓			✓	✓	✓		
[LOD18]			✓	✓	✓		✓	✓	✓		
[Lyu+17]			✓		✓		✓			✓	
[Aya+19]			✓		✓		✓			✓	

and they proved that the proposed algorithm has a bounded competitive ratio. The authors in [Ren+19] considered a system in which the communication resources are shared according to TDMA protocol. They assumed that the task of each device can be split into two parts that can be executed in an edge and a remote cloud, respectively. The authors formulated the problem of minimizing the completion time of all tasks under constraints on the overall communication and computing

resources. They then decomposed the original problem into the problem of allocating communication resources and the joint computing resource allocation and task splitting problem. The authors provided closed form solutions for the decomposed problems and proposed a resource allocation algorithm that can be implemented in a centralized manner.

The authors in [Zha+18] considered a single edge cloud computing system with limited communication, computing and storage resources. They assumed that the computational tasks can be executed locally or can be offloaded to the edge cloud; in order to execute a task, the edge cloud needs the task specific content (e.g., program code) that can be either stored locally in the edge cloud or requested from the Internet through the backhaul link. The authors modeled the popularity of the requested contents as a Zipf distribution and formulated the joint computation offloading, content caching, and resource allocation problem. Based on the generalized benders decomposition, they proposed an iterative polynomial complexity algorithm for solving the problem. The authors in [Guo+19] considered a three-tier edge computing system that consists of an edge cloud with a single server and a remote cloud with abundant computing resources. They considered the problem of joint optimization of the task placement and the allocation of uplink and downlink communication resources. The authors transformed the original problem into a piecewise convex problem and proposed an algorithm that minimizes the average completion times of the tasks. The authors in [EDF17] considered a system that consists of multiple sensors capturing periodically a sequence of frames that can be divided into multiple slices intended for offloading to the multiple processing nodes. The authors proved that the considered multi-sensor completion time minimization problem is NP-hard and they proposed centralized, decentralized and distributed solutions to the problem. The proposed centralized solution has a bounded approximation ratio and it allows a central entity to decide how the frames should be sliced and where the slices should be assigned for processing. According to the proposed decentralized solution, a central coordinator periodically optimizes the frame slicing and the slice placement decisions and the sensors are allowed to update only the frame slicing decisions. Finally, according to the proposed fully distributed solution, each sensor uses its local information to decide how to slice its own frames and where to process the slices.

Our work presented in Paper D [JD19b] and Paper E [JD20b] falls into the class of works that consider offline completion time minimization problems. In Paper D we considered an edge computing system that consists of a set of access points, a set of edge clouds and multiple wireless devices that can either perform their tasks locally or offload them to an edge cloud through a single access point. In Paper E we considered the same edge computing infrastructure under network slicing.

We used $\mathcal{A}$, $|\mathcal{A}| = A$ to denote the set of access points, $\mathcal{C}$, $|\mathcal{C}| = C$ to denote the set of edge clouds, and $\mathcal{S}$, $|\mathcal{S}| = S$ to denote the set of slices. In both papers we considered the joint task placement and resource allocation problem, which can be formulated as the following mixed-integer program

$$\min_{\mathbf{X},\mathbf{Y},\mathcal{P}_{\mathbf{X}},\mathcal{P}_{\mathbf{Y}}} \sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{N}} \Big(y_{i,i,s} T_i^l + \sum_{a\in\mathcal{A}}\sum_{c\in\mathcal{C}} x_{i,a,s} y_{i,c,s} T_{i,a}^c(\mathbf{X},\mathbf{Y},\mathcal{P}_{\mathbf{X}},\mathcal{P}_{\mathbf{Y}}) \Big) \tag{4.4}$$

$$\text{s.t.} \quad \sum_{s\in\mathcal{S}} \Big(y_{i,i,s} + \sum_{a\in\mathcal{A}}\sum_{c\in\mathcal{C}} x_{i,a,s} y_{i,c,s} \Big) = 1, \forall i \in \mathcal{N}, \tag{4.5}$$

$$\mathbf{X} \in \{0,1\}^{N\times A\times S}, \tag{4.6}$$

$$\mathbf{Y} \in \{0,1\}^{N\times (C+1)\times S}, \tag{4.7}$$

$$\mathcal{P}_{\mathbf{X}} : \mathbf{X} \to \mathbb{R}_{[0,1]}^{N\times A\times S}, \tag{4.8}$$

$$\mathcal{P}_{\mathbf{Y}} : \mathbf{Y} \to \mathbb{R}_{[0,1]}^{N\times (C+1)\times S}, \tag{4.9}$$

where the constraint (4.5) enforces that each device either performs the computation locally or it offloads the task through an access point to an edge cloud in a single slice, constraints (4.6) and (4.7) specify that the task placement decisions are integer variables, and constraints (4.8) and (4.9) describe policies for allocating communication and computing resources, respectively.

In both papers we considered that communication and computing resources are managed by a single network operator that decides about resource allocation policies. In Paper D we considered that the selfish devices make their own offloading decisions based on the information provided by the network operator that has a function of a central coordinator. Hence, in Paper D instead of solving (4.4)-(4.9) we used game theoretical tools to model and analyze the interaction between the network operator and the devices in the case of a single slice. We formulated the problem as a Stackelberg game in which the devices are leaders and the network operator is a follower and proved that the game has a subgame perfect equilibrium. We provided closed form solutions for the resource allocation policies, proposed a decentralized algorithm for computing the offloading decisions of the devices and proved that the algorithm has a bounded approximation ratio. In Paper E, we extended the model to consider a system with multiple slices and proved that the problem (4.4)-(4.9) is NP-hard. We provided closed form solutions for the inter-slice and intra-slice resource allocation policies and proposed an efficient approximation algorithm for solving the problem. The proposed algorithm can be implemented in a centralized manner in accordance with the proposed implementation of the network slicing technology [Ord+17; Red+19].

Energy consumption minimization

A large body of works addressed offline minimization of the energy consumption in a variety of edge computing infrastructures [You+16; TL17; Bar+16; SSB15; AlS+17; ZF19]. The authors in [You+16] considered a system that consists of an edge cloud that can be accessed through a single base station. They assumed that each task can be partitioned into a part for the local execution and a part

for offloading and they formulated two joint task partitioning and resource allocation problems for TDMA and OFDMA protocols, respectively. The authors first addressed the problems under the assumption that the system performance is independent of the congestion on edge computing resources. Under this assumption, they provided an optimal and a heuristic solution for minimizing the sum of energy consumptions of devices in the case of TDMA and OFDMA protocols, respectively. The authors then extended the model to consider the congestion on computing resources as well, and in the case of TDMA protocol they proposed a heuristic algorithm. In all cases, the proposed algorithms are designed for a centralized implementation. The authors in [TL17] considered an edge computing system in which each device can perform the computation locally or can offload it to an edge cloud or to a neighbouring device. They formulated the joint computation offloading and resource allocation problem under constraints on the latency and the amount of available communication and computing resources. Based on the successive convex approximation method, the authors proposed a centralized algorithm that finds a local optimal solution to the original problem by iteratively solving the sequence of approximated convex subproblems. In [Bar+16] the authors considered an IoT-cloud system that consists of multiple computing nodes that can be end users' devices, access points or edge clouds, possibly distributed across different hierarchical layers. They modeled the entire system as a directed graph and formulated the problem of minimizing the energy consumption of the system under constraints on the shared communication and computing resources and constraints on the latency, the reliability and the battery life time of the devices. The authors used linear programming to develop a centralized solution for splitting service flows over multiple paths.

Several works considered that devices cannot perform their tasks locally and that each device is associated with exactly one among multiple base stations [SSB15; AlS+17; ZF19]. In [SSB15; AlS+17] the authors considered that base stations are connected to the common edge cloud and they formulated the problem of optimizing the transmit powers of devices under delay constraints. They proposed an iterative algorithm for finding locally optimal allocations of communication and computing resources. The algorithm can be implemented in a decentralized manner and it requires limited signaling between base stations and the cloud. In [AlS+17] the authors extended the model from [SSB15] to consider not only the transmissions between devices and base stations, but also the transmissions between base stations and the cloud and they used the same approach as in [SSB15] to solve the extended version of the problem. The authors in [ZF19] considered that each base station is equipped with computing resources and assumed that each device is associated to the base station with the best signal-to-noise ratio. They formulated the problem of minimizing the sum of transmission powers of all devices under constraints on the minimum data rate requirements of devices and constraints on the available communication and computing resources. Based on the decomposition of the original problem, the authors proposed an iterative algorithm for computing optimal vectors of transmission powers and shares of communication and computing resources.

The algorithm can be implemented in a distributed way and it requires limited information exchange between the base stations.

Completion time and energy consumption minimization

Many works that considered offline optimization of computation offloading defined the optimization objective as a linear combination of two functions that are defined with respect to the task completion time and the energy consumption, respectively [Yan+19; Pha+19b; TP18; Lyu+16; Zha+17; CLD18; Du+18]. The authors in [Yan+19] considered a system in which the devices use a non-orthogonal multiple access technique for offloading their tasks to a single edge cloud. They formulated the problem of minimizing the sum of costs of all devices under constraints on task completion times, transmission powers, offloading data rates and available computing resources. The authors proposed a heuristic for computing shares of communication and computing resources and transmission powers of the devices and for partitioning the tasks into parts for offloading and parts for local executions. Next, they simplified the model to consider the completion time minimization problem only and showed that in this case the optimal solution can be obtained. The proposed algorithms are designed for a centralized implementation. The authors in [Pha+19b] considered a system that consists of a small base station that is connected to an edge cloud through a backhaul link. The authors assumed the communication model according to which a fraction of the total bandwidth is allocated for communication between the base station and the edge cloud and the rest of the bandwidth is shared among the offloading devices. The authors formulated the problem of minimizing the sum of costs of all devices and they decomposed the original problem into the task placement and the joint backhaul bandwidth and computing resource allocation problem. They developed a heuristic that solves these two problems iteratively in a centralized manner. The authors in [TP18] considered a system that consists of multiple base stations equipped with computing resources and they assumed that the devices are interested in maximizing their offloading benefits defined with respect to the local computing costs. They formulated the problem of maximizing the weighted sum of offloading benefits of devices under constraints on the maximum allowed transmission powers and constraints on the computing capabilities of the edge clouds. They decomposed the problem into the task offloading and the resource allocation problems and proposed a polynomial time heuristic for solving the original problem in a centralized manner. The authors in [Lyu+16] considered a system with a single edge cloud to which the devices can offload their tasks through a single shared wireless link. They proposed a heuristic that computes offloading decisions of devices, the allocation of computing resources and the uplink transmission powers. The algorithm is designed to minimize the sum of costs of all devices under constraints on the available communication and computing resources and it can be implemented in a decentralized manner. In [Zha+17] the authors assumed that there is a finite number of orthogonal subcarriers that the devices can use for communication with a

single edge cloud. The authors provided a game theoretical treatment of the problem according to which devices decide about their own offloading decisions and an edge cloud coordinator computes the policies for assigning transmission powers and allocating computing resources with the objective to minimize the total cost of all offloading devices. The proposed algorithm can be implemented in a decentralized manner and it relies on information exchange between the devices and the edge cloud coordinator.

Several works proposed centralized solutions to the offline computation offloading problems in three-tier edge computing systems in which the devices can perform their tasks locally or offload them to an edge or to a remote cloud [CLD18; Du+18; EL19]. The authors in [CLD18] formulated the problem of minimizing the sum of costs of devices under constraints on the available communication and computing resources. Based on the relaxation of the original problem, they proposed a heuristic for computing the offloading decisions of devices and allocating communication and computing resources. In [Du+18] the authors formulated the problem of minimizing the maximum cost among all devices under constraints on the task execution times, the available bandwidth and computing resources, and on the allowed transmission powers. By using semidefinite relaxation, fractional programming theory and Lagrangian dual decomposition, the authors designed a centralized heuristic for computing the task placement, the assignment of transmission powers and the allocation of communication and computing resources. The authors in [EL19] considered the same cost model as in the above works, but unlike these works they addressed the online task placement and resource allocation problem in a system in which the exact task complexities are unknown. Based on the relaxation of the problem, the authors proposed a centralized heuristic for computing offloading decisions of the wireless devices, shares of communication resources and rates at which the tasks should be executed locally and in the edge cloud.

Our work presented in Paper B [JD19c] and Paper C [JD20a] falls into the class of works that consider offline completion time and energy consumption minimization problems. We considered an edge computing system that consists of a set of access points, one edge cloud and multiple wireless devices that compete for both communication and computing resources. In Paper B we considered the task placement problem in which the tasks can be performed locally or they can be offloaded to the edge cloud through one of multiple access points. In Paper C we integrated the task placement problem into the scheduling problem by allowing devices not only to choose where to perform their tasks, but also in which time slot.

In both papers we defined the local execution cost C_i^l and the offloading cost $C_{i,a}^e$ of device $i \in \mathcal{N}$ as a linear combination of the task completion time and the energy consumption of the device, respectively

$$\textit{local execution:} \quad C_i^l = \gamma_i^T T_i^l + \gamma_i^E E_i^l,$$

$$\textit{offloading:} \quad C_{i,a}^c = \gamma_i^T T_{i,a}^c + \gamma_i^E E_{i,a}.$$

We used $\mathcal{A}$, $|\mathcal{A}| = A$ to denote the set of access points and $\mathcal{T}$, $|\mathcal{T}| = T$ to denote the set of time slots. Using this notation, the considered task placement problem can be defined as the following $0-1$ integer program

$$\min_{\mathbf{X},\mathbf{Y}} \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}} \Big(y_{i,i,t} C_i^l + \sum_{a\in\mathcal{A}} (1 - y_{i,i,t}) x_{i,a,t} C_{i,a}^{c,t}(\mathbf{X},\mathbf{Y}) \Big) \tag{4.10}$$

$$\text{s.t.} \quad \sum_{t\in\mathcal{T}} \Big(y_{i,i,t} + \sum_{a\in\mathcal{A}} (1 - y_{i,i,t}) x_{i,a,t} \Big) = 1, \forall i \in \mathcal{N}, \tag{4.11}$$

$$\mathbf{X} \in \{0,1\}^{N\times A\times T}, \tag{4.12}$$

$$\mathbf{Y} \in \{0,1\}^{N\times 2\times T}, \tag{4.13}$$

where the constraint (4.11) enforces that each device either performs the computation locally or it offloads the task through an access point in one of the time slots, and constraints (4.12) and (4.13) specify that the task placement decisions are integer variables.

In both papers we considered selfish devices that make their own offloading decisions and aim at minimizing their own costs. Therefore, instead of solving (4.10)-(4.13) we used game theoretical tools to analyze the task placement problem in the case of a single time slot (Paper B), and in the case of multiple time slots (Paper C). We defined the task placement problems as player-specific network congestion games, for which the existence of equilibria is not known in general. We proved the existence of equilibrium task assignments, and based on our constructive proofs we provided decentralized algorithms with bounded approximation ratios that can compute equilibrium assignments of tasks in polynomial time. By providing constructive equilibrium existence proofs and by characterizing the structure of an equilibrium task assignment, our work presented in Paper B and Paper C is also important from a game theoretical perspective.

Throughput Maximization

The throughput maximization problem has been addressed as an offline [Tär+17; Yan+18] and an online computation offloading problem [LOD18; Lyu+17; Aya+19]. The authors in [Tär+17] modeled an edge computing system as a tree graph, where the vertices are computing nodes and the edges are network links, each with limited amount of resources. They addressed the task placement problem from the perspective of an infrastructure provider that aims at minimizing its operational cost, which is defined as a function of the system throughput. The authors formulated the offline computation offloading problem and discussed two methods for solving the problem periodically in a centralized manner. The first method is an exhaustive search algorithm that is impractical due to its exponential complexity, and the second method is a tractable iterative search algorithm that computes a locally optimal task placement assignment. The authors in [Yan+18] considered a three-tier edge computing system in which the devices can perform their tasks

locally or can offload them to one of multiple edge clouds or to a remote cloud. They considered a slotted time model and assumed regular mobility patterns for multiple mobile devices. Assuming that a set of future devices is known and that each computational task can be modeled as a directed acyclic graph with multiple subtasks, the authors addressed offline maximization of the system throughput under constraints on the limited communication and time-shared edge computing resources. Based on Lagrange relaxation method, the authors proposed a polynomial time centralized heuristic for assigning subtasks and allocating bandwidth resources in a centralized manner.

The authors in [LOD18] considered a similar three-tier edge computing system and formulated the problem of maximizing the number of tasks that are processed by edge servers under constraints on limited bandwidth and edge computing resources. Based on the deep-learning approach, they proposed offline and online centralized task placement algorithms for which they provided approximation ratios. The authors in [Lyu+17] considered a system that consists of multiple time-shared wireless channels, a single edge cloud and multiple battery powered IoT devices equipped with energy harvesting capabilities. They assumed a time-slotted system and modeled the task arrivals, the energy harvesting and the per-slot availability of edge computing resources as three independent stochastic processes. The authors provided asymptotic analysis of proportionally fair scheduling of computational tasks and used Lyapunov optimization techniques to design an algorithm for maximizing the total amount of admitted data of all devices. They proposed a decentralized implementation of the algorithm that requires information exchange between the devices and the edge cloud. In [Aya+19] the authors proposed a framework according to which the radio and computing resources are managed in a decentralized way by multiple low-level schedulers over a short-time scale and by the main resource manager over a larger time scale. The objective of the main resource manager is to maximize the system throughput and to minimize the overall latency and the operational costs in the system. The authors modeled the resource management problem as a contextual bandit problem and used reinforcement learning technique to design a resource manager that learns the behaviour of low-level schedulers and manages them accordingly.

Chapter 5

Summary of Original Work

Paper A: Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems

Slađana Jošilo and György Dán

IEEE/ACM Transactions on Networking (ToN), vol. 27, no. 1, pp. 85-97, 2019.

Summary: In this paper we consider an edge computing system where multiple devices may offload their computational tasks to each other or to an edge cloud with the objective to improve their performance. We consider that devices are interested in minimizing the completion time of their own tasks, and we formulate the problem as a strategic game where each device plays a mixed strategy. We use variational inequality theory to prove the existence of an equilibrium task allocation in static mixed strategies, which we use to design an efficient algorithm for allocating the computational tasks in a decentralized way. The algorithm is based on average system parameters only, and thus it requires low signaling overhead. We perform simulations to evaluate the proposed algorithm, and we show that it achieves good system performance close to that of the greedy algorithm, which requires the full information about the system state, and close to that of an algorithm that allocates the tasks based on the socially optimal static mixed strategy.

Contribution: The author of this thesis developed the analytical model in collaboration with the second author of the paper. The author of this thesis proved the analytical results concerning the existence of static mixed strategy equilibrium, and carried out the simulations. The analysis of the resulting data was carried out in collaboration with the second author of the paper. The paper was written in collaboration with the second author.

Paper B: Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks

Slađana Jošilo and György Dán

IEEE Transactions on Mobile Computing (TMC), vol.18, no. 1, pp. 207-220, 2019. A shorter version of the paper appeared in Proc. of IEEE INFOCOM 2017.

Summary: In this paper we consider an edge computing system that consists of multiple access points, an edge cloud and multiple wireless devices that can either perform the computation locally or offload the computation to the edge cloud. We consider that each wireless device is selfish, and thus that it aims at minimizing its own cost, which we define as a linear combination of the time it takes to complete the computation and the corresponding energy consumption. In order to analyze interactions among wireless devices, we formulate the problem as a player-specific congestion game in which devices compete for communication and computing resources when offloading their tasks. We prove that a pure Nash equilibrium of the game exists, and we provide a polynomial complexity decentralized algorithm for computing it. We establish a bound on the price of anarchy of the game, and thus we show that the proposed algorithm has a bounded approximation ratio. We use extensive simulations to provide insight into the cost performance and the computational complexity of the proposed algorithm. We show that the proposed algorithm achieves the cost performance close to that of the optimal solution, and that the convergence time of the algorithm scales approximately linearly with the number of wireless devices.

Contribution: The author of this thesis developed the analytical model in collaboration with the second author of the paper. The author of this thesis proved the analytical results for the case of both elastic and non-elastic cloud models. The implementation of the simulations was carried out by the author of this thesis, and analysis of the resulting data was carried out in collaboration with the second author of the paper. The paper was written in collaboration with the second author.

Paper C: Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing

Slađana Jošilo and György Dán

IEEE/ACM Transactions on Networking (ToN), vol. 28, no. 2, pp. 667-680, 2020.

A shorter version of the paper appeared in Proc. of IFIP Networking 2018.

Summary: In this paper we consider periodic computation offloading problem in an edge computing system that serves multiple wireless devices. Each device can choose in which of multiple time slots to perform the computation, and within the time slot it can choose to perform its task locally or to offload the task to an edge cloud via one of multiple access points. The objective of each device is to minimize a linear combination of the time it takes to complete the computation and the corresponding energy consumption. We formulate the problem as a player-specific congestion game, and we prove the existence of pure strategy Nash equilibria. Based on the constructive equilibrium existence proof, we develop an efficient decentralized algorithm for computing an equilibrium of offloading decisions. We characterize the structure of computed equilibria, and by providing an upper bound on the price of anarchy of the game we establish an asymptotically tight bound on the approximation ratio of the proposed algorithm. Our numerical results show that the proposed algorithm can be used to compute an equilibrium task placement at polynomial computational complexity despite combinatorial nature of the problem. Finally, the results show that the algorithm computes equilibria with significant performance gain compared to the task placement that is uncoordinated over time.

Contribution: The author of this thesis developed the analytical model in collaboration with the second author of the paper. The second author proved the analytical results concerning the case of a single time slot, and the author of this thesis proved the analytical results concerning the case of multiple time slots. The implementation of the simulations and the analysis of the resulting data were carried out by the author of this thesis. The paper was written in collaboration with the second author.

Paper D: Joint Management of Wireless and Computing Resources for Computation Offloading in Mobile Edge Clouds

Slađana Jošilo and György Dán

IEEE Transactions on Cloud Computing (TCC), pp. 1-14, 2019.

A shorter version of the paper appeared in Proc. of IEEE INFOCOM 2019.

Summary: In this paper we consider an edge computing system that consists of multiple access points and multiple edge clouds. Each device can choose to perform its task locally or to offload the task to one of multiple edge clouds via one of multiple access points with the objective to minimize the computational time of its own task. We formulate the problem as a Stackelberg game in which the devices as leaders make their own offloading decisions and the network operator as a follower manages communication and computing resources. We prove the existence of a subgame perfect equilibria of the game and we provide a closed form solution for the optimal cost minimization resource allocation policies of the operator. We consider the interactions between devices under the optimal cost minimization and the time fair allocation policies, and we show that they can be modeled as a weighted congestion game with resource specific weights and a player-specific congestion game, respectively. We provide decentralized algorithms for computing equilibrium offloading decisions of the devices in both games, and by establishing the upper bounds on the price of anarchy of the games we prove that the proposed algorithms have bounded approximation ratios. Our numerical results show that the cost minimization policy can achieve significantly lower completion times compared to that of the time fair allocation policy, and that the convergence times of the proposed task assignment algorithms are approximately linear in the number of devices.

Contribution: The author of this thesis developed the analytical model in collaboration with the second author of the paper. The author of this thesis proved the analytical results concerning the optimal cost minimization and the time fair resource allocation policies, and the results concerning the existence of the task placement equilibria. The implementation of the simulations and the analysis of the resulting data were carried out by the author of this thesis. The paper was written in collaboration with the second author.

Paper E: Joint Wireless and Edge Computing Resource Management with Dynamic Network Slice Selection

Slađana Jošilo and György Dán

submitted to IEEE/ACM Transactions on Networking (ToN).

Summary: In this paper we consider an edge computing system under network slicing. The system consists of multiple edge clouds, multiple access points and a network operator that manages communication and computing resources within and across slices, and assigns the tasks generated by multiple wireless devices. Each task can be either performed locally or assigned to exactly one slice, one access point and one edge cloud in the case of offloading. We formulate the problem as a mixed-integer program in which the objective is to minimize completion times of the tasks, and we prove that the problem is NP-hard. We provide closed form solutions for the optimal policies for managing resources within and across slices, and based on a game theoretic treatment of the problem we propose an efficient task placement algorithm with a bounded approximation ratio. Our numerical results show that the proposed policies for sharing resources within and across slices can improve the system performance compared to no slicing and compared to equal slicing and that the convergence time of the proposed task assignment algorithm is approximately linear in the number of devices.

Contribution: The author of this thesis developed the analytical model in collaboration with the second author of the paper. The author of this thesis proved the analytical results concerning the NP-hardness of the problem, optimal resource allocation policies, and the task placement algorithm. The implementation of the simulations and the analysis of the resulting data were carried out by the author of this thesis. The paper was written in collaboration with the second author.

Publications not included in the thesis

1. Slađana Jošilo and György Dán. "Wireless and Computing Resource Allocation for Selfish Computation Offloading in Edge Computing". In: *Proc. of IEEE INFOCOM*. 2019, pp. 2467–2475

2. Slađana Jošilo and György Dán. "Joint allocation of computing and wireless resources to autonomous devices in mobile edge computing". In: *Proc. of MECOMM SIGCOMM*. 2018, pp. 13–18

3. Slađana Jošilo and György Dán. "Decentralized scheduling for offloading of periodic tasks in mobile edge computing". In: *Proc. of IFIP Networking*. 2018, pp. 1–9

4. Slađana Jošilo and György Dán. "A game theoretic analysis of selfish mobile computation offloading". In: *Proc. of IEEE INFOCOM*. 2017, pp. 1–9

5. Slađana Jošilo, Valentino Pacifici, and György Dán. "Distributed Algorithms for Content Placement in Hierarchical Cache Networks". In: *Elsevier Comput. Netw.* 125 (2017), pp. 160–171

6. Valentino Pacifici, Slađana Jošilo, and György Dán. "Distributed algorithms for content caching in mobile backhaul networks". In: *Proc. of ITC*. 2016, pp. 313–321

7. Slađana Jošilo et al. "Multicarrier waveforms with I/Q staggering: uniform and nonuniform FBMC formats". In: *Springer EURASIP J ADV SIG PR* 184.1 (2014), p. 167

8. Slađana Jošilo et al. "Widely linear filtering based kindred co - channel interference suppression in FBMC waveforms". In: *Proc. of IEEE ISWCS*. 2014, pp. 776–780

9. Slađana Jošilo and et al. "Non - uniform FBMC-a pragmatic approach". In: *Proc. of IEEE ISWCS*. 2013, pp. 1–5

Chapter 6

Conclusion and Future Work

In this thesis, we considered task placement and resource allocation problems in edge computing systems. We analyzed the problems from the perspective of the end users and from the perspective of the network operator. By using game theoretical tools, we designed efficient and scalable algorithms for managing the resources and for allocating the computational tasks in various edge computing infrastructures.

In the first part of this thesis, we considered a distributed edge computing system in which wireless devices can offload their tasks to an edge cloud and to each other. We modeled the transmission and the execution of computational tasks using queuing theory, and provided a game theoretical formulation of the task placement problem. We used variational inequality theory to address the question whether the devices can compute an efficient equilibrium task placement in static mixed strategies in a decentralized manner. We showed that an efficient decentralized solution can be developed, under the assumption that every device knows only the average statistics on task arrival intensities, transmission rates, and task parameters.

In the second part of the thesis, we considered an edge computing system in which wireless devices can offload their computational tasks to an edge cloud through one of multiple wireless links. We formulated the task placement problem as a strategic game and we investigated whether there is an efficient decentralized algorithm for computing a pure Nash equilibrium of the game. Furthermore, we extended our model to consider offloading of periodic tasks in the case of homogeneous task periodicities. We addressed the question whether a pure Nash equilibrium exists if devices can choose not only where to perform their tasks, but also in which time slot. In both cases, we showed that efficient decentralized task placement algorithms can be developed using game theoretic tools.

In the third part of the thesis, we considered an edge computing system in which wireless devices can offload their tasks to one of multiple edge clouds through one of multiple wireless links. We formulated the joint task placement and resource allocation problem as a Stackelberg game played by devices that decide about the

offloading strategies and by the network operator that decides about the resource allocation policies. We addressed two questions concerning the resource allocation policies and the task offloading strategies, respectively. First, we addressed the question whether the optimal completion time minimization policy for allocating communication and computing resources exists. Second, we addressed the question whether an equilibrium task placement exists under the completion time minimization and the time fair resource allocation policies, respectively. We then extended our model to consider the same edge computing system under network slicing, and addressed the question whether the results still hold in the case of multiple network slices. In both cases, we showed that the optimal completion time minimization policy can be expressed in closed form, and that an equilibrium task placement exists in all considered cases.

There are many open questions concerning the task placement and the resource allocation problems in edge computing systems. The first interesting question is whether efficient decentralized task placement algorithms exist in a system in which devices do not have the information about the resource allocation policies implemented by the network operator. The second interesting question is whether the network operator can allocate resources efficiently in the case of a system model which would allow for less signaling between the devices and the cloud/mobile network (e.g., in a system where the actual number of users and the characteristics of their devices and computational tasks are unknown to the network operator). The third interesting question is whether our results from the second part of the thesis can be extended to the case of heterogeneous tasks periodicities. Finally, the fourth interesting question is whether the proposed solutions can be extended to non-atomic models of computational tasks.

Bibliography

[AGH16] Khadija Akherfi, Micheal Gerndt, and Hamid Harroud. "Mobile cloud computing for computation offloading: Issues and challenges". In: *Applied Computing & Informatics* (2016).

[All15] NGMN Alliance. "5G White Paper". In: *White Paper* (2015).

[AlS+17] Ali Al-Shuwaili et al. "Joint uplink/downlink optimization for backhaul-limited mobile cloud computing with user scheduling". In: *IEEE T SIGNAL INF PR* 3.4 (2017), pp. 787–802.

[Aya+19] Jose A Ayala-Romero et al. "vrAIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs". In: *Proc. of ACM MobiCom.* 2019, pp. 1–16.

[Ba09] Niranjan Balasubramanian and et al. "Energy consumption in mobile phones: a measurement study and implications for network applications". In: *Proc. of ACM SIGCOMM.* 2009, pp. 280–293.

[Ban+19] Albert Banchs et al. "A 5G Mobile Network Architecture to Support Vertical Industries". In: *IEEE Commun. Mag.* 57.12 (2019), pp. 38–44.

[Bar+16] Marc Barcelo et al. "IoT-cloud service optimization in next generation smart environments". In: *IEEE JSAC* 34.12 (2016), pp. 4077–4090.

[BD14] Nik Bessis and Ciprian Dobre. "Big data and internet of things: a roadmap for smart environments". In: *Springer.* Vol. 546. 2014.

[Bia+09] Alessandro Biagioni et al. "Adaptive subcarrier allocation schemes for wireless OFDMA systems in WiMAX networks". In: *IEEE JSAC* 27.2 (2009), pp. 217–225.

[Bia00] Giuseppe Bianchi. "Performance analysis of the IEEE 802.11 distributed coordination function". In: *IEEE JSAC* 18.3 (2000), pp. 535–547.

[Bon+12] Flavio Bonomi et al. "Fog computing and its role in the internet of things". In: *Proc. of ACM, MCC Workshop.* 2012, pp. 13–16.

[Bon+14] Flavio Bonomi et al. “Fog computing: A platform for internet of things and analytics”. In: *Springer*. 2014, pp. 169–186.

[BZ18] Suzhi Bi and Ying Jun Zhang. “Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading”. In: *IEEE TWC* 17.6 (2018), pp. 4177–4190.

[Car+16] Valeria Cardellini et al. “A game-theoretic approach to computation offloading in mobile cloud computing”. In: *Springer MATH PROGRAM* 157.2 (2016), pp. 421–449.

[CC17] Huijin Cao and Jun Cai. “Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach”. In: *IEEE TVT* 67.1 (2017), pp. 752–764.

[CH18] Min Chen and Yixue Hao. “Task offloading for mobile edge computing in software defined ultra-dense network”. In: *IEEE JSAC* 36.3 (2018), pp. 587–597.

[Cha+17] Zheng Chang et al. “Energy efficient optimization for computation offloading in fog computing system”. In: *Proc. of IEEE GLOBECOM*. 2017, pp. 1–6.

[Che+16] Xu Chen et al. “Efficient multi-user computation offloading for mobile-edge cloud computing”. In: *IEEE/ACM ToN* 24.5 (2016), pp. 2795–2808.

[Che15] Xu Chen. “Decentralized computation offloading game for mobile cloud computing”. In: *IEEE TPDS* 26.4 (2015), pp. 974–983.

[Chu+11] Byung-Gon Chun et al. “Clonecloud: elastic execution between mobile device and cloud”. In: *Proc. of ACM EuroSys*. 2011, pp. 301–314.

[Cis17] Cisco. “Visual Networking Index: Global Mobile Data Traffic Forecast Update”. In: *Tech.rep* (2017).

[Cis20] Cisco. “Cisco Annual Internet Report (2018–2023)”. In: *White Paper* (2020).

[CL17] Jaya Prakash Champati and Ben Liang. “Single restart with time stamps for computational offloading in a semi-online setting”. In: *Proc. of IEEE INFOCOM*. 2017, pp. 1–9.

[CL19] Jaya Prakash Varma Champati and Ben Liang. “Delay and cost optimization in computational offloading systems with unknown task processing times”. In: *IEEE TCC* (2019).

[CLD18] Meng-Hsi Chen, Ben Liang, and Min Dong. “Multi-user multi-task offloading and resource allocation in mobile cloud systems”. In: *IEEE TWC* 17.10 (2018), pp. 6790–6805.

[Cue+10] Eduardo Cuervo et al. “MAUI: Making Smartphones Last Longer with Code Offload”. In: *Proc. of ACM MobiSys*. 2010, pp. 49–62.

[CZ16] Mung Chiang and Tao Zhang. "Fog and IoT: An overview of research opportunities". In: *IEEE IoT* 3.6 (2016), pp. 854–864.

[CZ17] Xu Chen and Junshan Zhang. "When D2D meets cloud: Hybrid mobile task offloadings in fog computing". In: *Proc. of IEEE ICC*. 2017, pp. 1–6.

[De16] Debashis De. *Mobile cloud computing: architectures, algorithms and applications*. CRC Press, 2016.

[Den+16] Ruilong Deng et al. "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption". In: *IEEE IoT* 3.6 (2016), pp. 1171–1181.

[Du+18] Jianbo Du et al. "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee". In: *IEEE TCOM* 66.4 (2018), pp. 1594–1608.

[EDF17] Emil Eriksson, György Dán, and Viktoria Fodor. "Coordinating Distributed Algorithms for Feature Extraction Offloading in Multi-Camera Visual Sensor Networks". In: *IEEE TCSVT* (2017).

[EL19] Nima Eshraghi and Ben Liang. "Joint Offloading Decision and Resource Allocation with Uncertain Task Computing Requirement". In: *Proc. of IEEE INFOCOM*. 2019, pp. 1414–1422.

[ETS] The European Telecommunications Standards Institute (ETSI). *Multi-access Edge Computing*. `http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing`.

[Fry17] Danny Frydman. "Mobile Edge Computing (MEC); Bandwidth Management API". In: *ETSI White Paper* v. 1.1.1 (2017).

[Gao+19] Bin Gao et al. "Winning at the starting line: Joint network selection and service placement for mobile edge computing". In: *Proc. of IEEE INFOCOM*. 2019, pp. 1459–1467.

[Ge+12] Yang Ge et al. "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system". In: *Proc. of ACM/IEEE ISLPED*. 2012, pp. 279–284.

[Giu+18] Fabio Giust et al. "MEC deployments in 4G and evolution towards 5G". In: *ETSI White Paper* 24 (2018), pp. 1–24.

[Gro17] Synergy Research Group. "The Leading Cloud Providers Continue to Run Away with the Market". In: *Tech.rep* (2017).

[Guo+16] Songtao Guo et al. "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing". In: *Proc. of IEEE INFOCOM*. 2016, pp. 1–9.

[Guo+19] Kai Guo et al. "Joint Computation Offloading and Bandwidth Assignment in Cloud-Assisted Edge Computing". In: *IEEE TCC* (2019).

[He+10] Qiming He et al. “Case study for running HPC applications in public clouds”. In: *Proc. of ACM HPDC*. 2010, pp. 395–401.

[Heu+03] Martin Heusse et al. “Performance anomaly of 802.11 b”. In: *Proc. of IEEE INFOCOM*. 2003, pp. 836–843.

[Hu+15] Yun Chao Hu et al. “Mobile edge computing—A key technology towards 5G”. In: *ETSI White Paper* 11.11 (2015), pp. 1–16.

[Hu+19] Miao Hu et al. “Learning driven computation offloading for asymmetrically informed edge computing”. In: *IEEE TPDS* 30.8 (2019), pp. 1802–1815.

[Hua+18] Liang Huang et al. “Distributed deep learning-based offloading for mobile edge computing networks”. In: *Springer MOBILE NETW APPL* (2018), pp. 1–8.

[HWN12] D. Huang, P. Wang, and D. Niyato. “A Dynamic Offloading Algorithm for Mobile Computing”. In: *IEEE TWC* 11.6 (2012), pp. 1991–1995.

[Ios+11] Alexandru Iosup et al. “Performance analysis of cloud computing services for many-tasks scientific computing”. In: *IEEE TPDS* 22.6 (2011), pp. 931–945.

[Ja13] Slađana Jošilo and et al. “Non - uniform FBMC-a pragmatic approach”. In: *Proc. of IEEE ISWCS*. 2013, pp. 1–5.

[Jac+10] Keith R Jackson et al. “Performance analysis of high performance computing applications on the amazon web services cloud”. In: *Proc. of IEEE CloudCom*. 2010, pp. 159–168.

[Jan+17] Insun Jang et al. “A Proxy-Based Collaboration System to Minimize Content Download Time and Energy Consumption”. In: *IEEE TMC* 16.8 (2017), pp. 2105–2117.

[JD17] Slađana Jošilo and György Dán. “A game theoretic analysis of selfish mobile computation offloading”. In: *Proc. of IEEE INFOCOM*. 2017, pp. 1–9.

[JD18a] Slađana Jošilo and György Dán. “Decentralized scheduling for offloading of periodic tasks in mobile edge computing”. In: *Proc. of IFIP Networking*. 2018, pp. 1–9.

[JD18b] Slađana Jošilo and György Dán. “Joint allocation of computing and wireless resources to autonomous devices in mobile edge computing”. In: *Proc. of MECOMM SIGCOMM*. 2018, pp. 13–18.

[JD19a] Slađana Jošilo and György Dán. “Decentralized algorithm for randomized task allocation in fog computing systems”. In: *IEEE/ACM ToN* 27.1 (2019), pp. 85–97.

[JD19b] Slađana Jošilo and György Dán. “Joint Management of Wireless and Computing Resources for Computation Offloading in Mobile Edge Clouds”. In: *IEEE TCC* (2019), pp. 1–14.

[JD19c] Slađana Jošilo and György Dán. "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks". In: *IEEE TMC* 18.1 (2019), pp. 207–220.

[JD19d] Slađana Jošilo and György Dán. "Wireless and Computing Resource Allocation for Selfish Computation Offloading in Edge Computing". In: *Proc. of IEEE INFOCOM*. 2019, pp. 2467–2475.

[JD20a] Slađana Jošilo and György Dán. "Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing". In: *IEEE/ACM ToN* 28.2 (2020), pp. 667–680.

[JD20b] Slađana Jošilo and György Dán. "Joint Wireless and Edge Computing Resource Management with Dynamic Network Slice Selection". In: *submitted to IEEE ToN* (2020).

[Jos+08] Tarun Joshi et al. "Airtime fairness for IEEE 802.11 multirate networks". In: *IEEE TMC* 7.4 (2008), pp. 513–527.

[Još+14a] Slađana Jošilo et al. "Multicarrier waveforms with I/Q staggering: uniform and nonuniform FBMC formats". In: *Springer EURASIP J ADV SIG PR* 184.1 (2014), p. 167.

[Još+14b] Slađana Jošilo et al. "Widely linear filtering based kindred co - channel interference suppression in FBMC waveforms". In: *Proc. of IEEE ISWCS*. 2014, pp. 776–780.

[JPD17] Slađana Jošilo, Valentino Pacifici, and György Dán. "Distributed Algorithms for Content Placement in Hierarchical Cache Networks". In: *Elsevier Comput. Netw.* 125 (2017), pp. 160–171.

[Kek+18] Sami Kekki et al. "MEC in 5G networks". In: *ETSI White Paper* 28 (2018), pp. 1–28.

[KL10a] K. Kumar and Y. H. Lu. "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" In: *IEEE Computer Mag.* 43.4 (2010), pp. 51–56.

[KL10b] Karthik Kumar and Yung-Hsiang Lu. "Cloud computing for mobile users: Can offloading computation save energy?" In: *IEEE Computer* 43.4 (2010), pp. 51–56.

[Kum+13] Karthik Kumar et al. "A survey of computation offloading for mobile systems". In: *Springer MOBILE NETW APPL* 18.1 (2013), pp. 129–140.

[LBH09] Tong Li, Dan Baumberger, and Scott Hahn. "Efficient and scalable multiprocessor fair scheduling using distributed weighted round-robin". In: *ACM Sigplan Notices* 44.4 (2009), pp. 65–74.

[Liu+18] Qiang Liu et al. "An edge network orchestrator for mobile augmented reality". In: *Proc. of IEEE INFOCOM*. 2018, pp. 756–764.

[LOD18] He Li, Kaoru Ota, and Mianxiong Dong. "Learning IoT in edge: Deep learning for the Internet of Things with edge computing". In: *IEEE Netw.* 32.1 (2018), pp. 96–101.

[Loe11] Bill Loeffler. "Cloud computing: what is infrastructure as a service". In: *TechNet Magazine* 10 (2011).

[LS01] Jacob R Lorch and Alan Jay Smith. "Improving dynamic voltage scaling algorithms with PACE". In: *Proc. of ACM SIGMETRICS PER.* Vol. 29. 1. 2001, pp. 50–61.

[Lyu+16] Xinchen Lyu et al. "Multiuser joint task offloading and resource optimization in proximate clouds". In: *IEEE TVT* 66.4 (2016), pp. 3435–3447.

[Lyu+17] Xinchen Lyu et al. "Optimal schedule of mobile edge computing for Internet of Things using partial information". In: *IEEE JSAC* 35.11 (2017), pp. 2606–2615.

[LZL11] Erwu Liu, Qinqing Zhang, and Kin K Leung. "Asymptotic analysis of proportionally fair scheduling in Rayleigh fading". In: *IEEE TWC* 10.6 (2011), pp. 1764–1775.

[Mao+17] Yuyi Mao et al. "A survey on mobile edge computing: The communication perspective". In: *IEEE Commun. Surv.* 19.4 (2017), pp. 2322–2358.

[Mar09] Eugene E Marinelli. "Hyrax: cloud computing on mobile devices using MapReduce". In: *Tech.rep* (2009).

[Mav17] Dimitris Mavrakis. "The Evolution of Network Slicing". In: *ABI Research* (2017).

[Mes+17] Erfan Meskar et al. "Energy Aware Offloading for Competing Users on a Shared Communication Channel". In: *IEEE TMC* 16.1 (2017), pp. 87–96.

[MN10] A. P. Miettinen and J. K. Nurminen. "Energy efficiency of mobile clients in cloud computing". In: *Proc. of USENIX HotCloud.* 2010, pp. 4–4.

[Mti+13] Abderrahmen Mtibaa et al. "Towards resource sharing in mobile device clouds: Power balancing across mobile devices". In: *ACM SIGCOMM COMP COM.* 2013, pp. 51–56.

[Net+18] José Leal D Neto et al. "ULOOF: a user level online offloading framework for mobile edge computing". In: *IEEE TMC* 17.11 (2018), pp. 2660–2674.

[OEY11] Stephan Olariu, Mohamed Eltoweissy, and Mohamed F Younis. "Towards autonomous vehicular clouds." In: *EAI Endorsed Trans. Mobile Communications Applications* 1.1 (2011).

[Ord+17] Jose Ordonez-Lucena et al. "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges". In: *IEEE Commun. Mag.* 55.5 (2017), pp. 80–87.

[Ost+09] Simon Ostermann et al. "A performance analysis of EC2 cloud computing services for scientific computing". In: *Proc. of Springer Cloud Computing*. 2009, pp. 115–131.

[Pha+19a] Quoc-Viet Pham et al. "Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation". In: *IEEE Access* 7 (2019), pp. 16444–16459.

[Pha+19b] Quoc-Viet Pham et al. "Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation". In: *IEEE Access* 7 (2019), pp. 16444–16459.

[PJD16] Valentino Pacifici, Slađana Jošilo, and György Dán. "Distributed algorithms for content caching in mobile backhaul networks". In: *Proc. of ITC*. 2016, pp. 313–321.

[Rah+15] M Reza Rahimi et al. "On optimal and fair service allocation in mobile cloud computing". In: *IEEE TCC* 6.3 (2015), pp. 815–828.

[Red+19] Simone Redana et al. "5G PPP Architecture Working Group: View on 5G Architecture". In: *5G PPP White Paper* (2019).

[Ren+19] Jinke Ren et al. "Collaborative cloud and edge computing for latency minimization". In: *IEEE TVT* 68.5 (2019), pp. 5031–5044.

[Rez+18] Alex Reznik et al. "MEC in an enterprise setting: A solution outline". In: *ETSI White Paper* (2018), pp. 1–20.

[RVM16] Bhaskar Prasad Rimal, Dung Pham Van, and Martin Maier. "Mobile-edge computing vs. centralized cloud computing in fiber-wireless access networks". In: *Proc. of IEEE INFOCOM Workshop*. 2016, pp. 991–996.

[Ryu+19] June-Woo Ryu et al. "Multi-Access Edge Computing Empowered Heterogeneous Networks: A Novel Architecture and Potential Works". In: *Symmetry* 11.7 (2019), p. 842.

[Sab+19a] Dario Sabella et al. "Developing software for multi-access edge computing". In: *ETSI White Paper* 20 (2019).

[Sab+19b] Dario Sabella et al. "Edge Computing: from standard to actual infrastructure deployment and software development". In: *ETSI White Paper* (2019).

[Sat+09] Mahadev Satyanarayanan et al. "The case for vm-based cloudlets in mobile computing". In: *IEEE PERVAS COMPUT* 8.4 (2009).

[Sat15] Mahadev Satyanarayanan. "A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets". In: *ACM GetMobile: Mobile Computing and Communications* 18.4 (2015), pp. 19–23.

[Shi+12a] Cong Shi et al. “Computing in cirrus clouds: the challenge of intermittent connectivity”. In: *Proc. of ACM, MCC Workshop*. 2012, pp. 23–28.

[Shi+12b] Cong Shi et al. “Serendipity: enabling remote computing among intermittently connected mobile devices”. In: *Proc. of ACM MobiHoc*. 2012, pp. 145–154.

[Sid+15] Uzma Siddique et al. “Wireless backhauling of 5G small cells: Challenges and solution approaches”. In: *IEEE Wirel. Commun.* 22.5 (2015), pp. 22–31.

[Sio+18] Dimitris Siomos et al. “5G Wireless Backhaul/X-Haul”. In: *ETSI White Paper* v. 1.1.1 (2018).

[SL18] Sowndarya Sundar and Ben Liang. “Offloading dependent tasks with communication delay and deadline constraint”. In: *Proc. of IEEE INFOCOM*. 2018, pp. 37–45.

[SLS03] Keng Siau, Ee-Peng Lim, and Zixing Shen. “Mobile commerce: Current states and future trends”. In: *Advances in mobile commerce technologies*. IGI Global, 2003, pp. 1–17.

[SMG02] Cem U Saraydar, Narayan B Mandayam, and David J Goodman. “Efficient power control via pricing in wireless data networks”. In: *IEEE T COMMUN* 50.2 (2002), pp. 291–303.

[SSB15] Stefania Sardellitti, Gesualdo Scutari, and Sergio Barbarossa. “Joint optimization of radio and computational resources for multicell mobile-edge computing”. In: *IEEE T SIGNAL INF PR* 1.2 (2015), pp. 89–103.

[SW18] Hamed Shah-Mansouri and Vincent WS Wong. “Hierarchical fog-cloud computing for IoT systems: A computation offloading game”. In: *IEEE IoT* 5.4 (2018), pp. 3246–3257.

[Tär+17] William Tärneberg et al. “Dynamic application placement in the mobile cloud network”. In: *Elsevier FUTURE GENER COMP SY* 70 (2017), pp. 163–177.

[TL17] Nguyen Ti Ti and Long Bao Le. “Computation offloading leveraging computing resources from edge cloud and mobile peers”. In: *Proc. of IEEE ICC*. 2017, pp. 1–6.

[TP18] Tuyen X Tran and Dario Pompili. “Joint task offloading and resource allocation for multi-server mobile-edge computing networks”. In: *IEEE TVT* 68.1 (2018), pp. 856–868.

[VR14] Luis M Vaquero and Luis Rodero-Merino. “Finding your way in the fog: Towards a comprehensive definition of fog computing”. In: *ACM SIGCOMM COMP COM* 44.5 (2014), pp. 27–32.

[Wan+17] Shuo Wang et al. “A survey on mobile edge networks: Convergence of computing, caching and communications”. In: *IEEE Access* 5 (2017), pp. 6757–6779.

[WLP13] Yanzhi Wang, Xue Lin, and Massoud Pedram. “A nested two stage game-based optimization framework in mobile cloud computing system”. In: *Proc. of IEEE SOSE*. 2013, pp. 494–502.

[WZL12] Y. Wen, W. Zhang, and H. Luo. “Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones”. In: *Proc. of IEEE INFOCOM*. 2012, pp. 2716–2720.

[XCZ18] Jie Xu, Lixing Chen, and Pan Zhou. “Joint service caching and task offloading for mobile edge computing in dense networks”. In: *Proc. of IEEE INFOCOM*. 2018, pp. 207–215.

[XHS19] Jiuyun Xu, Zhuangyuan Hao, and Xiaoting Sun. “Optimal Offloading Decision Strategies and Their Influence Analysis of Mobile Edge Computing”. In: *Sensors* 19.14 (2019), p. 3231.

[XLL15] Liyao Xiang, Baochun Li, and Bo Li. “Coalition Formation Towards Energy-Efficient Collaborative Mobile Computing”. In: *Proc. of IEEE ICCCN*. 2015, pp. 1–8.

[XLX13] Qiufen Xia, Weifa Liang, and Wenzheng Xu. “Throughput maximization for online request admissions in mobile cloudlets”. In: *Proc. of IEEE LCN*. 2013, pp. 589–596.

[XSC03] Mingbo Xiao, Ness B Shroff, and Edwin KP Chong. “A utility-based power-control scheme in wireless cellular systems”. In: *IEEE/ACM ToN* 11.2 (2003), pp. 210–221.

[XSM18] Jianbin Xue, Hua Shao, and Qing Ma. “Resource Allocation for System Throughput Maximization Based on Mobile Edge Computing”. In: *Proc. of ACM EEET*. 2018, pp. 177–181.

[Yan+13] Lei Yang et al. “A framework for partitioning and execution of data stream applications in mobile cloud computing”. In: *ACM SIGMETRICS PER* 40.4 (2013), pp. 23–32.

[Yan+15] Lei Yang et al. “Multi-user computation partitioning for latency sensitive mobile cloud applications”. In: *IEEE TC* 64.8 (2015), pp. 2253–2266.

[Yan+17] Lei Yang et al. “Joint Computation Partitioning and Resource Allocation for Latency Sensitive Applications in Mobile Edge Clouds”. In: *Proc. of IEEE Cloud Computing*. 2017, pp. 246–253.

[Yan+18] Lei Yang et al. “Network aware mobile edge computation partitioning in multi-user environments”. In: *IEEE T SERV COMPUT* (2018).

[Yan+19] Zhaohui Yang et al. “Efficient Resource Allocation for Mobile-Edge Computing Networks with NOMA: Completion Time and Energy Minimization”. In: *IEEE TCOM* 67.11 (2019), pp. 7771–7784.

[YN06] Wanghong Yuan and Klara Nahrstedt. “Energy-efficient CPU scheduling for multimedia applications”. In: *Proc. of ACM TOCS* 24.3 (2006), pp. 292–331.

[You+16] Changsheng You et al. “Energy-efficient resource allocation for mobile-edge computation offloading”. In: *IEEE TWC* 16.3 (2016), pp. 1397–1411.

[Zen+16] Deze Zeng et al. “Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system”. In: *IEEE TC* 65.12 (2016), pp. 3702–3712.

[ZF19] Ming Zeng and Viktoria Fodor. “Dynamic Spectrum Sharing for Load Balancing in Multi-Cell Mobile Edge Computing”. In: *IEEE WCL* (2019).

[Zha+16] Ke Zhang et al. “Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks”. In: *IEEE Access* 4 (2016), pp. 5896–5907.

[Zha+17] Jing Zhang et al. “Joint offloading and resource allocation optimization for mobile edge computing”. In: *Proc. of IEEE GLOBECOM*. 2017, pp. 1–6.

[Zha+18] Jiao Zhang et al. “Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching”. In: *IEEE IoT* 6.3 (2018), pp. 4283–4294.

[Zhe+18] Jianchao Zheng et al. “Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach”. In: *IEEE TMC* 18.4 (2018), pp. 771–786.

[Zho+18] Fuhui Zhou et al. “Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems”. In: *IEEE JSAC* 36.9 (2018), pp. 1927–1941.

[Zhu+18] Zhengfa Zhu et al. “Fair resource allocation for system throughput maximization in mobile edge computing”. In: *IEEE Access* 6 (2018), pp. 5332–5340.

Paper **A**

Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems

Slađana Jošilo and György Dán

IEEE/ACM Transactions on Networking (ToN), vol. 27, no. 1, pp. 85-97, 2019.

Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems

Slađana Jošilo and György Dán
School of Electrical Engineering and Computer Science
KTH, Royal Institute of Technology, Stockholm, Sweden
E-mail: {josilo, gyuri}@kth.se *

Abstract

Fog computing is identified as a key enabler for using various emerging applications by battery powered and computationally constrained devices. In this paper, we consider devices that aim at improving their performance by choosing to offload their computational tasks to nearby devices or to an edge cloud. We develop a game theoretical model of the problem, and we use variational inequality theory to compute an equilibrium task allocation in static mixed strategies. Based on the computed equilibrium strategy, we develop a decentralized algorithm for allocating the computational tasks among nearby devices and the edge cloud. We use extensive simulations to provide insight into the performance of the proposed algorithm, and we compare its performance with the performance of a myopic best response algorithm that requires global knowledge of the system state. Despite the fact that the proposed algorithm relies on average system parameters only, our results show that it provides good system performance close to that of the myopic best response algorithm.

Index terms— computation offloading, fog computing, game theory, task placement, decentralized resource management

1 Introduction

Fog computing is widely recognized as a key component of 5G networks and an enabler of the Internet of Things (IoT) [1,2]. The concept of fog computing extends the traditional centralized cloud computing architecture by allowing devices not only to use computing and storage resources of centralized clouds, but also resources distributed across the network including the resources of each other and resources located at the network edge [3].

*The work was partly funded by the Swedish Research Council through project 621-2014-6.

Traditional centralized cloud computing allows devices to offload the computation to a cloud infrastructure with significant computational power [4], [5,6]. Cloud offloading may indeed accelerate the execution of applications, but it may suffer from high communication delays, on the one hand due to the contention of devices for radio spectrum, on the other hand due to the remoteness of the cloud infrastructure. Thus, traditional centralized cloud computing may not be able to meet the delay requirements of emerging IoT applications [7, 8], [9, 10].

Fog computing addresses this problem by allowing collaborative computation offloading among nearby devices and distributed cloud resources close to the network edge [11]. The benefits of collaborative computation offloading are twofold. First, collaboration among devices can make use of device-to-device (D2D) communication, and thereby it can improve spectral efficiency and free up radio resources for other purposes [12–14]. Second, the proximity of devices to each other can enable low communication delays. Thus, fog computing allows to explore the tradeoff between traditional centralized cloud offloading, which ensures low computing time, but may suffer from high communication delay, and collaborative computation offloading, which ensures low communication delay, but may involve higher computing times.

One of the main challenges facing the design of fog computing systems is how to manage fog resources efficiently. Compared to traditional centralized cloud computing, where a device only needs to decide whether to offload the computation of a task, in the case of fog computing the number of offloading choices increases with the number of devices. Furthermore, today's devices are heterogeneous in terms of computational capabilities, in terms of what tasks they have to execute and how often. At the same time, some devices may be autonomous, and hence they would be interested in minimizing their own perceived completion times. Therefore, developing low complexity algorithms for efficient task allocation among nearby devices is an inherently challenging problem.

In this paper we address this problem by considering a fog computing system, where devices can choose either to perform their computation locally, to offload the computation to a nearby device, or to offload the computation to an edge cloud. We provide a game theoretical model of the completion time minimization problem. We show that an equilibrium task allocation in static mixed strategies always exists, i.e., if devices can choose at random whether to offload, and where to offload. Based on the game theoretical model we propose a decentralized algorithm that relies on average system parameters, and allocates the tasks according to a Nash equilibrium in static mixed strategies. We use the algorithm to address the important question whether efficient task allocation is feasible using an algorithm that requires low signaling overhead, and we compare the performance achieved by the proposed algorithm with the performance of a myopic best response algorithm that requires global knowledge of the system state. Our results show that the proposed decentralized algorithm, despite significantly lower signaling overhead, provides good system performance close to that of the myopic best response algorithm.

The rest of the paper is organized as follows. We present the system model in Section 2. We present two algorithms in Sections 3 and 4. In Section 5 we present numerical results and in Section 6 we review related work. Section 7 concludes the paper.

2 System Model and Problem Formulation

We consider a fog computing system that consists of a set $\mathcal{N}=\{1,2,...,N\}$ of devices, and an edge cloud. Device $i \in \mathcal{N}$ generates a sequence $(t_{i,1}, t_{i,2}, \ldots)$ of computational tasks. We consider that the size $D_{i,k}$ (e.g., in bytes) of task $t_{i,k}$ of device i can be modeled by a random variable D_i, and the number of CPU cycles $L_{i,k}$ required to perform the task by a random variable L_i. According to results reported in [15–17] the number X_i of CPU cycles per data bit can be approximated by a Gamma distribution, and thus we can model the relation between L_i and D_i as $L_i = D_i X_i$. Furthermore, assuming that the first moment $\overline{X}_i$ and the second moment ${}^2\overline{X}_i$ of X_i can be estimated based on the past, the statistics of the number of CPU cycles required to perform the task of device i can be easily obtained. Similar to other works [18–20], we assume that the task arrival process of device i can be modeled by a Poisson process with arrival intensity λ_i.

For each task $t_{i,k}$ device i can decide whether to perform the task locally, to offload it to a device $j \in \mathcal{N} \setminus \{i\}$ or to an edge cloud. Thus, device i chooses a member of the set $\mathcal{N} \cup \{0\}$, where 0 corresponds to the edge cloud. We allow for randomized policies, and we denote by $p_{i,j}(k)$ the probability that device i assigns its task $t_{i,k}$ to $j \in \mathcal{N} \cup \{0\}$, and we define the probability vector $p_i(k) = \{p_{i,0}(k), p_{i,1}(k), ..., p_{i,N}(k)\}$, where $\sum_{j \in \mathcal{N} \cup \{0\}} p_{i,j}(k) = 1$. Finally, we denote by $\mathcal{P}$ the set of probability distributions over $\mathcal{N} \cup \{0\}$, i.e., $p_i(k) \in \mathcal{P}$.

The above fog computing system relies on the assumption that all devices faithfully execute the tasks offloaded to them. To ensure this, the devices need to be incentivized to collaborate in executing each others' computational tasks, as discussed in [21]. The collaboration among devices in fog computing systems can be ensured with an adequate incentive scheme similar to those used in peer-to-peer systems [22–24]. These schemes ensure the collaboration among the peers through the reputation-based trust supporting mechanism. In the context of fog computing systems, the mechanism would result in an incentive scheme in which only devices that process offloaded tasks themselves are entitled to offload the tasks.

2.1 Communication model

We consider that the devices communicate using an orthogonal frequency division multiple access (OFDMA) framework in which there is an assignment of subcarriers to pairs of communicating nodes [25,26]. Furthermore, we consider that devices use dedicated bandwidth resources, i.e. node-to-node pairs do not share the bandwidth

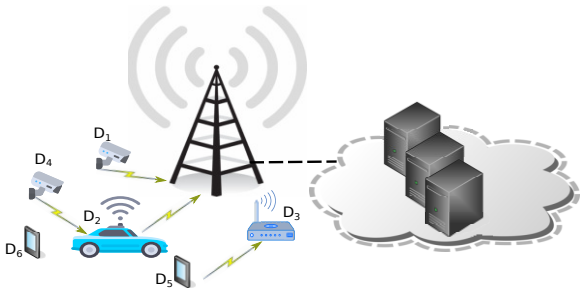

Figure 1: Fog computing system that consists of 6 devices and an edge cloud.

with each other and with the other cellular users [25]. This can be implemented by assigning an orthogonal subcarrier per transmission direction for each pair of communicating nodes, resulting in $N \times N$ subcarriers in total. We denote the transmission rate from device i to device j by $R_{i,j}$, and the transmission rate from device i to the edge cloud through a base station by $R_{i,0}$. Each device maintains N transmission queues, i.e., $N-1$ queues for transmitting to devices $j \in \mathcal{N} \setminus \{i\}$ and one for transmitting to the edge cloud, and the tasks are transmitted in FIFO order.

We consider that the time $T^t_{i,j}(k)$ needed to transmit a task $t_{i,k}$ from device i to $j \in \mathcal{N} \cup \{0\}$ is proportional to its size $D_{i,k}$, and is given by

$$T^t_{i,j}(k) = D_{i,k}/R_{i,j}.$$

Furthermore, the time $T^d_{i,j}(k)$ needed to deliver the input data $D_{i,k}$ from device i to $j \in \mathcal{N} \cup \{0\}$ is the sum of the transmission time $T^t_{i,j}(k)$ and of the waiting time (if any).

Similar to other works [27, 28], [29, 30], we consider that the time needed to transmit the results of the computation back to the device is negligible. This assumption is justified for many applications including face and object recognition, and anomaly detection, where the size of the result of the computation is much smaller than the size of the input data.

Observe that our system model can accommodate systems in which certain devices $i \in \mathcal{N}$ only serve for performing the computational tasks of others, by setting the arrival intensity $\lambda_i = 0$. These devices can be considered as micro-data centers located at the network edge, whose function in fog computing systems is to perform the computational tasks of the other devices [31,32]. Furthermore, our system model can accommodate systems in which certain devices $j \in \mathcal{N}$ are not supposed to perform the computational tasks of others, by setting the transmission rates $R_{i,j}$ from the other devices $i \in \mathcal{N} \setminus \{j\}$ to device j to low enough values.

Figure 1 illustrates a fog computing system that consists of six devices and one edge cloud; device 1 and device 2 offload their tasks through a base station to the cloud server, device 4 offloads its tasks to device 2, device 5 offloads its task to

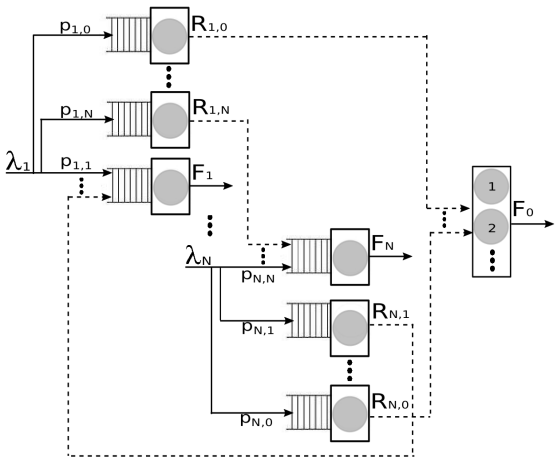

Figure 2: Fog computing system modeled as a queuing network.

device 3 that serves as a micro-data center, and device 6 performs computation locally.

2.2 Computation model

To model the time that is needed to compute a task in a device i, we consider that each device i maintains one execution queue with tasks served in FIFO order. We denote by F_i the computational capability of device i. Unlike devices, the cloud server has a large number of processors with computational capability F_0 each, and we assume that computing in the edge cloud begins immediately upon arrival of a task.

Similar to common practice [21, 27] we consider that the time $T^c_{i,j}(k)$ needed to compute a task $t_{i,k}$, on $j \in \mathcal{N} \cup \{0\}$ is proportional to its complexity $L_{i,k}$, and is given by

$$T^c_{i,j}(k) = L_{i,k}/F_j.$$

Furthermore, the execution time $T^e_{i,j}(k)$ of a task $t_{i,k}$ on device j is the sum of the computation time $T^c_{i,j}(k)$ and of the waiting time (if any). Figure 2 illustrates the queuing model of a computation offloading system.

2.3 Problem formulation

We define the cost C_i of device i as the mean completion time of its tasks. Given a sequence $(t_{i,1}, t_{i,2}, \ldots)$ of computational tasks, we can thus express the cost C_i as

$$C_i = \lim_{K\to\infty} \frac{1}{K}\Bigg[\sum_{k=1}^{K}\Bigg(p_{i,i}(k)T^e_{i,i}(k) + \sum\nolimits_{j\in\mathcal{N}\setminus\{i\}\cup\{0\}} p_{i,j}(k)\big(T^d_{i,j}(k)+T^e_{i,j}(k)\big)\Bigg)\Bigg]. \tag{1}$$

Since the devices are autonomous, we consider that each device aims at minimizing its cost by solving

$$\min C_i \quad \text{s.t.} \tag{2}$$

$$p_i(k) \in \mathcal{P}. \tag{3}$$

Since devices' decisions affect each other, the devices play a dynamic non-cooperative game, and we refer to the game as the *multi user computation offloading game* (MCOG). The game is closest to an undiscounted stochastic game with countably infinite state space, but the system state evolves according to a semi-Markov chain (instead of a Markov chain, depending on the distribution of D_i and L_i) and payoffs (the completion times) are unbounded. We are not aware of existence results for Markov equilibria for this class of problem, and even for the case when the state evolves according to a Markov chain with countable state space and unbounded payoffs, there are only a few results on the existence of equilibria in Markov strategies [33–35].

2.4 Decentralized solution supported by a centralized entity

Since fog computing architecture is decentralized in nature, and devices in fog computing systems are expected to be autonomous [11, 36] we are interested in developing decentralized algorithms that will allow devices to make their offloading decisions locally. Motivated by widely considered hierarchical fog computing architectures [37,38], we consider that there is a single central entity with a high level of hierarchy that collects and stores the information about the fog computing system. The entity need not be a single physical entity, but a single logically centralized entity that can handle high loads and can be resilient to failure.

Furthermore, we consider that the entity periodically sends the needed information to the devices and thus supports them in making their offloading decisions. Intuitively, more information about the system state will allow devices to make better offloading decisions, but at the cost of increased signaling overhead. Therefore, one important objective when developing decentralized algorithms for allocating the computational tasks is to achieve good system performance at the cost of an

$p_i(k) = MyopicBestResponse(t_{i,k})$

1: $p_{i,j}(k) = 0, \quad \forall j \in \mathcal{N} \cup \{0\}$
2: /* Estimate completion time of $t_{i,k}$ in $\forall j \in \mathcal{N} \cup \{0\}$ */
3: **for** $j = 0, \ldots, N$ **do**
4: **if** $j = i$ **then**
5: $ECompleteT(j) = T^e_{i,j}(k)$
6: **else**
7: $ECompleteT(j) = T^d_{i,j}(k) + T^e_{i,j}(k)$
8: **end if**
9: **end for**
10: /* Make a greedy decision */
11: $i' \leftarrow \arg\min_{\{j \in \mathcal{N} \cup \{0\}} ECompleteT(j)$
12: $p_{i,i'}(k) = 1$
13: **return** $p_i(k)$

Figure 3: Pseudo code of myopic best response.

acceptable signaling overhead. With this in mind, in what follows we propose and discuss two decentralized solutions for the MCOG problem in the form of a Markov strategy and in static mixed strategies, respectively.

3 Myopic best response

The first algorithm we consider, called *Myopic Best Response* (MBR), requires global knowledge of the system state, but decisions are made locally at the devices. Similar to the *WaterFilling* algorithm proposed in [39], in the MBR algorithm every device i makes a decision based on a myopic best response strategy, i.e., every device i chooses a node $j \in \mathcal{N} \cup \{0\}$ that minimizes the completion time of its task $t_{i,k}$, given the instantaneous state of the queuing network. The pseudo-code for computing the myopic best response strategy is shown in Figure 3. Note that since the devices make their decisions based on the instantaneous states of the queues, they do not take into account the tasks that may arrive to the other devices' execution queues while transmitting a task. Futhermore, if the devices' execution queues were stable if all devices perform all tasks locally, then under the MBR algorithm the queue lengths do not grow unbounded since each device chooses the destination node based on the instantaneous state of the queues.

Note that if we define the system state upon the arrival of task $t_{i,k}$ as the number of jobs in the transmission and execution queues, then the devices' decisions depend on the instantaneous system state only, and hence the myopic best response is a Markov strategy for the *MCOG*. Nonetheless, it is not necessarily a Markov perfect

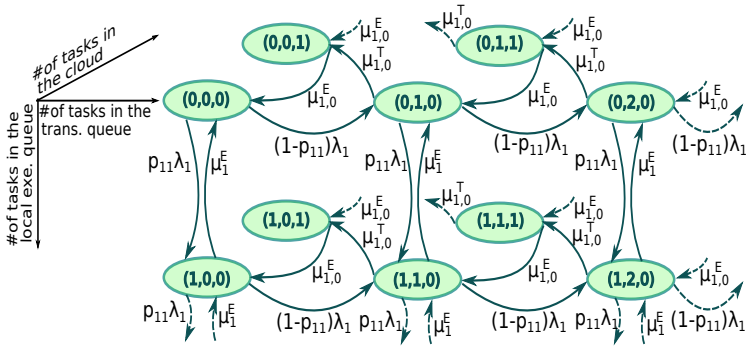

Figure 4: State transition diagram of the semi-Markov process induced by the offloading decisions for the single device case ($N = 1$).

equilibrium.

In a system with N devices we have $N \times N$ transmission queues and $N+1$ execution queues, and we can thus model the system as an $N \times (N+1)+1$ dimensional semi-Markov process.

Example 1. *Figure 4 shows the state transition diagram for a single device, i.e., $N = 1$, which is three dimensional. We use the triplet (n_l, n_t, n_0) to denote the system state, where n_l, n_t and n_0 stand for the number of tasks in the local execution queue, number of tasks in the transmission queue and the number of tasks in the cloud server, respectively. Since $N = 1$, a device only needs to decide whether to offload the computation to the edge cloud or to perform the computation locally and hence the transition intensities from state (n_l, n_t, n_0) to state $(n_l, n_t + 1, n_0)$ and from state (n_l, n_t, n_0) to state $(n_l + 1, n_t, n_0)$ are $(1 - p_{1,1})\lambda_1$ and $p_{1,1}\lambda_1$, respectively. In the case of computation offloading, the task with size D_1 and complexity L_1 needs to be transmitted to the edge cloud at rate $R_{1,0}$ and executed with computational capability F_0 and thus the transition intensities from state (n_l, n_t, n_0) to state $(n_l, n_t - 1, n_0 + 1)$ and from state (n_l, n_t, n_0) to state $(n_l, n_t, n_0 - 1)$ are $\mu^T_{1,0} = D_1/R_{1,0}$ and $\mu^E_{1,0} = n_0 L_1/F_0$, respectively. Finally, in the case of local execution the task with complexity L_1 needs to be executed locally with local computational capability F_1 and hence the transition intensity from state (n_l, n_t, n_0) to state $(n_l - 1, n_t, n_0)$ is $\mu^E_1 = L_1/F_1$.*

A significant detriment of the MBR algorithm is its signaling overhead, as it requires global information about the system state upon the arrival of each task. To reduce the signaling requirements, in what follows we propose an algorithm that is based on a strategy that relies on average system parameters only.

4 Equilibrium in Static Mixed Strategies

As a practical alternative to the MBR algorithm, in this section we propose a decentralized algorithm, which we refer to as the *Static Mixed Nash Equilibrium* (SM-NE) algorithm. The algorithm is based on an equilibrium $(p_i)_{i\in\mathcal{N}}$ in static mixed strategies, that is, device i chooses the node where to execute an arriving task at random according to the probability vector p_i, which is the same for all tasks. For computing a static mixed strategy, it is enough for a device to know the average task arrival intensities, transmission rates, and the first and second moments of the task size and the task complexity distribution. Therefore, the SM-NE algorithm requires significantly less signaling than the MBR algorithm.

In order to compute an equilibrium strategy, we start with expressing the (approximate) equilibrium cost of device i as a function of strategy profile $(p_i)_{i\in\mathcal{N}}$, i.e., the mean completion time of its tasks in steady state. Throughout the section we denote by $\overline{D}_i$ and ${}^2\overline{D}_i$ the first and the second moment of D_i, respectively, and by $\overline{L}_i$ and ${}^2\overline{L}_i$ the first and the second moment of L_i, respectively.

4.1 Transmission time in steady state

Since tasks arrive to each device as a Poisson process and we aim for a constant probability vector p_i as a solution, the arrival processes to the transmission queues are Poisson processes. If the transmission queues are sufficiently large, we can approximate them as infinite, similar to [20], and thus we can model each transmission queue as an $M/G/1$ system. Let us denote by $\overline{T^t}_{i,j}$ and ${}^2\overline{T^t}_{i,j}$ the mean and the second moment of the time needed to transmit a task from device i to $j\in\mathcal{N}\setminus\{i\}\cup\{0\}$, respectively. Then the mean time $\overline{T^d}_{i,j}$ needed to deliver the input data from device i to $j\in\mathcal{N}\setminus\{i\}\cup\{0\}$ is the sum of the mean waiting time in the transmission queue and the mean transmission time $\overline{T^t}_{i,j}$, and can be expressed as

$$\overline{T^d}_{i,j} = \frac{p_{i,j}\lambda_i {}^2\overline{T^t}_{i,j}}{2(1-p_{i,j}\lambda_i\overline{T^t}_{i,j})} + \overline{T^t}_{i,j}, \tag{4}$$

and the queue is stable as long as the offered load $\rho^t_{i,j} = p_{i,j}\lambda_i\overline{T^t}_{i,j} < 1$.

4.2 Computation time in steady state

Observe that if the input data size D_i follows an exponential distribution, then departures from the transmission queues can be modeled by a Poisson process, and thus tasks arrive to the devices' execution queues according to a Poisson process. In what follows we use the approximation that the tasks arrive according to a Poisson process even if D_i is not exponentially distributed. Furthermore, following common practice [19, 40], for analytical tractability we approximate the execution queues as being infinite. This approximation is reasonable if the queues are sufficiently large.

These two approximations allow us to model the execution queue of each device as an $M/G/1$ system, and the edge cloud as an $M/G/\infty$ system.

Let us denote by $\overline{T^c}_{i,j}$ and ${}^2\overline{T^c}_{i,j}$ the mean and the second moment of the time needed to compute device i's task on $j \in \mathcal{N} \cup \{0\}$, respectively. Then the mean time $\overline{T^e}_{i,j}$ that device $j \in \mathcal{N}$ needs to complete the execution of device i's task is the sum of the mean waiting time in the execution queue and the mean computation time $\overline{T^c}_{i,j}$, and can be expressed as

$$\overline{T^e}_{i,j} = \frac{\sum_{i' \in \mathcal{N}} p_{i',j} \lambda_{i'}^2 \overline{T^c}_{i',j}}{2(1 - \sum_{i' \in \mathcal{N}} p_{i',j} \lambda_{i'} \overline{T^c}_{i',j})} + \overline{T^c}_{i,j}, \tag{5}$$

and the queue is stable as long the offered load $\rho_j^e = \sum_{i' \in \mathcal{N}} p_{i',j} \lambda_{i'} \overline{T^c}_{i',j} < 1$.

Since computing in the edge cloud begins immediately upon arrival of a task, the mean time $\overline{T^e}_{i,0}$ that the cloud needs to complete the execution of device i's task is equal to the mean computation time $\overline{T^c}_{i,0}$, i.e.,

$$\overline{T^e}_{i,0} = \overline{L}_i / F_0. \tag{6}$$

4.3 Existence of Static Mixed Strategy Equilibrium

We can rewrite (1) to express the cost C_i of device i in steady state as a function of $(p_i)_{i \in \mathcal{N}}$,

$$C_i(p_i, p_{-i}) = p_{i,i} \overline{T^e}_{i,i} + \sum\nolimits_{j \in \mathcal{N} \setminus \{i\} \cup \{0\}} p_{i,j} \left(\overline{T^d}_{i,j} + \overline{T^e}_{i,j} \right),$$

where we use p_{-i} to denote the strategies of all devices except device i.

Observe that static mixed strategy profile $(p_i)_{i \in \mathcal{N}}$ of the devices has to ensure that the entire system is stable in steady state, and we assume that the load is such that there is at least one strategy profile that satisfies the stability condition of the entire system. Now, we can define the set of feasible strategies of device i as the set of probability vectors that ensure stability of the transmission and the execution queues

$$\mathcal{K}_i(p_{-i}) = \{ p_i \in \mathcal{P} | \rho_{i,j}^t \leq S_t, \rho_{i'}^e \leq S_t, \forall j \in \mathcal{N} \setminus \{i\} \cup \{0\}, \forall i' \},$$

where $0 < S_t < 1$ is the stability threshold associated with the transmission and the execution queues.

Note that due to the stability constraints the set of feasible strategies $\mathcal{K}_i(p_{-i})$ of device i depends on the other devices' strategies, and we are interested in whether there is a strategy profile $(p_i^*)_{i \in \mathcal{N}}$, such that

$$C_i(p_i^*, p_{-i}^*) \leq C_i(p_i, p_{-i}^*), \quad \forall p_i \in \mathcal{K}_i(p_{-i}^*).$$

We are now ready to formulate the first main result of the section.

Theorem 1. *The MCOG has at least one equilibrium in static mixed strategies.*

In the rest of this subsection we use *variational inequality* (VI) theory to prove the theorem and for computing an equilibrium. For a given set $\mathcal{K} \subseteq \mathbb{R}^n$ and a function $F : \mathcal{K} \to \mathbb{R}^n$, the $VI(\mathcal{K}, F)$ problem is the problem of finding a point $x^* \in \mathcal{K}$ such that $F(x^*)^T(x - x^*) \geq 0$, for $\forall x \in \mathcal{K}$. We define the set $\mathcal{K}$ as

$$\mathcal{K} = \{(p_i)_{i \in N} | p_i \in \mathcal{P}, \rho^t_{i,j} \leq S_t, \rho^e_i \leq S_t, j \in \mathcal{N} \backslash \{i\} \cup \{0\}, \forall i\}.$$

Before we prove the theorem, in the following we formulate an important result concerning the cost function $C_i(p_i, p_{-i})$.

Lemma 1. *$C_i(p_i, p_{-i})$ is a convex function of p_i for any fixed p_{-i} and $(p_i, p_{-i}) \in \mathcal{K}$.*

Proof. For notational convenience let us start the proof with introducing a few shorthand notations,

$$\gamma_{i,j} = p_{i,j}\lambda_i^2 \overline{T^t}_{i,j}, \;\; \delta_i = \sum_{j \in \mathcal{N}} p_{j,i}\lambda_j^2 \overline{T^c}_{j,i},$$
$$\epsilon_{i,j} = 1 - \rho^t_{i,j}, \;\; \zeta_i = 1 - \rho^e_i.$$

Using this notation we expand the cost $C_i(p_i, p_{-i})$ as

$$C_i(p_i, p_{-i}) = p_{i,i}\big(\frac{\delta_i}{2\zeta_i} + \overline{T^c}_{i,i}\big) + p_{i,0}\big(\frac{\gamma_{i,0}}{2\epsilon_{i,0}} + \overline{T^t}_{i,0} + \overline{T^c}_{i,0}\big) + \sum_{j \in \mathcal{N} \backslash \{i\}} p_{i,j}\big(\frac{\gamma_{i,j}}{2\epsilon_{i,j}} + \overline{T^t}_{i,j} + \frac{\delta_j}{2\zeta_j} + \overline{T^c}_{i,j}\big).$$

To prove convexity we proceed with expressing the first order derivatives $h_{i,j} = \frac{\partial C_i(p_i, p_{-i})}{\partial p_{i,j}}$,

$$h_{i,0} = \overline{T^t}_{i,0} + \overline{T^c}_{i,0} + \frac{\gamma_{i,0}}{2\epsilon_{i,0}} + p_{i,0}\lambda_i\big(\frac{^2\overline{T^t}_{i,0}}{2\epsilon_{i,0}} + \frac{\overline{T^t}_{i,0}\gamma_{i,0}}{2\epsilon^2_{i,0}}\big),$$

$$h_{i,i} = \overline{T^c}_{i,i} + \frac{\delta_i}{2\zeta_i} + p_{i,i}\lambda_i\big(\frac{^2\overline{T^c}_{i,i}}{2\zeta_i} + \frac{\overline{T^c}_{i,i}\delta_i}{2\zeta_i^2}\big),$$

$$h_{i,j}\big|_{j \neq i} = \overline{T^t}_{i,j} + \overline{T^c}_{i,j} + \frac{\gamma_{i,j}}{2\epsilon_{i,j}} + \frac{\delta_j}{2\zeta_j} + p_{i,j}\lambda_i\big(\frac{^2\overline{T^t}_{i,j}}{2\epsilon_{i,j}} + \frac{^2\overline{T^c}_{i,j}}{2\zeta_j} + \frac{\overline{T^t}_{i,j}\gamma_{i,j}}{2\epsilon^2_{i,j}} + \frac{\overline{T^c}_{i,j}\delta_j}{2\zeta_j^2}\big).$$

We can now express the Hessian matrix

$$H_i(p_i,p_{-i}) = \begin{pmatrix} h^i_{i,0} & 0 & \dots & 0 \\ 0 & h^i_{i,1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h^i_{i,N} \end{pmatrix},$$

where $h^i_{i,j} = \frac{\partial^2 C_i(p_i,p_{-i})}{\partial p^2_{i,j}}$, and

$$h^i_{i,0} = \frac{\lambda_i}{\epsilon_{i,0}}\big(^2\overline{T^t}_{i,0} + \frac{\gamma_{i,0}T^t_{i,0}}{\epsilon_{i,0}}\big)\big(1 + p_{i,0}\frac{\lambda_i T^t_{i,0}}{\epsilon_{i,0}}\big),$$

$$h^i_{i,i} = \frac{\lambda_i}{\zeta_i}\big(^2\overline{T^c}_{i,i} + \frac{\delta_i T^c_{i,i}}{\zeta_i}\big)\big(1 + p_{i,i}\frac{\lambda_i T^c_{i,i}}{\zeta_i}\big),$$

$$\begin{aligned} h^i_{i,j}\big|_{j\neq i} = &\frac{\lambda_i}{\epsilon_{i,j}}\big(^2\overline{T^t}_{i,j} + \frac{\gamma_{i,j}T^t_{i,j}}{\epsilon_{i,j}}\big)\big(1 + p_{i,j}\frac{\lambda_i T^t_{i,j}}{\epsilon_{i,j}}\big) + \\ &\frac{\lambda_i}{\zeta_j}\big(^2\overline{T^c}_{i,j} + \frac{\delta_j T^c_{i,j}}{\zeta_j}\big)\big(1 + p_{i,j}\frac{\lambda_i T^c_{i,j}}{\zeta_j}\big). \end{aligned}$$

Observe that all diagonal elements of $H_i(p_i,p_{-i})$ are nonnegative, and thus the Hessian matrix $H_i(p_i,p_{-i})$ is positive semidefinite on $\mathcal{K}$, which implies convexity. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. Let us define the generalized Nash equilibrium problem $\Gamma^s =< \mathcal{N}, (\mathcal{P})_{i\in\mathcal{N}}, (C_i)_{i\in\mathcal{N}} >$, subject to $(p_i)_{i\in\mathcal{N}} \in \mathcal{K}$. Γ^s is a strategic game, in which each device $i \in \mathcal{N}$ plays a *mixed strategy* $p_i \in \mathcal{K}_i(p_{-i})$, and aims at minimizing its cost C_i by solving

$$\min_{p_i} C_i(p_i,p_{-i}) \quad \text{s.t.} \tag{7}$$

$$p_i \in \mathcal{K}_i(p_{-i}). \tag{8}$$

Clearly, a pure strategy Nash equilibrium $(p^*_i)_{i\in\mathcal{N}}$ of Γ^s is an equilibrium of the *MCOG* in static mixed strategies, as

$$C_i(p^*_i,p^*_{-i}) \leq C_i(p_i,p^*_{-i}), \quad \forall p_i \in \mathcal{K}_i(p^*_{-i}).$$

We thus have to prove that Γ^s has a pure strategy Nash equilibrium.

To do so, let us first define the function

$$F = \begin{pmatrix} \nabla_{p_1} C_1(p_1, p_{-1}) \\ \vdots \\ \nabla_{p_N} C_N(p_N, p_{-N}) \end{pmatrix},$$

where $\nabla_{p_i} C_i(p_i, p_{-i})$ is the gradient vector given by

$$\nabla_{p_i} C_i(p_i, p_{-i}) = \begin{pmatrix} h_{i,0} \\ h_{i,1} \\ \vdots \\ h_{i,N} \end{pmatrix}.$$

We prove the theorem in two steps based on the VI($\mathcal{K}, F$) problem, which corresponds to Γ^s.

First, we prove that the solution set of the VI($\mathcal{K}, F$) problem is nonempty and compact. Since the first order derivatives $h_{i,j}$ are rational functions, the function F is infinitely differentiable at every point in $\mathcal{K}$, and hence it is continuous on $\mathcal{K}$. Furthermore, the set $\mathcal{K}$ is compact and convex. Hence, the solution set of the VI($\mathcal{K}, F$) problem is nonempty and compact (Corollary 2.2.5 in [41]).

Second, we prove that any solution of the VI($\mathcal{K}, F$) problem is an equilibrium of the MCOG. Since the function F is continuous on $\mathcal{K}$, it follows that $C_i(p_i, p_{-i})$ is continuously differentiable on $\mathcal{K}$. Furthermore, by Lemma 1 we know that $C_i(p_i, p_{-i})$ is a convex function. Therefore, any solution of the VI($\mathcal{K}, F$) problem is a pure strategy Nash equilibrium of Γ^s [42], and is thus an equilibrium in static mixed strategies of *MCOG*. This proves the theorem. □

Theorem 1 guarantees that the MCOG possesses at least one equilibrium in static mixed strategies, according to which the SM-NE algorithm allocates the tasks among the devices and the edge cloud. The next important question is whether there is an efficient algorithm for solving the VI problem, and hence for computing an equilibrium $(p_i^*)_{i \in \mathcal{N}}$ of the *MCOG* in static mixed strategies.

In what follows we show that an equilibrium can be computed efficiently under certain conditions. To do so, we show that the function F is monotone if the execution queue of each device can be modeled by an $M/M/1$ system and all task arrival intensities are equal. Monotonicity of F is a sufficient condition for various algorithms proposed for solving VIs [43], e.g., for the *Solodov-Tseng Projection-Contraction* (ST-PC) method.

Theorem 2. *If the task sizes and complexities are exponentially distributed, arrival intensities $\lambda_i = \lambda$ and*

$$\lambda \max_{j \in \mathcal{N}} \overline{Tc}_{j,i} \leq \frac{1 - S_t}{N}, \quad \forall i \in \mathcal{N},$$

then the function F is monotone.

The proof is given in Appendix A.1.

Note that the sufficient condition provided by Theorem 2 ensures stability of all execution queues in the worst case scenario, i.e., when $\overline{T^c}_{j,i} = \max_{j \in \mathcal{N}} \overline{T^c}_{j,i}$ for all devices. This condition is, however, not necessary for function F to be monotone in realistic scenarios. In fact, our simulations showed that the ST-PC method converges to an equilibrium for various considered scenarios.

5 Numerical Results

In what follows we show simulation results obtained using an event driven simulator, in which we implemented the MBR and SM-NE algorithms. For the *ST-PC* method we set $p_{i,i} = 1$, $\forall i \in \mathcal{N}$ as starting point, which corresponds to the strategy profile in which each device performs all tasks locally. The *ST-PC* method stops when the norm of the difference of two successive iterations is less than 10^{-4}.

Similar to [44, 45], we placed the devices at random on a *regular grid* with 10^4 points defined over a square area of $1km \times 1km$, and we placed the edge cloud at the center of the grid as in [44]. Unless otherwise noted, we consider that the wired link latency τ_c incurred during communication with the cloud server can be neglected since the cloud is located in close proximity of devices [46]. For simplicity, we consider a static bandwidth assignment for the simulations; we assign a bandwidth of $B_{i,j} = 5$ *MHz* for communication between device i and device j [47, 48], and for the device to cloud bandwidth assignment we consider two scenarios. In the *elastic* scenario the bandwidth $B_{i,0}$ assigned for communication between device i and the edge cloud is independent of the number of devices. In the *fixed* scenario the devices share a fixed amount of bandwidth B_0 when they want to offload a task to the edge cloud, and the bandwidth $B_{i,0}$ scales directly proportional with the number of devices, i.e., $B_{i,0} = \frac{1}{N} B_0$. We consider that the channel gain of device i to a node $j \in \mathcal{N} \setminus \{i\} \cup \{0\}$ is proportional to $d_{i,j}^{-\alpha}$, where $d_{i,j}$ is the distance between device i and node j, and α is the path loss exponent, which we set to 4 according to the path loss model in urban and suburban areas [49]. We set the data transmit power P_i^t of every device i to 0.4 W according to [50] and given the bandwidth $B_{i,j}$ available for the communication between nodes i and j we calculate the noise power P_n as $P_n = B_{i,j} N_0$, where $N_0 = 1.38065 \times 10^{-23} T$ is the spectral density for the termal noise at the temperature $T = 290K$. Finally, we calculate the transmission rate $R_{i,j}$ from device i to node $j \in \mathcal{N} \setminus \{i\} \cup \{0\}$ as $R_{i,j} = B_{i,j} log_2(1 + P_i^t d_{i,j}^{-\alpha} / P_n)$.

The input data size D_i follows a uniform distribution on $[a_i^d, b_i^d]$, where a_i^d and b_i^d are uniformly distributed on $[0.1, 1.4]$ *Mb* and on $[2.2, 3.4]$ *Mb*, respectively. The arrival intensity λ_i of the tasks of device i is uniformly distributed on $[0.01, 0.03]$ *tasks/s*, and the stability threshold is $S_t = 0.6$. Note that for the above set of parameters the maximum arrival intensity does not satisfy the sufficient condition of Theorem 2 already for $N = 20$ devices. Yet, our evaluation shows that the *ST-PC* method converges even for larger instances of the problem.

The computational capability F_i of device i is drawn from a continuous uniform distribution on $[1,4]$ *GHz*, while the computation capability of the edge cloud is $F_0 = 64$ *GHz* [51]. The task complexity L_i follows a uniform distribution on $[a_i^l, b_i^l]$, where a_i^l and b_i^l are uniformly distributed on $[0.2, 0.5]$ *Gcycles* and $[0.7, 1]$ *Gcycles*, respectively.

We use three algorithms as a basis for comparison. The first algorithm computes the socially optimal static mixed strategy profile $(\bar{p}_i)_{i\in\mathcal{N}}$ that minimizes the system cost $C = \frac{1}{N}\sum_{i\in\mathcal{N}} C_i$, i.e., $(\bar{p}_i)_{i\in\mathcal{N}} = \arg\min_{(p_i)_{i\in\mathcal{N}}} C$. We refer to this algorithm as the *Static Mixed Optimal* (SM-OPT) algorithm. The second algorithm considers that the devices are allowed to offload the tasks to the edge cloud only (i.e., $p_{i,i} + p_{i,0} = 1$), and we refer to this algorithm as the *Static Mixed Cloud Nash Equilibrium* (SMC-NE) algorithm. The third algorithm considers that all devices perform local execution (i.e., $p_{i,i} = 1$). Furthermore, we define the performance gain of an algorithm as the ratio between the system cost reached when all devices perform local execution and the system cost reached by the algorithm. For the SM-OPT algorithm the results are shown only up to 30 or 35 devices, because the computation of the socially optimal strategy profile was computationally infeasible for larger problem instances. The results shown in all figures are the averages of 50 simulations, together with 95% confidence intervals.

5.1 Performance gain

We start with evaluating the performance gain as a function of the number of devices. Figure 5 shows the performance gain for the MBR, SM-NE, SM-OPT and SMC-NE algorithms as a function of the number of devices for the two scenarios of device to cloud bandwidth assignment. For the *elastic* scenario $B_{i,0} = 0.2$ *MHz* and $B_{i,0} = 1.25$ *MHz*, and for the *fixed* scenario $B_0 = 12.5$ *MHz*.

The results show that the SM-NE and the SM-OPT algorithms perform close to the MBR algorithm, despite the fact that they are based on average system parameters only. We can also observe that when the device to cloud bandwidth is low (about 0.2 *MHz*), SMC-NE does not provide significant gain compared to local execution (the performance gain is close to one for all values of N). On the contrary, the MBR, SM-NE and SM-OPT algorithms, which allow collaborative offloading, provide a performance gain of about 50%, and the gain slightly increases with the number of devices. The reason for the slight increase of the gain is that when there are more devices, devices are closer to each other on average, which allows higher transmission rates between devices.

Compared to the case when $B_{i,0} = 0.2$ *MHz*, the results for $B_{i,0} = 1.25$ *MHz* show that all algorithms achieve very high performance gains (up to 300%). Furthermore, the performance gain of the SMC-NE algorithm is similar to that of the SM-NE and the SM-OPT algorithms, while the MBR algorithm performs slightly better. The reason is that for high device to cloud bandwidth in the static mixed equilibrium most devices offload to the edge cloud, as on average it is best to do so, even if given

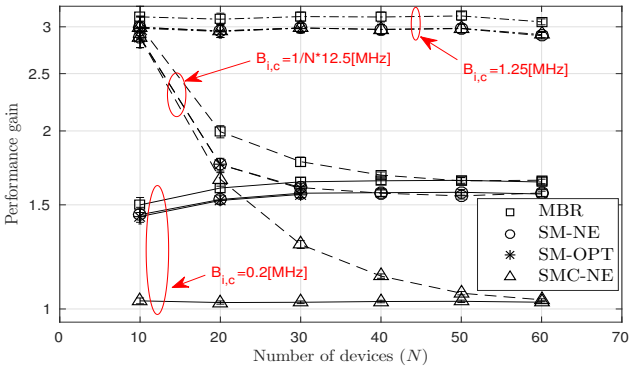

Figure 5: *Performance gain* vs. number of devices for $B_{i,0} = 0.2$ *MHz*, $B_{i,0} = 1.25$ *MHz* and $B_{i,0} = \frac{1}{N}12.5$ *MHz*.

the instantaneous system state it may be better to offload to a device, as done by the MBR algorithm. Furthermore, unlike for $B_{i,0} = 0.2$ *MHz*, for $B_{i,0} = 1.25$ *MHz* the performance gain becomes fairly insensitive to the number of devices, which is again due to the increased reliance on the cloud resources for computation offloading.

The results are fairly different for the *fixed* device to cloud bandwidth assignment scenario, as in this scenario the number of devices affects the device to cloud bandwidth. In this scenario collaboration among the devices improves the system performance (SMC-NE vs. SM-NE algorithms). We can also observe that as N increases, the curves for *fixed* scenario approach the curves for the *elastic* scenario for $B_{i,0} = 0.2$ *MHz*. This is due to that for large values of N the device to cloud bandwidth $B_{i,0}$ becomes low and the devices offload more to each other than to the edge cloud.

Finally, the results show that the gap between the *SM-NE* and the *SM-OPT* algorithms is almost negligible for all scenarios, and hence we can conclude that the price of stability of the MCOG game in static mixed strategies is close to one.

5.2 Impact of cloud availability

In order to analyse the impact of the possibility to offload to the edge cloud, in the following we vary the bandwidth $B_{i,0}$ between 0.2 *MHz* and 5.2 *MHz*.

Figure 6 shows the average and the median performance gain for the MBR, SM-NE, SM-OPT and SMC-NE algorithms as a function of the device to cloud bandwidth for 8 devices placed over a square area of $0.5km \times 0.5km$, for 30 devices placed over a square area of $1km \times 1km$, and for 60 devices placed over a square area of $1.41km \times 1.41km$. Note that the three scenarios have approximately the same density of devices. We first observe that the median performance gain is almost equal to the average performance gain for all algorithms and for all considered

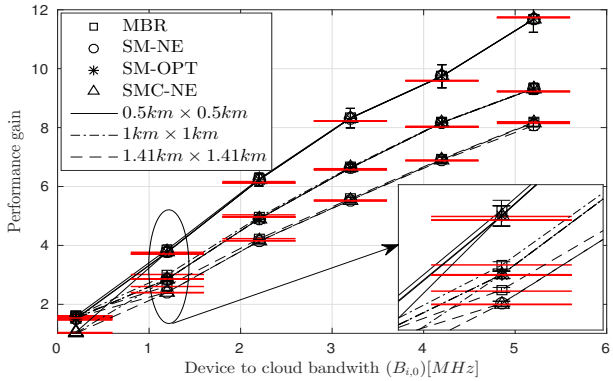

Figure 6: *Performance gain* vs. device to cloud bandwidth $B_{i,0}$ for $N = 8$ devices placed over $0.5km \times 0.5km$ square area, for $N = 30$ devices placed over $1km \times 1km$ square area, and for $N = 60$ devices placed over $1.41km \times 1.41km$ square area.

scenarios, which suggests that distribution of the completion times of the tasks is approximately symmetrical. The figure shows that the performance gain achieved by the algorithms increases with the bandwidth $B_{i,0}$. Furthermore, we observe that the gap between the algorithms decreases as the device to cloud bandwidth increases, and for reasonably high bandwidths the SM-NE algorithm performs almost equally well as the MBR algorithm. The results also show that collaboration among the devices has highest impact on the system performance when the bandwidth $B_{i,0}$ is low, and for $B_{i,0} = 1.2$ *MHz* offloading to the edge cloud only (SMC-NE) is as good as the SM-NE and SM-OPT algorithms.

Comparing the performance for different sized areas we observe that the performance gain decreases as the size of the area increases, which is due to that the devices are closer to the cloud server on average in a smaller area.

5.3 Impact of cloud remoteness

In order to evaluate the impact of the cloud access latency, in the following we vary the latency τ_c between 0 s and 0.4 s. A low latency ($0ms \leq \tau_c < 20ms$) would correspond to the case of an edge cloud or a home gateway, a moderate latency ($20ms \leq \tau_c < 100ms$) would correspond to an edge cloud located deeper in the network (e.g., metro network), and high latency ($100ms \leq \tau_c$) would correspond to remote cloud servers.

In Figure 7 we show the average performance gain as a function of the latency τ_c for the MBR, SM-NE, SM-OPT and SMC-NE algorithms in a fog computing system that serves $N = 30$ devices, each of them assigned a bandwidth of $B_{i,0} = 1.25$ *MHz* for communication with the cloud. The figure shows that the performance gain of all algorithms decreases as the latency to the cloud server increases. Furthermore,

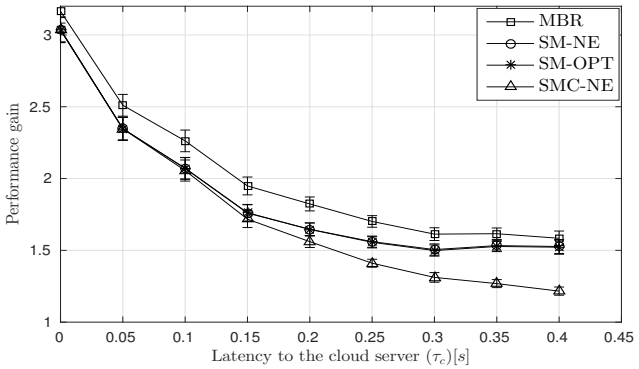

Figure 7: *Performance gain* vs. latency τ_c to the cloud server, for $N = 30$ devices placed over $1km \times 1km$ square area, and $B_{i,0} = 1.25$ *MHz*.

we observe that the performance gain of the SMC-NE algorithm approaches one, as in the case of a high latency it is better for most of devices to perform the computation locally. On the contrary, the performance gain of the MBR, SM-NE and SM-OPT algorithms remains slightly above 1.5 even for high values of the latency ($\tau_c \geq 300ms$), which additionally confirms that devices can decrease the average completion times of their tasks through collaboration even in systems where they cannot entirely rely on the cloud resources.

5.4 Performance gain perceived per device

In order to evaluate the performance gain perceived per device, we use the notion of ex-ante and ex-post individual rationality. These are important in situations when the devices are allowed to decide whether or not to participate in the collaboration before and after learning their types (i.e., the exact size and complexity of their tasks), respectively. The results in Figure 5 show that on average the devices benefit from collaboration, as the performance gain is greater than one, and hence collaboration among the devices is ex-ante individually rational. In order to investigate whether collaboration among the devices is ex-post individually rational, in Figure 8 we plot the CDF of the performance gain for the *elastic* device to cloud bandwidth assignment scenario with 30 devices and for $B_{i,0} = 0.2$ *MHz*, $B_{i,0} = 0.8$ *MHz*, and $B_{i,0} = 1.25$ *MHz*.

The results for $B_{i,0} = 0.2$ *MHz* show that the SMC-NE algorithm is ex-post individually rational, as devices always gain compared to local computation. At the same time, the SM-NE and MBR algorithms achieve a performance gain below one for a small fraction of the devices, and hence collaboration among devices is not ex-post individually rational. On the contrary, the results for $B_{i,0} = 0.8$ *MHz* show that the MBR algorithm is ex-post individually rational, since the performance

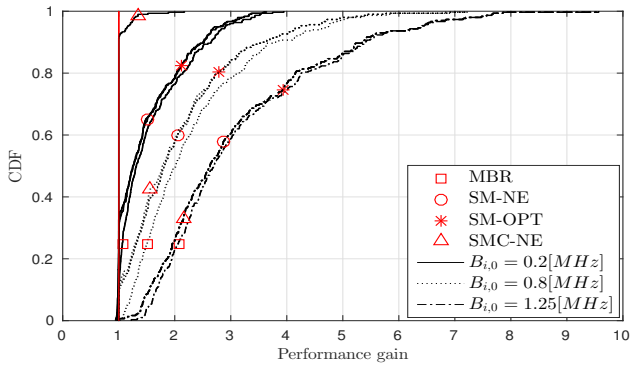

Figure 8: Distribution of the performance gain for $N = 30$ devices, $B_{i,0} = 0.2$ *MHz*, $B_{i,0} = 0.8$ *MHz* and $B_{i,0} = 1.25$ *MHz*.

gain of every device is larger than one, but the SM-NE is not. Finally, the results for $B_{i,0} = 1.25$ *MHz* show that all algorithms ensure that every device achieves a performance gain at least one, and hence for $B_{i,0} = 1.25$ *MHz* collaboration among devices is ex-post individually rational using all algorithms.

The above results show that collaboration among the devices is ex-post individually rational only if sufficient bandwidth is provided for communication to the edge cloud. Thus, if ex-post individual rationality is important then the device to cloud bandwidth has to be managed appropriately.

5.5 Utilization ratio of collaboration among devices

In order to evaluate the impact of collaboration on the system performance, we consider the ratio of the tasks executed at different nodes in the system. To obtain this ratio, we simulated stochastic task arrivals over a period of $10^4 s$. We recorded the N_t tasks generated in the system during this period, and for an algorithm $A \in \{\text{MBR}, \text{SM-NE}, \text{SM-OPT}\}$ we recorded N_l^A and N_c^A, the number of tasks executed locally and the number of tasks executed in the edge cloud, respectively. Figure 9 shows the ratio $\frac{N_l^A}{N_t}$ of the tasks executed locally, and the ratio $\frac{N_t - N_c^A}{N_t}$ of the tasks executed either locally or at one of the other devices for the MBR, SM-NE and SM-OPT algorithms as a function of the number of devices for $B_{i,0} = \frac{1}{N} 12.5$ *MHz*.

The results in Figure 9 show that for $N = 10$, i.e., when the bandwidth assigned to each device for communication with the edge cloud is 1.25 *MHz*, the devices offload more tasks to the edge cloud in the case of the SM-NE and SM-OPT algorithms than in the case of the MBR algorithm, which coincides with the observation made in Figure 5 for $B_{i,0} = 1.25$ *MHz*. On the contrary, when $N \geq 20$ the devices

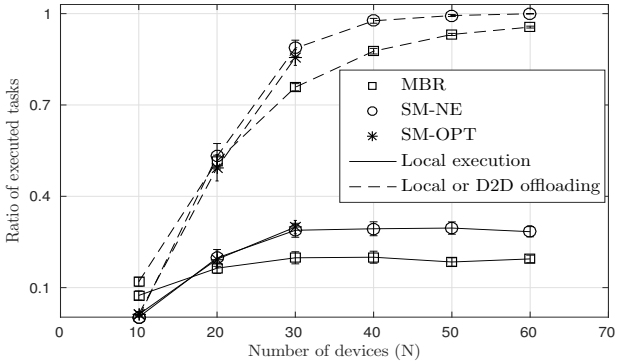

Figure 9: Ratio of the tasks executed locally and the tasks executed at any of the devices for $B_{i,0} = \frac{1}{N}12.5\ MHz$.

offload more tasks to the edge cloud in the case of the MBR algorithm than in the case of the SM-NE and SM-OPT algorithms that achieve approximately the same performance. Furthermore, we observe that while the ratio of the tasks executed locally increases up to 30 users and remains constant for more devices, the ratio of the tasks executed either locally or at one of the other devices continues to increase with the number of devices for all algorithms. These results confirm the observation made for $B_{i,0} = \frac{1}{N}12.5\ MHz$ in Figure 5 that the collaboration among the devices improves the system performance.

5.6 Computational efficiency of the SM-NE algorithm

Recall that the SM-NE algorithm is based on the static mixed strategy equilibrium, and that the SM-OPT algorithm is based on the socially optimal static mixed strategy profile. In order to assess the computational efficiency of the SM-NE algorithm we measured the time needed to compute a static mixed strategy equilibrium by the *ST-PC* method and the time needed to compute a socially optimal static mixed strategy profile by the quasi-Newton method. Figure 10 shows the measured times as a function of the number of devices. We observe that the time needed to compute the socially optimal static mixed strategy profile increases exponentially with the number of devices at a fairly high rate, and already for 30 devices it is more than an order of magnitude faster to compute a static mixed strategy equilibrium than to compute the socially optimal static mixed strategy profile. Therefore, we conclude that the SM-NE algorithm, which is based on an equilibrium in static mixed strategies, is a computationally efficient solution for medium to large scale collaborative computation offloading systems.

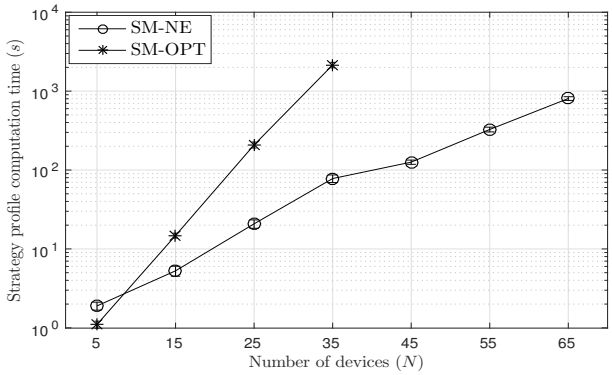

Figure 10: Time needed to compute a static mixed strategy equilibrium and a socially optimal static mixed strategy profile for $B_{i,0} = 1.25\ MHz$.

6 Related Work

There is a large body of work on augmenting the execution of computationally intensive applications using cloud resources [52], [53], [27, 54–56]. In [52] the authors studied the problem of maximizing the throughput of mobile data stream applications through partitioning, and proposed a genetic algorithm as a solution. The authors in [53] considered multiple QoS factors in a 2-tiered cloud infrastructure, and proposed a heuristic for minimizing the users' cost. In [54] the authors proposed an iterative algorithm that minimizes the users' overall energy consumption, while meeting latency constraints. The authors in [55] considered the joint optimization of the offloading decisions, and the allocation of communication and computation resources, proved the NP-hardness of the problem and proposed a heuristic offloading decision algorithm for minimizing the completion time and the energy consumption of devices. The authors in [27] considered a single wireless link and an elastic cloud, provided a game theoretic treatment of the problem of minimizing completion time and showed that the game is a potential game. The authors in [56] considered multiple wireless links, elastic and non-elastic cloud, provided a game theoretic analysis of the problem and proposed a polynomial complexity algorithm for computing an equilibrium allocation. In [19] the authors considered a three-tier cloud architecture with stochastic task arrivals, provided a game theoretical formulation of the problem, and used a variational inequality to prove the existence of a solution and to provide a distributed algorithm for computing an equilibrium. Unlike these works, we allow devices to offload computations to each other as well.

A few recent works considered augmenting the execution of computationally intensive applications using the computational power of nearby devices in a collaborative way [18, 39, 57–59]. The authors in [57] modeled the collaboration among mobile devices as a coalition game, and proposed a heuristic method for solving a

$0-1$ integer quadratic programing problem that minimizes the overall energy consumption. In [58] the authors formulated the resource allocation problem among neighboring mobile devices as a multi-objective optimization that aims to minimize the completion times of the tasks as well as the overall energy consumption, and as a solution proposed a two-stage approach based on enumerating Pareto optimal solutions. In [59] the authors formulated the problem of maximizing the probability of computing tasks before their deadlines through mobility-assisted opportunistic computation offloading as a convex optimization problem, and used the barrier method to solve the problem. The authors in [18] considered a collaborative cloudlet that consists of devices that can perform shared offloading, and proposed two heuristic allocation algorithms that minimize the average relative usage of all the nodes in the cloudet. The authors in [39] proposed an architecture that enables a mobile device to remotely access computational resources on other mobile devices, and proposed two greedy algorithms that require complete information about devices' states, for minimizing the job completion time and the energy consumption, respectively. Our work differs from these works, as we consider computation offloading to an edge cloud and nearby devices, and provide a non-cooperative game theoretic treatment of the problem.

Only a few recent works considered the computation offloading problem in fog computing systems [60–63]. The authors in [60] considered a fog computing system in which the tasks can be performed locally at the devices, at a fog node or at a remote cloud server, and proposed a suboptimal algorithm for computing the offloading decisions and allocating resources with the objective to minimize the delay and the energy consumption of devices. In [61] the authors considered a fog computing system, where devices may offload their computation to small cell access points that provide computation and storage capacities, and designed a heuristic for a joint optimization of radio and computational resources with the objective of minimizing the energy consumption. Unlike this work, we consider stochastic task arrivals, and we provide a game theoretical treatment of the completion time minimization problem. In [62] authors formulated the power consumption-delay tradeoff problem in fog computing system that consists of a set of fog devices and a set of cloud servers, and proposed a heuristic for allocating the workload among fog devices and cloud servers. In [63] the authors considered the joint optimization problem of task allocation and task image placement in a fog computing system that consists of a set of storage srevers, a set of computation servers and a set of users, and proposed a low-complexity three-stage algorithm for the task completion time minimization problem. Our work differs from these works, as we consider heterogeneous computational tasks, and our queueing system model captures the contention for both communication and computational resources.

To the best of our knowledge ours is the first work based on a game theoretical analysis that proposes a decentralized algorithm with low signaling overhead for solving the completion time minimization problem in fog computing systems.

7 Conclusion

We have provided a game theoretical analysis of a fog computing system. We proposed an efficient decentralized algorithm based on an equilibrium task allocation in static mixed strategies. We compared the performance achieved by the proposed algorithm that relies on average system parameters with the performance of a myopic best response algorithm that requires global knowledge of the system state. Our numerical results show that the proposed algorithm achieves good system performance, close to that of the myopic best response algorithm, and could be a possible solution for coordinating collaborative computation offloading with low signaling overhead. There is a number of interesting extensions of our model. First, one could consider a communication model in which devices share the bandwidth with each other. Another direction is to consider the energy cost of offloading, e.g., use it as a constraint for offloading optimization.

References

[1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, pp. 854–864, 2016.

[2] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, pp. 112–116, 2016.

[3] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for internet of things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.

[4] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. of ACM MobiSys*, 2010, pp. 49–62.

[5] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[6] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. of IEEE INFOCOM*, 2012, pp. 2716–2720.

[7] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5G be?" *IEEE J-SAC*, pp. 1065–1082, 2014.

[8] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, pp. 64–70, 2014.

[9] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *Proc. of IEEE Networks and Communications (EuCNC)*, 2017, pp. 1–6.

[10] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[11] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[12] G. Fodor, E. Dahlman, G. Mildh, S. Parkvall, N. Reider, G. Miklós, and Z. Turányi, "Design aspects of network assisted device-to-device communications," *IEEE Communications Magazine*, vol. 50, no. 3, 2012.

[13] K. Doppler, C.-H. Yu, C. B. Ribeiro, and P. Janis, "Mode selection for device-to-device communication underlaying an lte-advanced network," in *Proc. of IEEE WCNC*, 2010, pp. 1–6.

[14] M. Zulhasnine, C. Huang, and A. Srinivasan, "Efficient resource allocation for device-to-device communication underlaying lte network," in *Proc. of IEEE WiMob*, 2010, pp. 368–375.

[15] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of USENIX Conference on Hot Topics in Cloud Computing*, 2010, pp. 4–4.

[16] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 1. ACM, 2001, pp. 50–61.

[17] W. Yuan and K. Nahrstedt, "Energy-efficient cpu scheduling for multimedia applications," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 3, pp. 292–331, 2006.

[18] S. Bohez, T. Verbelen, P. Simoens, and B. Dhoedt, "Discrete-event simulation for efficient and stable resource allocation in collaborative mobile cloudlets," *Simulation Modelling Practice and Theory*, vol. 50, pp. 109–129, 2015.

[19] V. Cardellini, V. De Nitto Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, 2015.

[20] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *Service Oriented System Engineering*, Mar. 2013, pp. 494–502.

[21] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J-SAC*, vol. 34, no. 12, pp. 3887–3901, 2016.

[22] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proc. of ACM International Conference on Information and Knowledge Management*, 2001, pp. 310–317.

[23] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *Proc. of ACM International Conference on World Wide Web*, 2003, pp. 640–651.

[24] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, pp. 843–857, 2004.

[25] P. Mach, Z. Becvar, and T. Vanek, "In-band device-to-device communication in OFDMA cellular networks: A survey and challenges," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 1885–1922, 2015.

[26] S. Sharma, N. Gupta, and V. A. Bohara, "OFDMA-based device-to-device communication frameworks: Testbed deployment and measurement results," *IEEE Access*, vol. 6, pp. 12 019–12 030, 2018.

[27] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[28] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[29] S. Jošilo and G. Dan, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Transactions on Mobile Computing*, 2018.

[30] S. Jošilo and G. Dán, "Decentralized scheduling for offloading of periodic tasks in mobile edge computing," in *Proc. of IFIP NETWORKING*, 2018.

[31] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything*. Springer, 2018, pp. 103–130.

[32] A. Brogi, S. Forti, A. Ibrahim, and L. Rinaldi, "Bonsai in the fog: an active learning lab with fog computing," in *Proc. of IEEE International Conference on Fog and Mobile Edge Computing*, Apr. 2018.

[33] L. I. Sennott, "Nonzero-sum stochastic games with unbounded costs: discounted and average cost cases," *Mathematical Methods of Operations Research*, vol. 40, no. 2, pp. 145–162, 1994.

[34] E. Altman, A. Hordijk, and F. Spieksma, "Contraction conditions for average and α-discount optimality in countable state markov games with unbounded rewards," *Mathematics of Operations Research*, vol. 22, no. 3, pp. 588–618, 1997.

[35] A. S. Nowak, "Sensitive equilibria for ergodic stochastic games with countable state spaces," *Mathematical Methods of Operations Research*, vol. 50, no. 1, pp. 65–76, 1999.

[36] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 120–128, 2016.

[37] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proc. of the ASE BigData & SocialInformatics*. ACM, 2015, p. 28.

[38] O. Consortium *et al.*, "OpenFog reference architecture for fog computing," Tech. Rep., Feb. 2017.

[39] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: enabling remote computing among intermittently connected mobile devices," in *Proc. of ACM MobiHoc*, 2012, pp. 145–154.

[40] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2018.

[41] F. Facchinei and J.-S. Pang, *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007.

[42] F. Facchinei, A. Fischer, and V. Piccialli, "On generalized nash games and variational inequalities," *Operations Research Letters*, vol. 35, no. 2, pp. 159–164, 2007.

[43] F. Tinti, "Numerical solution for pseudomonotone variational inequality problems by extragradient methods," in *Variational Analysis and Applications*. Springer, 2005, pp. 1101–1128.

[44] E. Balevi and R. D. Gitlin, "Optimizing the number of fog nodes for cloud-fog-thing networks," *IEEE Access*, vol. 6, pp. 11 173–11 183, 2018.

[45] S. Sigg, P. Jakimovski, and M. Beigl, "Calculation of functions on the RF-channel for IoT," in *Proc. of IEEE IOT*, 2012, pp. 107–113.

[46] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, March 2017.

[47] Y.-L. Chung, "Rate-and-power control based energy-saving transmissions in OFDMA-based multicarrier base stations," *IEEE Systems Journal*, vol. 9, no. 2, pp. 578–584, 2015.

[48] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu, "Device-to-device communication in 5G cellular networks: challenges, solutions, and future directions," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 86–92, 2014.

[49] A. Aragon-Zavala, *Antennas and propagation for wireless communication systems*. John Wiley & Sons, 2008.

[50] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proc. of ACM Internet Measurement Conference (IMC)*, 2009, pp. 280–293.

[51] M. Satyanarayanan, "A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets," *GetMobile: Mobile Computing and Communications*, pp. 19–23, 2015.

[52] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, Apr. 2013.

[53] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, "On optimal and fair service allocation in mobile cloud computing," *IEEE Transactions on Cloud Computing*, 2015.

[54] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[55] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[56] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of IEEE INFOCOM*, 2017.

[57] L. Xiang, B. Li, and B. Li, "Coalition formation towards energy-efficient collaborative mobile computing," in *Proc. of IEEE ICCCN*, 2015, pp. 1–8.

[58] S. Ghasemi-Falavarjani, M. Nematbakhsh, and B. S. Ghahfarokhi, "Context-aware multi-objective resource allocation in mobile cloud," *Computers & Electrical Engineering*, vol. 44, pp. 218–240, 2015.

[59] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Communications Letters*, vol. 18, no. 10, pp. 1779–1782, 2014.

[60] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, 2017.

[61] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. of IEEE Vehicular Technology Conference*, 2015, pp. 1–6.

[62] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[63] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[64] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 2012.

[65] D. S. Bernstein, *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory*. Princeton University Press Princeton, 2005, vol. 41.

A Appendix

A.1 Proof of Theorem 2

Observe that if $\lambda_i = \lambda$ then the cost C_i can equivalently be defined as $N_i = \lambda C_i$, i.e., the number of tasks in the system. Furthermore, since task complexities are assumed to be exponentially distributed, the execution queues are $M/M/1$ systems. We can thus rewrite $\overline{T^e}_{i,j}$ as

$$\overline{T^e}_{i,j} = \frac{\overline{T^c}_{i,j}}{1-\rho_j^e}, \tag{9}$$

and the cost $N_i(p_i, p_{-i})$ of device i as

$$\begin{aligned} N_i(p_i,p_{-i}) =& p_{i,i}\lambda\frac{\overline{T^c}_{i,i}}{\zeta_i} + p_{i,0}\lambda\big(\frac{\gamma_{i,0}}{2\epsilon_{i,0}} + \overline{T^t}_{i,0} + \overline{T^c}_{i,0}\big) \\ &+ \sum_{j\in\mathcal{N}\setminus\{i\}} p_{i,j}\lambda\big(\frac{\gamma_{i,j}}{2\epsilon_{i,j}} + \overline{T^t}_{i,j} + \frac{\overline{T^c}_{i,j}}{\zeta_j}\big). \end{aligned}$$

Next, we express the first order derivatives $h_{i,j}$ of $N_i(p_i, p_{-i})$ as

$$h_{i,0} = \lambda\big(\overline{T^t}_{i,0} + \overline{T^c}_{i,0} + \frac{\gamma_{i,0}}{2\epsilon_{i,0}}\big) + p_{i,0}\lambda^2\big(\frac{{}^2\overline{T^t}_{i,0}}{2\epsilon_{i,0}} + \frac{\overline{T^t}_{i,0}\gamma_{i,0}}{2\epsilon_{i,0}^2}\big),$$

$$h_{i,i} = \lambda\frac{\overline{T^c}_{i,i}}{\zeta_i} + p_{i,i}\lambda^2\frac{\overline{T^c}_{i,i}^2}{\zeta_i^2},$$

$$\begin{aligned} h_{i,j}|_{j\neq i} =& \lambda\big(\overline{T^t}_{i,j} + \frac{\gamma_{i,j}}{2\epsilon_{i,j}} + \frac{\overline{T^c}_{i,j}}{\zeta_j}\big) \\ &+ p_{i,j}\lambda^2\big(\frac{{}^2\overline{T^t}_{i,j}}{2\epsilon_{i,j}} + \frac{\overline{T^t}_{i,j}\gamma_{i,j}}{2\epsilon_{i,j}^2} + \frac{\overline{T^c}_{i,j}^2}{\zeta_j^2}\big). \end{aligned}$$

In order to prove the monotonicity of the function F in what follows we show that the Jacobian J of F is positive semidefinite. The Jacobian J has the following structure

$$\begin{pmatrix} h^1_{1,0} & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & h^1_{1,1} & \dots & 0 & 0 & h^1_{2,1} & \dots & 0 & \dots & 0 & h^1_{N,1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & h^1_{1,N} & 0 & 0 & \dots & h^1_{2,N} & \dots & 0 & 0 & \dots & h^1_{N,N} \\ \vdots & & & & & & & & & & & & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & h^N_{N,0} & 0 & \dots & 0 \\ 0 & h^N_{1,1} & \dots & 0 & 0 & h^N_{2,1} & \dots & 0 & \dots & 0 & h^N_{N,1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & h^N_{1,N} & 0 & 0 & \dots & h^N_{2,N} & \dots & 0 & 0 & \dots & h^N_{N,N} \end{pmatrix},$$

where the second order derivatives can be expressed as

$$h^i_{i,0} = \frac{\lambda^2}{\epsilon_{i,0}}\big({}^2\overline{T^t}_{i,0} + \frac{\gamma_{i,0}\overline{T^t}_{i,0}}{\epsilon_{i,0}}\big)\big(1 + p_{i,0}\frac{\lambda\overline{T^t}_{i,0}}{\epsilon_{i,0}}\big)$$

$$h^i_{i,i} = \left(\frac{\lambda\overline{T^c}_{i,i}}{\zeta_i}\right)^2\big(2 + 2\frac{\lambda}{\zeta_i}p_{i,i}\overline{T^c}_{i,i}\big),$$

$$h^i_{i,j}\big|_{j\neq i} = \left(\frac{\lambda\overline{T^c}_{i,j}}{\zeta_j}\right)^2\big(2 + 2\frac{\lambda}{\zeta_j}p_{i,j}\overline{T^c}_{i,j}\big) + h^t_{i,j},$$
$$\text{where } h^t_{i,j} = \frac{\lambda^2}{\epsilon_{i,j}}\big({}^2\overline{T^t}_{i,j} + \frac{\gamma_{i,j}\overline{T^t}_{i,j}}{\epsilon_{i,j}}\big)\big(1 + p_{i,j}\frac{\lambda\overline{T^t}_{i,j}}{\epsilon_{i,j}}\big),$$

and

$$h^i_{i',j}\Big|_{i'\neq i} = \frac{\lambda\overline{T^c}_{i,j}\lambda\overline{T^c}_{i',j}}{\zeta_j^2}\big(1 + 2\frac{\lambda}{\zeta_j}p_{i,j}\overline{T^c}_{i,j}\big).$$

Reordering the rows and columns, the Jacobian J can be rewritten as

$$J = \begin{pmatrix} C & 0 & \dots & 0 \\ 0 & M_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M_N \end{pmatrix},$$

where

$$C = \begin{pmatrix} h^1_{1,0} & 0 & \dots & 0 \\ 0 & h^2_{2,0} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h^N_{N,0} \end{pmatrix}, M_i = \begin{pmatrix} h^1_{1,i} & h^1_{2,i} & \dots & h^1_{N,i} \\ h^2_{1,i} & h^2_{2,i} & \dots & h^2_{N,i} \\ \vdots & \vdots & \ddots & \vdots \\ h^N_{1,i} & h^N_{2,i} & \dots & h^N_{N,i} \end{pmatrix}.$$

Observe that all diagonal elements of C are nonnegative, and thus the matrix C is positive definite. In order to show that J is positive semidefinite we have to show that the symmetric matrix $M^s_i = \frac{1}{2}(M_i^T + M_i)$ is positive semidefinite.

The diagonal elements ${}^dh^s_{j,i}$ of M^s_i are given by

$${}^dh^s_{j,i}\Big|_{j=i} = \left(\frac{\lambda\overline{T^c}_{i,i}}{\zeta_i}\right)^2\big(2 + 2\frac{\lambda}{\zeta_i}p_{i,i}\overline{T^c}_{i,i}\big),$$

$${}^dh^s_{j,i}\Big|_{j\neq i} = \left(\frac{\lambda\overline{T^c}_{j,i}}{\zeta_i}\right)^2\big(2 + 2\frac{\lambda}{\zeta_i}p_{j,i}\overline{T^c}_{j,i}\big) + h^t_{j,i},$$
$$\text{where } h^t_{j,i} = \frac{\lambda^2}{\epsilon_{j,i}}\big({}^2\overline{T^t}_{j,i} + \frac{\gamma_{j,i}\overline{T^t}_{j,i}}{\epsilon_{j,i}}\big)\big(1 + p_{j,i}\frac{\lambda\overline{T^t}_{j,i}}{\epsilon_{j,i}}\big),$$

and the off-diagonal elements ${}^{o}h^{s}_{j,i} = \frac{1}{2}(h^{i}_{j,i} + h^{j}_{i,i})\Big|_{j\neq i}$ are given by

$$ {}^{o}h^{s}_{j,i} = \frac{\lambda\overline{Tc}_{i,i}\lambda\overline{Tc}_{j,i}}{\zeta_i^2}\big(1 + \frac{\lambda}{\zeta_i}(p_{i,i}\overline{Tc}_{i,i} + p_{j,i}\overline{Tc}_{j,i})\big) $$

Let us define the vector $\overline{Tc}_i = (\overline{Tc}_{1,i}\ \overline{Tc}_{2,i}\dots\overline{Tc}_{N,i})^T$ and matrix $\overline{Tt}_i$

$$ \overline{Tt}_i = \begin{pmatrix} \mathrm{diag}(h^{t}_{j,i})\big|_{j\in\mathcal{N}\setminus\{i\}} & 0 \\ 0 & 0 \end{pmatrix}. $$

Furthermore, let us define matrix T^p_i as

$$ \begin{pmatrix} p_{1,i}\overline{Tc}_{1,i} & \frac{p_{1,i}\overline{Tc}_{1,i}+p_{2,i}\overline{Tc}_{2,i}}{2} & \dots & \frac{p_{1,i}\overline{Tc}_{1,i}+p_{N,i}\overline{Tc}_{N,i}}{2} \\ \frac{p_{2,i}\overline{Tc}_{2,i}+p_{1,i}\overline{Tc}_{1,i}}{2} & p_{2,i}\overline{Tc}_{2,i} & \dots & \frac{p_{2,i}\overline{Tc}_{2,i}+p_{N,i}\overline{Tc}_{N,i}}{2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{p_{N,i}\overline{Tc}_{N,i}+p_{1,i}\overline{Tc}_{1,i}}{2} & \frac{p_{N,i}\overline{Tc}_{N,i}+p_{2,i}\overline{Tc}_{2,i}}{2} & \dots & p_{N,i}\overline{Tc}_{N,i} \end{pmatrix}. $$

Now, matrix M_i can be rewritten as

$$ M_i = \frac{\lambda^2}{\zeta_i^2}\left(\overline{Tc}_i\ \overline{Tc}_i^T \circ \left(I + E + \frac{2\lambda}{\zeta_i}T^p_i\right)\right) + \overline{Tt}_i, $$

where $\circ$ denotes the Hadamard product, i.e., the component-wise product of two matrices.

It is well known that the identity I and unit E matrices are positive definite, while positive definiteness of matrix $\overline{Tc}_i\ \overline{Tc}_i^T$ follows from the definition. Observe that matrix $\overline{Tt}_i$ is positive semidefinite as well, since it is a diagonal matrix with non-negative elements. Since the sum of two positive semidefinite matrices is positive semidefinite and the Hadamard product of two positive semidefinite matrices is also positive semidefinite [64], the proof reduces to showing that matrix $I + E + \frac{2\lambda}{\zeta_i}T^p_i$ is positive semidefinite. To do so, we will show that the minimum eigenvalue of the matrix $\frac{2\lambda}{\zeta_i}T^p_i$ is greater than or equal to -1. To do so, let us denote by e the all-ones vector and define the vector $t^p_i = (p_{1,i}\overline{Tc}_{1,i}\ p_{2,i}\overline{Tc}_{2,i}\dots p_{N,i}\overline{Tc}_{N,i})$. Now, we can express matrix T^p_i as

$$ T^p_i = \frac{1}{2}\big(t^p_i e^T + e(t^p_i)^T\big). $$

The characteristic polynomial of the symmetric matrix T^p_i is given by [65]

$$ \frac{k^{N-2}}{2}\big(k^2 - 2(e^T t^p_i)k + (e^T t^p_i)^2 - N\|t^p_i\|^2\big). $$

We observe that T^p_i has $N-2$ zero eigenvalues, and one non-negative and one non-positive eigenvalue given by $k_+ = \big(e^T t^p_i + \sqrt{N}\|t^p_i\|\big)/2$ and $k_- = \big(e^T t^p_i -$

$\sqrt{N}\|t_i^p\|\big)/2$, respectively. Therefore, the minimum eigenvalue of the matrix $\frac{2\lambda}{\zeta_i}T_i^p$ is greater than -1 if

$$\frac{\lambda}{\zeta_i}\big(\sqrt{N}\|t_i^p\| - e^T t_i^p\big) \leq 1. \tag{10}$$

Since t_i^p is a vector with non-negative elements, we have that $e^T t_i^p \geq \|t_i^p\|$ and it also holds that $\|t_i^p\| \leq \sqrt{N}\max_{j\in\mathcal{N}} t_{j,i}$. Therefore, the following inequalities hold

$$\begin{aligned}
&\frac{\lambda}{\zeta_i}\big(\sqrt{N}\|t_i^p\| - e^T t_i^p\big) \leq \frac{\lambda}{\zeta_i}\big(\sqrt{N}\max_{j\in\mathcal{N}} t_{j,i}(\sqrt{N}-1)\big)\\
&\leq \frac{N\lambda}{\zeta_i}\max_{j\in\mathcal{N}} t_{j,i} \leq \frac{N\lambda}{\zeta_i}\max_{j\in\mathcal{N}} \overline{T^c}_{j,i}.
\end{aligned}$$

Since $\rho_i^e \leq S_t$, we have that $\zeta_i \geq 1 - S_t$, and therefore

$$\frac{N\lambda}{\zeta_i}\max_{j\in\mathcal{N}} \overline{T^c}_{j,i} \leq \frac{N\lambda}{1-S_t}\max_{j\in\mathcal{N}} \overline{T^c}_{j,i}. \tag{11}$$

Based on (11) a sufficient condition for (10) is that $\lambda\max_{j\in\mathcal{N}} \overline{T^c}_{j,i} \leq \frac{1-S_t}{N}$. This proves the theorem.

Paper B

Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks

Slađana Jošilo and György Dán

IEEE Transactions on Mobile Computing (TMC), vol.18, no. 1, pp. 207-220, 2019.

Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks

Slađana Jošilo and György Dán
School of Electrical Engineering and Computer Science
KTH, Royal Institute of Technology, Stockholm, Sweden
E-mail: {josilo, gyuri}@kth.se *

Abstract

Offloading computation to a mobile cloud is a promising solution to augment the computation capabilities of mobile devices. In this paper we consider selfish mobile devices in a dense wireless network, in which individual mobile devices can offload computations through multiple access points or through the base station to a mobile cloud so as to minimize their computation costs. We provide a game theoretical analysis of the problem, prove the existence of pure strategy Nash equilibria, and provide an efficient decentralized algorithm for computing an equilibrium. For the case when the cloud computing resources scale with the number of mobile devices we show that all improvement paths are finite. Furthermore, we provide an upper bound on the price of anarchy of the game, which serves as an upper bound on the approximation ratio of the proposed decentralized algorithms. We use simulations to evaluate the time complexity of computing Nash equilibria and to provide insights into the price of anarchy of the game under realistic scenarios. Our results show that the equilibrium cost may be close to optimal, and the convergence time is almost linear in the number of mobile devices.

Index terms— computation offloading, mobile edge computing, Nash equilibria, decentralized algorithms

1 Introduction

Mobile handsets are increasingly used for various computationally intensive applications, including augmented reality, natural language processing, face, gesture and

*The work was partly funded by SSF through the Modane project and by the Swedish Research Council through project 621-2014-6.

object recognition, and various forms of user profiling for recommendations [1, 2]. Executing such computationally intensive applications on mobile handsets may result in slow response times, and can also be detrimental to battery life, which may limit user acceptance.

Mobile cloud computing has emerged as a promising solution to serve the computational needs of these computationally intenstive applications, while potentially relieving the battery of the mobile handsets [3, 4]. In the case of mobile cloud computing the mobile devices offload the computations via a wireless network to a cloud infrastructure, where the computations are performed, and the result is sent back to the mobile handset. While computation offloading to general purpose cloud infrastructures, such as Amazon EC2, may not be able to provide sufficiently low response times for many applications, emerging mobile edge computing (MEC) resources may provide sufficient computational power close to the network edge to meet all application requirements [5].

Computation offloading to a mobile edge cloud can significantly increase the computational capability of individual mobile handsets, but the response times may suffer when many handsets attempt to offload computations to the cloud simultaneously, on the one hand due to the competition for possibly constrained edge cloud resources, on the other hand due to contention in the wireless access [6, 7]. The problem is even more complex in the case of a dense deployment of access points, e.g., cellular femtocells or WiFi access points, when each mobile user can choose among several access points to connect to. Good system performance in this case requires the coordination of the offloading choices of the indvidual mobile handsets, while respecting their individual performance objectives, both in terms of response time and energy consumption.

In this paper we consider the problem of resource allocation for computation offloading by self-interested mobile users to a mobile cloud. The objective of each mobile user is to minimize its cost, which is a linear combination of its response time and its energy consumption for performing a computational task, by choosing whether or not to offload a task via a wireless network to a mobile cloud. Clearly, the choice of a mobile user affects the cost of other mobile users. If too many mobile users choose offloading, the response times could be affected by the contention between the mobile devices for MEC computing resources and for wireless communication resources. Hence, the fundamental question is whether a self-enforcing resource allocation among mobile users exists, and if it exists, whether it can be computed by mobile users in a decentralized manner.

In order to answer this question, we formulate the computation offloading problem as a non-cooperative game and we make three important contributions. First, based on a game theoretical treatment of the problem, we propose an efficient decentralized algorithm for coordinating the offloading decisions of the mobile devices, and prove convergence of the algorithm to a pure strategy Nash equilibrium when the computational capability assigned to a mobile device by the cloud is a non-increasing function of the number of mobile users that offload. Second, we show

that a simple decentralized algorithm can be used for computing equilibria when the cloud computing resources scale directly proportional with the number of mobile users. Finally, we provide a bound on the price of anarchy for both models of cloud resources. We provide numerical results based on extensive simulations to illustrate the computational efficiency of the proposed algorithms and to evaluate the price of anarchy for scenarios of practical interest.

The rest of the paper is organized as follows. We present the system model in Section 2. We present the algorithms and prove their convergence in Sections 3 and 4, respectively. We provide a bound on the price of anarchy in Section 5 and present numerical results in Section 6. Section 7 discusses related work and Section 8 concludes the paper.

2 System Model and Problem Formulation

We consider a mobile cloud computing system that serves a set $\mathcal{N}=\{1,2,...,N\}$ of mobile users (MUs). Each MU has a computationally intensive task to perform, and can decide whether to perform the task locally or to offload the computation to a cloud server. The computational task is characterized by the size D_i of the input data (e.g., in bytes), and by the number L_i of the instructions required to perform the computation. Recent work has shown that a task's work requirement X_i per data bit can be approximated by a Gamma distribution [8, 9]. We can thus model $L_i = D_i X_i$, where X_i is a random variable with mean $\overline{X}_i$. Furthermore, we can express the expected number of instructions required for performing MU i's task as $\overline{L}_i = D_i \overline{X}_i$.

Each MU can decide whether to perform the task locally or to offload the computation to a cloud server through one of a set of access points (APs) denoted by $\mathcal{A}=\{1,2,...,A\}$ or through a base station (BS) denoted by B.

2.1 Decentralized mobile cloud computing architecture

Motivated by the emergence of mobile edge clouds, we consider that the cloud, besides providing computational resources, acts as a centralized entity that stores information about the mobile cloud computing system, e.g., achievable data rates and the number of MUs that offload. Furthermore, we consider that the cloud sends this information to the MUs so that the offloading decisions can be made in a decentralized manner. The motivation for such a decentralized implementation is twofold. First, the cloud can be relieved from complex centralized management. Second, the MUs may be autonomous entities with individual interests, and using a decentralized algorithm they would not need to reveal all their parameters to the cloud, but they only need to report their offloading decisions, which helps in protecting privacy and confidentiality.

Despite using a decentralized algorithm, devices still have to send the data pertaining to their tasks through a shared communication link, and thus to avoid

eavesdropping and integrity attacks, cryptographic protection is necessary. Protocols for securing computation offloading have been proposed in [10,11], and are out of scope for our work.

To enable a meaningful analysis of the resource allocation problem, we make the common assumption that the set of MUs does not change during the computation offloading period, i.e., in the order of seconds [4,12–15].

2.2 Communication model

If an MU i decides to offload the computation to the cloud server, it has to transmit D_i amount of data pertaining to its task to the cloud through one of the APs or through the BS. Thus, together with local computing MU i can choose an action from the set $\mathfrak{D}_i=\{0,1,2,...,A,B\}$, where 0 corresponds to local computing, i.e., no offloading. We denote by $d_i \in \mathfrak{D}_i$ the decision of MU i, and refer to it as her strategy. We refer to the collection $\mathbf{d}=(d_i)_{i\in\mathcal{N}}$ as a strategy profile, and we denote by $\mathfrak{D}=\times_{i\in\mathcal{N}}\mathfrak{D}_i$ the set of all feasible strategy profiles.

For a strategy profile $\mathbf{d}$ we denote by $n_o(\mathbf{d})$ the number of MUs that use $o \in \mathcal{A}\cup\{B\}$ for computation offloading, and by $n(\mathbf{d})=\sum_{o\in\mathcal{A}\cup\{B\}} n_o(\mathbf{d})$ the number of MUs that offload. Similarly, we denote by $O_o(\mathbf{d}) = \{i|d_i = o\}$ the set of MUs that offload through $o \in \mathcal{A}\cup\{B\}$, and we define the set of offloaders as $O(\mathbf{d}) = \cup_{o\in\mathcal{A}\cup\{B\}} O_o(\mathbf{d})$.

We denote by $R_{i,o}$ the PHY rate of MU i on $o \in \mathcal{A}\cup\{B\}$, which depends on the physical layer signal characteristics and the corresponding channel gain. We denote by $\omega_i^o(\mathbf{d})$ the uplink rate that MU i receives when she offloads through $o \in \mathcal{A}\cup\{B\}$. For the case of offloading through an AP a we consider that $\omega_i^a(\mathbf{d})$ depends on the PHY rate $R_{i,a}$ and on the number $n_a(\mathbf{d})$ of MUs that offload via AP a

$$\omega_i^a(\mathbf{d}) = R_{i,a}/n_a(\mathbf{d}). \tag{1}$$

This model of the uplink rate can be used to model the bandwidth sharing in TDMA and OFDMA based MAC protocols [16].

For the case of offloading through the BS we consider that the uplink data rate $\omega_i^B(\mathbf{d})$ of MU i is independent of the number of MUs that offload through the BS, i.e., $\omega_i^B(\mathbf{d}) = R_{i,B}$.

The uplink rate $\omega_i^o(\mathbf{d})$ together with the input data size D_i determines the transmission time $T_{i,o}^c(\mathbf{d})$ of MU i for offloading through $o \in \mathcal{A}\cup\{B\}$,

$$T_{i,o}^c(\mathbf{d}) = D_i/\omega_i^o(\mathbf{d}). \tag{2}$$

To model the energy consumption of the MUs, we consider that every MU i knows the transmit power $P_{i,o}$ that it would use to transmit the data through $o \in \mathcal{A}\cup\{B\}$, e.g., determined using an algorithm as the ones proposed in [17, 18]. Thus, the energy consumption of MU i for offloading the input data of size D_i through $o \in \mathcal{A}\cup\{B\}$ is

$$E_{i,o}^c(\mathbf{d}) = P_{i,o}T_{i,o}^c(\mathbf{d}). \tag{3}$$

2.3 Computation model

In what follows we introduce our model of the time and energy consumption of performing the computation locally and in the cloud server.

2.3.1 Local computing

In the case of local computing data need not be transmitted, but the task has to be processed using local computing power. We consider that the expected time it takes to complete MU i's task locally consists of two parts [19]. The first part is the expected CPU execution time $\overline{T}_i^{0,exe}=\overline{L}_i \cdot CPI_i \cdot CC_i$, where CPI_i and CC_i are the average number of cycles per instruction and the clock cycle time, respectively. The second part is the expected time $\overline{T}_i^{0,ot}$ the processor spends executing other tasks, including disk and memory management, I/O and operating system activities. Finally, we express the expected time it takes to complete MU i's task locally as

$$\overline{T}_i^0 = \overline{T}_i^{0,exe} + \overline{T}_i^{0,ot}, \tag{4}$$

where $\overline{T}_i^{0,ot}$ may or may not depend on $\overline{L}_i$.

This model essentially implies that the expected execution time of a task is an affine function of $\overline{L}_i$. In practice the MUs can maintain a history of past execution times and can use this history for estimating the parameters of the affine function, which allows to predict the mean execution time as a function of $\overline{L}_i$.

In order to model the expected energy consumption of local computing we denote by v_i the consumed energy per CPU cycle, and thus we obtain

$$\overline{E}_i^0 = v_i \overline{L}_i CPI_i. \tag{5}$$

2.3.2 Cloud computing

In the case of cloud computing, after the data are transmitted through one of the APs or through the BS, processing is done at the cloud server. In order to express the expected time it takes to complete MU i's task in the cloud server, we use a similar model as in the case of local computing, but we consider that the expected time $\overline{T}_i^{c,ot}(\mathbf{d})$ the cloud server spends executing other tasks, besides from the system-related tasks, may also depend on the tasks of the other MUs that offload. We make the reasonable assumption that $\overline{T}_i^{c,ot}(\mathbf{d})$ is a non-decreasing function of the number $n(\mathbf{d})$ of MUs that offload, and thus the computation capability assigned to MU i by the cloud is a non-increasing function of $n(\mathbf{d})$. Consequently, we can express the expected time it takes to complete MU i's task in the cloud server as

$$\overline{T}_i^c(\mathbf{d}) = \overline{T}_i^{c,exe} + \overline{T}_i^{c,ot}(\mathbf{d}). \tag{6}$$

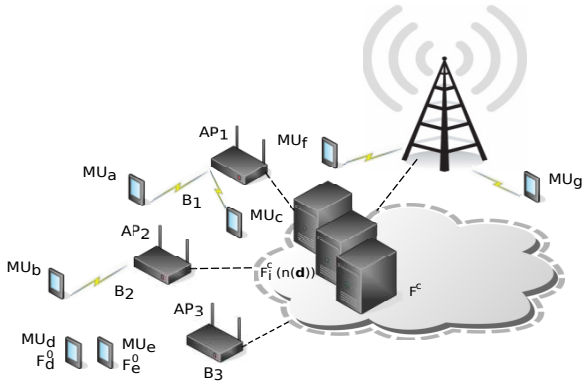

Figure 1: An example of a mobile cloud computing system

where $\overline{T}_i^{c,exe} = \overline{L}_i \cdot CPI_c \cdot CC_c$ and $\overline{T}_i^{c,ot}(\mathbf{d})$ may or may not depend on $\overline{L}_i$.

Figure 1 shows an example of a mobile cloud computing system in which 3 of 7 MUs offload their task through one of 3 APs, 2 MUs offload their task through the BS, and 2 MUs perform local computation.

2.4 Cost Model

In order to express the expected cost of MU i we denote by γ_i^E the weight attributed to energy consumption and by γ_i^T the weight attributed to the time it takes to finish the computation, $0 \leq \gamma_i^E, \gamma_i^T \leq 1$.

Using this notation, for the case of local computing the expected cost of MU i can be modeled as a linear combination of the expected time it takes to complete the task locally and the corresponding expected energy consumption,

$$C_i^0 = \gamma_i^T \overline{T}_i^0 + \gamma_i^E \overline{E}_i^0 = \gamma_i^T (\overline{T}_i^{0,exe} + \overline{T}_i^{0,ot}) + \gamma_i^E v_i \overline{L}_i CPI_i. \tag{7}$$

For the case of offloading we consider a subscription-based pricing mechanism in which MUs pay a flat fee F in order to use cloud resources [20]. Consequently, in the case of offloading through $o \in \mathcal{A} \cup \{B\}$ the expected cost of MU i can be modeled as a linear combination of the expected time it takes to complete MU i's task in the cloud server, the transmission time, the corresponding transmit energy, and a fee for using cloud resources,

$$\begin{aligned} C_{i,o}^c(\mathbf{d}) &= \gamma_i^T (\overline{T}_i^c(\mathbf{d}) + T_{i,o}^c(\mathbf{d})) + \gamma_i^E E_{i,o}^c(\mathbf{d}) + F \\ &= (\gamma_i^T + \gamma_i^E P_{i,o}) \frac{D_i}{\omega_i^o(\mathbf{d})} + \gamma_i^T \overline{T}_i^c(\mathbf{d}) + F. \end{aligned} \tag{8}$$

Similar to previous works [7, 21, 22], we do not model the time needed to transmit the results of the computation from the cloud server to the MU, as for typical applications like face and speech recognition, the size of the result of the computation is much smaller than D_i.

For notational convenience let us define the indicator function $I(d_i, o)$ for MU i as

$$I(d_i, o) = \begin{cases} 1, & \text{if } d_i = o, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

We can then express the expected cost of MU i in strategy profile $\mathbf{d}$ as

$$C_i(\mathbf{d}) = C_i^0 I(d_i, 0) + \sum_{o \in \mathcal{A} \cup \{B\}} C_{i,o}^c(\mathbf{d}) I(d_i, o). \tag{10}$$

Finally, we define the total expected cost $C(\mathbf{d}) = \sum_{i \in \mathcal{N}} C_i(\mathbf{d})$.

2.5 Computation Offloading Game

We consider that each MU aims at minimizing its expected cost (10), i.e., it aims at finding a strategy

$$d_i^* \in \arg\min_{d_i \in \mathfrak{D}_i} C_i(d_i, d_{-i}), \tag{11}$$

where we use d_{-i} to denote the strategies of all MUs except MU i. This problem formulation is not only reasonable when MUs are autonomous, but as we show later, our algorithms also serve as polynomial-time approximations for solving the problem of minimizing the total cost $C(\mathbf{d})$.

It may seem natural to model the interaction between the MUs as a Bayesian game, since the number of instructions L_i are random variables. However, it follows from (10) that the solution to (11) is determined by the expectation $\overline{L}_i$ and the number of MUs that offload. We can thus model the interaction between the MUs as a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$, in which the players are the MUs that aim at minimizing their expected cost. We refer to the game as the *multi access point computation offloading game* (MCOG), and we are interested in whether the MUs can compute a strategy profile in which no MU can further decrease her expected cost by changing her strategy, i.e., a Nash equilibrium of the game Γ.

Definition 1. A Nash equilibrium (NE) of the strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$ is a strategy profile $\mathbf{d}^*$ such that

$$C_i(d_i^*, d_{-i}^*) \leq C_i(d_i, d_{-i}^*), \ \forall d_i \in \mathfrak{D}_i.$$

Given a strategy profile (d_i, d_{-i}) we say that strategy d_i' is an improvement step for MU i if $C_i(d_i', d_{-i}) < C_i(d_i, d_{-i})$. We call a sequence of improvement steps in which one MU changes her strategy at a time an *improvement path*. Furthermore, we say that a strategy d_i^* is a best reply to d_{-i} if it solves (11), and we call an

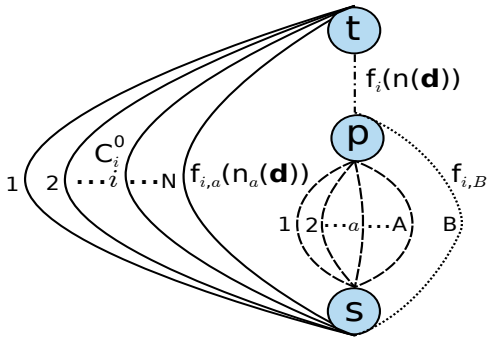

Figure 2: The computation offloading game modeled as a player-specific congestion game. The source node s represents N MUs and the sink node t corresponds to the execution of a computation task.

improvement path in which all improvement steps are best replies a *best improvement path*. Observe that in a NE all MUs play their best replies to each others' strategies.

It is important to note that we do not consider mixed strategy NE. Since the MCOG is a finite game, Nash equilibria in mixed strategies are guaranteed to exist, but they are impractical for two reasons. First, a mixed strategy would make it hard for a cloud scheduler to allocate computational resources. Second, mixed strategy equilibria are computationally hard to find in general. Hence, our focus is on finding pure strategy NE.

2.6 The MCOG as a player-specific congestion game

In the case of offloading through an AP a the cost $C_{i,a}^c(\mathbf{d})$ depends on the number of MUs that offload through the same AP a and on the total number of MUs that offload, while in the case of offloading through BS the cost $C_{i,B}^c(\mathbf{d})$ only depends on the total number of MUs that offload. Hence, the MCOG can be modeled as a *player-specific congestion* game as illustrated in Figure 2. The source node s represents N MUs, the sink node t corresponds to the execution of the computation task, and the intermediate node p corresponds to offloading. A solid edge i corresponds to local computing by MU i and has a cost of C_i^0, a dashed edge a corresponds to using AP a, a dotted edge B corresponds to using the BS, and the dash-dotted edge that connects node p to the sink node t corresponds to cloud computing. For a strategy profile $\mathbf{d}$ the cost $f_{i,a}(n_a(\mathbf{d}))$ of the dashed edge that corresponds to AP a is a function of the number of MUs that offload via AP a, the cost $f_{i,B}$ of the dotted edge that corresponds to the BS is independent of the other MUs' strategies, and the cost $f_i(n(\mathbf{d}))$ of the dash-dotted edge is a function of the total number of MUs that offload.

Unfortunately, general pure equilibrium existence results are not known for

d = *ImprovementPath*(**d**)

1: **while** $\exists i \in \mathcal{N}$ s.t. $\exists d_i', C_i(d_i', d_{-i}) < C_i(d_i, d_{-i})$ **do**
2: $\quad$ **d** $= (d_i', d_{-i})$
3: **end while**
4: **return d**

Figure 3: Pseudo code of the *ImprovementPath* algorithm.

player-specific congestion games. Therefore, a natural question is whether the MCOG possesses a pure NE, and if it possesses, whether there is a low complexity decentralized algorithm for computing it. In what follows we answer these questions.

3 Equilibria and the JPBR Algorithm

We start the analysis with the definition of the set of congested communication links and of the notion of the reluctance to offload.

Definition 2. For a strategy profile **d** we define the set $D_{O\to O}(\mathbf{d})$ of congested communication links as the set of communication links with at least one MU for which changing to another communication link is a best reply,

$$D_{O\to O}(\mathbf{d}) = \{o \in \mathcal{A}\cup\{B\} | \exists i \in O_o(\mathbf{d}), \exists b \in \mathcal{A}\cup\{B\}\setminus\{o\}, C^c_{i,o}(o, d_{-i}) > C^c_{i,b}(b, d_{-i})\}.$$

Similarly, for a strategy profile **d** we define the set $D_{O\to L}(\mathbf{d})$ of communication links with at least one MU for which local computing is a best reply,

$$D_{O\to L}(\mathbf{d}) = \{o \in \mathcal{A}\cup\{B\} | \exists i \in O_o(\mathbf{d}), C^c_{i,o}(o, d_{-i}) > C^0_i\}$$

Definition 3. The reluctance to offload via $o \in \mathcal{A}\cup\{B\}$ of MU i in a strategy profile **d** is $\rho_i(\mathbf{d}) = C^c_{i,o}(\mathbf{d})/C^0_i$.

To facilitate the analysis, for a strategy profile **d** we rank the MUs that play the same strategy in decreasing order of their reluctance to offload, and we use the tuple (o,l) to index the MU that in the strategy profile **d** occupies position l in the ranking for $o \in \mathcal{A}\cup\{B\}$, i.e., $\rho_{(o,1)}(\mathbf{d}) \geq \rho_{(o,2)}(\mathbf{d}) \geq \ldots \geq \rho_{(o,n_o(\mathbf{d}))}(\mathbf{d})$. Note that for $o \in \mathcal{A}\cup\{B\}$ it is MU $(o,1)$ that can gain most by changing her strategy to local computing among all MUs $i \in O_o(\mathbf{d})$.

3.1 The ImprovementPath Algorithm

Using these definitions, let us start with investigating whether the simple *ImprovementPath* algorithm shown in Figure 3 can be used for computing a NE. To do so, we analyze the finiteness of improvement paths, and as a first step, we show that improvement paths may be infinite in the MCOG, even in the case when the transmit power $P_{i,o}$ of every MU i is the same for all APs.

d_i	d_a	d_b	d_c	d_d	d_e
$\mathbf{d}(0)$	1	2	1	0	0
$\mathbf{d}(1)$	1	2	2	0	0
$\mathbf{d}(2)$	1	0	2	0	0
$\mathbf{d}(3)$	1	0	2	2	0
$\mathbf{d}(4)$	1	0	2	2	2
$\mathbf{d}(5)$	1	0	1	2	2
$\mathbf{d}(6)$	1	3	1	2	2
$\mathbf{d}(7)$	1	3	1	2	0
$\mathbf{d}(8)$	1	3	1	0	0
$\mathbf{d}(9)$	1	2	1	0	0

$$R_{c,2} > R_{c,1} \quad (1)$$
$$\frac{2}{R_{b,2}}(\gamma_b^T + \gamma_b^E P_b) D_b + 3\gamma_b^T \overline{L}_b CC_c > C_b^0 \quad (2)$$
$$C_d^0 > \frac{2}{R_{d,2}}(\gamma_d^T + \gamma_d^E P_d) D_d + 3\gamma_d^T \overline{L}_d CC_c \quad (3)$$
$$C_e^0 > \frac{3}{R_{e,2}}(\gamma_e^T + \gamma_e^E P_e) D_e + 4\gamma_e^T \overline{L}_e CC_c \quad (4)$$
$$R_{c,1} > \frac{2}{3} R_{c,2} \quad (5)$$
$$C_b^0 > \frac{1}{R_{b,3}}(\gamma_b^T + \gamma_b^E P_b) D_b + 5\gamma_b^T \overline{L}_b CC_c \quad (6)$$
$$\frac{2}{R_{e,2}}(\gamma_e^T + \gamma_e^E P_e) D_e + 5\gamma_e^T \overline{L}_e CC_c > C_e^0 \quad (7)$$
$$\frac{1}{R_{d,2}}(\gamma_d^T + \gamma_d^E P_d) D_d + 4\gamma_d^T \overline{L}_d CC_c > C_d^0 \quad (8)$$
$$R_{b,2} > R_{b,3} \quad (9)$$

Figure 4: A cyclic improvement path in a computation offloading game with 5 MUs, 3 APs and 1 BS. Rows correspond to strategy profiles, columns to MUs. An arrow between adjacent rows indicates the MU that performs the improvement step. The cycle consists of 9 improvement steps and the inequalities on the right show the condition under which the change of strategy is an improvement step.

Example 1. *Consider a MCOG with* $\mathcal{N} = \{a,b,c,d,e\}$, $\mathcal{A} = \{1,2,3\}$ *and the BS. Furthermore, let the expected time it takes to complete MU i's task in the cloud server be linear in* $n(\boldsymbol{d})$, $\overline{L}_i$ *and* CC_c, *i.e.,* $\overline{T}_i^c(\boldsymbol{d}) = n(\boldsymbol{d})\overline{L}_i CC_c$, *and assume that the transmit power* $P_{i,o}$ *of every MU i is the same for all APs, i.e.,* $P_{i,o} = P_i$. *Figure 4 shows a cyclic improvement path starting from the strategy profile* $(1,2,1,0,0)$, *in which MUs a and c are connected to AP 1, MU b is connected to AP 2 and MUs d and e perform local computation.*

Starting from the initial strategy profile $(1,2,1,0,0)$, MU c revises its strategy to AP 2, which is an improvement step if $R_{c,2} > R_{c,1}$, as shown in inequality (1) in the figure. Observe that after 9 improvement steps the MUs reach the initial strategy profile. For each step the inequality on the right provides the condition for being an improvement. It follows from inequalities (1), (5) and (9) that $R_{c,2} > R_{c,1}$, $R_{c,1} > \frac{2}{3}R_{c,2}$ and $R_{b,2} > R_{b,3}$, respectively. Since $\frac{1}{R_{b,3}}(\gamma_b^T + \gamma_b^E P_b) D_b + 5\gamma_b^T \overline{L}_b CC_c > \frac{1}{R_{b,3}}(\gamma_b^T + \gamma_b^E P_b) D_b + 3\gamma_b^T \overline{L}_b CC_c$ holds, from inequalities (2) and (6) it follows that $R_{b,3} > \frac{1}{2}R_{b,2}$. Combining inequalities (3) and (8) we have that $\gamma_d^T \overline{L}_d CC_c > \frac{1}{R_{d,2}}(\gamma_d^T + \gamma_d^E P_d) D_d$. Similarly, it follows from inequalities (4) and (7) that $\gamma_e^T \overline{L}_e CC_c > \frac{1}{R_{e,2}}(\gamma_e^T + \gamma_e^E P_e) D_e$. Given these constraints, an instance of the example can be formulated easily, even in the case of homogeneous PHY rates, i.e., $R_{i,a} = R_{i',a}$ for every $i, i' \in \mathcal{N}$, $i \neq i'$.

Example 1 illustrates that the improvement paths may be cyclic, and thus the MCOG does not have the finite improvement property, which implies that the

d = *ImproveOffloading*(**d**)

1: **while** $D_{O\to O}(\mathbf{d}) \neq \emptyset$ **do**
2: $(i', a') \leftarrow \underset{\{i\in O(\mathbf{d}), \exists o\in\mathcal{A}\cup\{B\}, C_i(o,d_{-i})<C_i(\mathbf{d})\}}{\arg\max} \frac{C_i(\mathbf{d})}{C_i(o,d_{-i})}$
3: $\mathbf{d} = (a', d_{-i'})$
4: **end while**
5: **return d**

Figure 5: Pseudo code of the *ImproveOffloading* algorithm.

MCOG does not allow a generalized ordinal potential function [23]. Consequently, the *ImprovementPath* algorithm in which the MUs perform improvement steps iteratively cannot be used for computing a NE.

3.2 The ImproveOffloading Algorithm

Although improvement paths may cycle, as we next show, improvement paths are finite if we only allow the MUs to change between APs or to change between APs and the BS but not to start or to stop offloading. We refer to this algorithm as the *ImproveOffloading* algorithm, and show its pseudo code in Figure 5. Our first result shows that all improvement paths generated by the *ImproveOffloading* algorithm are finite.

Lemma 1. *The ImproveOffloading algorithm terminates after a finite number of improvement steps.*

Proof. Let us define the function

$$\Phi(\mathbf{d}) = \sum_{a'=1}^{A} \sum_{n=1}^{n_{a'}(\mathbf{d})} log(n) - \sum_{o\in\mathcal{A}\cup\{B\}} \sum_{i'=1}^{N} log(S_{i',o}) I(d_{i'}, o),$$

where $S_{i,o} \triangleq \frac{R_{i,o}}{D_i(\gamma_i^T + \gamma_i^E P_{i,o})}$.

We prove the lemma by showing that the function $\Phi(\mathbf{d})$ decreases strictly at every improvement step generated by the *ImproveOffloading* algorithm.

First, let us consider an improvement step made by MU i in which she changes from offloading via AP b to offloading via AP a. Observe that after this improvement step the number $n(\mathbf{d})$ of MUs that offload remains unchanged. Hence, according to (8) and (10), the condition $C_i(a, d_{-i}) < C_i(b, d_{-i})$ implies $n_a(a, d_{-i})/n_b(b, d_{-i}) < S_{i,a}/S_{i,b}$. Since $n_a(a, d_{-i})$, $n_b(b, d_{-i}) > 0$ and $S_{i,a}$,$S_{i,b} > 0$ this is equivalent to

$$\log(n_a(a, d_{-i})) - \log(n_b(b, d_{-i})) < \log(S_{i,a}) - \log(S_{i,b}). \tag{12}$$

Let us rewrite Φ by separating the terms for APs a and b,

$$\Phi(a,d_{-i}) = \sum_{n=1}^{n_a(a,d_{-i})} \log(n) + \sum_{n=1}^{n_b(a,d_{-i})} \log(n) + \sum_{a' \neq a,b} \sum_{n=1}^{n_{a'}(a,d_{-i})} \log(n) - \log(S_{i,a}) - \sum_{o \in \mathcal{A} \cup \{B\}} \sum_{i' \neq i} \log(S_{i',o}) I(d_{i'},o).$$

Since $n_a(a,d_{-i}) = n_a(b,d_{-i}) + 1$ and $n_b(b,d_{-i}) = n_b(a,d_{-i}) + 1$, we have that

$$\Phi(a,d_{-i}) - \Phi(b,d_{-i}) = \log(n_a(a,d_{-i})) - \log(n_b(b,d_{-i})) - (\log(S_{i,a}) - \log(S_{i,b})).$$

It follows from (13) that $\Phi(a,d_{-i}) - \Phi(b,d_{-i}) < 0$.

Second, let us consider an improvement step made by MU i in which she changes from offloading via AP a to offloading via BS B. Observe that after this improvement step the number $n(\mathbf{d})$ of MUs that offload remains unchanged. Hence, according to (8) and (10), the condition $C_i(B,d_{-i}) < C_i(a,d_{-i})$ implies $n_a(a,d_{-i}) > S_{i,a}/S_{i,B}$. Since $n_a(a,d_{-i}), S_{i,a}, S_{i,B} > 0$ this is equivalent to

$$\log(n_a(a,d_{-i})) > \log(S_{i,a}) - \log(S_{i,B}). \tag{13}$$

Since $n_a(a,d_{-i}) = n_a(B,d_{-i}) + 1$, we have that

$$\Phi(B,d_{-i}) - \Phi(a,d_{-i}) = -\log(n_a(a,d_{-i})) + \log(S_{i,a}) - \log(S_{i,B}).$$

It follows from (13) that $\Phi(B,d_{-i}) - \Phi(a,d_{-i}) < 0$. Similarly, we can show that $C_i(a,d_{-i}) < C_i(B,d_{-i})$ implies $\Phi(a,d_{-i}) < \Phi(B,d_{-i})$.

Since the number of strategy profiles is finite, $\Phi(\mathbf{d})$ can not decrease infinitely and the *ImproveOffloading* algorithm terminates after a finite number of improvement steps. □

Thus, if MUs can only change between APs and between APs and the BS, they terminate after a finite number of improvement steps.

3.3 The JPBR Algorithm

In what follows we use the *ImproveOffloading* algorithm as a building block for proving that a NE always exists in the MCOG even if it does not allow a generalized ordinal potential function.

Theorem 1. *The MCOG possesses a pure strategy Nash equilibrium.*

Proof. We use induction in the number N of MUs in order to prove the theorem. We denote by $N^{(t)} = t$ the number of MUs that are involved in the game in induction step t.

Update phase of JPBR algorithm

/* Corresponds to case (i) */
Let $\mathbf{d}'(t) = \text{ImproveOffloading}(\mathbf{d}(t))$
/* Corresponds to case (ii) */
if $a' \in D_{O\to L}(\mathbf{d}'(t)), n_{a'}(\mathbf{d}'(t)) = n_{a'}(\mathbf{d}^*(t-1))+1$ **then**
  Let $i' \leftarrow (a', 1)$
  Let $\mathbf{d}'(t) = (0, d'_{-i'}(t))$/* Best reply by MU i' */
else
  while $D_{O\to L}(\mathbf{d}'(t)) \neq \emptyset$ **do**
    $b \leftarrow \arg\max_{a\in D_{O\to L}} \rho_{(a,1)}(\mathbf{d}'(t))$
                /*Link with MU with highest reluctance to offload */
    Let $i' \leftarrow (b, 1)$
    Let $\mathbf{d}'(t) = (0, d'_{-i'}(t))$/*Best reply by MU $(b,1)$*/
    if $\exists i \in \mathcal{N} \setminus O(\mathbf{d}'(t))$ s.t. $C_i^0 > C_i(b, d'_{-i}(t))$ **then**
      $i' \leftarrow \underset{\{i\in\mathcal{N}\setminus O(\mathbf{d}'(t)) | C_i^0 > C_i(b,d'_{-i}(t))\}}{\arg\min} \rho_i(b, d'_{-i}(t))$
                /*MU with lowest reluctance to offload*/
      Let $\mathbf{d}'(t) = (b, d'_{-i'}(t))$/*Best reply by MU i'*/
    else
      Let $\mathbf{d}'(t) = \text{ImproveOffloading}(\mathbf{d}'(t))$
    end if
  end while
end if

Figure 6: Pseudo code of the update phase of the JPBR algorithm.

For $N^{(1)}=1$ the only participating MU plays her best reply $d_i^*(1)$. Since there are no other MUs, $\mathbf{d}^*(1)$ is a NE. Observe that if $d_i^*(1) = 0$, MU i would never have an incentive to deviate from this decision, because the number of MUs that offload will not decrease as more MUs are added. Similarly, if $d_i^*(1)=B$, MU i would never have an incentive to offload using one of the APs, because the number of MUs that offload using any of the APs will not decrease as more MUs are added. Otherwise, if MU i decides to offload using one of the APs, she would play her best reply which is given by $d_i^*(1) = \arg\max_{a\in\mathcal{A}} S_{i,a}$.

Assume now that for $t-1>0$ there is a NE $\mathbf{d}^*(t-1)$. Upon induction step t one MU enters the game; we refer to this MU as MU $N^{(t)}$. Let MU $N^{(t)}$ play her best reply $d^*_{N^{(t)}}(t)$ with respect to the NE strategy profile of the MUs that already participated in induction step $t-1$, i.e., with respect to $d_{-N^{(t)}}(t) = \mathbf{d}^*(t-1)$. After that, MUs can perform best improvement steps one at a time starting from the strategy profile $\mathbf{d}(t) = (d^*_{N^{(t)}}(t), d_{-N^{(t)}}(t))$, following the algorithm shown in Figure 6. We refer to this as the update phase. In order to prove that there is a NE in induction step t, in the following we show that the MUs will perform a finite number of best improvement steps in the update phase.

Observe that if $d^*_{N^{(t)}}(t) = 0$, then $n_a(\mathbf{d}(t)) = n_a(\mathbf{d}^*(t-1))$ for every $a \in \mathcal{A}$ and thus $\mathbf{d}(t)$ is a NE. If $d^*_{N^{(t)}}(t) = o \in \mathcal{A} \cup \{B\}$, but none of the MUs want to deviate from their strategy in $\mathbf{d}^*(t-1)$ then $\mathbf{d}(t)$ is a NE. Otherwise, we can have one or both of the following cases: (i) for some MUs $i \in O_o(\mathbf{d}(t))$ offloading using $b \in \mathcal{A} \cup \{B\} \setminus \{o\}$ becomes a best reply, (ii) for some MUs $i \in O(\mathbf{d}(t))$ local computing becomes a best reply. Note that case (i) can happen only if $o \neq B$, as otherwise MU i would be able to gain by changing her strategy to offloading using on of the APs in $\mathbf{d}^*(t-1)$.

Let us first consider case (i) and let MUs execute the *ImproveOffloading* algorithm. Recall that by Lemma 1 the MUs will reach a strategy profile in which there is no MU that can further decrease her cost by changing her strategy between APs or between APs and the BS. In the resulting strategy profile the number of MUs that offload will be $n(\mathbf{d}^*(t-1))+1$. Furthermore, there will be one communication link (denoted by a') for which the number of offloaders is $n_{a'}(\mathbf{d}^*(t-1))+1$, while for the other communication links $a \neq a'$ it is $n_a(\mathbf{d}^*(t-1))$. As a consequence, there can be no MU that wants to start offloading in the resulting strategy profile if she did not want to do so in $\mathbf{d}^*(t-1)$.

If in this strategy profile no MU wants to stop offloading either, i.e., $|D_{O\to L}(\mathbf{d}(t))| = 0$, then we reached a NE. Otherwise $|D_{O\to L}(\mathbf{d}(t))| > 0$, which is the same as case (ii) above. Note that if case (i) did not happen, i.e. $|D_{O\to O}\mathbf{d}(t)| = 0$, then communication link a' is the same communication link o that was chosen by MU $N^{(t)}$ when she was added. Now if $a' \in D_{O\to L}(\mathbf{d}(t))$, let MU $(a',1)$ perform an improvement step and let $\mathbf{d}'(t)$ be the resulting strategy profile. Since MU $(a',1)$ changed her strategy from offloading through a' to local computation, $n_o(\mathbf{d}'(t)) = n_o(\mathbf{d}^*(t-1))$ holds for every $o \in \mathcal{A} \cup \{B\}$ and $\mathbf{d}'(t)$ is a NE.

Otherwise, if $a' \notin D_{O\to L}$ and $|D_{O\to L}| > 0$, we have that there is an MU i that wants to change her strategy from offloading through $b \in \mathcal{A} \cup \{B\} \setminus \{a'\}$ to local computing. Note that the only reason why MU i would want to change to local computing is that the number of MUs that offload was incremented by one, i.e., $n(\mathbf{d}(t)) = n(\mathbf{d}^*(t-1))+1$. Among all MUs that would like to change to local computing, let us allow the MU i with highest reluctance to perform the improvement step (note that this is MU $(b,1)$, $b \neq a'$). We denote the resulting strategy profile by $\mathbf{d}'(t)$. Due to this improvement step we have that $n(\mathbf{d}'(t)) = n(\mathbf{d}^*(t-1))$. Observe that if $b = B$ there can be no MU that wants to start offloading because $n_a(\mathbf{d}'(t)) = n_a(\mathbf{d}^*(t-1))$ for every AP $a \in \mathcal{A}$, and there is no more MU that would like to stop offloading either because $n(\mathbf{d}'(t)) = n(\mathbf{d}^*(t-1))$, and $\mathbf{d}'(t)$ is a NE. Otherwise, if $b \neq B$ we have that $n_b(\mathbf{d}'(t)) = n_b(\mathbf{d}^*(t-1))-1$ and some MUs that perform local computation may be able to decrease their cost by connecting to AP b. If there is no MU $i \in \mathcal{N} \setminus O(\mathbf{d}'(t))$ that would like to start offloading, there is no more MU that would like to stop offloading either because $n(\mathbf{d}'(t)) = n(\mathbf{d}^*(t-1))$ and we reached a NE. Otherwise, among all MUs $i \in \mathcal{N} \setminus O(\mathbf{d}'(t))$ that would like to start offloading, let MU i' with lowest reluctance to offload, i.e., $\rho_{i'}(b, d'_{-i'}(t))$, connect to AP b. We now repeat these steps starting from Line 8 until no more MUs want to stop offloading. Note

that when one MU is replaced by another MU, the number of MUs that offload through any of the APs does not change. Therefore, offloading cost of the MU that starts to offload will not increase in the following update steps and she will not want to stop to offload. Since the MU that starts to offload will not have an incentive to stop to offload and the number of MUs is finite, the sequence of stopping to offload and starting to offload is finite too.

Let b be the AP that the last MU that stopped offloading was connected to. If the last MU that stopped offloading was replaced by an MU that did not offload before, then we reached a NE. Otherwise some MUs that offload via $o \in \mathcal{A} \cup \{B\} \setminus \{b\}$ may want to connect to AP b, and we let them execute the *ImproveOffloading* algorithm, which by Lemma 1 terminates in a finite number of improvement steps. Now, no MU wants to stop offloading, and if there is no MU that wants to start offloading either then we reached a NE. Otherwise, if there is a MU that wants to start to offload, we repeat the steps starting from Line 8. Let us recall that the MU that starts to offload would not want to stop to offload and as a consequence the size of the set $D_{O \to L}$ will decrease every time when a MU stops to offload. Therefore, after a finite number of steps, the MUs will reach either an equilibrium in which the number of offloaders is the same as in the strategy profile $\mathbf{d}^*(t-1)$ or an equilibrium in which the number of offloaders is incremented by 1, which proves the inductive step. □

Consider now that we add one MU at a time and for every new MU we compute a NE following the proof of Theorem 1. We refer to the resulting algorithm as the *Join and Play Best Replies (JPBR)* algorithm. In what follows we provide a bound on the complexity of this algorithm.

Proposition 2. *When MU $N^{(t)}$ enters the game in an equilibrium $\mathbf{d}^*(t-1)$, a new Nash equilibrium can be computed in $O((A+2)N^{(t)} - 2A)$ time.*

Proof. In the worst case scenario $|O(\mathbf{d}^*(t-1))| = N^{(t)} - 2$, $d^*_{N^{(t)}}(t) = a \in \mathcal{A}$ and case (i) happens such that in the next $N^{(t)}-2$ update steps all MUs $i \in O(\mathbf{d}^*(t-1))$, i.e., $N^{(t)}-2$ MUs change between APs before they reach the strategy profile in which there is no MU that can decrease her offloading cost. Furthermore, in the worst case scenario, this is followed by a sequence of update steps in which $N^{(t)}-2$ MUs stop to offload and $N^{(t)}-3$ MUs start to offload and between every stop to offload and start to offload update step, MUs change between the APs. When a MU stops to offload, the sequence in which MUs change between APs consists of at most $A-1$ update steps. Hence, a NE is reached after at most $(N^{(t)}-2)+(N^{(t)}-2)+(N^{(t)}-3)+(N^{(t)}-2)(A-1)$ updates. □

Since we add one MU at a time, we can formulate the following result.

Corollary 1. *The* JPBR *algorithm terminates in an equilibrium allocation in $O((A+2)N^2/2-(A-1)N)$ time.*

So far we have shown that starting from a NE and adding a new MU, a new NE can be computed. We now show a similar result for the case when a MU leaves.

Theorem 3. *Consider the MCOG and assume that the system is in a NE. If a MU leaves the game and the remaining MUs play their best replies one at a time, they converge to a NE after a finite number of updates.*

Proof. Let us consider that MU i leaves the game when the system is in a NE. If the strategy of MU i was to perform local computation, none of the remaining MUs would have an incentive to change their strategies. If the strategy of MU i was to offload using one of the APs or using the BS, we can consider MU i as an MU that after changing its strategy from offloading to local computing would have no incentive to offload again. Recall from the proof of Theorem 1 that when an MU changes her strategy from offloading to local computing the game converges to a NE after a finite number of updates. This proves the theorem. □

Observe that Theorem 1 and Theorem 3 allow for the efficient computation of Nash equilibria even if the number of MUs changes, if the time between MU arrivals and departures is sufficient to compute a new equilibrium. Furthermore, the computation can be done in a decentralized manner, by letting MUs perform best improvements one at a time. The advantage of such a decentralized implementation compared to a centralized solution could be that MUs do not have to reveal their parameters.

4 The Case of an Elastic Cloud

The JPBR algorithm can be used for computing an equilibrium for the MCOG with polynomial complexity. In what follows we show that a much simpler algorithm can be used for computing an equilibrium if we assume that the expected time needed for the cloud to complete MU i's task is independent of the other MUs' strategies, and thus of the number of MUs that offload. This is a reasonable assumption for large cloud computing infrastructures, in which the cloud computing resources scale with the number of MUs. We refer to the resulting model as the *elastic* cloud model, and we express the expected time needed for the cloud to complete MU i's task as $\overline{T}_i^c = \overline{T}_i^{c,exe} + \overline{T}_i^{c,ot}$. Consequently, the cost function of MU i in the case of offloading is simpler because the part of MU i's cost concerning the time needed for performing its task in the cloud does not depend on the strategy profile $\mathbf{d}$. Therefore, in the case of the *elastic* cloud model the cost function of MU i when it offloads its task through AP a can be expressed as

$$C_{i,a}^c(\mathbf{d}) = (\gamma_i^T + \gamma_i^E P_{i,a}) D_i \frac{n_a(\mathbf{d})}{R_{i,a}} + \gamma_i^T \overline{T}_i^c + F, \tag{14}$$

which only depends on the number of MUs that offload through the same AP a. Furthermore, in the case of offloading through BS B the cost function is independent

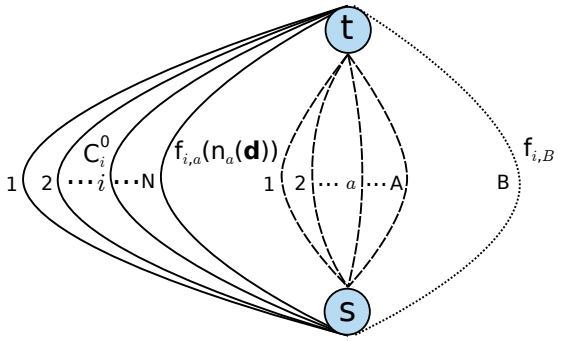

Figure 7: The computation offloading game in the case of an elastic cloud modeled as a player-specific singleton congestion game. The source node s represents N MUs, and the sink node t corresponds to the execution of a computation task.

of the other MUs' strategies and can be expressed as

$$C_{i,B}^c = (\gamma_i^T + \gamma_i^E P_{i,B}) \frac{D_i}{R_{i,B}} + \gamma_i^T \overline{T}_i^c + F, \tag{15}$$

Let us recall that when the expected time needed for the cloud to complete MU i's task depends on the other MUs' strategies, the MUs have to share both communication and computing resources (c.f., Figure 2). On the contrary, in the case of the *elastic* cloud model, the MUs only have to share communication resources when they offload their tasks through one of the APs, and thus the MCOG can be modeled as a *player-specific singleton congestion* game as illustrated in Figure 7. Compared to the graph shown in Figure 2, the graph in Figure 7 has only the source node s and the sink node t. A solid edge i corresponds to local computing by MU i and has a cost of C_i^0, a dashed edge a corresponds to using AP a, and the dotted edge B corresponds to using the BS. Every MU can either choose the solid edge or the dotted edge that corresponds to itself or one of the dashed edges, thus one of $A+2$ edges. For a strategy profile $\mathbf{d}$ the cost $f_{i,a}(n_a(\mathbf{d}))$ of the dashed edge that corresponds to AP a is a function of the number of MUs that offload through AP a, while the cost $f_{i,B}$ of the dotted edge that corresponds to the BS is independent of the other MUs' strategies.

For *player-specific singleton congestion* games it is known that a pure strategy Nash equilibrium always exsists, even if potential function may not exist [24]. Before we formulate the theorem, let us recall the definition of a generalized ordinal potential from [23].

Definition 4. A function $\Phi : \times \mathfrak{D}_i \to \mathbb{R}$ is a generalized ordinal potential function for the strategic game $\Gamma^s =< \mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i >$ if for an arbitrary strategy profile

(d_i, d_{-i}) and for any corresponding improvement step d_i' it holds that

$$C_i(d_i', d_{-i}) - C_i(d_i, d_{-i}) < 0 \Rightarrow \Phi(d_i', d_{-i}) - \Phi(d_i, d_{-i}) < 0.$$

Theorem 4. *The MCOG with elastic cloud admits the generalized ordinal potential function*

$$\Phi(\boldsymbol{d}) = \sum_{a'=1}^{A} \sum_{n=1}^{n_{a'}(\boldsymbol{d})} log(n) - \sum_{o \in \mathcal{A} \cup \{B\}} \sum_{i'=1}^{N} log(M_{i'} S_{i',o}) I(d_{i'}, o), \tag{16}$$

and hence it possesses a pure strategy Nash equilibrium, if $F < \gamma_i^E v_i \overline{L}_i CPI_i + \gamma_i^T (\overline{T}_i^0 - \overline{T}_i^c)$.

Proof. To prove that $\Phi(\mathbf{d})$ is a generalized ordinal potential function, we first show that $C_i(a, d_{-i}) < C_i(0, d_{-i})$ implies $\Phi(a, d_{-i}) < \Phi(0, d_{-i})$.

According to (7), (10) and (14), the condition $C_i(a, d_{-i}) < C_i(0, d_{-i})$ implies that

$$(\gamma_i^T + \gamma_i^E P_{i,a}) D_i \frac{n_a(a, d_{-i})}{R_{i,a}} + \gamma_i^T \overline{T}_i^c + F < \gamma_i^T \overline{T}_i^0 + \gamma_i^E v_i \overline{L}_i CPI_i. \tag{17}$$

After algebraic manipulations we obtain

$$n_a(a, d_{-i}) < M_i S_{i,a}, \tag{18}$$

where $S_{i,a} \triangleq \frac{R_{i,a}}{D_i(\gamma_i^T + \gamma_i^E P_{i,a})}$ and $M_i \triangleq \gamma_i^T (\overline{T}_i^0 - \overline{T}_i^c) + \gamma_i^E v_i \overline{L}_i CPI_i - F$.

Since $n_a(a, d_{-i}) > 0$ and $M_i S_{i,a} > 0$, (17) implies that

$$\log(n_a(a, d_{-i})) < \log(M_i S_{i,a}). \tag{19}$$

For the strategy profile (a, d_{-i}) it holds that

$$\Phi(a, d_{-i}) = \sum_{n=1}^{n_a(a, d_{-i})} \log(n) + \sum_{a' \neq a} \sum_{n=1}^{n_{a'}(a, d_{-i})} \log(n)$$
$$- \log(M_i S_{i,a}) - \sum_{o \in \mathcal{A} \cup \{B\}} \sum_{i' \neq i} \log(M_{i'} S_{i',o}) I(d_{i'}, o),$$

and for the strategy profile $(0, d_{-i})$

$$\Phi(0, d_{-i}) = \sum_{n=1}^{n_a(0, d_{-i})} \log(n) + \sum_{a' \neq a} \sum_{n=1}^{n_{a'}(0, d_{-i})} \log(n)$$
$$- \sum_{o \in \mathcal{A} \cup \{B\}} \sum_{i' \neq i} \log(M_{i'} S_{i',o}) I(d_{i'}, o).$$

Since $n_a(a,d_{-i}) = n_a(0,d_{-i})+1$, we obtain $\Phi(a,d_{-i}) - \Phi(0,d_{-i}) = \log(n_a(a,d_{-i})) - \log(M_i S_{i,a})$. It follows from (19) that $\Phi(a,d_{-i}) - \Phi(0,d_{-i}) < 0$. Similarly, we can show that $C_i(0,d_{-i}) < C_i(a,d_{-i})$ implies $\Phi(0,d_{-i}) < \Phi(a,d_{-i})$.

Second, we show that $C_i(a,d_i) < C_i(b,d_i)$ implies $\Phi(a,d_i) < \Phi(b,d_i)$. According to (10) and (14), the condition $C_i(a,d_i) < C_i(b,d_i)$ implies that

$$(\gamma_i^T + \gamma_i^E P_{i,a}) D_i \frac{n_a(a,d_{-i})}{R_{i,a}} < (\gamma_i^T + \gamma_i^E P_{i,b}) D_i \frac{n_b(b,d_{-i})}{R_{i,b}},$$

which is equivalent to

$$\frac{n_a(a,d_{-i})}{n_b(b,d_{-i})} < \frac{S_{i,a}}{S_{i,b}}. \tag{20}$$

Since $n_a(a,d_{-i}), n_b(b,d_{-i}) > 0$ and $S_{i,a}, S_{i,b} > 0$, (20) implies

$$\log(n_a(a,d_{-i})) - \log(n_b(b,d_{-i})) < \log(S_{i,a}) - \log(S_{i,b}). \tag{21}$$

Let us rewrite Φ by separating the terms for APs a and b,

$$\begin{aligned}\Phi(a,d_{-i}) = &\sum_{n=1}^{n_a(a,d_{-i})} \log(n) + \sum_{n=1}^{n_b(a,d_{-i})} \log(n) + \sum_{a' \neq a,b} \sum_{n=1}^{n_{a'}(a,d_{-i})} \log(n) \\ &- \log(M_i S_{i,a}) - \sum_{o \in \mathcal{A} \cup \{B\}} \sum_{i' \neq i} \log(M_{i'} S_{i',o}) I(d_{i'}, o).\end{aligned}$$

Since $n_a(a,d_{-i}) = n_a(b,d_{-i})+1$ and $n_b(b,d_{-i}) = n_b(a,d_{-i})+1$, we have that $\Phi(a,d_{-i}) - \Phi(b,d_{-i}) = \log(n_a(a,d_{-i})) - \log(n_b(b,d_{-i})) - (\log(S_{i,a}) - \log(S_{i,b}))$. It follows from (21) that $\Phi(a,d_{-i}) - \Phi(b,d_{-i}) < 0$.

Third, we show that $C_i(B,d_i) < C_i(0,d_i)$ implies $\Phi(B,d_i) < \Phi(0,d_i)$. According to (7), (10) and (15), the condition $C_i(B,d_i) < C_i(0,d_i)$ implies that

$$(\gamma_i^T + \gamma_i^E P_{i,B}) \frac{D_i}{R_{i,B}} + \gamma_i^T \overline{T}_i^c + F < \gamma_i^T \overline{T}_i^0 + \gamma_i^E v_i \overline{L}_i CPI_i, \tag{22}$$

which is equivalent to

$$1 < M_i S_{i,B}. \tag{23}$$

For the strategy profile (B,d_{-i}) it holds that

$$\begin{aligned}\Phi(B,d_{-i}) = &\sum_{a'=1}^{A} \sum_{n=1}^{n_{a'}(B,d_{-i})} \log(n) - \log(M_i S_{i,B}) \\ &- \sum_{o \in \mathcal{A} \cup \{B\}} \sum_{i' \neq i} \log(M_{i'} S_{i',o}) I(d_{i'}, o).\end{aligned}$$

Since $n_a(B, d_{-i}) = n_a(0, d_{-i})$ for every AP a, we have that $\Phi(B, d_{-i}) - \Phi(0, d_{-i}) = -\log(M_i S_{i,B})$, and since $M_i S_{i,B} > 0$, it follows from (23) that $\Phi(B, d_{-i}) - \Phi(0, d_{-i}) < 0$. Similarly, we can show that $C_i(0, d_{-i}) < C_i(B, d_{-i})$ implies $\Phi(0, d_{-i}) < \Phi(B, d_{-i})$.

Finally, we show that $C_i(B, d_i) < C_i(a, d_i)$ implies $\Phi(B, d_i) < \Phi(a, d_i)$. According to (10), (14) and (15), the condition $C_i(B, d_i) < C_i(a, d_i)$ implies that

$$(\gamma_i^T + \gamma_i^E P_{i,B}) \frac{D_i}{R_{i,B}} < (\gamma_i^T + \gamma_i^E P_{i,a}) D_i \frac{n_a(a, d_{-i})}{R_{i,a}},$$

which is equivalent to

$$n_a(a, d_{-i}) > \frac{S_{i,a}}{S_{i,B}}. \tag{24}$$

Since $n_a(a, d_{-i}) > 0$ and $S_{i,a}, S_{i,B} > 0$, (24) implies

$$\log(n_a(a, d_{-i})) > \log(S_{i,a}) - \log(S_{i,B}). \tag{25}$$

Since $n_a(a, d_{-i}) = n_a(B, d_{-i}) + 1$, we have that $\Phi(B, d_{-i}) - \Phi(a, d_{-i}) = -\log(n_a(a, d_{-i})) + \log(S_{i,a}) - \log(S_{i,B})$. It follows from (25) that $\Phi(B, d_{-i}) - \Phi(a, d_{-i}) < 0$. Similarly, we can show that $C_i(a, d_{-i}) < C_i(B, d_{-i})$ implies $\Phi(a, d_{-i}) < \Phi(B, d_{-i})$, which proves the theorem. □

It is well known that in a finite strategic game that admits a generalized ordinal potential all improvement paths are finite [25]. Therefore, the existence of a generalized ordinal potential function allows us to use the *ImprovementPath* Algorithm (c.f., Figure 3) for computing a NE.

Corollary 2. *The* ImprovementPath *algorithm terminates in a NE after a finite number of improvement steps for the MCOG with elastic cloud.*

In what follows, we prove that if MUs aim at minimizing only their energy consumption, the *ImprovementPath* algorithm can be used for computing a NE not only for the MCOG with elastic cloud, but also in the general case, i.e., if the time needed for performing the task of an MU in the cloud depends on the strategy profile $\mathbf{d}$.

Proposition 5. *The* ImprovementPath *algorithm terminates in a NE after a finite number of improvement steps for the MCOG with* $\gamma_i^T = 0$*, i.e., if each MU aims at minimizing its energy consumption.*

Proof. Observe that if $\gamma_i^T = 0$, MU i spends the energy only to transmit the data through one of the APs in the case of offloading, and thus the cost function of MU i only depends on the number of MUs that offload through the same AP, and it is independent of the total number of MUs that offload. Hence, the MCOG in which $\gamma_i^T = 0$ for all MUs can be modeled by a congestion game on parallel links as shown in Figure 7. Furthermore, (16) is a generalized ordinal potential function of the game with $\gamma_i^T = 0$, which proves the proposition. □

5 Price of Anarchy

We have so far shown that a NE exists and provided low complexity algortihms for computing it. We now address the important question how far the system performance would be from optimal in a NE. To quantify the difference from the optimal performance we use the price of anarchy (PoA), defined as the ratio of the worst case NE cost and the minimal cost

$$PoA = \frac{\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d}\in\mathfrak{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})}. \tag{26}$$

In what follows we give an upper bound on the PoA.

Theorem 6. *The price of anarchy for the computation offloading game is upper bounded by*

$$\frac{\sum_{i\in\mathcal{N}} C_i^0}{\sum_{i\in\mathcal{N}} \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}, C_{i,B}^{\bar{c}}\}},$$

both in the case of elastic cloud and in the case of non-elastic cloud.

Proof. First we show that if there is a NE in which all players perform local computation then it is the worst case NE. To show this let $\mathbf{d}^*$ be an arbitrary NE. Observe that $C_i(d_i^*, d_{-i}^*) \leq C_i^0$ holds for every player $i \in \mathcal{N}$. Otherwise, if $\exists i \in \mathcal{N}$ such that $C_i(d_i^*, d_{-i}^*) > C_i^0$, player i would have an incentive to deviate from decision d_i^*, which contradicts our initial assumption that $\mathbf{d}^*$ is a NE. Thus, in any NE $\sum_{i\in\mathcal{N}} C_i(d_i^*, d_{-i}^*) \leq \sum_{i\in\mathcal{N}} C_i^0$ holds, and if all players performing local computation is a NE then it is the worst case NE.
Now we derive a lower bound for the optimal solution of the computation offloading game. Let us consider an arbitrary decision profile $(d_i, d_{-i}) \in \mathfrak{D}$. If $d_i = 0$, then $C_i(d_i, d_{-i}) = C_i^0$. Otherwise, if $d_i = o$ for some $o \in \mathcal{A} \cup \{B\}$, we have that in the best case $d_{i'} = 0$ for every $i' \in \mathcal{N} \setminus \{i\}$, and thus $n(\mathbf{d}) = 1$. Therefore, $\omega_i^o(d_i, d_{-i}) \leq R_{i,o}$ and $\overline{T}_i^c(n(d_i, d_{-i})) \leq \overline{T}_i^c$, which implies

$$\begin{aligned}&(\gamma_i^T + \gamma_i^E P_{i,o}) \frac{D_i}{\omega_i^o(d_i, d_{-i})} + \gamma_i^T \overline{T}_i^c(n(d_i, d_{-i})) \\ &\geq (\gamma_i^T + \gamma_i^E P_{i,o}) \frac{D_i}{R_{i,o}} + \gamma_i^T \overline{T}_i^c = C_{i,o}^{\bar{c}}.\end{aligned}$$

Hence, $C_i(d_i, d_{-i}) \geq \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}, C_{i,B}^{\bar{c}}\}$ and $\sum_{i\in\mathcal{N}} C_i(d_i, d_{-i}) \geq \sum_{i\in\mathcal{N}} \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}, C_{i,B}^{\bar{c}}\}$. Using these expressions we can establish the following bound

$$PoA = \frac{\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d}\in\mathfrak{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})} \leq \frac{\sum_{i\in\mathcal{N}} C_i^0}{\sum_{i\in\mathcal{N}} \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}, C_{i,B}^{\bar{c}}\}},$$

which proves the theorem. □

In the following we provide an upper bound on the PoA in the case of homogeneous MUs, that is, when all MUs have the same parameters.

Proposition 7. *In the case of homogeneous MUs and when the expected time* $\overline{T}_i^{c,ot}(\boldsymbol{d})$ *the cloud server spends executing other tasks is an affine function of the number of MUs that offload, the price of anarchy for the MCOG is at most* $\frac{5}{2}$.

Proof. When the MUs are homogeneous and $\overline{T}_i^{c,ot}(\mathbf{d})$ is an affine function of the number of MUs that offload, the MCOG falls into the category of congestion games with affine cost functions. It follows from [26] that congestion games with affine cost functions are smooth games with a PoA at most $\frac{5}{2}$, which proves the proposition. □

Observe that the PoA is in fact a bound on the approximation ratio of the decentralized algorithms used for computing a NE.

6 Numerical Results

We use extensive simulations to evaluate the cost performance and the computational time of the *JPBR* algorithm. We placed the MUs and the APs at random on a regular grid with 10^4 points defined over a square area of $1km \times 1km$. We chose the channel gain of MU i to AP a to be proportional to $d_{i,a}^{-\alpha}$, where $d_{i,a}$ is the distance between MU i and AP a, and α is the path loss exponent, which we set to 4 according to the path loss model in urban and subrurban areas [27]. The channel bandwidth B_a of every AP a was set to 5 *MHz*, while the data transmit power $P_{i,a}$ of every MU i and for every AP a was set to to $0.4\,W$ according to [28]. Given the noise power P_n we calculate the PHY rate $R_{i,a}$ as $R_{i,a} = B_a \log(1 + P_{i,a} d_{i,a}^{-\alpha}/P_n)$.

The clock rate CR_i of MU i was drawn from a continuous uniform distribution with parameters $[0.5, 1]$ *GHz* based on the specification of NVIDIA Tegra 2, which is the reference platform for Android OS [29], while the clock rate CR_c of the cloud was set to 100 *GHz* [30]. Unless otherwise noted, the input data size D_i was uniformly distributed on $[0.42, 2]$ *Mb*. We drew the total number of CPU cycles required to perform the computation $(L_i \cdot CPI)$ from a continuous uniform distribution with parameters $[0.1, 0.8]$ *Gcycles*. The consumed energy per CPU cycle v_i was set to $10^{-11}(CR_i)^2$ according to measurements reported in [4,31]. The weights attributed to energy consumption γ_i^E and the response time γ_i^T were drawn from a continuous uniform distribution on $[0, 1]$.

In order to evaluate the cost performance of the equilibrium strategy profile $\mathbf{d}^*$ computed by the *JPBR* algorithm, we computed the optimal strategy profile $\bar{\mathbf{d}}$ that minimizes the total cost, i.e., $\bar{\mathbf{d}} = \arg\min_{\mathbf{d}} \sum_{i \in \mathcal{N}} C_i(\mathbf{d})$. Furthermore, as a baseline for comparison we computed the system cost that can be achieved if all MUs execute their computation tasks locally, which coincides with the bound on the PoA. Unless otherwise noted, the MUs are added at random in the induction steps of the JPBR

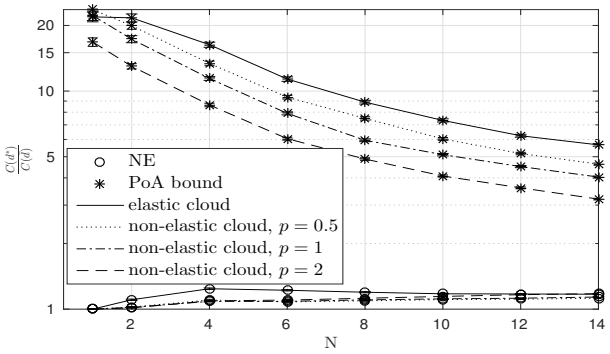

Figure 8: The *cost ratio* and the upper bound on the PoA for the *elastic* and *non-elastic* cloud ($p = 0.5, 1, 2$), $A = 3$ APs.

algorithm. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

6.1 Optimal vs. Equilibrium Cost

Figure 8 shows the *cost ratio* $C(\mathbf{d}^*)/C(\bar{\mathbf{d}})$ vs. the number of MUs. The results are shown for the case of the *elastic* cloud as well as for the case when the expected time it takes to complete MU i's task is linear in the number of MUs that offload, i.e., $\overline{T}_i^{c,exe}(\mathbf{d}) = pn(\mathbf{d})\overline{L}_i CPI_c CC_c$. We refer to this latter case as a *non-elastic* cloud and to the coefficient p as the *cloud provisioning coefficient*. A coefficient of $p = 1$ corresponds to a cloud with fixed amount of resources, $p < 1$ to resources that scale slower than the demand, while $p > 1$ corresponds to a cloud with backup resources that scale with the demand.

To make the computation of the optimal strategy profile $\bar{\mathbf{d}}$ feasible, unless otherwise noted, we considered a scenario with $A = 3$ APs and we show the *cost ratio* $C(\mathbf{d}^*)/C(\bar{\mathbf{d}})$ as a function of the number of MUs. We consider the *non-elastic* cloud model that does not implement redundancy mechanisms for three values of the *cloud provisioning coefficient* ($p = 0.5, 1$ and 2).

The results in Fig. 8 show that the performance of *JPBR* is close to optimal (*cost ratio* is close to 1) in all cases, and the *cost ratio* is fairly insenstive to the number of MUs, which is due to the number of MUs that choose to offload, as we will see later. The results for the bound on the PoA additionally confirm that the *JPBR* algorithm performs good in terms of the *cost ratio*. It is interesting to note that the gap between the PoA bound and the actual *cost ratio* decreases with increasing number of MUs. This is due to the benefit of offloading decreases as the number of MUs increases, and as a result the optimal solution and the *JPBR* algorithm will converge to a strategy profile in which most of the MUs perform local computation.

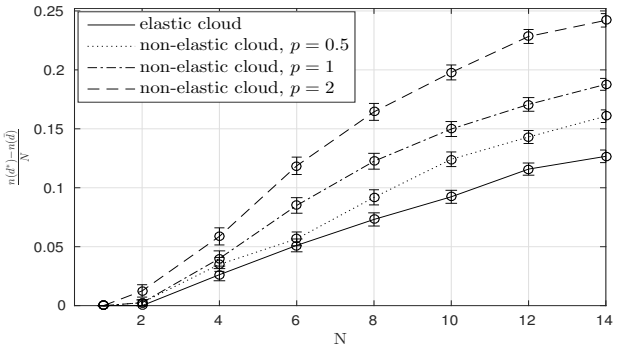

Figure 9: *Offloading difference ratio* vs. number of MUs N for the *elastic* and *non-elastic* cloud ($p = 0.5, 1, 2$), $A = 3$ APs.

We can also observe that the upper bound on the PoA decreases as p increases, and thus the problem becomes computationally easier for larger values of p.

In order to gain insight in the structure of the equilibrium strategy profiles $\mathbf{d}^*$, it is interesting to compare the number of MUs that offload in equilibrium $\mathbf{d}^*$ and the number of MUs that offload in the optimal solution $\bar{\mathbf{d}}$. We define the *offloading difference ratio* $(n(\mathbf{d}^*) - n(\bar{\mathbf{d}}))/N$, and show it in Figure 9 for the same set of parameters as in Figure 8. The results show that the *offloading difference ratio* increases with the number of MUs, which explains the increased *cost ratio* observed in Figure 8, as more offloaders reduce the achievable rate, which in turn leads to increased costs. The observation that the number of MUs that offload is higher in equilibrium than in the optimal solution is consistent with the theory of the tragedy of the commons in the economic literature [32]. The results also show that the *offloading difference ratio* is slightly lower in the case of the *elastic* cloud, which is due that a higher proportion of MUs offload in the optimal solution for the *elastic* cloud.

6.2 Impact of the order of adding MUs

Since the order in which the MUs are added in induction steps of the JPBR algorithm can be arbitrarily chosen, in the following we investigate in which order the controller should add the MUs, so that their costs are minimized. To do so, we consider five orderings of adding MUs: *random* where MUs are added at random, *least reluctance first* (LRF) where MUs are added in increasing order of their ratio $\frac{D_i}{C_i^0 L_i CPI_i}$, *most reluctance first* (MRF) where MUs are added in decreasing order of their ratio $\frac{D_i}{C_i^0 L_i CPI_i}$, *least clock rate first* ($LCRF$) where MUs are added in

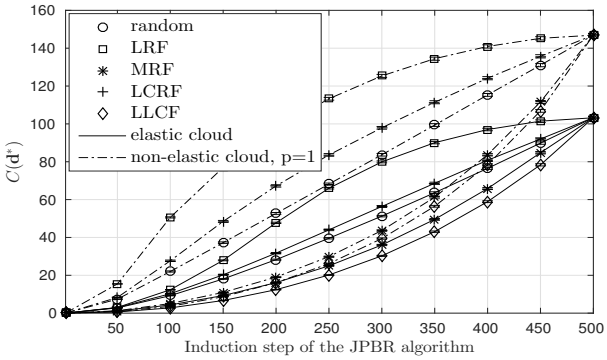

Figure 10: The system cost vs. the induction step for the *elastic* and *non-elastic* cloud ($p = 1$), $A = 10$ APs, $N = 500$ MUs.

increasing order of their clock rate CR_i, and *least local cost first* ($LLCF$) where MUs are added in increasing order of their local cost C_i^0. Figure 10 shows the system cost as a function of the number of induction steps performed (i.e., MUs added) by the JPBR algorithm, for the *elastic* and the *non-elastic* cloud (p=1), $N = 500$ MUs, and $A = 10$ APs. The results show that the order of adding MUs affects the system cost during the induction steps, but the system cost $C(\mathbf{d}^*)$ in the equilibrium $\mathbf{d}^*$ computed upon the last induction step is almost the same for all considered orderings, and thus the order of adding MUs does not affect the system performance significantly.

To gain insight in the impact of the order of adding MUs on the cost of the MUs, we define the *individual cost ratio* for a particular order of entry, compared to the LLCF order of entry. Figure 11 shows the CDF of the *individual cost ratio* in the equilibrium $\mathbf{d}^*$ computed upon the last induction step for the same set of parameters as in Figure 10. The results show that the individual costs slightly differ for different orderings only for a few MUs, and thus changing the order of adding MUs does not affect the performance of individual MUs either.

6.3 Impact of the input data size

In order to analyse the impact of the input data size we considered three distributions with the same mean for the input data size, uniform (lower limit fixed to 0.42 and upper limit scales with the mean), exponential, and Weibull (shape parameter 0.5), and considered that all MUs have to offload a task that requires a computation of $L_i \cdot CPI$=0.45 *Gcycles*. Figure 12 shows the *cost ratio* $C(\mathbf{d}^*)/C(\bar{\mathbf{d}})$ and the upper bound on the PoA as a function of the mean input data size. The results are shown for the *non-elastic* cloud (p=1), N=12 MUs and A=3 APs, and show that while the *cost ratio* does not change, the upper bound on the PoA decreases with

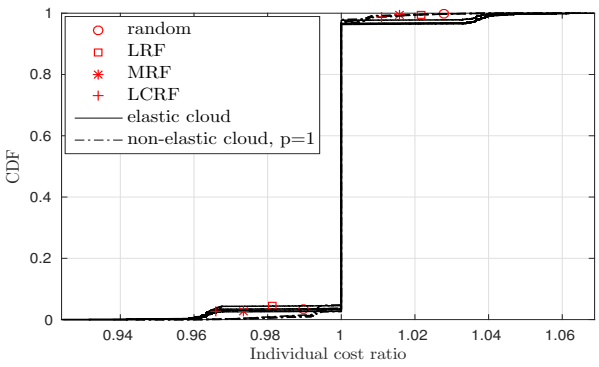

Figure 11: Distribution of the *individual cost ratio* for the *elastic* and *non-elastic* cloud ($p = 1$), $A = 10$ APs, $N = 500$ MUs.

the mean input data size and for large data sizes it reaches the *cost ratio*. This is due to the transmission time increases with the input data size and if the MUs have to offload a large amount of data, it becomes optimal for most of them to perform local computation, which coincides with the worst case equilibrium. Note that the upper bound on the PoA decreases slower in the case of the Weibull distribution because for the same mean it has a median that is smaller than that of the uniform and exponential distributions.

6.4 Computational Complexity

In order to evaluate the computational complexity of the *JPBR* algorithm, we consider the number of iterations, the total number of update steps over all induction steps plus the number of induction steps, to compute the strategy profile $\mathbf{d}^*$ for the *elastic* cloud and for the *non-elastic* cloud ($p = 1$), $A = 10$ and $A = 100$ APs. Figure 13 shows the number of iterations as a function of the number of MUs for the *random* and the *LRF* order of adding MUs. Intuitively, one would expect that the *LRF* order results in a smaller number of iterations, since the MUs with lower $\frac{D_i}{C_i^0 L_i CPI_i}$ ratio have lower computational capability to execute computationally more demanding tasks with smaller offloading data size than the MUs with higher $\frac{D_i}{C_i^0 L_i CPI_i}$. However, the simulation results show that the number of iterations is fairly insensitive to the order of adding the MUs and mostly depends on the number of MUs. This insensitivity allows for a very low-overhead decentralized solution, as the coordinator need not care about the order in which the MUs are added for computing the equilibrium allocation. The results also show that the number of iterations scales approximately linearly with the number of MUs, and indicates that the worst case scenario described in Corollary1 is unlikely to happen. Thus *JPBR* is

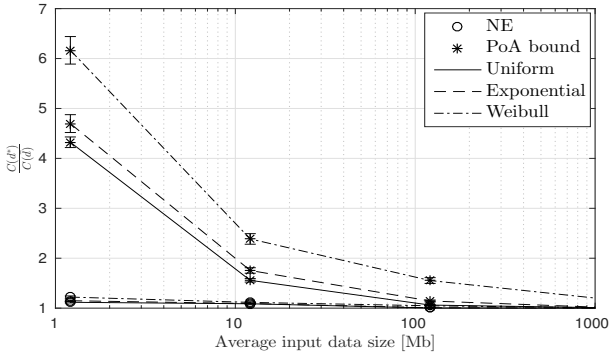

Figure 12: The *cost ratio* and the upper bound on the PoA for the *elastic* and *non-elastic* cloud ($p = 1$), uniform, exponential and Weibull distributions of the input data sizes, $A = 3$ APs, $N = 12$ MUs.

an efficient decentralized algorithm for coordinating computation offloading among autonomous MUs.

7 Related Work

There is a significant body of works that deals with the design of energy efficient computation offloading for a single mobile user [3, 4, 6, 21, 33–35]. The experimental results in [34] showed that significant battery power savings can be achieved by computation offloading. [6] studied the commmunication overhead of computation offloading and the impact of bandwidth availability on an experimental platform. [3] proposed a code partitioning solution for fine-grained energy-aware computation offloading. [21] proposed an algorithm for offloading partitioned code under bandwidth and delay constraints. [4] proposed CPU frequency and transmission power adaptation for energy-optimal computation offloading under delay constraints. [35] modeled the offloading problem under stochastic task arrivals as a Markov decision process and provided a near-optimal offloading policy.

A number of recent works considered the problem of joint energy minimization for multiple mobile users [13, 14, 36]. [13] studies computation partitioning for streaming data processing with the aim of maximizing throughput, considering sharing of computation instances among multiple mobile users, and proposes a genetic algorithm as a heuristic for solving the resulting optimization problem. [36] models computation offloading to a tiered cloud infrastructure under user mobility in a location-time workflow framework, and proposes a heuristic for minimizing the users' cost. [14] aims at minimizing the mobile users' energy consumption by joint allocation of radio resources and cloud computing power, and provides an iterative

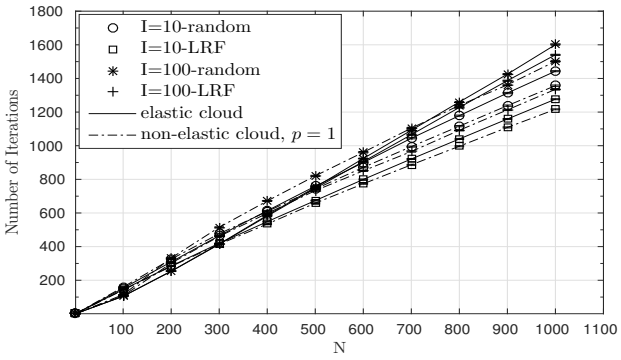

Figure 13: Number of iterations vs. number of MUs N for the *elastic* and *non-elastic* cloud ($p = 1$), A =10 and 100 APs.

algorithm to find a local minimum of the optimization problem.

A few recent works provided a game theoretic treatment of computation offloading in a game theoretical setting [7, 37–41]. [37] considers a two-stage problem, where first each mobile user decides what share of its task to offload so as to minimize its energy consumption and to meet its delay deadline, and then the cloud allocates computational resources to the offloaded tasks. [38] considers a two-tier cloud infrastructure and stochastic task arrivals and proves the existence of equilibria and provides an algorithm for computing and equilibrium. [40] considers tasks that arrive simultaneously, a single wireless link, and elastic cloud, and show the existence of equilibria when all mobile users have the same delay budget. Our work differs from [37] in that we consider that the allocation of cloud resources is known to the mobile users, from [38] in that we take into account contention in the wireless access, and from [40] in that we consider multiple wireless links and a non-elastic cloud.

Most related to our work are the problems considered in [7, 12, 39, 41]. [12] considers a system where multiple devices can offload their tasks to a non-elastic cloud through one of multiple shared heterogeneous wireless links. Different from [12], we consider that devices besides offloading through shared heterogeneous wireless links can offload their tasks through a non-shared link, and we consider that devices use different transmit powers for different wireless links when they offload their tasks to the cloud. [7] considers contention on a single wireless link and an elastic cloud, assumes upload rates to be determined by the Shannon capacity of an interference channel, and shows that the game is a potential game. [39] extends the model to multiple identical wireless links, shows that the game is still a potential game, and that the same algorithm as in [7] can be used for computing an equilibrium allocation. Unlike these works, we consider heterogeneous wireless links, fair bandwidth sharing and a non-elastic cloud. [41] considers multiple wireless links, fair

bandwidth sharing and a non-elastic cloud, and claims the game to have an exact potential. In our work we on the one hand extend the model to an elastic cloud, on the other hand we show that an exact potential cannot exist in case of a non-elastic cloud, but at the same time we prove the existence of an equilibrium allocation, provide an efficient algorithm with quadratic complexity for computing one, and provide a bound on the price of anarchy.

Besides providing efficient distributed algorithms for computing equilibria, the importance of our contribution lies in the fact that while games with an elastic cloud are player-specific singleton congestion games for which the existence of equilibria is known [24], the non-elastic cloud model does not fall in this category of games and thus no general equilibrium existence result exists.

8 Conclusion

We have provided a game theoretic analysis of selfish mobile computation offloading. We proposed a polynomial complexity algorithm for computing equilibrium allocations of the wireless and cloud resources, and provided a bound on the price of anarchy, which serves as an approximation ratio bound for the optimization problem. Our numerical results show that the proposed algorithms and the obtained equilibria provide good system performance irrespective of the number of mobile users and access points, for various distributions of the input data size and task complexity, and confirm the low complexity of the proposed algorithms.

References

[1] M. Hakkarainen, C. Woodward, and M. Billinghurst, "Augmented assembly using a mobile phone," in *Proc. of IEEE/ACM ISMAR*, Sept 2008, pp. 167–168.

[2] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uwave: Accelerometer-based personalized gesture recognition and its applications," in *Proc. of IEEE PerCom*, March 2009, pp. 1–9.

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. of ACM MobiSys*, 2010, pp. 49–62.

[4] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. of IEEE INFOCOM*, March 2012, pp. 2716–2720.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI White Paper*, pp. 1–16, 2015.

[6] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. of IEEE INFOCOM*, 2013, pp. 1285–1293.

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE TPDS*, vol. 26, no. 4, pp. 974–983, 2015.

[8] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 1. ACM, 2001, pp. 50–61.

[9] W. Yuan and K. Nahrstedt, "Energy-efficient cpu scheduling for multimedia applications," *ACM TOCS*, pp. 292–331, 2006.

[10] X. Li, Q. Xue, and M. C. Chuah, "Casheirs: Cloud assisted scalable hierarchical encrypted based image retrieval system," in *INFOCOM 2017*. IEEE, 2017, pp. 1–9.

[11] X. Zhang, J. Schiffman, S. Gibbs, A. Kunjithapatham, and S. Jeong, "Securing elastic applications on mobile devices for cloud computing," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 127–134.

[12] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of IEEE INFOCOM*, May 2017.

[13] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, pp. 23–32, Apr. 2013.

[14] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE T-SIPN*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[15] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, "An iterative double auction for mobile data offloading," in *Proc. of WiOpt*, May 2013, pp. 154–161.

[16] T. Joshi, A. Mukherjee, Y. Yoo, and D. P. Agrawal, "Airtime fairness for ieee 802.11 multirate networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 4, pp. 513–527, 2008.

[17] M. Xiao, N. B. Shroff, and E. K. Chong, "A utility-based power-control scheme in wireless cellular systems," *IEEE/ACM Transactions on networking*, vol. 11, no. 2, pp. 210–221, 2003.

[18] C. U. Saraydar, N. B. Mandayam, and D. J. Goodman, "Efficient power control via pricing in wireless data networks," *IEEE transactions on Communications*, vol. 50, no. 2, pp. 291–303, 2002.

[19] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 4th ed. Morgan Kaufmann Publishers Inc., 2011.

[20] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stößer, "Cloud computing–a classification, business models, and research directions," *Business & Information Systems Engineering*, vol. 1, no. 5, pp. 391–399, 2009.

[21] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[22] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer Mag.*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[23] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[24] I. Milchtaich, "Congestion games with player-specific payoff functions," *Games and Economic Behavior*, vol. 13, no. 1, pp. 111 – 124, 1996.

[25] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124 – 143, 1996.

[26] T. Roughgarden, "Intrinsic robustness of the price of anarchy," *Journal of the ACM (JACM)*, vol. 62, no. 5, p. 32, 2015.

[27] A. Aragon-Zavala, *Antennas and propagation for wireless communication systems*. John Wiley & Sons, 2008.

[28] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proc. of the 9th ACM SIGCOMM conference*, 2009, pp. 280–293.

[29] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.

[30] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *ISCC*, 2012, pp. 59–66.

[31] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of the 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 4–4.

[32] G. Hardin, "The tragedy of the commons," *Science*.

[33] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mob. Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb 2013.

[34] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *ACM Mob. Comput. Commun. Rev.*, pp. 19–26, Jan 1998.

[35] E. Hyytiä, T. Spyropoulos, and J. Ott, "Offload (only) the right jobs: Robust offloading using the Markov decision processes," in *Proc. of IEEE WoWMoM*, Jun. 2015, pp. 1–9.

[36] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "MuSIC: Mobility-aware optimal service allocation in mobile cloud computing," in *Proc. of IEEE CLOUD*, Jun. 2013, pp. 75–82.

[37] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *SOSE*, Mar. 2013, pp. 494–502.

[38] V. Cardellini, V. De Nitto Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, 2015.

[39] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM ToN*, pp. 2795–2808, 2016.

[40] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy efficient offloading for competing users on a shared communication channel," in *Proc. of IEEE ICC*, Jun. 2015, pp. 3192–3197.

[41] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. of ACM MSWiM*, 2015, pp. 271–278.

Paper C

Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing

Slađana Jošilo and György Dán

IEEE/ACM Transactions on Networking (ToN), vol. 28, no. 2, pp. 667-680, 2020.

Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing

Slađana Jošilo and György Dán
Division of Network and Systems Engineering,
School of Electrical Engineering and Computer Science
KTH, Royal Institute of Technology, Stockholm, Sweden
E-mail: {josilo, gyuri}@kth.se *

Abstract

Motivated by various delay sensitive applications, we address the problem of coordinating the offloading decisions of wireless devices that periodically generate computationally intensive tasks. We consider autonomous devices that aim at minimizing their own cost by choosing when to perform their tasks and whether or not to offload their tasks to an edge cloud through one of the multiple wireless links. We develop a game theoretical model of the problem, prove the existence of pure strategy Nash equilibria and propose a polynomial complexity algorithm for computing an equilibrium. Furthermore, we characterize the structure of the equilibria, and by providing an upper bound on the price of anarchy of the game we establish an asymptotically tight bound on the approximation ratio of the proposed algorithm. Our simulation results show that the proposed algorithm achieves significant performance gain compared to uncoordinated computation offloading at a computational complexity that is on average linear in the number of devices.

Index terms— computation offloading, edge computing, game theory, decentralized resource management

1 Introduction

The emergence of affordable wireless sensors, such as cameras, has given rise to a variety of Internet of Things (IoT) applications, including surveillance [1], tracking [2] and traffic monitoring [3]. These applications typically involve the periodic collection of sensory data, which need to be processed in a timely manner to ensure the stability of potential feedback control loops. Processing often involves some form of machine

*The work was partly funded by the Swedish Research Council through project 621-2014-6.

learning, e.g., visual analysis, which may be too computationally intensive to be performed in the sensors.

A promising solution to enable the timely processing of computationally intensive IoT tasks is mobile edge computing (MEC) [4]. Opposed to traditional remote cloud infrastructures such as Amazon and Azure [5], MEC provides computing resources close to the end users, i.e., at the network edge, which makes it a better candidate for meeting the requirements of delay sensitive IoT applications.

By offloading the computation to nearby edge clouds [6], devices may be able to significantly reduce their response times and energy consumption and thus to extend the lifetime of their batteries. Nevertheless, without coordination of their offloading decisions, devices in MEC systems might experience poor performance due to contention for communication and computing resources. Therefore, in order to use MEC systems to their full potential the autonomous devices need to coordinate their offloading decisions over time and across communication and computing resources.

There are several challenges facing the coordination of offloading decisions of autonomous devices in MEC systems. First, edge clouds are not as computationally powerful as remote clouds, and thus the response time and energy consumption of devices may be affected by contention for both communication and computing resources [7–9]. Second, MEC systems are likely to combine heterogeneous communication and computing resources, and thus coordination of offloading decisions involves not only deciding whether or not to offload the tasks, but also which of the communication and computing resources to use in the case of offloading. Third, IoT devices such as vehicles, drones and manufacturing machines [10] may be autonomous entities with different computing capabilities and tasks, and thus with individual interests in terms of response time and energy consumption requirements [11, 12]. Finally, besides the allocation of edge resources, coordination for offloading periodic tasks involves deciding when to process the collected sensory data and it may also affect the optimal time for sensing, so as to minimize the age of information [13, 14]. All these challenges make the problem of coordinating the offloading decisions of autonomous devices inherently difficult.

In this paper we address this problem for a MEC system in which the devices aim at minimizing their cost defined as a linear combination of the task completion time and the energy consumption. Each device can choose autonomously the time slot for performing its periodic task and in the chosen time slot it can decide whether to perform the task locally or to offload it to an edge cloud through one of multiple heterogeneous wireless links.

We make three important contributions to solve the problem. First, based on a game theoretical treatment of the problem, we propose a polynomial time decentralized algorithm for coordinating the offloading decisions of the devices, and prove the convergence of the algorithm to a pure strategy Nash equilibrium. Second, we characterize the structure of the computed equilibrium. Third, we establish an asymptotically tight upper bound on the price of anarchy of the game, and by doing so we show that a the proposed algorithm has a bounded approximation ratio. We

use simulations to assess the performance of the proposed algorithm in a variety of scenarios. Our results show that the algorithm can efficiently coordinate the offloading decisions of autonomous devices with periodic tasks at low computational complexity.

The rest of the paper is organized as follows. In Section 2 we present the system model and the problem formulation. In Section 4 we present the algorithm, prove its convergence and characterize the structure of computed equilibria. In Section 5.2 we provide a bound on the approximation ratio and in Section 6 we show numerical results. In Section 7 we discuss related work and in Section 8 we conclude the paper.

2 System Model

We consider an edge computing system that consists of N devices, A access points (APs) and an edge cloud. We denote by $\mathcal{N}=\{1,2,...,N\}$ and $\mathcal{A}=\{1,2,...,A\}$ the set of devices and the set of APs, respectively. For ease of reference, the key notations used in the paper are summarized in Table 1.

We consider that device i generates a computationally intensive task periodically every T time units, and we characterize the task by the mean size D_i of the input data and by the mean number of CPU cycles L_i required to perform the computation. Similar to related works on mobile cloud computing [15–17], we define the mean number of CPU cycles as $L_i{=}D_iE[X]$, where $E[X]$ is the mean number of CPU cycles required per data bit. We assume that $E[X]$ is known from previous measurements, which is reasonable for periodically generated tasks; in fact X is often modeled by a Gamma distribution [18–20], for which unbiased estimators exist [21]. It is important to note that our model is based on the average complexity and thus it does not depend on the distribution. Furthermore, the assumption of homogeneous task periodicities is reasonable for modeling the surveillance of homogeneous physical phenomena and in manufacturing systems as discussed in [22]. We leave the case of heterogeneous periodicities to be subject of future work.

We consider that time is slotted and we denote by $\mathcal{T}=\{1,2,...,T\}$ the set of time slots. Every device can choose a time slot $t \in \mathcal{T}$ for performing the whole task $< D_i, L_i >$ and in the chosen time slot t it can decide whether to perform the task locally or to offload the computation to the cloud server through an AP $a \in \mathcal{A}$. Thus, device $i \in \mathcal{N}$ can choose one element of the discrete set $\mathfrak{D}_i = \mathcal{T} \times \{\mathcal{A} \cup \{i\}\}$, where i corresponds to local computing. We denote by $d_i \in \mathfrak{D}_i$ the decision of device i, and refer to it as its strategy. We refer to the collection $\mathbf{d}=(d_i)_{i\in\mathcal{N}}$ as a strategy profile, and we denote by $\mathfrak{D}=\times_{i\in\mathcal{N}}\mathfrak{D}_i$ the set of all feasible strategy profiles.

For a strategy profile $\mathbf{d} \in \mathfrak{D}$ we denote by $O_{(t,a)}(\mathbf{d}) = \{i | d_i = (t,a)\}$ the set of devices that offload using AP a in time slot t, and we denote by $n_{(t,a)}(\mathbf{d}) = |O_{(t,a)}(\mathbf{d})|$ the number of devices that use AP a in time slot t. Furthermore, we define the set of all devices that offload in time slot t as $O_{(t,c)}(\mathbf{d}) = \cup_{a\in\mathcal{A}} O_{(t,a)}(\mathbf{d})$, and the total number of devices that offload in time slot t as $n_{(t,c)}(\mathbf{d}){=}\sum_{a\in\mathcal{A}} n_{(t,a)}(\mathbf{d})$. Finally,

Table 1: Summary of key notations

Notation	Description
$\mathcal{N}$	Set of N devices
$\mathcal{A}$	Set of A APs
$\mathcal{T}$	Set of T time sots
D_i	Mean size of the input data for device i
L_i	Mean task complexity for device i
F_i^0	Computational capability of device i
v_i	Energy consumption of device i per CPU cycle
γ_i^T	Completion time weight for device i
γ_i^E	Energy consumption weight for device i
T_i^0	Local execution time for device i
E_i^0	Local execution energy consumption for device i
C_i^0	Local computing cost for device i
$R_{i,a}$	Uplink PHY rate of device i towards AP a
$P_{i,a}$	Transmit power of device i towards AP a
F^c	Cloud computing capability
$\mathfrak{D}_i$	Set of feasible decisions for device i
d_i	Decision of device i, $d_i \in \mathfrak{D}_i$
$\mathbf{d}$	Strategy profile
$O_{(t,a)}(\mathbf{d})$	Set of $n_{(t,a)}(\mathbf{d})$ devices $i \in \mathcal{N}$ s.t. $d_i = (t,a)$ in $\mathbf{d}$
$O_{(t,c)}(\mathbf{d})$	Set of $n_{(t,c)}(\mathbf{d})$ devices offloading in time slot t in $\mathbf{d}$
$O(\mathbf{d})$	Set of all devices that offload their tasks in $\mathbf{d}$
$F_{i,t}^c(\mathbf{d})$	Cloud computing capability assigned to device i in $\mathbf{d}$
$\omega_{i,(t,a)}(\mathbf{d})$	Uplink rate of device i, $d_i = (t,a)$ in $\mathbf{d}$
$T_{i,(t,a)}^{tx}(\mathbf{d})$	Transmission time of device i, $d_i = (t,a)$ in $\mathbf{d}$
$E_{i,(t,a)}^{tx}(\mathbf{d})$	Energy consumption of device i, $d_i = (t,a)$ in $\mathbf{d}$
$T_{i,(t,c)}^{exe}(\mathbf{d})$	Cloud execution time for device i in time slot t in $\mathbf{d}$
$T_{i,(t,a)}^{off}(\mathbf{d})$	Total offloading time for device i, $d_i = (t,a)$ in $\mathbf{d}$
$C_{i,(t,a)}^{c}(\mathbf{d})$	Offloading cost of device i, $d_i = (t,a)$ in $\mathbf{d}$
$C_i(\mathbf{d})$	Cost of device i in $\mathbf{d}$

we denote by $O(\mathbf{d}) = \cup_{t \in \mathcal{T}} O_{(t,c)}(\mathbf{d})$ the set of all devices that offload in strategy profile $\mathbf{d}$. In what follows we introduce our model of sharing communication and computing resources among devices that offload their tasks.

2.1 Wireless resource sharing

We denote by $R_{i,a}$ the achievable uplink rate of device i towards AP (i.e., if device i was the only transmitter). We consider that $R_{i,a}$ depends on the the average channel conditions, modulation and coding scheme and the transmit power $P_{i,a}$. Thus, $R_{i,a}$

depends both on the device i and the AP a. We denote by $\omega_{i,(t,a)}(\mathbf{d})$ the allocated uplink rate of device i at AP a in time slot t and we consider that $\omega_{i,(t,a)}(\mathbf{d})$ is a non-increasing function $f_a(n_{(t,a)}(\mathbf{d}))$ of the number $n_{(t,a)}(\mathbf{d})$ of devices that use the same AP a in time slot t,

$$\omega_{i,(t,a)}(\mathbf{d}) = R_{i,a} \times f_a(n_{(t,a)}(\mathbf{d})). \tag{1}$$

This model is a good approximation for throughput sharing mechanisms in TDMA and OFDMA based MAC protocols [23].

Based on the uplink rate $\omega_{i,(t,a)}(\mathbf{d})$ we can express the time needed for device i to transmit the input data of size D_i through AP a in time slot t as

$$T^{tx}_{i,(t,a)}(\mathbf{d}) = D_i / \omega_{i,(t,a)}(\mathbf{d}). \tag{2}$$

We consider that every device i knows the transmit power $P_{i,a}$ that it would use to transmit the data through AP a, where $P_{i,a}$ may be determined using one of the power control algorithms proposed in [24, 25]. The transmit power $P_{i,a}$ and the transmission time $T^{tx}_{i,(t,a)}(\mathbf{d})$ determine the energy consumption of device i for transmitting the input data of size D_i through AP a in time slot t

$$E^{tx}_{i,(t,a)}(\mathbf{d}) = P_{i,a} T^{tx}_{i,(t,a)}(\mathbf{d}). \tag{3}$$

2.2 Computing resource sharing

We denote by F^c the computational capability of the edge cloud, and we consider that the computational capability $F^c_{i,t}(\mathbf{d})$ that device i receives from the cloud in time slot t is a non-increasing function $f_i(n_{(t,c)}(\mathbf{d}))$ of the total number $n_{(t,c)}(\mathbf{d})$ of devices that offload in time slot t

$$F^c_{i,t}(\mathbf{d}) = F^c \times f_i(n_{(t,c)}(\mathbf{d})). \tag{4}$$

This model could be used for edge computing systems in which the computing resources are shared according to a time fair resource allocation policy.

Observe that the time needed for performing device i's task in the cloud depends on the chosen time slot through the number of offloaders, and can be expressed as

$$T^{exe}_{i,(t,c)}(\mathbf{d}) = L_i / F^c_{i,t}(\mathbf{d}). \tag{5}$$

We consider that a single time slot is long enough for performing each user's task both in the case of local computing and in the case of computation offloading. This assumption is reasonable in the case of real time applications, where the worst-case task completion time must be less than a fraction of the periodicity.

We can use (2) and (5) to express the task completion time in the case of offloading as the sum of the transmission time and the execution time,

$$T^{off}_{i,(t,a)}(\mathbf{d}) = T^{tx}_{i,(t,a)}(\mathbf{d}) + T^{exe}_{i,(t,c)}(\mathbf{d}). \tag{6}$$

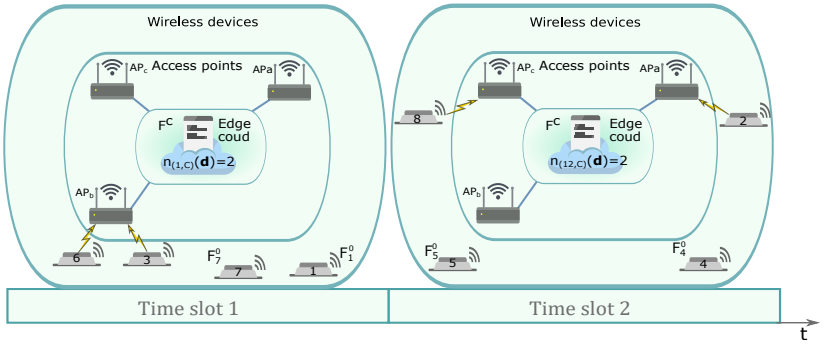

Figure 1: An example of a mobile cloud computing system that consists of an edge cloud, $A = 3$ APs, $T = 2$ time slots and $N = 8$ devices.

In (6) we made the common assumption that the time needed to transmit the result of the computation from the edge cloud to the device can be neglected [7, 26–30], because for many applications (e.g., object recognition, tracking) the size of the output data is significantly smaller than the size D_i of the input data.

2.3 Local computing

Devices that perform local computing use their own computing resources only. We denote by F_i^0 the computational capability of device i, and we consider that the computational capability F_i^0 of device i is constant over time, and hence the time needed for device i to perform its task can be expressed as

$$T_i^0 = L_i / F_i^0. \tag{7}$$

To model the corresponding energy consumption, we denote by v_i the energy consumption of device i per CPU cycle, which can be obtained through the measurement method proposed in [31]. We then express the expected energy consumption of device i for a task that requires on average L_i CPU cycles as

$$E_i^0 = v_i L_i. \tag{8}$$

Fig. 1 shows an example of a mobile edge computing system where devices can choose one out of two time slots to perform the computation. A device can offload its task to the cloud through one of three APs in its chosen time slot, if it decides to offload, e.g., in time slot 1 devices 3 and 6 offload their tasks through AP b, and devices 1 and 7 perform the computation locally.

3 Problem Formulation

In what follows we introduce the cost model of the devices and we formulate the multi-slot computation offloading problem.

3.1 Device Cost Model

To allow for flexibility in modeling the cost, we use the completion time γ_i^T and the energy consumption γ_i^E weights ($0 \leq \gamma_i^T, \gamma_i^E \leq 1$) to capture devices' preferences over the task completion time and the energy consumption, respectively. The weights can also be used to account for different units, which allows us to express the cost of device i as a linear combination of its completion time and its energy consumption, i.e.,

$$C_i^0 = \gamma_i^T T_i^0 + \gamma_i^E E_i^0. \tag{9}$$

Similarly we define the cost of device i in the case of offloading through AP a in time slot t as

$$C^c_{i,(t,a)}(\mathbf{d}) = \gamma_i^T T^{off}_{i,(t,a)}(\mathbf{d}) + \gamma_i^E E^{tx}_{i,(t,a)}(\mathbf{d}). \tag{10}$$

By (9) and (10) the cost of device i in strategy profile $\mathbf{d}$ is

$$C_i(\mathbf{d}) = \sum_{d_i \in \mathcal{T} \times \{i\}} \mathbf{1}_{(t,i)}(d_i) \cdot C_i^0 + \sum_{d_i \in \mathcal{T} \times \mathcal{A}} \mathbf{1}_{(t,a)}(d_i) \cdot C^c_{i,(t,a)}(\mathbf{d}), \tag{11}$$

where $\mathbf{1}_{(t,d)}(d_i)$ is the indicator function, i.e., $\mathbf{1}_{(t,d)}(d_i) = 1$ if $d_i = (t,d)$ and $\mathbf{1}_{(t,d)}(d_i) = 0$ otherwise.

Observe that in the above cost model, devices can adjust their objectives to the specific application requirements and to their current battery state by changing the values of the parameters γ_i^T and γ_i^E.

3.2 Multi-slot computation offloading game

We consider that devices are autonomous entities that follow their individual interests, and thus we consider that the objective of each device is to minimize its own cost (11), i.e., to find a strategy

$$d_i^* \in \arg\min_{d_i \in \mathfrak{D}_i} C_i(d_i, d_{-i}), \tag{12}$$

where $C_i(d_i, d_{-i})$ is the cost of device i if it chooses strategy d_i given the strategies d_{-i} of the other devices. Given the autonomy of the devices, we model the problem as a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$, in which the set of players is the set of devices (we use these two terms interchangeably). We refer to the game as the *multi-slot computation offloading game* (MSCOG). The MSCOG is a player specific network congestion game, as illustrated in Fig. 2.

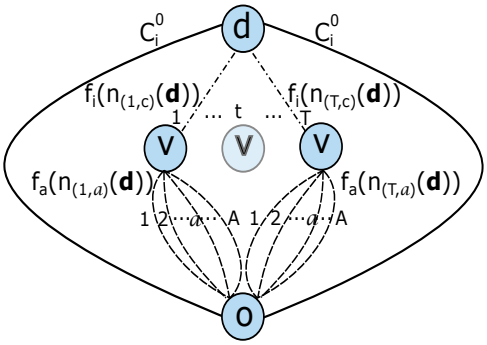

Figure 2: Network model of the MSCOG.

Our objective is to study fundamental questions related to the existence and computability of strategy profiles from which no device would want to deviate, i.e., pure strategy Nash equilibria, defined as follows [32].

Definition 1. A pure strategy Nash equilibrium (NE) is a strategy profile $\mathbf{d}^*$ in which all players play their best replies to each others' strategies, that is,

$$C_i(d_i^*, d_{-i}^*) \leq C_i(d_i, d_{-i}^*), \forall d_i \in \mathfrak{D}_i, \forall i \in \mathcal{N}.$$

Definition 2. Given a strategy profile $d=(d_i', d_{-i})$, an *improvement step* of device i is a strategy d_i' such that $C_i(d_i', d_{-i}) < C_i(d_i, d_{-i})$. A *best improvement* step is an improvement step that is a best reply. A *(best) improvement path* is a sequence of strategy profiles in which one device at a time changes its strategy through performing a *(best) improvement step*. We refer to the device that makes the best improvement step as the *deviator*. Observe that, by definition, no device can perform a (best) improvement step in a NE. A game in which every (best) improvement path is finite is said to have the *finite (best) improvement property*.

4 Computing Equilibria

We start with the analysis of the finiteness of improvement paths. Clearly, if a game has the finite improvement path property then it has a NE. As a first step, we show that this is not the case for the MSCOG, that is, improvement paths may be infinite, even in the case of a single time slot, i.e., $T = 1$.

Lemma 1. *The MSCOG does not have the finite improvement property.*

Proof. The proof is given in Appendix A.1. □

In what follows we show that although improvement paths may be cyclic, the MSCOG possesses a NE and a NE can be computed in polynomial time.

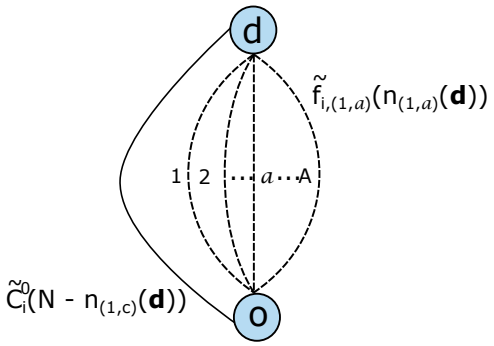

Figure 3: Network model of the MSCOG for $T = 1$.

4.1 Single time slot ($T = 1$)

We start with considering a single time slot case.

Theorem 1. *The MSCOG for $T = 1$ possesses a pure strategy Nash equilibrium.*

Proof. We prove the result by showing that the game is best response equivalent to a player specific congestion game $\tilde{\Gamma}$ on a parallel network, i.e., a singleton player specific congestion game [33]. Observe that if for $T=1$ we contract the edge (v,d) in the network shown in Fig. 2, i.e., if we replace the edge (v,d) and its two end vertices v and d by a single vertex, then we obtain a parallel network shown in Fig. 3. Let us define the local computation cost of player i in $\tilde{\Gamma}$ as $\tilde{C^0}_i(N-n_{(1,c)}(\mathbf{d}))=C^0_i-f_i(1+n_{(1,c)}(\mathbf{d}))+c$, and the cost of offloading through AP a as $\tilde{f}_{i,(1,a)}(n_{(1,a)}(\mathbf{d}))=f_{i,(1,a)}(n_{(1,a)}(\mathbf{d}))+c$, where c is a suitably chosen constant to make all costs non-negative. Observe that due to the contraction of the edge (v,d) the offloading cost is $\tilde{C^c}_{i,(1,a)}=C^c_{i,(1,a)}-f_i(n_{(1,c)}(\mathbf{d}))$, and thus the difference between the cost function of player i in $\tilde{\Gamma}$ and that in Γ only depends on the strategies of the other players. This in fact implies that $\tilde{\Gamma}$ and Γ are best-response equivalent, and thus they have identical sets of pure strategy Nash equilibria. Since $\tilde{\Gamma}$ is a singleton player specific congestion game, it has a NE, and so does Γ, which proves the result. □

Furthermore, a Nash equilibrium of the MSCOG can be found in polynomial time.

Corollary 1. *Consider a MSCOG with $T = 1$ and N players. Let $\mathbf{d}^*$ be a Nash equilibrium of the game, and consider that a new player is added to the game. Then there is a sequence of best responses that leads to a NE.*

Proof. The result follows from the best response equivalence to $\tilde{\Gamma}$, and from the proof of Theorem 2 in [32]. □

Unfortunately, the contraction technique used in the proof of Theorem 1 cannot be applied for $T > 1$, as the resulting game would no longer be a congestion game.

4.2 Multiple time slots ($T \geq 1$)

In order to answer the question for $T \geq 1$ we first show that if a pure strategy NE exists for $T \geq 1$ then its structure cannot be arbitrary.

Theorem 2. *Assume that $\boldsymbol{d}^*$ is a NE of the MSCOG with $T \geq 1$. Then the following must hold*
(i) $\min_{t' \in \mathcal{T}} n_{(t',c)}(\boldsymbol{d}^*) \leq n_{(t,c)}(\boldsymbol{d}^*) \leq \min_{t' \in \mathcal{T}} n_{(t',c)}(\boldsymbol{d}^*)+1$ *for* $\forall t,t' \in \mathcal{T}$,
(ii) if $n_{(t,c)}(\boldsymbol{d}^*) = n_{(t',c)}(\boldsymbol{d}^*)+1$ *for some* $t' \in \mathcal{T} \setminus \{t\}$, *then* $n_{(t,a)}(\boldsymbol{d}^*) \leq n_{(t',a)}(\boldsymbol{d}^*)+1$ *for every AP* $a \in \mathcal{A}$, *and*
(iii) if $n_{(t,a)}(\boldsymbol{d}^*) = n_{(t',a)}(\boldsymbol{d}^*) - k$ *for* $k > 1$ *and* $t' \neq t$, *then* $n_{(t',c)}(\boldsymbol{d}^*) \leq n_{(t,c)}(\boldsymbol{d}^*) \leq n_{(t',c)}(\boldsymbol{d}^*)+1$.

Proof. Clearly, all statements hold for $T=1$. Assume that $T>1$ and $\exists t,t' \in \mathcal{T}$ such that $n_{(t,c)}(\mathbf{d}^*) > n_{(t',c)}(\mathbf{d}^*)+1$. Then $\exists a \in \mathcal{A}$ such that $n_{(t,a)}(\mathbf{d}^*) \geq n_{(t',a)}(\mathbf{d}^*)+1$. Therefore, player $i \in O_{(t,a)}(\mathbf{d}^*)$ could decrease her cost by changing the strategy to offloading through AP a in time slot t'. This contradicts $\mathbf{d}^*$ being a NE and proves (i).

We continue by proving (ii). Assume that there is an AP a such that $n_{(t,a)}(\mathbf{d}^*) > n_{(t',a)}(\mathbf{d}^*)+1$ holds. Since $n_{(t,c)}(\mathbf{d}^*) = n_{(t',c)}(\mathbf{d}^*)+1$, we have that player $i \in O_{(t,a)}(\mathbf{d}^*)$ could decrease her cost by changing the strategy from (t,a) to (t',a). This contradicts $\mathbf{d}^*$ being a NE and proves (ii).

Finally, we prove (iii). First, assume that $n_{(t,c)}(\mathbf{d}^*) < n_{(t',c)}(\mathbf{d}^*)$. Since $n_{(t,a)}(\mathbf{d}^*) < n_{(t',a)}(\mathbf{d}^*)-1$, we have that player $i \in O_{(t',a)}(\mathbf{d}^*)$ could decrease her cost by changing the strategy from (t',a) to (t,a). This contradicts $\mathbf{d}^*$ being a NE and proves that $n_{(t,c)}(\mathbf{d}^*) \geq n_{(t',c)}(\mathbf{d}^*)$. Second, assume that $n_{(t,c)}(\mathbf{d}^*) > n_{(t',c)}(\mathbf{d}^*)+1$ holds. Since $n_{(t,a)}(\mathbf{d}^*) < n_{(t',a)}(\mathbf{d}^*) - 1$, there is at least one AP $b \neq a$ such that $n_{(t,b)}(\mathbf{d}^*) \geq n_{(t',b)}(\mathbf{d}^*) + 1$, and thus player $i \in O_{(t,b)}(\mathbf{d}^*)$ could decrease her cost by changing the strategy to (t',b). This contradicts $\mathbf{d}^*$ being a NE and proves that $n_{(t,c)}(\mathbf{d}^*) \leq n_{(t',c)}(\mathbf{d}^*) + 1$ must hold. □

We are now ready to formulate our main result concerning the existence of an equilibrium in the general case.

Theorem 3. *The MSCOG for $T \geq 1$ possesses a pure strategy Nash equilibrium.*

We provide the proof in the rest of the section, based on the *MyopicBest* (MB) algorithm shown in Fig. 4. The MB algorithm adds players one at a time, and lets them play their best replies given the other players' strategies in a particular order. Our proof is based on an induction in the number N of players, and starts with the following result.

Theorem 4. *The MB algorithm terminates in a NE after N steps for the MSCOG with $N \leq T$ players.*

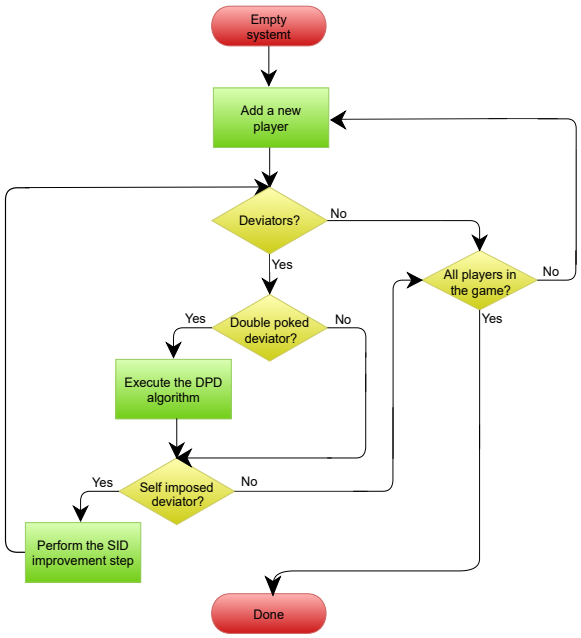

Figure 4: Flow chart of the MB algorithm.

Proof. It is easy to see that if a strategy profile $\mathbf{d}^*(N)$ is a NE for $N \leq T$ then by Theorem 2 there is at most one player per time slot. The MB algorithm computes such a strategy profile in N steps since each player upon it is added into the game chooses an empty time slot to perform its best reply and it has no incentive to deviate from the chosen strategy in the following induction steps. □

We continue by considering the case $N > T$. Let us assume that for $N-1 \geq T$ there is a NE $\mathbf{d}^*(N-1)$ and that upon induction step N a new player i enters the game and plays her best reply d_i^* with respect to $\mathbf{d}^*(N-1)$. After that, players can make best improvement steps one at a time starting from the strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$. If $d_i^* = (t,i)$, then $n_{(t,a)}(\mathbf{d}) = n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for every $(t,a) \in \mathcal{T} \times \mathcal{A}$, and thus $\mathbf{d}$ is a NE. Otherwise, if $d_i^* = (t,a)$, for some $a \in \mathcal{A}$, some players $j \in O_{(t,a)}(\mathbf{d})$ may have an incentive to make an improvement step because their communication and cloud computing costs have increased, and some players $j \in O_{(t,c)}(\mathbf{d}) \setminus O_{(t,a)}(\mathbf{d})$ may have an incentive to make an improvement step because their cloud computing cost has increased.

In order to define one of the possible best improvement paths that can be

$(\mathbf{d}, t, A') = DPD(\mathbf{d}, \mathbf{d}^*(N-1), (t,a), A')$

/*Players that want to stop to offload*/
$D_1' = \{j | d_j = (t,a), (t,j) = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d_{-j})\}$
/*Player that want to change offloading strategy*/
$D_2' = \{j | d_j = (t,a), (t',b) = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d_{-j}) \notin A',$
$(t,a) \neq (t',b)\}$
while $|D_1' \cup D_2'| > 0$ **do**
  /*Players that want to stop to offload have priority*/
  if $|D_1'| > 0$ **then**
    Take $i \in D_1'$
    $d_i = (t,i)$
  else
    Take $i \in D_2'$
    Let $d_i = \arg\min_{d \in \mathcal{T} \times \mathcal{A}} C_i(d, d_{-i})$
    Let $(t,a) \leftarrow d_i$
  end if
  Let $\mathbf{d} \leftarrow (d_i, d_{-i})$
  Update A', D_1', D_2'
end while
return $(\mathbf{d}, t, A')$

Figure 5: Pseudo code of the DPD algorithm.

generated starting form the strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$ we introduce the term deviator to denote a player that changes its strategy profile.

Definition 3. Consider two successive strategy profiles $\mathbf{d}'$ and $\mathbf{d}''$ in a best improvement path D that starts from an initial strategy profile $\mathbf{d}$. We say that the path D is a poke-new-deviator best improvement path if the following conditions hold:

1. If $\mathbf{d}'' = (d_i'', d'_{-i})$, then either $d_i'' = (t,i)$ or $d_i'' = (t,a)$ such that $n_{(t,a)}(\mathbf{d}') \geq n_{(t,a)}(\mathbf{d})$ (i.e., all deviators are either changing from offloading to local computing or from an offloading strategy to another offloading strategy for which the number of offloaders is at least as in the initial strategy profile $\mathbf{d}$).

2. If $\mathbf{d}'' = (d_i'', d'_{-i})$, then the next best improvement step can be performed only by a player $j \in \mathcal{N} \setminus \{i\}$ such that $d_j'' = d_i''$ (i.e. every next deviator has to be a player that wants to deviate from the strategy that has been chosen by the previous deviator).

Among all players that want to deviate from strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$, the MB algorithm allows players $j \in O_{(t,a)}(\mathbf{d})$ to perform best improvement steps, using the *DoublePokeDeviator* (DPD) algorithm, which creates poke-new-deviator best improvement paths. The pseudo code of the DPD algorithm is shown in Fig. 5. According to the definition of a poke-new-deviator best improvement path, there

are two types of players that can make a best improvement step using the DPD algorithm. The first type are players $j \in O_{(t,a)}(\mathbf{d})$ for which a best reply is to stop to offload. The second type are players $j \in O_{(t,a)}(\mathbf{d})$ for which a best reply is an offloading strategy $(t',b) \in \mathcal{T} \times \mathcal{A} \setminus \{(t,a)\}$ for which the number of offloaders in $\mathbf{d}$ is not smaller than the number of offloaders in the NE $\mathbf{d}^*(N-1)$. In each iteration, the DPD algorithm allows either one player of the first type, or one player of the second type to perform a best improvement step. In what follows we prove that the poke-new-deviator best improvement paths are finite and we provide an upper bound on the convergence of the DPD algorithm.

Proposition 1. *In a poke-new-deviator best improvement path generated by the DPD algorithm each player deviates at most once.*

Proof. Let us denote by $\mathbf{d}'$ a strategy profile after a player $j \in O_{(t,a)}(\mathbf{d})$ performs its best improvement step. First, observe that if player j's best improvement step is to stop to offload, then the resulting poke-new-deviator path terminates since only players that play the same strategy as the previous deviator are allowed to perform best improvement steps.

Otherwise, if player j's best improvement step is $(t',b) \in \mathcal{T} \times \mathcal{A} \setminus \{(t,a)\}$, then $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d}) + 1$ holds, and we can have one of the following: (1) there is no player $j' \in O_{(t',b)}(\mathbf{d})$ that wants to deviate from (t',b), (2) there is a player $j' \in O_{(t',b)}(\mathbf{d})$ that wants to deviate from (t',b).

If case (1) happens then the resulting poke-new-deviator path terminates because none of the players playing the same strategy as the previous deviator want to deviate. Otherwise, if case (2) happens then a new best improvement step can be triggered, which will bring the system to a state where $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d})$ holds.

In what follows we show that none of the players that has changed its offloading strategy in one of the previous best improvement steps would have an incentive to deviate again. Let us consider a player j' that changed its strategy from (t',b) to another offloading strategy, and let us assume that in one of the subsequent best improvement steps one of the players changes its offloading strategy to (t',b), and thus it brings the system to a state where $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d}) + 1$ holds. We observe that player j that has changed its strategy from (t,a) to (t',b) before player j' deviated from (t',b) would have no incentive to deviate from its strategy (t',b) after a new player starts offloading through AP b in time slot t'. This is because (t',b) was its best response while player j' was still offloading through AP b in time slot t', i.e, while $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d}) + 1$ was true. Therefore, a new best improvement step can be triggered only if there is another player that wants to change from (t',b) to another offloading strategy. If this happens, $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d})$ will hold again, and thus the maximum number of players that offload through AP b in time slot t' will be at most $n_{(t',b)}(\mathbf{d}) + 1$ in all subsequent best improvement steps. Consequently, player j would have no incentive to leave AP b in time slot t' in the subsequent steps. Therefore, each player deviates at most once in a poke-new-deviator best improvement path generated by the DPD algorithm, which proves the proposition. □

Corollary 2. *The length of a poke-new-deviator best improvement path generated by the DPD algorithm is at most* $N-1$.

Proof. It follows from Proposition 1 that the DPD algorithm can generate a longest poke-new-deviator best improvement path upon induction step N if every player $j \in O(\mathbf{d}^*(N-1))$ performs an improvement step, which proves the result. □

The DPD algorithm may be called multiple times during the execution of the MB algorithm, but as we show next for any fixed N, it is called a finite number of times.

Proposition 2. *The DPD algorithm is executed a finite number of times for any particular* N.

Proof. Let us assume that the DPD algorithm has been called at least once during the execution of the MB algorithm, and let us denote by $\mathbf{d}'$ the most recent strategy profile computed by the DPD algorithm. Now, let us assume that in the next best improvement step generated by the MB algorithm a player $i \in O(\mathbf{d}') \cup L(\mathbf{d}')$ changes its strategy to $(t,a) \in \mathcal{T} \times \mathcal{A}$. Starting from a strategy profile $\mathbf{d} = ((t,a), d'_{-i})$ players $j \in O_{(t,a)}(\mathbf{d})$ are allowed to perform the next best improvement step using the DPD algorithm.

Observe that players $j' \in O_{(t,a)}(\mathbf{d}')$ that in the previous best improvement steps changed their strategy to (t,a) using the DPD algorithm and triggered one of the players to leave the same strategy (t,a) would have no incentive to perform a best improvement step using the DPD algorithm. This is because the previous deviators $j' \in O_{(t,a)}(\mathbf{d}')$ brought $n_{(t,a)}(\mathbf{d}')$ to its maximum, that is to $n_{(t,a)}(\mathbf{d}^*(N-1))+1$, which decreased again to $n_{(t,a)}(\mathbf{d}^*(N-1))$ after the next deviator left strategy (t,a). Since the number of previous deviators $j' \in O_{(t,a)}(\mathbf{d}')$ that have no incentive to perform a new best improvement step using the DPD algorithm increases with every new best improvement path generated by the DPD algorithm, players will stop performing best improvement steps using the DPD algorithm eventually, which proves the proposition. □

So far we have proved that the DPD algorithm generates a finite number of poke-new-deviator best improvement paths, each of them with a length of at most $N-1$. In the following we use this result for proving the convergence of the MB algorithm. The pseudo code of the algorithm is shown in Fig. 6.

Proof of Theorem 3. We continue with considering all conditions under which the DPD algorithm may have terminated. First, let us assume that the last deviator's best improvement step is a strategy within time slot t'. The proof of Proposition 2 shows that the DPD algorithm terminates if one of the following happens: (i) starting from a strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$ all players performed their best improvement steps, (ii) some players did not deviate and the last deviator's strategy was $(t',0)$, i.e., the last deviator changed to local computing in time slot t', (iii) some players did

$\mathbf{d}^* = MB(\mathcal{N}, \mathcal{T}, \mathcal{A}, F^c, F_i^0)$

1: Let $N \leftarrow 1$
2: **for** $N = 1 \dots |\mathcal{N}|$ **do**
3: Let $A' \leftarrow \emptyset$ /*APs with decreased number of offloaders*/
4: Let $i \leftarrow N$
5: $d_i^* = \arg\min_{d \in \mathfrak{D}_i} C_i(d, \mathbf{d}^*(N-1))$
6: Let $\mathbf{d} \leftarrow (d_i^*, \mathbf{d}^*(N-1))$
7: **if** $d_i^* = (t, a)$ s.t. $a \in \mathcal{A}$ **then**
8: /*Players $j \in O_{(t,a)}(\mathbf{d})$ play best replies*/
9: $(\mathbf{d}', t', A') = DPD(\mathbf{d}, \mathbf{d}^*(N-1), (t, a), A')$
10: **if** $\exists j \in O_{(t',c)}(\mathbf{d}'), \exists d_j \in \mathfrak{D}_j$ s.t. $C_j(d_j, d'_{-j}) < C_j(d'_j, d'_{-j})$ **then**
11: /*Players $j \in O_{(t,c)}(\mathbf{d}')$ play best replies*/
12: $d_j = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d'_{-j})$
13: Let $\mathbf{d} \leftarrow (d_j, d'_{-j})$, update A'
14: **if** $\exists i \in O_{d_i}(\mathbf{d}), d_i \neq \arg\min_{d \in \mathfrak{D}_i} C_i(d, d_{-i}) \notin A'$ **then**
15: Let $(t, a) \leftarrow d_j$, **go to** 9
16: **else**
17: Let $\mathbf{d}' \leftarrow \mathbf{d}$
18: **end if**
19: **end if**
20: **if** $A' \neq \emptyset$ **then**
21: /*Players $j \in O(\mathbf{d}') \cup L(\mathbf{d}')$ play best replies*/
22: $(\mathbf{d}, (t, a), A') = SID(\mathbf{d}', A')$
23: **if** $\exists i \in O_{(t,a)}(\mathbf{d}), d_i \neq \arg\min_{d \in \mathfrak{D}_i} C_i(d, d_{-i}) \notin A'$ **then**
24: **go to** 9
25: **else if** $\exists i \in O(\mathbf{d}) \cup L(\mathbf{d}), d_i \neq \arg\min_{d \in \mathfrak{D}_i} C_i(d, d_{-i}) \in A'$ **then**
26: Let $\mathbf{d}' \leftarrow \mathbf{d}$, **go to** 22
27: **end if**
28: **end if**
29: **end if**
30: Let $\mathbf{d}^*(N) \leftarrow \mathbf{d}'$
31: **end for**
32: **return** $\mathbf{d}^*(N)$

Figure 6: Pseudo code of the MB algorithm.

not deviate and there was no player that wanted to change from the last deviator's strategy $(t', b) \in \mathcal{T} \times \mathcal{A}$.

Let us first consider case (i), and the last deviator that performed its best improvement step. If its best improvement step was to stop to offload, $n_{(t,a)}(\mathbf{d}') = n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for every $(t, a) \in \mathcal{T} \times \mathcal{A}$. Otherwise, if a best improvement step of the last deviator was to change its offloading strategy to (t', b), we have that $n_{(t,a)}(\mathbf{d}') \geq n_{(t,a)}(\mathbf{d}^*(N-1))$ for every $(t, a) \in \mathcal{T} \times \mathcal{A}$, where the strict inequality holds

only for (t',b), and $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d}^*(N-1))+1$. Since there is no offloading strategy for which the number of offloaders is less than the number of offloaders in the NE $\mathbf{d}^*(N-1)$, there is no player $j \in O(\mathbf{d}')$ that can decrease its offloading cost. Furthermore, there is no player that wants to change its strategy from local computing to offloading, and thus a strategy profile computed by the DPD algorithm is a NE.

If case (ii) or case (iii) happen the MB algorithm allows players that offload in the same time slot as the last deviator to perform any type of best improvement steps. Furthermore, if case (ii) happens and there are no APs with decreased number of offloaders compared with the NE $\mathbf{d}^*(N-1)$, i.e., $n_{(t,a)}(\mathbf{d}') = n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for every $(t,a) \in \mathcal{T} \times \mathcal{A}$, then the strategy profile $\mathbf{d}'$ computed by the DPD algorithm is a NE. Observe that $n_{(t,a)}(\mathbf{d}') = n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for every $(t,a) \in \mathcal{T} \times \mathcal{A}$ if strategy profile $\mathbf{d}'$ is obtained by the DPD algorithm starting from strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$.

Otherwise, if case (ii) happens such that there is a strategy $(t,a) \in \mathcal{T} \times \mathcal{A}$ for which $n_{(t,a)}(\mathbf{d}') < n_{(t,a)}(\mathbf{d}^*(N-1))$ holds, then players $j \in O_{(t',c)}(\mathbf{d}')$ that offload in the same time slot as the last deviator may want to change their offloading strategy to (t,a). Let us assume that there is a player $j \in O_{(t',c)}(\mathbf{d}')$ that wants to change its offloading strategy to (t,a) and let us denote by $\mathbf{d}$ a resulting strategy profile. Since $n_{(t,a)}(\mathbf{d}) = n_{(t,a)}(\mathbf{d}')+1$ and $n_{(t,c)}(\mathbf{d}) = n_{(t,c)}(\mathbf{d}')+1$ hold, some players $j \in O_{(t,a)}(\mathbf{d})$ may want to perform a best improvement step using the DPD algorithm, which can happen only a finite number of times accoring to Proposition 2.

We continue the analysis by considering case (iii). Observe that if there is a strategy (t,a) for which $n_{(t,a)}(\mathbf{d}') < n_{(t,a)}(\mathbf{d}^*(N-1))$ players $j \in O_{(t',c)}(\mathbf{d}')$ that offload in the same time slot as the last deviator may want to change their offloading strategy to (t,a). Furthermore, players $j \in O_{(t',c)}(\mathbf{d}') \setminus O_{(t',b)}(\mathbf{d}')$ may want to stop to offload or to change to any offloading strategy $(t,a) \in \mathcal{T} \times \mathcal{A} \setminus \{(t',b)\}$ since their cloud computing cost increased. Let us assume that there is a player $j \in O_{(t',c)}(\mathbf{d}')$ that wants to change its offloading strategy to $(t,a) \in \mathcal{T} \times \mathcal{A} \setminus \{(t',b)\}$ and let us denote by $\mathbf{d}$ the resulting strategy profile. Since $n_{(t,a)}(\mathbf{d}) = n_{(t,a)}(\mathbf{d}')+1$ and $n_{(t,c)}(\mathbf{d}) = n_{(t,c)}(\mathbf{d}')+1$ hold, some players $j \in O_{(t,a)}(\mathbf{d})$ may want to perform a best improvement step using the DPD algorithm, which can happen only a finite number of times according to Proposition 2.

If case (ii) or case (iii) happens and there is no player $j \in O_{(t',c)}(\mathbf{d}')$ that wants to deviate, the MB algorithm allows players from the other time slots $t \in \mathcal{T} \setminus \{t'\}$ to perform best improvement steps using *SelfImposedDeviator* (SID) algorithm shown in Fig. 7. Observe that players from time slots $t \in \mathcal{T} \setminus \{t'\}$ are not poked to deviate by the other players, and only reason why they would have an incentive to deviate is that $n_{(t,a)}(\mathbf{d}') < n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for some strategies $(t,a) \in \mathcal{T} \times \mathcal{A}$. The SID algorithm first allows one of the players $j \in O(\mathbf{d}') \setminus O_{(t',c)}(\mathbf{d}')$ that already offloads to perform a best improvement step, and if there is no such player the SID algorithm allows one of the players $j \in L(\mathbf{d}')$ that performs computation locally to

$(\mathbf{d},(t,a),A') = SID(\mathbf{d},A')$

/*Players that offload and can decrease their offloading cost*/
$D_1 = \{j \in O(\mathbf{d}) | (t,a) = \arg\min_{d \in \mathfrak{D}_j} C_j(d,d_{-j}) \in A', d_j \neq (t,a)\}$
/*Players that compute locally and want to start to offload*/
$D_2 = \{j \in L(\mathbf{d}) | (t,a) = \arg\min_{d \in \mathfrak{D}_j} C_j(d,d_{-j}) \in A'\}$
if $|D_1 \cup D_2| \neq \emptyset$ **then**
  /*Players that offload have priority*/
  if $D_1 \neq \emptyset$ **then**
    Take $i \in D_1$
  else if $D_2 \neq \emptyset$ **then**
    Take $i \in D_2$
  end if
  $d_i' = \arg\min_{d \in \mathfrak{D}_i} C_i(d,d_{-i})$
  Let $\mathbf{d} \leftarrow (d_i', d_{-i})$
  Let $(t,a) \leftarrow d_i'$
  Update A'
end if
return $(\mathbf{d},(t,a),A')$

Figure 7: Pseudo code of the SID algorithm.

start to offload. Let us assume that there is a strategy (t,a) for which $n_{(t,a)}(\mathbf{d}') < n_{(t,a)}(\mathbf{d}^*(N-1))$ holds and that there is a player $j \in O(\mathbf{d}') \setminus O_{(t',c)}(\mathbf{d}') \cup L(\mathbf{d}')$ that wants to deviate to strategy (t,a). We denote by $\mathbf{d}$ the resulting strategy profile, after player j performs its best improvement step. Since $n_{(t,a)}(\mathbf{d}) = n_{(t,a)}(\mathbf{d}') + 1$ and $n_{(t,c)}(\mathbf{d}) = n_{(t,c)}(\mathbf{d}') + 1$ hold, some players $j \in O_{(t,a)}(\mathbf{d})$ may want to perform a best improvement step using the DPD algorithm, which can happen only a finite number of times according to Proposition 2. Finally, let us consider case (iii) such that there is a player $j \in O_{(t',c)}(\mathbf{d}') \setminus O_{(t',b)}(\mathbf{d}')$ that wants to stop to offload because its cloud computing cost increased. Let us denote by $\mathbf{d}$ a strategy profile after player j changes its strategy from $(t',a) \neq (t',b)$ to local computing. We have that $n_{(t',a)}(\mathbf{d}) = n_{(t',a)}(\mathbf{d}') - 1$, and if $n_{(t',a)}(\mathbf{d}') = n_{(t',a)}(\mathbf{d}^*(N-1))$ we have that players $j' \in O(\mathbf{d}) \setminus O_{(t',a)}(\mathbf{d})$ may have an incentive to change their offloading strategy to (t',a) if doing so decreases their offloading cost. We have seen that a best improvement step of this type can trigger the DPD algorithm a finite number of times according to Proposition 2. Now, let us assume that a player $j' \in O_{(t,b)}(\mathbf{d})$, where $(t,b) \in \mathcal{T} \times \mathcal{A} \setminus \{(t',a)\}$, changes its offloading strategy from (t,b) to (t',a), and that by doing so it does not trigger the DPD algorithm. The resulting strategy profile $\mathbf{d} = ((t',a), d_{-j'})$ is such that $n_{(t,b)}(\mathbf{d}) = n_{(t,b)}(\mathbf{d}') - 1$ holds, and if $n_{(t,b)}(\mathbf{d}') = n_{(t,b)}(\mathbf{d}^*(N-1))$ some players may have an incentive to change their offloading strategy to (t,b) if doing so decreases their offloading cost.

We continue by considering the case where all subsequent best improvement steps are such that deviators change to a strategy for which the number of offloaders

is less than the number of offloaders in the NE $\mathbf{d}^*(N-1)$ and by doing so they do not trigger the DPD algorithm. Therefore, the resulting best improvement path is such that the cost of each deviator decreases with every new best improvement step it makes. Assume now that after $k \geq 2$ improvement steps player j' wants to return back to strategy (t,b). By the definition of the resulting best improvement path, the cost of player j' in the $(k+1)$-th improvement step is not only less than the cost in the k-th best improvement step, but also less than its cost in the first best improvement step. Therefore, player j' will not return to a strategy it deviated from, and thus it will deviate at most $T \times A - 1$ times. Consequently, when there are no players that can trigger the DPD algorithm, players that change their startegy from local computing to offloading using the SID algorithm, can only decrease their offloading cost in the subsequent best improvement steps, and thus they would have no incentive to stop to offload. Since the number of players is finite, the players will stop changing from local computing to offloading eventually, which proves the theorem. □

Even though the convergence proof of the MB algorithm is fairly involved, the algorithm itself is computationally efficient, as we show next.

Theorem 5. *When a new player i enters the game in an equilibrium $\boldsymbol{d}^*(N-1)$, the MB algorithm computes a new equilibrium $\boldsymbol{d}^*(N)$ after at most $N \times T \times A - 2$ best improvement steps.*

Proof. In the worst case scenario the DPD algorithm generates an $N-2$ steps long best improvement path, and a player that offloads in the same time slot as the last deviator, but not through the same AP changes to local computing, because its cloud computing cost increased. Observe that the worst case scenario can happen only if $|O(\mathbf{d}^*(N-1))| = N-1$ holds. Furthermore, $N-2$ players will have an opportunity to deviate using the DPD algorithm and a player that offloads in the same time slot as the last deviator will have an opportunity to stop to offload only if $n_{(t,a)}(\mathbf{d}^*(N-1)) = n_{(t',b)}(\mathbf{d}^*(N-1))$ holds for every $(t,a),(t',b) \in \mathcal{T} \times \mathcal{A}$. Furthermore, in the worst case scenario, the best improvement path generated by the DPD algorithm is followed by an $N \times (T \times A - 1)$ long best improvement path, in which deviators change to a strategy for which the number of offloaders is less than the number of offloaders in the NE $\mathbf{d}^*(N-1)$ and by doing so they do not trigger the DPD algorithm. Therefore, a NE can be computed in at most $N-2+N \times (T \times A - 1)$ best improvement steps. □

By adding players one at a time, it follows that the MB algorithm has quadratic worst case complexity.

Theorem 6. *The MB algorithm computes a NE allocation in $O(N^2 \times T \times A)$ time.*

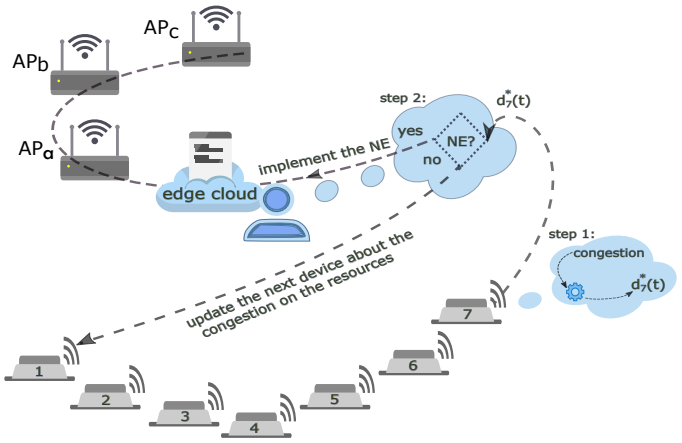

Figure 8: Example of the interaction between the centralized entity and devices 1 and 7 in a decentralized implementation of the MB algorithm.

4.3 Implementation considerations

Motivated by potential use cases that rely on the autonomy of devices [10], we consider that the MB algorithm can be implemented in a decentralized manner, by letting devices perform the best improvement steps one at a time. For computing a best response, besides its local parameters (e.g. D_i, L_i, F_i^0), each device i requires information about achievable uplink rates, available cloud resources, and the number of users sharing the APs and the cloud. In practice these information can be provided by a centralized entity that is the part of the infrastructure, e.g., the cloud, and that stores information about the mobile cloud computing system. The advantages of such a decentralized implementation compared to a centralized one are threefold. First, it supports the autonomy of the devices in MEC systems by allowing them to make their own offloading decisions based on the information provided by the centralized entity. Second, it can relieve the cloud from complex centralized management. Third, devices do not need to reveal their parameters, but only their most recent decisions.

Fig. 8 illustrates the interaction between the centralized entity located in the cloud and the devices during the process of computing a NE in a decentralized way using the MB algorithm. The centralized entity sends the information about the system (i.e. the information about the resources and the congestion on the resources in each time slot) to the devices that are allowed to update their best responses in a certain order. Given the most recent information provided by the centralized entity, each device upon its turn computes a set of best responses and sends its best offloading decision back to the centralized entity. After receiving the offloading decision of the device, the centralized entity sends the updated information about the congestion on the resources to the next device that is supposed to update its

offloading decision. Observe that at some point in time, according to Theorem 3 and Theorem 6, the offloading decisions of all devices will be the same as the ones that they reported in the previous iteration and the reached state will be a NE of the MSCOG. Once such a state is reached, the centralized entity can implement the computed NE by allocating the communication and cloud resources to the devices according to their equilibrium offloading decisions.

5 NE Structure and Price of Anarchy

In what follows we characterize the structure of a NE computed by the MB algorithm and provide a bound on the price of anarchy of the game.

5.1 Equilibrium characterization

Recall that by Theorem 2, if a NE exists for $T \geq 1$ then the number of players is balanced across the time slots. Our next result characterizes a NE computed by the MB algorithm from the perspective of the number of offloaders per AP.

Lemma 2. *Consider a NE $\boldsymbol{d}^*(N-1)$ computed by the MB algorithm upon an induction step for some $T < N \leq |\mathcal{N}|$. Assume that a new player i enters the game and given the NE $\boldsymbol{d}^*(N-1)$ plays its best reply $d_i^*(N) = (t', a)$. If $n_{(t',a)}(\boldsymbol{d}^*(N-1)) > n_{(t,a)}(\boldsymbol{d}^*(N-1))$ for the same AP a and a time slot $t \in \mathcal{T} \setminus \{t'\}$, then the following holds*
(i) $n_{(t,c)}(\boldsymbol{d}^*(N-1)) = n_{(t',c)}(\boldsymbol{d}^*(N-1)) + 1$,
(ii) $A > 2$,
(iii) $n_{(t,a)}(\boldsymbol{d}^*(N-1) \geq n_{(t',a)}(\boldsymbol{d}^*(N-1)) - (A-2)$, and
(iv) if $n_{(t',a)}(\boldsymbol{d}^*(N-1)) - n_{(t,a)}(\boldsymbol{d}^*(N-1)) = k_a$ for $1 \leq k_a \leq A-2$ holds for every AP $a \in \mathcal{B} \subset \mathcal{A}$, $\mathcal{B} \neq \emptyset$, then $\sum_{a \in \mathcal{B}} k_a \leq A-2$ must hold.

Proof. We start with proving (i). The only reason why player i would choose to offload through AP a in time slot t', which is more congested than AP a in time slot t is that the cloud in time slot t is more congested than the cloud in time slot t', that is, $n_{(t,c)}(\mathbf{d}^*(N-1)) = n_{(t',c)}(\mathbf{d}^*(N-1)) + 1$ must hold, which proves (i).

We continue by proving (ii). It is easy to see that if $\mathcal{A} = \{a\}$, then player i's best reply has to be a strategy (t', a) for which $n_{(t',a)}(\mathbf{d}^*(N-1)) \leq n_{(t,a)}(\mathbf{d}^*(N-1))$ holds. Now, let us assume that $\mathcal{A} = \{a, b\}$. It follows from $n_{(t',a)}(\mathbf{d}^*(N-1)) > n_{(t,a)}(\mathbf{d}^*(N-1))$ and (i) that $n_{(t,b)}(\mathbf{d}^*(N-1)) > n_{(t',b)}(\mathbf{d}^*(N-1)) + 1$, and thus player $j \in O_{(t,b)}(\mathbf{d}^*(N-1))$ could decrease its cost by changing the strategy to (t', b). This contradicts $\mathbf{d}^*(N-1)$ being a NE, and proves (ii).

Next, we prove (iii). Assume that $n_{(t,a)}(\mathbf{d}^*(N-1)) < n_{(t',a)}(\mathbf{d}^*(N-1)) - (A-2)$ holds, which is equivalent to $n_{(t,c)}(\mathbf{d}^*(N-1)) - \sum_{b \neq a} n_{(t,b)}(\mathbf{d}^*(N-1)) < n_{(t',c)}(\mathbf{d}^*(N-1)) - \sum_{b \neq a} n_{(t',b)}(\mathbf{d}^*(N-1)) - (A-2)$. It follows from (i) that $n_{(t,c)}(\mathbf{d}^*(N-1)) = n_{(t',c)}(\mathbf{d}^*(N-1)) + 1$, and thus we have that $\sum_{b \neq a} n_{(t,b)}(\mathbf{d}^*(N-1)) > \sum_{b \neq a} n_{(t',b)}(\mathbf{d}^*$

$(N-1))+A-1$ holds. Therefore, there is at least one AP $b \neq a$ such that $n_{(t,b)}(\mathbf{d}^*(N-1)) > n_{(t',b)}(\mathbf{d}^*(N-1))$, and thus player $j \in O_{(t,b)}(\mathbf{d}^*(N-1))$ could decrease its cost by changing the strategy to (t',b). This contradicts $\mathbf{d}^*(N-1)$ being a NE, and proves (iii).

Finally, we prove (iv). Assume that $\sum_{a\in\mathcal{B}} k_a > A-2$, which implies that $\sum_{a\in\mathcal{B}} \left(n_{(t',a)}(\mathbf{d}^*(N-1)) - n_{(t,a)}(\mathbf{d}^*(N-1))\right) > A-2$ holds. It follows from (i) that $n_{(t,c)}(\mathbf{d}^*(N-1)) = n_{(t',c)}(\mathbf{d}^*(N-1))+1$, and thus we have that $\sum_{b\in\mathcal{A}\setminus\mathcal{B}} n_{(t,b)}(\mathbf{d}^*(N-1)) > \sum_{b\in\mathcal{A}\setminus\mathcal{B}} n_{(t',b)}(\mathbf{d}^*(N-1)) + A - 1$ holds. Therefore, there is at least one AP $b \in \mathcal{A}\setminus\mathcal{B}$ such that $n_{(t,b)}(\mathbf{d}^*(N-1)) > n_{(t',b)}(\mathbf{d}^*(N-1))$, and thus player $i \in O_{(t,b)}(\mathbf{d}^*(N-1))$ could decrease its cost by changing the strategy to (t',b). This contradicts $\mathbf{d}^*(N-1)$ being a NE, and proves (iv). □

Observe that Theorem 2 provides an upper bound on the number of offloaders per AP for every NE, while Lemma 2 provides a lower bound on the number of offloaders per AP for a NE computed by the MB algorithm. Therefore, if $n_{(t,c)}(\mathbf{d}^*) = n_{(t',c)}(\mathbf{d}^*)+1$, then by Theorem 2 we have that $n_{(t,a)}(\mathbf{d}^*) \leq n_{(t',a)}(\mathbf{d}^*)+1$ and by Lemma 2 we have that $n_{(t,a)}(\mathbf{d}^*) \geq n_{(t',a)}(\mathbf{d}^*) - (A-1)$.

5.2 Price of Anarchy Bound

We have so far shown that a NE of the MSCOG can be computed in polynomial time and we characterized the structure of the computed NE. We now address the important question how far the system performance would be from optimal in a NE. We do so by quantifying the worst case difference between the system performance in a NE and the optimal performance using the price of anarchy (PoA). The PoA of the game is defined as the ratio of the worst case NE cost and the minimal cost, and it can be expressed as

$$PoA = \frac{\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d}\in\mathfrak{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})}. \tag{13}$$

We first provide a bound of the PoA for $N \leq T$, in fact we show that a NE is optimal in this case.

Theorem 7. *A NE of the MSCOG for $N \leq T$ is the socially optimal strategy profile, i.e., $PoA = 1$ for $N \leq T$.*

Proof. We start with deriving a lower bound for an optimal solution $\overline{\mathbf{d}}$ of the MSCOG. The minimum offloading cost $\bar{C}_{i,a}$ that player i can achieve in $\overline{\mathbf{d}}$ is the cost when it offloads alone in a time slot $t\in\mathcal{T}$, i.e., $n_t(\overline{\mathbf{d}})=1$, through an AP a that provides maximum achievable uplink rate, i.e., $a=\arg\max_{b\in\mathcal{A}} R_{i,b}$. Consequently, we have that in the case of offloading $C_i(\overline{\mathbf{d}}) \geq \bar{C}_{i,a}$ holds. Otherwise, in the case of local computing, i.e., if $d_i = (t,i)$, we have that $C_i(\overline{\mathbf{d}}) = C_i^0$. Hence, we have

that $C_i(\overline{\mathbf{d}}) \geq \min\{C_i^0, \bar{C}c_{i,1}, \ldots, \bar{C}c_{i,A}\}$ holds, and thus a lower bound on the optimal solution cost is given by $\sum_{i\in\mathcal{N}} \min\{C_i^0, \bar{C}c_{i,1}, \ldots, \bar{C}c_{i,A}\}$, i.e., $\sum_{i\in\mathcal{N}} C_i(\overline{\mathbf{d}}) \geq \sum_{i\in\mathcal{N}} \min\{C_i^0, \bar{C}c_{i,1}, \ldots, \bar{C}c_{i,A}\}$ and the equality holds for $N \leq T$.

We continue with characterizing the cost in a NE $\mathbf{d}^*$ for $N \leq T$. From Theorem 4 it follows that for $N \leq T$ there is at most one player per time slot, and thus in a NE $\mathbf{d}^*$ each player $i \in \mathcal{N}$ plays its best reply d_i^*, where $C_i(d_i^*, d_{-i}^*) = \min\{C_i^0, \bar{C}c_{i,1}, \ldots, \bar{C}c_{i,A}\}$. Therefore, $PoA = 1$ for $N \leq T$, which proves the theorem. □

In what follows we give an upper bound on the PoA of the MSCOG for $N > T$. We start with the definition of the set $\mathcal{R} = \mathcal{T} \times \{\mathcal{A} \cup \{c\} \cup \mathcal{N}\}$ of all resources in the system, and the set $O_{(t,i)}(\mathbf{d}) = \{i | d_i = (t,i)\}$ of players that use local computing resource i in time slot t. Observe that either $O_{(t,i)}(\mathbf{d}) = \emptyset$ or $O_{(t,i)}(\mathbf{d}) = \{i\}$, i.e., $n_{(t,i)}(\mathbf{d}) = |O_{(t,i)}(\mathbf{d})| \in \{0,1\}$, since players do not share their local computing resources. Furthermore, $\sum_{t\in\mathcal{T}} n_{(t,i)}(\mathbf{d}) \leq 1$ must hold since every player $i \in \mathcal{N}$ can choose only one time slot $t \in \mathcal{T}$ to perform its task. Next, we introduce the notion of player specific constants

$$w_{i,(t,a)} \triangleq D_i/R_{i,a}, w_{i,(t,c)} \triangleq L_i/F^c, w_{i,(t,i)} \triangleq L_i/F_i^0.$$

For a strategy profile $\mathbf{d}$ we define the total weight $w_{(t,a)}(\mathbf{d})$ associated with AP a in time slot t as $w_{(t,a)}(\mathbf{d}) \triangleq \sum_{j\in O_{(t,a)}(\mathbf{d})} w_{j,(t,a)}$, the total weight $w_{(t,c)}(\mathbf{d})$ associated with cloud c in time slot t as $w_{(t,c)}(\mathbf{d}) \triangleq \sum_{j\in O_{(t,c)}(\mathbf{d})} w_{j,(t,c)}$ and the total weight $w_{(t,i)}(\mathbf{d})$ associated with local computing resource i in time slot t as $w_{(t,i)}(\mathbf{d}) \triangleq \sum_{j\in O_{(t,i)}(\mathbf{d})} w_{j,(t,i)}$. Note that either $w_{(t,i)}(\mathbf{d}) = w_{i,(t,i)} = C_i^0$ or $w_{(t,i)}(\mathbf{d}) = 0$ must hold since $j \notin O_{(t,i)}(\mathbf{d})$ for $j \neq i$. Next, using the above notation we can express the system cost $C(\mathbf{d})$ as

$$C(\mathbf{d}) = \sum_{r\in\mathcal{R}} \sum_{i\in O_r(\mathbf{d})} n_r(\mathbf{d}) w_{i,r} = \sum_{r\in\mathcal{R}} n_r(\mathbf{d}) w_r(\mathbf{d}). \tag{14}$$

Finally, given an optimal strategy profile $\overline{\mathbf{d}}$, we can express the PoA of the MSCOG as

$$PoA = \frac{\max_{\mathbf{d}^*\in\mathfrak{D}} \sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*) w_r(\mathbf{d}^*)}{\sum_{r\in\mathcal{R}} n_r(\overline{\mathbf{d}}) w_r(\overline{\mathbf{d}})}. \tag{15}$$

Theorem 8. *Consider the MSCOG with $N > T$. Then $PoA \leq N + 1$.*

Proof. Let us denote by $\mathcal{R}_{d_i} \subset \mathcal{R}$ the set of resources that player i uses in strategy profile $\mathbf{d}$. Then, from the definition of a NE $\mathbf{d}^*$ we have the following

$$\sum_{r\in\mathcal{R}_{d_i^*}} n_r(\mathbf{d}^*) w_{i,r} \leq \sum_{r\in\mathcal{R}_{d_i^*}\cap\mathcal{R}_{\overline{d}_i}} n_r(\mathbf{d}^*) w_{i,r} + \tag{16}$$

$$\sum_{r\in\mathcal{R}_{d_i^*}\setminus\mathcal{R}_{\overline{d}_i}} \big(n_r(\mathbf{d}^*)+1\big) w_{i,r} \leq \sum_{r\in\mathcal{R}_{\overline{d}_i}} \big(n_r(\mathbf{d}^*)+1\big) w_{i,r}.$$

By summing inequality (16) over all players $i \in \mathcal{N}$ we obtain

$$\sum_{i\in\mathcal{N}}\sum_{r\in\mathcal{R}_{d_i^*}} n_r(\mathbf{d}^*)w_{i,r} \leq \sum_{i\in\mathcal{N}}\sum_{r\in\mathcal{R}_{\overline{d}_i}} \big(n_r(\mathbf{d}^*)+1\big)w_{i,r}, \tag{17}$$

and by reordering summations in (17) we obtain

$$\sum_{r\in\mathcal{R}}\sum_{i\in O_r(\mathbf{d}^*)} n_r(\mathbf{d}^*)w_{i,r} \leq \sum_{r\in\mathcal{R}}\sum_{i\in O_r(\overline{\mathbf{d}})} \big(n_r(\mathbf{d}^*)w_{i,r}+w_{i,r}\big). \tag{18}$$

By using the definition of the total weight $w_r(\mathbf{d})$ associated with resource $r \in \mathcal{R}$ in strategy profile $\mathbf{d}$, we can rewrite (18) as

$$\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)w_r(\mathbf{d}^*) \leq \sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)w_r(\overline{\mathbf{d}}) + \sum_{r\in\mathcal{R}} w_r(\overline{\mathbf{d}}). \tag{19}$$

Next, observe that $n_r(\mathbf{d}) \leq N$ must hold for any feasible strategy profile $\mathbf{d}$ and for every resource $r \in \mathcal{R}$, and that $|O_r(\mathbf{d})| \geq 1$ implies $n_r(\mathbf{d}) \geq 1$. Therefore, we have that $\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)w_r(\overline{\mathbf{d}}) \leq N\sum_{r\in\mathcal{R}} n_r(\overline{\mathbf{d}})w_r(\overline{\mathbf{d}})$ and $\sum_{r\in\mathcal{R}} w_r(\overline{\mathbf{d}}) \leq \sum_{r\in\mathcal{R}} n_r(\overline{\mathbf{d}})w_r(\overline{\mathbf{d}})$. By using these observations in (19) we obtain the following inequality

$$\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)w_r(\mathbf{d}^*) \leq (N+1)\sum_{r\in\mathcal{R}} n_r(\overline{\mathbf{d}})w_r(\overline{\mathbf{d}}). \tag{20}$$

Finally, since $\sum_{r\in\mathcal{R}} n_r(\overline{\mathbf{d}})w_r(\overline{\mathbf{d}}) > 0$ must hold, we can divide the right and the left side of inequality (20) by $\sum_{r\in\mathcal{R}} n_r(\overline{\mathbf{d}})w_r(\overline{\mathbf{d}})$ to obtain

$$\frac{\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)w_r(\mathbf{d}^*)}{\sum_{r\in\mathcal{R}} n_r(\overline{\mathbf{d}})w_r(\overline{\overline{\mathbf{d}}})} \leq N+1. \tag{21}$$

Since (21) holds for any NE of the MSCOG, it also holds for the worst case NE, and thus from (15) we have that

$$PoA \leq N+1. \tag{22}$$

which proves the theorem. □

In what follows we investigate the tightness of the above bound on the PoA of the MSCOG.

Proposition 3. *There is an infinite family of instances of the MSCOG for which* $PoA = N - \epsilon$*, where*

$$\epsilon = \frac{(N-1)\sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}{\frac{1}{N}C_k^0 + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}. \tag{23}$$

Proof. The proof is given in Appendix A.2. □

Proposition 3 allows us to formulate the following result.

Corollary 3. *The upper bound $N+1$ on the PoA of the MSCOG is asymptotically tight.*

Proof. The proof is given in Appendix A.3. □

6 Numerical Results

In the following we show simulation results to evaluate the cost performance and the computational efficiency of the MB algorithm. Similar to [34, 35] we consider that the devices are placed uniformly at random over a square area of $1km \times 1km$, while the APs are placed at random on a *regular grid* with A^2 points defined over the area. We consider that the channel gain of device i to AP a is proportional to $d_{i,a}^{-\alpha}$, where $d_{i,a}$ is the distance between device i and AP a, and α is the path loss exponent, which we set to 4 according to the path loss model in urban and suburban areas [36]. For simplicity we assign a bandwidth of $B_a = 5$ *MHz* to every AP a, and the data transmit power of $P_{i,a}$ is drawn from a continuous uniform distribution on $[0.05, 0.18]$ *W* according to measurements reported in [37]. Given the noise power P_n we calculate the PHY rate $R_{i,a}$ as $R_{i,a} = B_a \log_2(1 + P_{i,a} d_{i,a}^{-\alpha} / P_n)$. We consider that the uplink rate of a device connected to an AP a scales directly proportional with the number of devices offloading through AP a. According to the specification reported in [38], the clock rate achievable for NVIDIA Tegra 2 is up to 1 *GHz*, and thus we consider that the computational capability F_i^0 of device i is uniformly distributed on $[0.5, 1]$ *GHz*. Based on the approximate relative computational parameters for devices and clouds reported in [39], we consider that the computation capability of the cloud is $F^c = 100$ *GHz* and we assume that the computational capability that a device receives from the cloud scales inversely proportional with the number of devices that offload. The input data size D_i and the number L_i of CPU cycles required to perform the computation are uniformly distributed on $[0.42, 2]$ *Mb* and $[0.1, 0.8]$ *Gcycles*, respectively. The consumed energy per CPU cycle v_i is set to $10^{-11}(F_i^0)^2$ according to measurements reported in [31]. The weights attributed to energy consumption γ_i^E and the response time γ_i^T are drawn from a continuous uniform distribution on $[0, 1]$.

We use four algorithms as a basis for comparison for the proposed *MB* algorithm. In the first algorithm devices choose a time slot at random, and implement an equilibrium allocation within their chosen time slots. We refer to this algorithm as the *RandomSlot* (RS) algorithm. The second algorithm considers that all devices perform local execution. The third algorithm is a worst case scenario where all devices choose the same time slot and implement an equilibrium allocation within that time slot. Observe that this corresponds to $T = 1$. In the fourth algorithm, the offloading decisions of the devices are made according to a socially optimal strategy

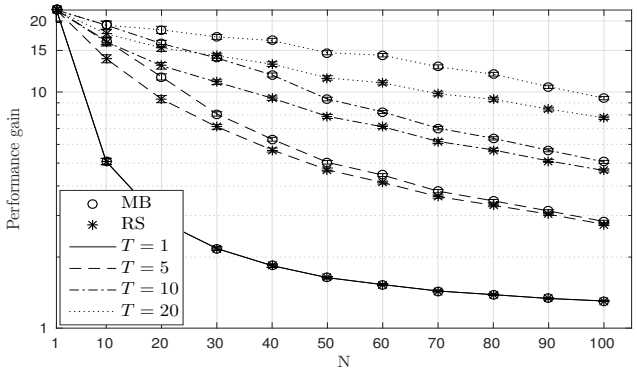

Figure 9: *Performance gain* vs. the number of devices (N).

profile $\mathbf{d}^*$. We define the *performance gain* of an algorithm as the ratio between the system cost reached when all devices perform local execution and the system cost reached by the algorithm and we define the *cost-approximation ratio* of an algorithm as the ratio between the system cost reached by the algorithm and the system cost reached when all devices choose offloading decisions according to the socially optimal strategy profile. Unfortunately, computing the socially optimal strategy profile was computationally feasible only for small problem instances due to the combinatorial nature of the corresponding system cost minimization problem, which is a 0-1 non-linear program. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

6.1 Performance gain vs. number of devices

We start with evaluating the *performance gain* as a function of the number N of devices for $A = 4$ APs. Fig. 9 shows the *performance gain* of the MB algorithm, the RS algorithm and the deterministic worst case $T = 1$. The results show that the *performance gain* decreases with the number of devices for all algorithms. This is due to that the APs and the cloud get congested as the number of devices increases. The performance gain of the MB algorithm is up to 50% higher than that of the RS algorithm for $T > 1$; the gap between the two algorithms is largest when the ratio N/T is approximately equal to 4. The reason is that as T increases the average number of offloaders per time slot remains balanced in the case of the MB algorithm. On the contrary, in the case of the RS algorithm some time slots may be more congested than others, since the players choose their time slot at random. However, the average imbalance in the number of offloaders per time slot decreases as the number of devices increases, thus the results are similar for large values of N. At the same time, the performance gain of the MB algorithm compared to that of the deterministic worst case $T = 1$ is almost proportional to the number T of time

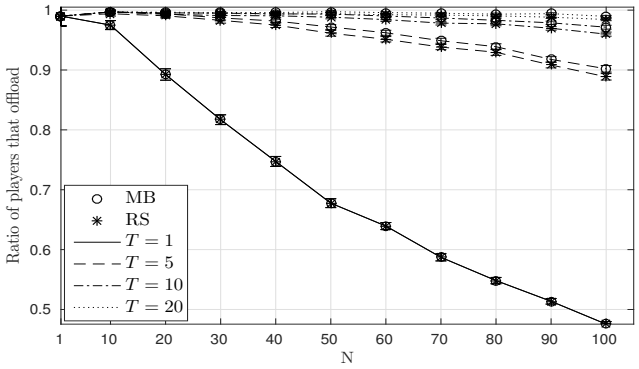

Figure 10: Ratio of offloaders vs. the number of devices (N).

slots, and shows that coordination is essential for preventing severe performance degradation. It is also interesting to note that for $T = 1$ the *performance gain* decreases with N at a much higher rate than for $T > 1$, which is due to the fast decrease of the number of offloaders, as we show next.

Fig. 10 shows the ratio of players that offload for the same set of parameters as in Fig. 9. The results show that in the worst case, for $T = 1$, the ratio of players that offload decreases almost linearly with N, which explains the fast decrease of the *performance gain* observed in Fig. 9. On the contrary, for larger values of T the ratio of players that offload appears less sensitive to N. We observe that the ratio of players that offload is in general higher in equilibrium than in the strategy profile computed by the RS algorithm, which explains the superior performance of MB observed in Fig. 9.

6.2 Performance gain vs number of APs

In order to evaluate the *performance gain* as a function of the number A of APs, we consider a system that consists of $N = 50$ devices. Fig. 11 shows the *performance gain* of the MB algorithm, the RS algorithm and the deterministic worst case $T = 1$. We observe that the *performance gain* achieved by the algorithms increases monotonically with the number of APs for all values of T with a decreasing marginal gain. The reason is that once $T \times A \geq N$ every device can offload its task through its favorite AP without sharing it, and hence the largest part of the offloading cost comes from the computing cost in the cloud. However, a small change in the *performance gain* is still present even for very large values of A because the density of the APs over a region becomes larger as A increases, and hence the channel gain, which depends on the distance between the device and the APs becomes larger on average. The results also show that MB always outperforms RS, and its *performance gain* compared to that of RS increases with T. Most importantly, the number of APs required for a

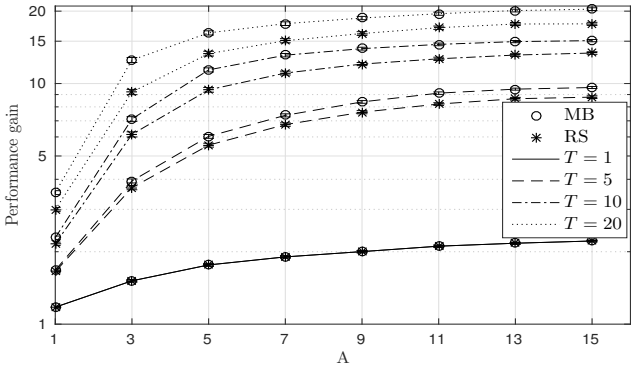

Figure 11: *Performance gain* vs. the number of APs (A).

certain performance gain is almost 50% lower using the MB algorithm compared to the RS algorithm for higher values of T, i.e., significant savings can be achieved in terms of infrastructural investments.

6.3 Cost-approximation ratio vs. number of devices

Fig. 12 shows the average *cost-approximation ratio* for the MB and the RS algorithms, and the computed *price of anarchy* (PoA) and the *price of stability* (PoS) as a function of the number N of devices. The results are shown for three values of the number T of time slots (i.e. $T \in \{1, 2, 3\}$) for a system with $A = 4$ APs. We show the results only up to 7 devices, because the computation of the socially optimal strategy profile was infeasible for larger problem instances. The results show that the MB and the RS algorithms have the same *cost-approximation ratio* for all values of N when $T = 1$, which is because the two algorithms are equivalent in this case, and thus they compute the same equilibrium strategy profiles. On the contrary, for $T > 1$ and for $N > 1$, a strategy profile computed by the RS algorithm is not an equilibrium, and the *cost-approximation ratio* of the RS algorithm is higher than that of the MB algorithm. We can also observe that the gap between the algorithms increases with T, which is because the imbalance in the average congestion per time slot increases with T in the case of the RS algorithm due to the random time slot selection. The results also show that for $T = 1$ the computed PoA increases linearly with N with a slope lower than 1 and that for $T > 1$ the computed PoA remains close to 2. Therefore, the results indicate that the worst case scenario for which the PoA of the game is asymptomatically close to N (c.f. Theorem 8 and Proposition 3 from Section 5.2) is not likely to happen. Finally, we observe that the computed PoS is equal to 1 in all cases, which suggests that the MB algorithm is able to compute a socially optimal equilibrium of offloading decisions.

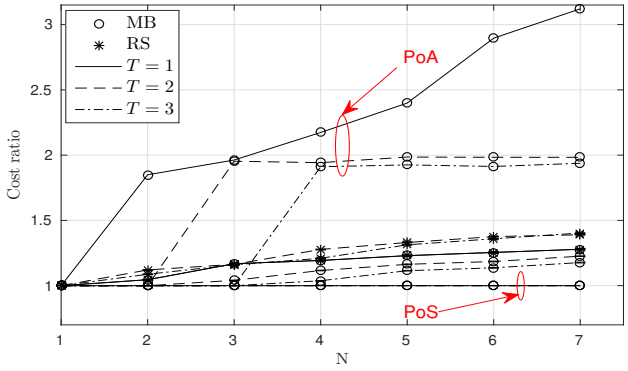

Figure 12: *Cost ratio* vs. the number of devices (N).

6.4 Computational Complexity

In order to assess the computational efficiency of the algorithms we consider the number of iterations, defined as the number of induction steps plus the total number of update steps over all induction steps needed to compute a strategy profile. Fig. 13 and Fig. 14 show the average number of iterations and the corresponding average strategy profile computation time for the MB and the RS algorithms, respectively. The results are shown as a function of the number N of devices in a system with $A = 4$ APs for four different values of the number T of time slots (i.e., $T \in \{1, 5, 10, 20\}$). The results show that the number of iterations scales approximately linearly with N for both algorithms, and indicates that the worst case scenario considered in Theorem 6 is unlikely to happen. The first interesting feature of Fig. 13 is that the number of iterations is slightly less in the case of the MB algorithm than in the case of the RS algorithm for all values of T, except for $T = 1$ for which the two algorithms are equivalent. These results coincide with the results in Fig. 14 that show that the MB algorithm is at least as good as the RS algorithm (i.e. the two algorithms consume the same amount of time for computing a strategy profile when $T = 1$ and the MB algorithm performs better than the RS algorithm for $T > 1$). The reason is that in the case of the MB algorithm the number of offloaders per time slot is more balanced, and hence the devices have less incentive to deviate when a new device enters the system, and their updates are always at least as good as in the case of RS algorithm, since the MB algorithm allows devices to change between time slots. On the contrary, in the case of the RS algorithm some of the time slots may be very congested, and the devices that offload within these time slots have a higher incentive to deviate when a new device enters the system. The second interesting feature of Fig. 13 is that the number of iterations is smaller for larger values of T for smaller values of N, but for larger values of N the results are reversed. The reason is that for smaller values of N the time slots are less congested on average as T

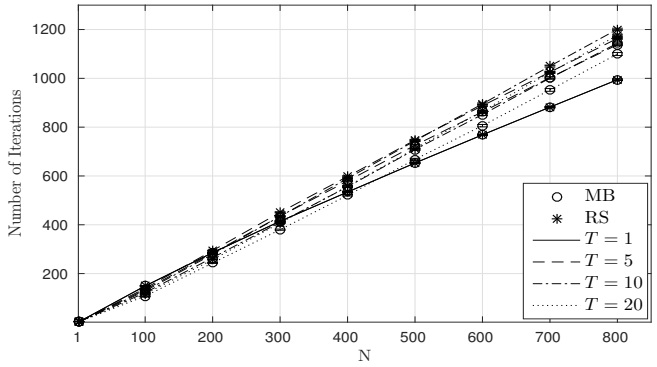

Figure 13: Number of iterations vs. the number of devices (N).

increases, and hence the devices do not want to update their strategies so often. On the contrary, as N increases the benefit of large values of T becomes smaller, because the congestion per time slots increases, and hence devices may want to update their strategies more often.

Finally, Fig. 14 shows the wall clock computing time of the MB and RS algorithms as a function of the number of devices N. The results show that the MB algorithm is faster than the RS algorithm in all cases, and the computing times are slightly super-linear due to the increasing time to compute a best response. Overall, we conclude that the proposed MB algorithm can compute efficient equilibrium allocations for periodic task offloading at low computational complexity.

7 Related Work

The scheduling of periodic tasks received significant attention for real-time systems [40, 41], but without considering communications. Similarly, the scheduling of communication resources has been considered without considering computation [42]. Most works that considered both communication and computation focused on a single device case [8,31,43–45], and thus they do not consider the sharing of communication and computing resources.

Related to our work are recent works on energy efficient computation offloading for multiple mobile users [46–48]. [46] proposed a genetic algorithm for maximizing the throughput in a partitioning problem for mobile data stream applications, while [47] considered a two-tiered cloud infrastructure with user mobility in a location-time workflow framework and proposed a heuristic for minimizing the cost of users. [48] considered minimizing mobile users' energy consumption by joint allocation of wireless and cloud resources, and proposed an iterative algorithm.

A few recent works provided a game theoretic treatment of the mobile computation

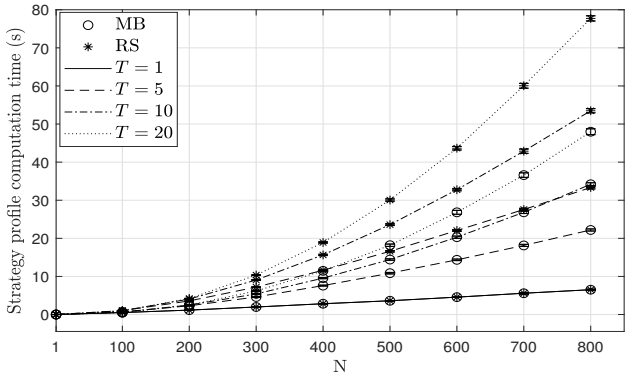

Figure 14: Strategy profile computing time vs. the number of devices (N).

offloading problem for a single time slot [7, 49–55]. Compared to [49], we characterize the structure of the computed equilibrium, prove the bound on the price of anarchy and show an example of a better reply cycle. [50] considered a two-stage game, where first each mobile user chooses the parts of its task to offload with the objective to minimize the energy consumption and the task completion time, and then the cloud allocates computational resources to the offloaded parts with the objective to maximize its profit. [51] considered a three-tier cloud architecture with stochastic task arrivals, and provided a distributed algorithm for the computing a mixed strategy equilibrium. [53] considered tasks that arrive simultaneously and a single wireless link, and showed the existence of equilibria when all mobile users have the same delay budget. [7] showed that assuming a single wireless link and link rates determined by the Shannon capacity of an interference channel, the resulting game is a potential game. [52] extended the model to multiple wireless links and showed that the game is still a potential game under the assumption that a mobile user experiences the same channel gain for all links. [55] considered multiple wireless links, equal bandwidth sharing and a non-elastic cloud, and provided a polynomial time algorithm for computing equilibria. Our work differs significantly from these works, as we consider the problem of scheduling periodic tasks over time and across heterogeneous communication resources and to the best of our knowledge, this is the first work that bridges the gap between early works on scheduling [41] and recent works on the resource allocation in computation offloading systems [7, 55].

From a game theoretical perspective the importance of our contribution is the analysis of a player-specific network congestion game for which the existence of equilibria is not known in general [33], thus the proposed algorithm and our proof of existence advance the state of the art in the study of equilibria in network congestion games.

8 Conclusion

We provided a game theoretical analysis of computation offloading in a mobile edge computing system where devices generate tasks periodically. We proved the existence of pure strategy Nash equilibria, characterized their structure and based on our constructive proof we proposed a decentralized algorithm for computing an equilibrium allocation of offloading decisions. We proved that the proposed algorithm has a bounded approximation ratio and quadratic worst case complexity. Our numerical results show that the performance in an equilibrium computed by the proposed algorithm is significantly better than in a strategy profile in which offloading decisions are not coordinated over time. An interesting open question is whether our results can be extended to devices with heterogeneous periodicities, we leave this question subject of future work.

References

[1] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "Pipeneta wireless sensor network for pipeline monitoring," in *Proc. of IPSN*, 2007, pp. 264–273.

[2] L. Xiao and Z. Wang, "Internet of things: A new application for intelligent traffic monitoring system," *Journal of networks*, vol. 6, no. 6, p. 887, 2011.

[3] M. Ayazoglu, B. Li, C. Dicle, M. Sznaier, and O. I. Camps, "Dynamic subspace-based coordinated multicamera tracking," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2462–2469.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," Sep. 2015.

[5] S. R. Group *et al.*, "The leading cloud providers continue to run away with the market," Tech. rep, Tech. Rep., 2017.

[6] A. Reznik, R. Arora, M. Cannon, L. Cominardi, W. Featherstone, R. Frazao, F. Giust, S. Kekki, A. Li, D. Sabella *et al.*, "Developing software for multi-access edge computing," *ETSI, White Paper*, no. 20, 2017.

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *Proc. of IEEE PDS*, pp. 974–983, 2015.

[8] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. of IEEE INFOCOM*, April 2013, pp. 1285–1293.

[9] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE TMC*, vol. 18, no. 1, pp. 207–220, 2018.

[10] "IoT ONE: Use Cases," https://www.iotone.com/usecases.

[11] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[12] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[13] Q. He, G. Dán, and V. Fodor, "Minimizing age of correlated information for wireless camera networks," in *INFOCOM WKSHPS*, 2018, pp. 547–552.

[14] I. Kadota, A. Sinha, and E. Modiano, "Optimizing age of information in wireless networks with throughput constraints," in *IEEE INFOCOM*, 2018, pp. 1844–1852.

[15] Z. Sheng, C. Mahapatra, V. C. Leung, M. Chen, and P. K. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," vol. 6, no. 1, pp. 114–126, 2018.

[16] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[17] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J-SAC*, vol. 34, no. 5, pp. 1757–1771, 2016.

[18] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *ACM SIGMETRICS Perf. Eval. Rev.*, vol. 29, no. 1, 2001, pp. 50–61.

[19] ——, "Pace: A new approach to dynamic voltage scaling," *IEEE Transactions on Computers*, no. 7, pp. 856–869, 2004.

[20] W. Yuan and K. Nahrstedt, "Energy-efficient CPU scheduling for multimedia applications," *ACM TOCS*, vol. 24, no. 3, pp. 292–331, 2006.

[21] A.-B. Shaibu and H. A. Muttlak, "Estimating the parameters of the normal, exponential and gamma distributions using median and extreme ranked set samples," *Statistica*, vol. 64, no. 1, pp. 75–98, 2004.

[22] T. Hoßfeld, F. Metzger, and P. E. Heegaard, "Traffic modeling for aggregated periodic iot data," in *Proc. of IEEE ICIN (Workshop)*, 2018, pp. 1–8.

[23] T. Joshi, A. Mukherjee, Y. Yoo, and D. P. Agrawal, "Airtime fairness for ieee 802.11 multirate networks," *IEEE Trans. on Mobile Computing*, vol. 7, no. 4, pp. 513–527, 2008.

[24] C. U. Saraydar, N. B. Mandayam, and D. J. Goodman, "Efficient power control via pricing in wireless data networks," *IEEE Trans. on Communications*, vol. 50, no. 2, pp. 291–303, 2002.

[25] M. Xiao, N. B. Shroff, and E. K. Chong, "A utility-based power-control scheme in wireless cellular systems," *IEEE/ACM Trans. on Networking*, vol. 11, no. 2, pp. 210–221, 2003.

[26] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[27] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer Mag.*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[28] S. Jošilo and G. Dán, "Joint allocation of computing and wireless resources to autonomous devices in mobile edge computing," in *Proc. of ACM MECOMM*, 2018, pp. 13–18.

[29] ——, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. of IEEE INFOCOM*, 2019, pp. 2467–2475.

[30] S. Jošilo and G. Dán, "Joint management of wireless and computing resources for computation offloading in mobile edge clouds," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[31] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. of IEEE INFOCOM*, March 2012, pp. 2716–2720.

[32] I. Milchtaich, "Congestion games with player-specific payoff functions," *Games and Economic Behavior*, vol. 13, no. 1, pp. 111 – 124, 1996.

[33] ——, "The equilibrium existence problem in finite network congestion games," in *Proc. of WINE*, 2006, pp. 87–98.

[34] E. Balevi and R. D. Gitlin, "Optimizing the number of fog nodes for cloud-fog-thing networks," *IEEE Access*, vol. 6, pp. 11 173–11 183, 2018.

[35] S. Sigg, P. Jakimovski, and M. Beigl, "Calculation of functions on the rf-channel for iot," in *2012 3rd IEEE International Conference on the Internet of Things*. IEEE, 2012, pp. 107–113.

[36] A. Aragon-Zavala, *Antennas and propagation for wireless communication systems*. John Wiley & Sons, 2008.

[37] E. Casilari, J. M. Cano-García, and G. Campos-Garrido, "Modeling of current consumption in 802.15. 4/zigbee sensor motes," *Sensors*, vol. 10, no. 6, pp. 5443–5468, 2010.

[38] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.

[39] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *ISCC*, 2012, pp. 59–66.

[40] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Trans. on Computers*, vol. 39, pp. 1175–1185, Sep. 1990.

[41] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-Time Syst.*, vol. 28, no. 2-3, pp. 101–155, Nov. 2004.

[42] I. H. Hou, "Packet scheduling for real-time surveillance in multihop wireless sensor networks with lossy channels," *IEEE Trans. on Wireless Comm.*, vol. 14, no. 2, pp. 1071–1079, Feb 2015.

[43] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. of ACM MobiSys*, 2010, pp. 49–62.

[44] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mob. Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb 2013.

[45] E. Hyytiä, T. Spyropoulos, and J. Ott, "Offload (only) the right jobs: Robust offloading using the Markov decision processes," in *Proc. of IEEE WoWMoM*, Jun. 2015, pp. 1–9.

[46] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Apr. 2013.

[47] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "MuSIC: Mobility-aware optimal service allocation in mobile cloud computing," in *Proc. of IEEE CLOUD*, Jun. 2013, pp. 75–82.

[48] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE T-SIPN*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[49] S. Jošilo and G. Dán, "Decentralized scheduling for offloading of periodic tasks in mobile edge computing," in *Proc of IFIP NETWORKING*, 2018.

[50] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *Proc. of IEEE SOSE*, Mar. 2013, pp. 494–502.

[51] V. Cardellini et al., "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, 2015.

[52] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[53] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy efficient offloading for competing users on a shared communication channel," in *Proc. of IEEE ICC*, Jun. 2015, pp. 3192–3197.

[54] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. of ACM MSWiM*, 2015, pp. 271–278.

[55] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of IEEE INFOCOM*, May 2017.

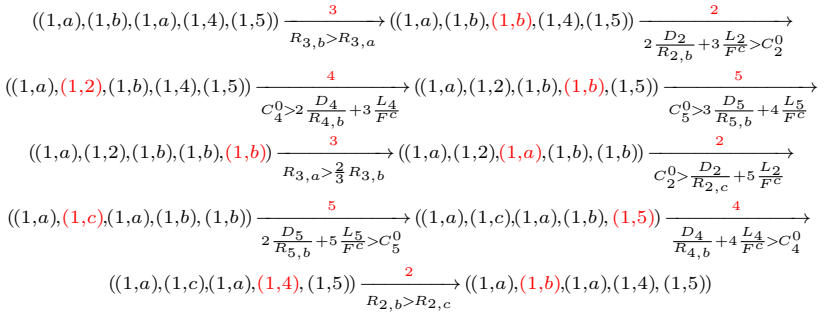

Figure 15: A cyclic improvement path in a MSCOG with $N=5$ devices, $A=3$ APs, one cloud and $T=1$. Labeled arrows between strategy profiles indicate better improvement steps. A label above the arrow indicates a player that makes the improvement step, and a label below the arrow indicates the condition under which the performed action is an improvement step.

A Appendix

A.1 Proof of Lemma 1

We prove the lemma through the following example.

Example 1. *Consider a MSCOG with* $\mathcal{N}=\{1,2,3,4,5\}$ *players,* $\mathcal{A}=\{a,b,c\}$ *APs, one edge cloud and* $\mathcal{T}=\{1\}$*. Communication and computing resources are shared equally among the devices, i.e.* $\omega_{i,(1,a)}(\boldsymbol{d})=\frac{R_{i,a}}{n_{(1,a)}(\boldsymbol{d})}$ *and* $F^c_{i,1}(\boldsymbol{d})=\frac{F^c}{n_{(1,c)}(\boldsymbol{d})}$*, respectively. Furthermore, consider that devices aim at mininizing their completion times only, i.e.,* $\gamma_i^T=1$ *and* $\gamma_i^E=0$ *for every* $i\in\mathcal{N}$*.*

Fig. 15 shows a cyclic improvement path starting from the strategy profile $((1,a),(1,b),\ (1,a),(1,4),(1,5))$, in which devices 1 and 3 offload through AP a, device 2 offloads through AP b and devices 4 and 5 perform the computation locally. The cycle shown in Fig. 15 consists of 9 improvement steps, each imposing a constraint on the system parameters. Given these constraints, an instance of the example can be formulated easily. Without loss of generality, we use the following set of parameters to illustrate the proof of the lemma: $D_i=2$ *Mb* for every $i\in\{2,3,4,5\}$, $L_2=L_3=3$ *Gcycles*, $L_4=5$ *Gcycles*, $L_5=10$ *Gcycles*, $R_{2,b}=\frac{1}{2}$ *Mb/s*, $R_{2,c}=\frac{1}{3}$ *Mb/s*, $R_{3,a}=5$ *Mb/s*, $R_{3,b}=6$ *Mb/s*, $R_{4,b}=8$ *Mb/s*, $R_{5,b}=4$ *Mb/s*, $F_2^0=\frac{3}{8}$ *GHz*, $F_4^0=\frac{16}{5}$ *GHz*, $F_5^0=\frac{50}{21}$ *GHz*, $F^c=15$ *GHz*.

A.2 Proof of Proposition 3

Consider a MSCOG with $\mathcal{T}=\{t\}$, $\mathcal{N}=\{1,2,...,N\}$, $\mathcal{A}=\{a\}$ and cloud c. Furthermore, let us consider a strategy profile $\mathbf{d}^*$ in which $d_i^*=(t,a)$ for every player

$i \in \mathcal{N}$. Given the minimum offloading cost $\bar{C}_{i,a} = \gamma_i^T(\frac{D_i}{R_{i,a}} + \frac{L_i}{F^c}) + \gamma_i^E P_{i,a}\frac{D_i}{R_{i,a}}$ that player i can achieve when it offloads alone, we can express the cost $C_i(\mathbf{d}^*)$ of player i and the system cost $C(\mathbf{d}^*)$ in strategy profile $\mathbf{d}^*$ as $C_i(\mathbf{d}^*) = N\bar{C}_{i,a}$ and $C(\mathbf{d}^*) = N\sum_{i\in\mathcal{N}}\bar{C}_{i,a}$, respectively. Next, let us assume that $C_i(\mathbf{d}^*) = N\bar{C}_{i,a} = C_i^0$ holds for every player $i \in \mathcal{N}$. It is easy to see that $\mathbf{d}^*$ is a NE of the MSCOG since there is no player $i \in \mathcal{N}$ that can decrease its cost by changing the strategy to local computing. Furthermore, it is easy to see that $\mathbf{d}^*$ is the worst case NE since all players achieve the same cost as they achieve in the case of local computing.

Now, let us assume that there is a player $k \in \mathcal{N}$ such that $C_k^0 - (N-1)\bar{C}_{k,a} \geq \sum_{i\in\mathcal{N}\setminus\{k\}}(C_i^0 - \bar{C}_{i,a})$ and $2\bar{C}_{k,a} \geq \sum_{i\in\mathcal{N}\setminus\{k\}}(C_i^0 - \bar{C}_{i,a})$ hold. From the first inequality it follows that the smallest cost saving that player k can achieve through offloading is larger than the largest cost saving that all other players can achieve together and thus in an optimal solution $\overline{\mathbf{d}}$ we have that $k \in O(\overline{\mathbf{d}})$ must hold. Furthermore, from the second inequality it follows that the smallest increase in the offloading cost of player k is higher that the largest cost saving that all other players can achieve together, and thus in an optimal solution $O(\overline{\mathbf{d}}) = \{k\}$ must hold. Therefore, the minimum system cost $C(\overline{\mathbf{d}})$ is achieved if player k is the only one offloading its computation, i.e., $C(\overline{\mathbf{d}}) = \bar{C}_{k,a} + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0$.

Next, let us find a constant ϵ for which $C(\mathbf{d}^*) = (N-\epsilon)C(\overline{\mathbf{d}})$ is satisfied, i.e,

$$N\sum\nolimits_{i\in\mathcal{N}}\bar{C}_{i,a} = (N-\epsilon)(\bar{C}_{k,a} + \sum\nolimits_{i\in\mathcal{N}\setminus\{k\}} C_i^0). \tag{24}$$

From (24) we obtain the following

$$\epsilon = \frac{N\sum_{i\in\mathcal{N}\setminus\{k\}}(C_i^0 - \bar{C}_{i,a})}{\bar{C}_{k,a} + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}. \tag{25}$$

Since according to our first assumption $\bar{C}_{i,a} = \frac{1}{N}C_i^0$ for every $i \in \mathcal{N}$, we can express ϵ as a function of local computing costs, i.e.,

$$\epsilon = \frac{(N-1)\sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}{\frac{1}{N}C_k^0 + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}, \tag{26}$$

which proves the proposition.

A.3 Proof of Corollary 3

To show the tightness of the bound, let consider a MSCOG with $\mathcal{T} = \{1\}, \mathcal{N} = \{1,2,3\}$, $\mathcal{A} = \{a\}$ and cloud c where players aim at minimizing the completion time of their tasks only, i.e., $\gamma_i^T = 1$ and $\gamma_i^E = 0$ for every $i \in \mathcal{N}$. Furthermore, let us consider the system with the following set of parameters: $F^c = 5$ *GHz*, $L_1 = 300$ *Gcycles*, $L_2 = 3$ *Gcycles*, $L_3 = 6$ *Gcycles*, $D_1 = 80$ *Mb*, $D_2 = 2$ *Mb*, $D_3 = 4$ *Mb*, $R_{1,a} = 2$ *Mb/s*,

$R_{2,a} = R_{3,a} = 5$ *Mb/s*, $F_i^0 = 1$ *GHz*, for $i \in \mathcal{N}$. Hence, we have $C_1^0 = 300$, $C_2^0 = 3$, $C_3^0 = 6$, $\bar{C}_{1,a} = 100$, $\bar{C}_{2,a} = 1$, $\bar{C}_{3,a} = 2$, $\epsilon = \frac{18}{109}$.

It is easy to verify that $\mathbf{d}^* = \big((1,a),(1,a),(1,a)\big)$ is a NE in which $C_i(\mathbf{d}^*) = 3\bar{C}_{i,a} = C_i^0$ and $C(\mathbf{d}^*) = 309$ hold. Furthermore, it is easy to see that $\overline{\mathbf{d}} = \big((1,a),(1,2),(1,3)\big)$ is an optimal solution in which $C(\overline{\mathbf{d}}) = 109$ holds. Hence, we have $\frac{C(\mathbf{d}^*)}{C(\overline{\mathbf{d}})} = N - \epsilon = \frac{309}{109} \approx 2.84$.

Paper D

Joint Management of Wireless and Computing Resources for Computation Offloading in Mobile Edge Clouds

Slađana Jošilo and György Dán

IEEE Transactions on Cloud Computing (TCC), pp. 1-14, 2019.

Joint Management of Wireless and Computing Resources for Computation Offloading in Mobile Edge Clouds

Slađana Jošilo and György Dán
School of Electrical Engineering and Computer Science
KTH, Royal Institute of Technology, Stockholm, Sweden
E-mail: {josilo, gyuri}@kth.se *

Abstract

We consider the computation offloading problem in an edge computing system in which an operator jointly manages wireless and computing resources across devices that make their offloading decisions autonomously with the objective to minimize their own completion times. We develop a game theoretical model of the interaction between the devices and an operator that can implement one of two resource allocation policies, a cost minimizing or a time fair resource allocation policy. We express the optimal cost minimizing resource allocation policy in closed form and prove the existence of Stackelberg equilibria for both resource allocation policies. We propose two efficient decentralized algorithms that devices can use for computing equilibria of offloading decisions under the cost minimizing and the time fair resource allocation policies. We establish bounds on the price of anarchy of the games played by the devices and by doing so we show that the proposed algorithms have bounded approximation ratios. Our simulation results show that the cost minimizing resource allocation policy can achieve significantly lower completion times than the time fair allocation policy. At the same time, the convergence time of the proposed algorithms is approximately linear in the number of devices, and thus they could be effectively implemented for edge computing resource management.

Index terms— edge computing, resource management, computation offloading, game theory, decentralized algorithms

1 Introduction

The evolution of wireless access and the Internet of Things are driving the development of a variety of mobile applications such as face and object recognition, mobile augmented reality, and cognitive assistance [1–3]. These emerging human-in-the-loop

*The work was partly funded by the Swedish Research Council through project 621-2014-6.

applications have delay and computational requirements that often surpass the capabilities of handheld devices [4].

A promising approach to support these emerging applications is mobile edge computing (MEC) [5]. The key idea of MEC is to move cloud resources towards the network edge so as to overcome the issue of high end-to-end transmission delays, which are inherent to computation offloading to remote centralized clouds such as Microsoft Azure or Amazon EC2 [6]. Owing to the proximity of computing resources to the end users, MEC has the potential to significantly reduce response times for individual devices by allowing them to offload the computationally intensive tasks through a wireless network to nearby edge clouds. However, computation offloading to edge clouds imposes a huge load on limited wireless and computing resources, and thus the response times might be adversely affected by the contention for MEC resources.

In order to keep response times as low as possible, it is thus essential to jointly manage the wireless and the computing resources. Nonetheless, joint resource management in a MEC system is inherently challenging for various reasons. First, it requires one to take into consideration the heterogeneity of the devices and their workloads. For example, the devices can differ in terms of their computing capabilities, the amount of data they need to offload and the delay and the computational requirements of the tasks they generate. Second, devices in MEC systems are likely to be autonomous entities, and thus they may be interested in maximizing their own performance [7,8]. Finally, MEC systems may consist of multiple heterogeneous communication and computing resources, e.g., wireless access points with different bandwidths and edge clouds with different computing capabilities. Therefore, the joint management of wireless and computing resources in MEC systems should be performed in accordance with the individual interest of the heterogeneous devices, the characteristics of their tasks and the heterogeneity of the infrastructure.

In this paper we consider devices that aim at minimizing the completion times of their own tasks, and we address the corresponding computation offloading problem by considering the interaction between an operator that jointly manages the wireless and computing resources, and devices that decide autonomously whether or not to offload the computations and in the case of offloading which of multiple heterogeneous wireless and computing resources to use. We model the problem as a multiple-leader common-follower Stackelberg game, in which devices are leaders and the operator is the follower. We consider two resource allocation policies for the operator, called the cost minimizing and the time fair resource allocation policy. We show that the resulting games played by the devices can be transformed into a weighted congestion game and into a player-specific congestion game under the cost minimizing and the time fair policy, respectively. We provide a closed form solution for the optimal cost minimizing resource allocation policy and we prove that Stackelberg equilibria exist for both policies. Based on our constructive equilibrium existence proofs, we propose two efficient decentralized algorithms that devices can use for computing offloading decisions under the cost minimizing and the time fair policy of the operator, respectively. We provide upper bounds on the price of anarchy of the games played by the devices, and thus we show that our proposed algorithms serve as approximation

algorithms for the completion time minimization problems defined for the cost minimizing and the time fair resource allocation policies. Our analytical results show that the cost minimizing policy can guarantee better performance in terms of the worst case system cost and that the time fair policy can guarantee better performance in terms of the worst case computational complexity. Finally, we use simulations to show that the completion times achieved under the cost minimizing policy are significantly lower than the completion times achieved under the time fair policy and that the complexity of computing an equilibrium is on average almost linear in the number of devices for both policies.

The rest of the paper is organized as follows. We present the system model and the problem formulation in Section 2. We present the cost minimizing resource allocation policy and prove the existence of Stackelberg equilibria in Section 3. We present the time fair resource allocation policy and prove the existence of Stackelberg equilibria in Section 4. We provide a bound on the price of anarchy of the games in Section 5 and present numerical results in Section 6. We discuss related work in Section 7 and conclude the paper in Section 8.

2 System Model

We consider an edge computing system that consists of a set $\mathcal{N}=\{1,2,\ldots,N\}$ of wireless devices (WDs), a set $\mathcal{A}=\{1,2,\ldots,A\}$ of access points (APs), a set $\mathcal{C}=\{1,2,\ldots,C\}$ of edge clouds (ECs), and an operator that manages the allocation of the wireless and computing resources. We denote by $\mathcal{A}_i\subseteq\mathcal{A}$ the set of APs through which WD $i\in\mathcal{N}$ can communicate with the ECs. For ease of reference, the key notations used in the paper are summarized in Table 1.

Each WD $i\in\mathcal{N}$ generates computationally intensive tasks, which can be characterized by two parameters, the size D_i of the input data and the expected number L_i of CPU cycles required to perform the computation (e.g., in bits). As shown by recent works, the number X of CPU cycles required per data bit can be approximated by a Gamma distribution [9, 10]. Hence, based on the empirical mean $E[X]$, the relationship between L_i and D_i can be expressed as $L_i=D_iE[X]$. To make the analysis tractable, we make the common assumption that the set of WDs is known (e.g., through signaling) [11] [12].

Each WD $i\in\mathcal{N}$ can decide whether to perform the computation locally or to offload the computation to one of the ECs $c\in\mathcal{C}$ through one of the APs $a\in\mathcal{A}_i$. Thus, the set of feasible decisions for WD i is $\mathfrak{D}_i=\{i\}\cup\{(a,c)|a\in\mathcal{A}_i,c\in\mathcal{C}\}$, where i corresponds to local computing and (a,c) to offloading through AP a to EC c. We refer to decision $d_i\in\mathfrak{D}_i$ of WD i as its strategy, and we refer to the collection $\mathbf{d}=(d_i)_{i\in\mathcal{N}}$ as a strategy profile, i.e., $\mathbf{d}\in\times_{i\in\mathcal{N}}\mathfrak{D}_i=\mathfrak{D}$. For a strategy profile $\mathbf{d}\in\mathfrak{D}$, we define the set $O_a(\mathbf{d})\triangleq\{i|d_i=(a,\cdot)\}$ of WDs that offload their tasks through AP a and we denote by $n_a(\mathbf{d})\triangleq|O_a(\mathbf{d})|$ the number of WDs that offload their tasks through AP a. Similarly, we define the set $O_c(\mathbf{d})\triangleq\{i|d_i=(\cdot,c)\}$ of WDs that offload their tasks to EC c and we denote by $n_c(\mathbf{d})\triangleq|O_c(\mathbf{d})|$ the number of WDs that offload their tasks to EC c. Finally, we define the set $O_{(a,c)}(\mathbf{d})\triangleq O_a(\mathbf{d})\cap O_c(\mathbf{d})$ of

Table 1: Summary of key notations

Notation	Description
$\mathcal{N}$	Set of N WDs
$\mathcal{A}$	Set of A APs
$\mathcal{A}_i$	Set of APs available for offloading to WD i
$\mathcal{C}$	Set of C ECs
$\mathcal{P}_c$	Operator's computing resource allocation policy
$\mathcal{P}_r$	Operator's rate allocation policy
D_i	Mean size of the input data for WD i
L_i	Mean task complexity for WD i
F_i^l	Computational capability of WD i
C_i^l	Local computing cost for WD i
$R_{i,a}$	Uplink PHY rate of WD i towards AP a
$u_{i,a}$	Uplink access provisioning coefficient, $(i,a) \in \mathcal{N} \times \mathcal{A}_i$
F^c	Computing capability of EC c
$p_{i,c}$	Computing power provisioning coefficient, $(i,c) \in \mathcal{N} \times \mathcal{C}$
$\mathfrak{D}_i$	Set of feasible decisions for WD i
d_i	Decision of WD i, $d_i \in \mathfrak{D}_i$
$\mathbf{d}$	Strategy profile
$O_a(\mathbf{d})$	Set of $n_a(\mathbf{d})$ WDs offloading through AP a in $\mathbf{d}$
$O_c(\mathbf{d})$	Set of $n_c(\mathbf{d})$ WDs offloading to EC c in $\mathbf{d}$
$O_{(a,c)}(\mathbf{d})$	Set of WDs offloading through AP a to EC c in $\mathbf{d}$
$O(\mathbf{d})$	Set of all WDs that offload their tasks in $\mathbf{d}$
$\omega_{i,a}(\mathbf{d}, \mathbf{u}_a)$	Uplink rate of WD $i \in O_a(\mathbf{d})$ for $\mathbf{u}_a$
$F_i^c(\mathbf{d}, \mathbf{p}_c)$	Computing capability of WD $i \in O_c(\mathbf{d})$ for $\mathbf{p}_c$
$C_{i,a}^c(\mathbf{d}, \mathbf{u}_a, \mathbf{p}_c)$	Offloading cost of WD i, $d_i = (a,c)$ for $\mathbf{u}_a$ and $\mathbf{p}_c$ in $\mathbf{d}$
$C_i(\mathbf{d}, \mathbf{u}, \mathbf{p})$	Cost of WD i for $\mathbf{u}$ and $\mathbf{p}$ in $\mathbf{d}$
$C(\mathbf{d}, \mathbf{u}, \mathbf{p})$	Total cost in the system for $\mathbf{u}$ and $\mathbf{p}$ in $\mathbf{d}$

WDs that offload their tasks through AP a to EC c and the set $O(\mathbf{d}) \triangleq \cup_{c \in \mathcal{C}} O_c(\mathbf{d})$ of all WDs that offload their tasks.

Fig. 1 shows an example of a MEC system that consists of $N = 5$ WDs, $C = 2$ ECs and $A = 3$ APs. WD 1 performs the computation locally, WDs 2 and 3 offload their tasks to EC c_1 through AP a, WDs 4 and 5 offload their tasks to EC c_2 through APs b and c, respectively. In what follows we discuss our models of computing and wireless resource management.

2.1 Computing Resource Management

A WD that chooses local computing performs its task using its local computing resources. We denote by F_i^l the computational capability of WD $i \in \mathcal{N}$ (e.g., CPU cycles/second).

A WD that chooses offloading has to transmit the data through an AP a, after

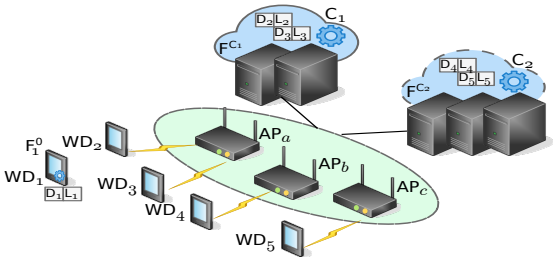

Figure 1: Example of an edge computing system with $N = 5$ WDs, $C = 2$ ECs and $A = 3$ APs. Transmission rates and cloud computing power may be actively managed by the operator.

which the task is performed in an EC c. We denote by F^c the computing capability of EC c. We consider that the computing capability allocated to WDs $i \in O_c(\mathbf{d})$ is determined by the operator's *computing resource allocation policy* $\mathcal{P}_c : \mathfrak{D} \to \mathbb{R}_{\geq 0}^{|\mathcal{C}|x|\mathcal{N}|}$. The policy sets for every strategy profile $\mathbf{d} \in \mathfrak{D}$ the computing power provisioning coefficients $(p_{i,c})_{i\in\mathcal{N},c\in\mathcal{C}}$, akin to the weight of a job in generalized processor sharing (GPS). Using the shorthand notation $\mathbf{p}_c = (p_{i,c})_{i\in\mathcal{N}}$, we can express the computing capability allocated to WD i by EC c as

$$F_i^c(\mathbf{d}, \mathbf{p}_c) = F^c \frac{p_{i,c}}{\sum_{j\in O_c(\mathbf{d})} p_{j,c}}. \tag{1}$$

Observe that for a policy that sets $p_{i,c} = 1$, $\forall i \in O_c(\mathbf{d})$,$\forall \mathbf{d} \in \mathfrak{D}$, the computing power is shared equally. While GPS is an ideal scheduler, several process schedulers exist to approximate it in practice, e.g., DWRR [13].

2.2 Wireless Resource Management

The wireless medium of AP a is shared by the WDs that choose to offload through AP a. We denote by $R_{i,a}$ the achievable PHY rate of WD i through AP a, which is determined by the physical characteristics of the wireless medium, distance, etc. The actual rate at which WD i can offload its data through AP a is determined by the operator's *rate allocation policy* $\mathcal{P}_r : \mathfrak{D} \to \mathbb{R}_{\geq 0}^{|\mathcal{A}|x|\mathcal{N}|}$. The policy sets for every strategy profile the uplink access provisioning coefficients $(u_{i,a})_{i\in\mathcal{N},a\in\mathcal{A}}$, akin to the weight of a flow in GPS. Using the shorthand notation $\mathbf{u}_a = (u_{i,a})_{i\in\mathcal{N}}$, we can express the uplink rate assigned to WD i at AP a as

$$\omega_{i,a}(\mathbf{d}, \mathbf{u}_a) = R_{i,a} \frac{u_{i,a}}{\sum_{j\in O_a(\mathbf{d})} u_{j,a}}. \tag{2}$$

Observe that for a policy that sets $u_{i,a}(\mathbf{d}) = 1, \forall i \in O_a(\mathbf{d})$ we obtain the model that describes the time-fair throughput sharing mechanisms in TDMA and OFDMA

based MAC protocols [14].

2.3 Cost Model

We define the cost of a WD as the completion time of its task. In what follows we introduce our cost model in the case of computation offloading and in the case of local computing.

Computation offloading: In the case of computation offloading the completion time of WD i's task consists of two parts. The first part is the time needed to transmit D_i amount of data, and the second part is the time needed to perform L_i CPU cycles at the cloud server. Thus, if in strategy profile $\mathbf{d}$ WD i offloads to EC $c \in \mathcal{C}$ through AP $a \in \mathcal{A}_i$ then its cost can be expressed as

$$C^c_{i,a}(\mathbf{d}, \mathbf{u}_a, \mathbf{p}_c) = D_i/\omega_{i,a}(\mathbf{d}, \mathbf{u}_a) + L_i/F^c_i(\mathbf{d}, \mathbf{p}_c). \tag{3}$$

In (3) we made the common assumption that the time needed to transmit the results from the cloud to the device can be neglected [11, 15–17], as for typical applications (e.g., face and object recognition), the size of the result of the computation is much smaller than D_i.

Local computing: In the case of local computing the completion time of WD i's task is determined by the number L_i of CPU cycles pertaining to the task and by the computing capability F^l_i. Thus, the local computing cost can be expressed as

$$C^l_i = L_i/F^l_i. \tag{4}$$

Total cost: To define the total cost, we first define the shorthand notation $\mathbf{u} \triangleq (\mathbf{u}_a)_{a\in\mathcal{A}}$ and $\mathbf{p} \triangleq (\mathbf{p}_c)_{c\in\mathcal{C}}$, and express the cost of WD i

$$C_i(\mathbf{d}, \mathbf{u}, \mathbf{p}) = \sum_{(a,c)\in\mathcal{A}_i\times\mathcal{C}} I_{d_i,(a,c)} C^c_{i,a}(\mathbf{d}, \mathbf{u}_a, \mathbf{p}_c) + I_{d_i,i} C^l_i, \tag{5}$$

where $I_{d_i,r} = 1$ if $d_i = r$ and $I_{d_i,r} = 0$ otherwise. Finally, we define the system cost $C(\mathbf{d}, \mathbf{u}, \mathbf{p})$ as

$$C(\mathbf{d}, \mathbf{u}, \mathbf{p}) = \sum_{i\in\mathcal{N}} \sum_{(a,c)\in\mathcal{A}_i\times\mathcal{C}} I_{d_i,(a,c)} C^c_{i,a}(\mathbf{d}, \mathbf{u}_a, \mathbf{p}_c) + \sum_{i\in\mathcal{N}} I_{d_i,i} C^l_i. \tag{6}$$

2.4 Operator Policies and Problem Formulation

We consider that in the edge computing system each WD is allowed to make an offloading decision so as to minimize its own cost. On the one hand, this assumption is motivated by the potential autonomy of WDs in edge computing systems [7, 8]. On the other hand, the obtained decentralized algorithms can serve as a good approximation for the optimal solution. Nonetheless, the decisions of the WDs interact with the computing resource and rate allocation policies of the operator, and hence we model the problem as a multiple-leader common-follower Stackelberg game,

in which WDs are leaders and the operator is the follower. We consider two variants of the game, which differ in the set of operator policies. In the first game the set of feasible decisions for the operator is $\mathfrak{A}_c = \{(\mathbf{u},\mathbf{p}) | \mathbf{u} \in \mathbb{R}_{\geq 0}^{|\mathcal{A}|x|\mathcal{N}|}, \mathbf{p} \in \mathbb{R}_{\geq 0}^{|\mathcal{C}|x|\mathcal{N}|}\}$; we refer to this as the *cost minimizing* (CM) operator. In the second game the set of feasible decisions for the operator is $\mathfrak{A}_t = \{(\mathbf{u},\mathbf{p}) | u_{i,a} = 1, p_{i,c} = 1, \forall i \in \mathcal{N}, a \in \mathcal{A}, c \in \mathcal{C}\}$; we refer to this as the *time fair* (TF) operator.

Given a strategy profile $\mathbf{d}$ chosen by the WDs, the objective of the operator is to minimize the system cost by jointly optimizing the allocation of wireless and computing resources. It does so by computing a best response $(\mathbf{u}^*, \mathbf{p}^*) \in \mathfrak{A}_o$, $o \in \{c, t\}$ to $\mathbf{d}$ through solving

$$\min_{(\mathbf{u},\mathbf{p}) \in \mathfrak{A}_o} C(\mathbf{d}, \mathbf{u}, \mathbf{p}). \tag{7}$$

We denote by $(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ the optimal policy of the CM operator, i.e., the collection of best responses of the CM operator for every $\mathbf{d} \in \mathfrak{D}$, and we denote by $(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ the optimal policy of the TF operator, i.e., the collection of best responses of the TF operator for every $\mathbf{d} \in \mathfrak{D}$.

The objective of every WD is to minimize its own completion time (5), given the announced allocation policy $(\mathcal{P}_r^*, \mathcal{P}_c^*)$ of the operator, through solving

$$\min_{d_i \in \mathfrak{D}_i} C_i(d_i, d_{-i}, \mathcal{P}_r^*(d_i, d_{-i}), \mathcal{P}_c^*(d_i, d_{-i})), \tag{8}$$

where we use d_{-i} to denote the strategies of all WDs except WD i. We refer to the game played between the WDs and the CM operator as the *cost minimizing computation offloading game* (CM-COG) and to the game played between the WDs and the TF operator as the *time fair computation offloading game* (TF-COG).

In this paper we address three fundamental questions for these games. First, we address whether there is a combination of computation offloading strategy profile and allocation policy from which neither the WDs nor the operator have an incentive to deviate, i.e., a subgame perfect equilibrium of the Stackelberg game.

Definition 1 (SPE)**.** Let $(\mathcal{P}_r^*, \mathcal{P}_c^*)$ be a solution of (7), and d_i^* be a solution of (8). Then the point $(\mathbf{d}^*, \mathcal{P}_r^*, \mathcal{P}_c^*)$ is a subgame perfect equilibrium (SPE) of the game $\Gamma \in \{\text{CM-COG, TF-COG}\}$ if for any feasible $(\mathbf{d}, \mathcal{P}_r, \mathcal{P}_c)$ point the following holds

$$C(\mathbf{d}^*, \mathcal{P}_r^*, \mathcal{P}_c^*) \leq C(\mathbf{d}^*, \mathcal{P}_r, \mathcal{P}_c),$$

$$C_i(d_i^*, d_{-i}^*, \mathcal{P}_r^*, \mathcal{P}_c^*) \leq C_i(d_i, d_{-i}^*, \mathcal{P}_r^*, \mathcal{P}_c^*), \forall d_i \in \mathfrak{D}_i, \forall i \in \mathcal{N}.$$

If the game $\Gamma \in \{\text{CM-COG}, \text{TF-COG}\}$ admits an SPE, the second question is whether an SPE can be computed efficiently. Third, we address whether the system cost in an SPE is efficient compared to a centrally optimized system. Before we answer these questions we recall the following definition from game theory.

Definition 2. (**Pure NE and Best reply (BR)**) A pure strategy Nash equilibrium (NE) is a strategy profile $\mathbf{d}^*$ in which all players play their best replies to each others'

strategies, that is,

$$C_i(d_i^*, d_{-i}^*) \leq C_i(d_i, d_{-i}^*), \forall d_i \in \mathfrak{D}_i, \forall i \in \mathcal{N}.$$

Given a strategy profile $d = (d_i, d_{-i})$, a better reply of WD i is a strategy d_i' such that $C_i(d_i', d_{-i}) < C_i(d_i, d_{-i})$, and a best reply of WD i is a better reply d_i^* such that $C_i(d_i^*, d_{-i}) \leq C_i(d_i, d_{-i}), \forall d_i \in \mathfrak{D}_i$.

3 Equilibria Under the Cost Minimizing Operator

We start the analysis by considering problem (7) solved by the CM operator, i.e.,

$$\min_{(\mathbf{u},\mathbf{p}) \in \mathfrak{A}_c} C(\mathbf{d}, \mathbf{u}, \mathbf{p}), \tag{9}$$

followed by problem (8) solved by the WDs.

3.1 Optimal Resource Allocation Policy of the CM Operator

Recall that an optimal resource allocation policy is essentially a collection of best responses $(\mathbf{u}^*, \mathbf{p}^*) \in \mathfrak{A}_c$ of the CM operator to the strategy profiles $\mathbf{d} \in \mathfrak{D}$ played by the WDs. In what follows we show that a best response of the CM operator to a strategy profile $\mathbf{d}$ is unique up to a scale factor and can be expressed in closed form.

Theorem 1. *Let $\boldsymbol{d}$ be a strategy profile played by the WDs. The optimal allocation policy $(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ of the CM operator assigns to $\boldsymbol{d}$ the uplink access provisioning and computing power provisioning coefficients*

$$u_{i,a}^* = \frac{\sqrt{D_i / R_{i,a}}}{\sum_{j \in O_a(\boldsymbol{d})} \sqrt{D_j / R_{j,a}}}, \forall a \in \mathcal{A}, \forall i \in O_a(\boldsymbol{d}), \tag{10}$$

and

$$p_{i,c}^* = \frac{\sqrt{L_i / F^c}}{\sum_{j \in O_c(\boldsymbol{d})} \sqrt{L_j / F^c}}, \forall c \in \mathcal{C}, \forall i \in O_c(\boldsymbol{d}). \tag{11}$$

Proof. The proof is given in Appendix A.1. □

It is important to note that following the optimal resource allocation policy, the CM operator allocates resources to the WDs depending on the characteristics of their tasks (i.e., D_i and L_i). Furthermore, the resource allocation policy of the operator can be made known a priori to the WDs, which allows us to analyze the computation offloading problem of the WDs.

3.2 Computing Equilibrium Offloading Decisions

Observe that for an arbitrary resource allocation policy $(\mathcal{P}_r, \mathcal{P}_c)$ the interaction between the WDs can be modeled by a player-specific weighted congestion game $\Gamma(\mathcal{P}_r, \mathcal{P}_c) = < \mathcal{N}, (\mathfrak{D}_i)_{i\in\mathcal{N}}, (C_i)_{i\in\mathcal{N}} >$, as (5) is both a function of the WDs' parameters and of the resource provisioning coefficients. Unfortunately, for this class of games general equilibrium existence results are not available. In what follows we show that under the optimal resource allocation policy of the CM operator the game can be transformed into a weighted congestion game.

Theorem 2. *Consider that the CM operator uses the optimal policy* $(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$*, i.e.,* $\boldsymbol{u}^*$ *and* $\boldsymbol{p}^*$ *are the collections of the optimal provisioning coefficients given by (10) and (11), respectively. Then, the strategic interaction of the WDs can be modeled as a congestion game with resource-dependent weights* $w_{i,r}, \forall (i,r) \in \mathcal{N} \times \{\mathcal{A}_i \cup \mathcal{C}\}$*, in which the cost of WD* i *is given by*

$$\bar{C}_i(\boldsymbol{d}) = \sum_{(a,c)\in\mathcal{A}_i\times\mathcal{C}} I_{d_i,(a,c)} \Big(w_{i,a} w_a(\boldsymbol{d}) + w_{i,c} w_c(\boldsymbol{d}) \Big) + I_{d_i,i} C_i^l, \tag{12}$$

where $w_r(\boldsymbol{d}) = \sum_{j\in O_r(\boldsymbol{d})} w_{j,r}$.

Proof. Let us first substitute (10) and (11) into (3) in order to obtain the offloading cost of WD i through AP a to EC c under the optimal resource allocation policy $(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$,

$$\bar{C^c}_{i,a}(\mathbf{d}) = \sqrt{\frac{D_i}{R_{i,a}}} \sum_{j\in O_a(\mathbf{d})} \sqrt{\frac{D_j}{R_{j,a}}} + \sqrt{\frac{L_i}{F^c}} \sum_{j\in O_c(\mathbf{d})} \sqrt{\frac{L_j}{F^c}}. \tag{13}$$

Second, let us define the weight $w_{i,a} \triangleq \sqrt{D_i/R_{i,a}}$ for each tuple $(i,a) \in \mathcal{N} \times \mathcal{A}_i$ and the weight $w_{i,c} \triangleq \sqrt{L_i/F^c}$ for each tuple $(i,c) \in \mathcal{N} \times \mathcal{C}$. Observe that the offloading cost (13) in strategy profile $\mathbf{d}$ depends on the total weight $w_a(\mathbf{d}) = \sum_{j\in O_a(\mathbf{d})} w_{j,a}$ associated to AP a and on the total weight $w_c(\mathbf{d}) = \sum_{j\in O_c(\mathbf{d})} w_{j,c}$ associated to EC c. Thus, the interaction between the WDs can be modeled as a *weighted congestion game with resource-dependent weights*. This proves the theorem. □

We refer to the resulting strategic game as $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*}) = < \mathcal{N}, (\mathfrak{D}_i)_{i\in\mathcal{N}}, (\bar{C}_i)_{i\in\mathcal{N}} >$, in which the players are WDs with the objective to minimize their costs given by (12). Observe that the game $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ is the CM-COG expressed in strategic form and thus if $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ has a NE then the CM-COG has an SPE. Hence, in what follows we focus on the existence and computability of pure NE for $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$.

Before we formulate our next result let us recall the definition of an exact potential function from [18].

Definition 3. A function $\Phi : \times_i(\mathfrak{D}_i) \to \mathbb{R}$ is an exact potential for a finite strategic game $\Gamma = < \mathcal{N}, (\mathfrak{D}_i)_i, (\bar{C}_i)_i >$ if for an arbitrary strategy profile (d_i, d_{-i}) and for any

better reply d_i' the following holds

$$\bar{C}i(d_i', d_{-i}) - \bar{C}_i(d_i, d_{-i}) = \Phi(d_i', d_{-i}) - \Phi(d_i, d_{-i}). \tag{14}$$

Given an arbitrary ordering of WDs, let us introduce the following shorthand notation,

$$w_a^{\leq i}(\mathbf{d}) = \sum_{\{j \in O_a(\mathbf{d}) | j \leq i\}} w_{j,a}, \quad w_a^{>i}(\mathbf{d}) = \sum_{\{j \in O_a(\mathbf{d}) | j > i\}} w_{j,a},$$

and

$$w_c^{\leq i}(\mathbf{d}) = \sum_{\{j \in O_c(\mathbf{d}) | j \leq i\}} w_{j,c}, \quad w_c^{>i}(\mathbf{d}) = \sum_{\{j \in O_c(\mathbf{d}) | j > i\}} w_{j,c}.$$

Theorem 3. *The game* $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ *has the exact potential function*

$$\Phi(\boldsymbol{d}) = \sum_{i \in \mathcal{N}} \left(\sum_{a \in \mathcal{A}} \Phi_{i,a}(\boldsymbol{d}) + \sum_{c \in \mathcal{C}} \Phi_{i,c}(\boldsymbol{d}) + \Phi_{i,i}(\boldsymbol{d}) \right), \tag{15}$$

where $\Phi_{i,a}(\boldsymbol{d}) = I_{d_i,(a,\cdot)} w_{i,a} w_a^{\leq i}(\boldsymbol{d})$, $\Phi_{i,c}(\boldsymbol{d}) = I_{d_i,(\cdot,c)} w_{i,c} w_c^{\leq i}(\boldsymbol{d})$, *and* $\Phi_{i,i}(\boldsymbol{d}) = I_{d_i,i} C_i^l$.

Proof. Let us define function $\Phi_i(\mathbf{d}) = \sum_{a \in \mathcal{A}} \Phi_{i,a}(\mathbf{d}) + \sum_{c \in \mathcal{C}} \Phi_{i,c}(\mathbf{d}) + \Phi_{i,i}(\mathbf{d})$, and rewrite $\Phi(\mathbf{d}) = \sum_{i \in \mathcal{N}} \Phi_i(\mathbf{d})$. To prove that $\Phi(\mathbf{d})$ is an exact potential function, let us consider strategy profiles $\mathbf{d}$ and $\mathbf{d}'$ such that $\mathbf{d} = (d_k, d_{-k})$ and $\mathbf{d}' = (d_k', d_{-k})$, and consider the following two cases.

Case 1: Changing offloading strategy: We start with considering the case when WD k offloads its task in both strategy profiles $\mathbf{d}$ and $\mathbf{d}'$. Let us denote by $d_k = (a, c)$ and $d_k' = (a', c')$ the offloading decisions of WD k in $\mathbf{d}$ and $\mathbf{d}'$, respectively. If $a \neq a'$ and $c \neq c'$ then the difference between the cost of WD k in $\mathbf{d}$ and that in $\mathbf{d}'$ is given by

$$\bar{C}_k(\mathbf{d}) - \bar{C}_k(\mathbf{d}') = w_{k,a} w_a(\mathbf{d}) + w_{k,c} w_c(\mathbf{d}) - w_{k,a'} w_{a'}(\mathbf{d}) - w_{k,c'} w_{c'}(\mathbf{d}).$$

To compute the change of the potential, observe that $\Phi_{i,i}(\mathbf{d}) = \Phi_{i,i}(\mathbf{d}')$ for all WDs $i \in \mathcal{N}$, since the set of WDs that perform the computation locally is the same in $\mathbf{d}$ and $\mathbf{d}'$. We also have that $\Phi_{i,r}(\mathbf{d}) = \Phi_{i,r}(\mathbf{d}')$ for every resource $r \in \mathcal{A} \cup \mathcal{C} \setminus \{a, a', c, c'\}$ since $O_r(\mathbf{d}) = O_r(\mathbf{d}')$. Furthermore, we observe that $\Phi_i(\mathbf{d}) = \Phi_i(\mathbf{d}')$ for all WDs $i < k$. For WDs $i > k$ that offload their tasks through APs a and a' we have that $\Phi_{i,a}(\mathbf{d}) - \Phi_{i,a}(\mathbf{d}') = w_{i,a} w_{k,a}$ and $\Phi_{i,a'}(\mathbf{d}) - \Phi_{i,a'}(\mathbf{d}') = -w_{i,a'} w_{k,a'}$, respectively. Similarly, for WDs $i > k$ that offload their tasks to ECs c and c' we have that $\Phi_{i,c}(\mathbf{d}) - \Phi_{i,c}(\mathbf{d}') = w_{i,c} w_{k,c}$ and $\Phi_{i,c'}(\mathbf{d}) - \Phi_{i,c'}(\mathbf{d}') = -w_{i,c'} w_{k,c'}$, respectively. For WD k we have the following

$$\Phi_k(\mathbf{d}) - \Phi_k(\mathbf{d}') = w_{k,a} w_a^{\leq k}(\mathbf{d}) + w_{k,c} w_c^{\leq k}(\mathbf{d}) - w_{k,a'} w_{a'}^{\leq k}(\mathbf{d}) - w_{k,c'} w_{c'}^{\leq k}(\mathbf{d}).$$

We hence obtain the equality

$$\Phi(\mathbf{d})-\Phi(\mathbf{d}')=w_{k,a}w_a^{>k}(\mathbf{d})+w_{k,c}w_c^{>k}(\mathbf{d})-w_{k,a'}w_{a'}^{>k}(\mathbf{d})-$$
$$w_{k,c'}w_{c'}^{>k}(\mathbf{d})+w_{k,a}w_a^{\leq k}(\mathbf{d})+w_{k,c}w_c^{\leq k}(\mathbf{d})-w_{k,a'}w_{a'}^{\leq k}(\mathbf{d})-$$
$$w_{k,c'}w_{c'}^{\leq k}(\mathbf{d}) = w_{k,a}w_a(\mathbf{d})+w_{k,c}w_c(\mathbf{d})-w_{k,a'}w_{a'}(\mathbf{d})-$$
$$w_{k,c'}w_{c'}(\mathbf{d}) = \bar{C}_k(\mathbf{d})-\bar{C}_k(\mathbf{d}').$$

Similarly, we can show that $\Phi(\mathbf{d})-\Phi(\mathbf{d}')=\bar{C}_k(\mathbf{d})-\bar{C}_k(\mathbf{d}')$ if WD k changes only the AP, i.e., if $d_k=(a,c)$ and $d'_k=(a',c)$, $a\neq a'$ or if WD k changes only the EC, i.e., if $d_k=(a,c)$ and $d'_k=(a,c')$, $c\neq c'$.

Case 2: Changing between offloading and local computing: We continue with considering the case when WD k offloads its task in one of the strategy profiles $\mathbf{d}$ and $\mathbf{d}'$ and it performs the computation locally in the other strategy profile. Let us first consider that WD k offloads its task in strategy profile $\mathbf{d}$, and denote by $d_k=(a,c)$ its offloading decision, and that WD k performs the computation locally in strategy profile $\mathbf{d}'$, i.e., $d'_k=0$. Then the difference between the cost of WD k in $\mathbf{d}$ and that in $\mathbf{d}'$ is given by

$$\bar{C}_k(\mathbf{d})-\bar{C}_k(\mathbf{d}')=w_{k,a}w_a(\mathbf{d})+w_{k,c}w_c(\mathbf{d})-C_k^l.$$

For the potential, we know that $\Phi_{i,i}(\mathbf{d})=\Phi_{i,i}(\mathbf{d}')$ for all WDs $i\in\mathcal{N}\setminus\{k\}$ and we also have that $\Phi_{i,r}(\mathbf{d})=\Phi_{i,r}(\mathbf{d}')$ for every resource $r\in\mathcal{A}\cup\mathcal{C}\setminus\{a,c\}$. Furthermore, we observe that $\Phi_i(\mathbf{d})=\Phi_i(\mathbf{d}')$ for all $i<k$. For WDs $i>k$ that offload their tasks through AP a we have that $\Phi_{i,a}(\mathbf{d})-\Phi_{i,a}(\mathbf{d}')=w_{i,a}w_{k,a}$. Similarily, for WDs $i>k$ that offload their tasks to EC c we have that $\Phi_{i,c}(\mathbf{d})-\Phi_{i,c}(\mathbf{d}')=w_{i,c}w_{k,c}$. Finally, for WD k we have

$$\Phi_k(\mathbf{d})-\Phi_k(\mathbf{d}')=w_{k,a}w_a^{\leq k}(\mathbf{d})+w_{k,c}w_c^{\leq k}(\mathbf{d})-C_k^l.$$

We hence obtain the equality

$$\Phi(\mathbf{d})-\Phi(\mathbf{d}')=w_{k,a}w_a^{>k}(\mathbf{d})+w_{k,c}w_c^{>k}(\mathbf{d})+w_{k,a}w_a^{\leq k}(\mathbf{d})$$
$$+w_{k,c}w_c^{\leq k}(\mathbf{d})-C_k^l=w_{k,a}w_a(\mathbf{d})+w_{k,c}w_c(\mathbf{d})-C_k^l=$$
$$\bar{C}_k(\mathbf{d})-\bar{C}_k(\mathbf{d}').$$

Similarily, we can show that $\Phi(\mathbf{d})-\Phi(\mathbf{d}')=\bar{C}_k(\mathbf{d})-\bar{C}_k(\mathbf{d}')$ if WD k changes its strategy from local computing in $\mathbf{d}$ to offloading to EC c through AP a in $\mathbf{d}'$, i.e., if $d_k=0$ and $d'_k=(a,c)$, which proves the theorem. □

The existence of an exact potential function implies that $\Gamma(\mathcal{P}_r^{c,*},\mathcal{P}_c^{c,*})$ has a pure NE [18]. We can thus formulate the following result.

Corollary 1. *The game $\Gamma(\mathcal{P}_r^{c,*},\mathcal{P}_c^{c,*})$ has a pure strategy NE $\boldsymbol{d}^*$. Hence, an SPE $(\boldsymbol{d}^*,\mathcal{P}_r^{c,*},\mathcal{P}_c^{c,*})$ for the CM-COG exists.*

AU(**d**)

while $\exists$ *WD* j *s.t.* $d_j \neq \arg\min_{d'_j \in \mathfrak{D}_j} \bar{C}_j(d'_j, d_{-j})$
  $d_j^* = \arg\min_{d'_j \in \mathfrak{D}_j} \bar{C}_j(d'_j, d_{-j})$
  $\mathbf{d} = (d_j^*, d_{-j})$
end

Figure 2: Pseudo code of the *AsynchronousUpdates* (AU) algorithm.

There are a variety of algorithms that are known to converge to an equilibrium for exact potential games, such as fictitious play [18], joint strategy fictitious play [19], and the best and better reply dynamics [18]. Nonetheless, they have exponential worst case complexity in general [20, 21]. Thus, the second fundamental question we address in this paper is whether a NE of $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ (and thus an SPE of the CM-COG) can be computed efficiently.

In what follows we propose the *ImproveLocalComputing* (ILC) algorithm to address this important question. The ILC algorithm starts from a strategy profile in which all WDs perform computation locally. Let us first denote by $\mathcal{N}'$ the set of WDs that have never changed their strategy from local computing to computation offloading (note that at the beginning $\mathcal{N}' = \mathcal{N}$). The ILC algorithm consists of two phases that are executed alternatingly. In the first phase, among all WDs $i \in \mathcal{N}'$ that can decrease their cost by starting to offload, a WD with the maximum task complexity L_i is allowed to perform a best reply. In the second phase, which we refer to as the update phase, WDs $i \in \mathcal{N} \setminus \mathcal{N}'$ are allowed to update their best replies according to the AU algorithm shown in Fig. 2.

In what follows we show that by letting WDs to start to offload in non-increasing order of their task complexities, the ILC algorithm reduces the number of iterations compared to the best reply dynamic that lets WDs to start using cloud resources in an arbitrary order.

Proposition 1. *Let us consider a strategy profile* $\boldsymbol{d}$ *in which all WDs* $j \in \mathcal{N} \setminus \mathcal{N}'$ *perform best replies and let us assume that there is a WD* $i \in \mathcal{N}'$ *that can decrease its cost by starting to offload to one of the ECs. Then upon WD* i *performs its best reply, WDs* $j \in O(\boldsymbol{d})$ *will not have an incentive to change between ECs.*

Proof. Let us assume that a best reply of WD $i \in \mathcal{N}'$ is offloading to an EC c, i.e., for any EC $c' \in \mathcal{C} \setminus \{c\}$ the following holds

$$\Big(w_c(\mathbf{d}) + w_{i,c}\Big) w_{i,c} < \Big(w_{c'}(\mathbf{d}) + w_{i,c'}\Big) w_{i,c'}. \tag{16}$$

Let us next assume that upon WD i performs its best reply, a WD $j \in O_c(\mathbf{d})$ can decrease its offloading cost by changing its strategy from $(\cdot, c)$ to $(\cdot, c')$, i.e., that the

following holds

$$\Big(w_c(\mathbf{d})+w_{i,c}\Big)w_{j,c} > \Big(w_{c'}(\mathbf{d})+w_{j,c'}\Big)w_{j,c'}. \tag{17}$$

In order to have (16) and (17) satisfied $\sqrt{L_i} > \sqrt{L_j}$ must hold, which contradicts the fact that the ILC algorithm allows WDs $i \in \mathcal{N}'$ to start to offload in non-increasing order of their task complexities L_i. This proves the result. □

Note that WDs can change between ECs only if the congestion in an EC decreases, i.e., if one of the WDs changes its strategy from offloading to local computing. This is, however, rarely the case, and as we show later, the number of iterations needed to compute an equilibrium allocation of offloading decisions using the ILC algorithm is on average almost linear in the number of WDs.

4 Equilibria Under the Time Fair Operator

We have so far shown that the the game played by the WDs under the resource allocation policy $(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ of the CM operator can be transformed into a weighted congestion game, and we have proven that the CM-COG has an SPE. In what follows we show that under the resource allocation policy $(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ of the TF operator the game played by the WDs can be transformed into a player-specific congestion game.

Proposition 2. *Consider that the TF operator uses the time fair resource allocation policy $(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$. Then, the strategic interaction of the WDs can be modeled as a player-specific congestion game, in which the cost of WD i is given by*

$$\widetilde{C}_i(\boldsymbol{d}) = \sum_{(a,c)\in\mathcal{A}\times\mathcal{C}} I_{d_i,(a,c)}\Big(\frac{D_i}{R_{i,a}}n_a(\boldsymbol{d})+\frac{L_i}{F^c}n_c(\boldsymbol{d})\Big)+I_{d_i,i}C_i^l \tag{18}$$

Proof. Given the equal sharing of resources, it follows from (1) and (2) that the offloading cost (3) of WD i through AP a to EC c can be expressed as

$$\widetilde{C}_{i,a}(\mathbf{d}) = \frac{D_i}{R_{i,a}}n_a(\mathbf{d})+\frac{L_i}{F^c}n_c(\mathbf{d}). \tag{19}$$

Observe that the offloading cost (19) depends on the total number $n_a(\mathbf{d})$ of WDs sharing the AP a, the total number $n_c(\mathbf{d})$ of WDs sharing the EC c, and on the characteristics of WD i's task. Thus, the interaction between the WDs can be modeled as a *player-specific congestion game*. This proves the result. □

4.1 Computing Equilibrium Offloading Decisions

We refer to the resulting strategic game as $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*}) = < \mathcal{N}, (\mathfrak{D}_i)_{i\in\mathcal{N}}, (\widetilde{C}_i)_{i\in\mathcal{N}} >$, in which the players are WDs with the objective to minimize their cost given by (18).

Observe that the game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ is the TF-COG expressed in strategic form and thus if $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ has a NE then the TF-COG has an SPE. Hence, in what follows we focus on the existence and computability of pure NE for $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$.

In what follows we prove our result concerning the existence of a pure strategy NE under the TF operator. Our proof is based on the *JoinAndPlayAsynchronousUpdates* (JPAU) algorithm, which we propose for computing a NE of the game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$. The pseudo code of the JPAU algorithm is shown in Fig. 3. The algorithm adds WDs one at a time, and lets them play their best replies given the other WDs' strategies, and thus the following result is based on an induction in the number N of WDs.

Theorem 4. *The game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ has a pure strategy NE.*

Proof. The proof is given in Appendix A.2. □

Even though the proof of Theorem 4 is fairly involved, the JPAU algorithm itself is computationally efficient, as we show next.

Proposition 3. *When a new WD enters the game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ in a NE $\boldsymbol{d}^*(n-1)$, a new NE can be computed in $O((A-2)|\mathcal{N}_{n-1}|^2 - (A-3)|\mathcal{N}_{n-1}|)$ time.*

Proof. The proof is given in Appendix A.3. □

By adding WDs one at a time, it follows that the JPAU algorithm computes a NE of the game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ in polynomial time.

Corollary 2. *The JPAU algorithm terminates in a NE of the game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ in $O((A-2)N^3 - (A-3)N^2)$ time.*

Finally, since $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ is the TF-COG expressed in strategic form, we can formulate the following result.

Corollary 3. *The game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ has a pure strategy NE $\boldsymbol{d}^*$. Hence, an SPE $(\boldsymbol{d}^*, \mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ for the TF-COG exists.*

4.2 Implementation Considerations

In what follows we discuss how the SPE can be implemented in practice. Given the information about the resource allocation policy adopted by the operator, WDs perform best replies one at a time according to the ILC and JPAU algorithms in the case of the CM-COG and TF-COG, respectively. Upon its turn, a WD computes the set of its best replies based on the information about the congestion on resources, as provided by the operator. If it can improve its current offloading decision then it reports one of its best replies to the operator, otherwise it reports its current offloading decision. The operator then sends the updated information about the congestion on the resources to the next WD that is supposed to update its offloading decision.

$\mathbf{d}^* = JPAU(\mathcal{N}, \mathcal{A}, \mathcal{C})$

/*First WD enters the game*/
Let $\mathbf{d} \leftarrow \emptyset, i \leftarrow 1$
$d_i^*(1) = \arg\min_{d_i \in \mathfrak{D}_i} C_i(d_i, d_{-i})$
$\mathbf{d}^*(1) = d_i^*(1)$
for $n = 2 : N$ **do**
  /*Corresponds to induction phase*/
  Let $i \leftarrow n$
  $d_i^*(n) = \arg\min_{d_i \in \mathfrak{D}_i} C_i(d_i, d^*_{-i}(n-1))$
  $\mathbf{d}(n) = (d_i^*(n), \mathbf{d}^*(n-1))$
  if $d_i^*(n) = (a, c)$
    /*Corresponds to update phase*/
    if $\exists j \in O_{(a,c)}(\boldsymbol{d}(n))$ *for which a BR is local computing*
      /*Corresponds to case (i)*/
      $\mathbf{d}'(n) = (j, d_{-j}(n))$
    end
    else if $\exists j \in O_{(a,c')}(\boldsymbol{d}(n)), c' \neq c$ *for which a BR is local computing*
      /*Corresponds to case (ii)*/
      $\mathbf{d}'(n) = (j, d_{-j}(n))$
      $k \leftarrow O_c(\mathbf{d}'(n))$
      $\mathbf{d}'(n) = ((\cdot, c'), d'_{-k}(n))$
    else if $\exists j \in O_a(\boldsymbol{d}(n)), a' \neq a$ *for which a BR is changing to AP* a'
      /*Corresponds to case (iii)*/
      $\mathbf{d}'(n) = ((a', \cdot), d_{-j}(n))$
      while $\exists j \in O(\boldsymbol{d}'(n))$ *that can decrease its offloading cost*
        $d_j^*(n) = \arg\min_{d'_j \in \mathfrak{D}_j} C_j(d'_j, d'_{-j}(n))$
        $\mathbf{d}'(n) = (d_j^*(n), d'_{-j}(n))$
      end
    else
      $\mathbf{d}'(n) = \mathbf{d}(n)$
    end
    $a'' \leftarrow a$ for which $n_a(\mathbf{d}'(n)) = n_a(\mathbf{d}^*(n-1)) + 1$
    $c \leftarrow c'$ for which $n_{c'}(\mathbf{d}'(n)) = n_{c'}(\mathbf{d}^*(n-1)) + 1$
    if $\exists j \in O_{(a'',c)}(\boldsymbol{d}'(n), a' \neq a''$ *for which a BR is local computing*
      /*Corresponds to case (iv)*/
      $\mathbf{d}'(n) = (j, d'_{-j}(n))$
      if $\exists j \in O_{a''}(\boldsymbol{d}'(n))$ *for which a BR is changing to AP* a'
        $k \leftarrow O_{a''}(\mathbf{d}'(n))$
        $\mathbf{d}'(n) = ((a', \cdot), d'_{-k}(n))$
      else
        while $\exists j \in O(\boldsymbol{d}'(n))$ *that can decrease its offloading cost*
          $d_j^*(n) = \arg\min_{d'_j \in \mathfrak{D}_j} C_j(d'_j, d'_{-j}(n))$
          $\mathbf{d}'(n) = (d_j^*(n), d'_{-j}(n))$
        end
        if $\exists k \in \mathcal{N}_n \setminus O(\boldsymbol{d}'(n))$ *for which a BR is* (a', c)
          $\mathbf{d}'(n) = ((a', c), d'_{-k}(n))$
        end
        if $\exists j \in O_{(a',c)}(\boldsymbol{d}'(n)), a' \neq a$ *for which a BR is local computing*
          go to 35
        end
      end
    end
    $\mathbf{d}^*(n) = \mathbf{d}'(n)$
end
return $\boldsymbol{d}^*(N)$

Figure 3: Pseudo code of the $JPAU$ algorithm.

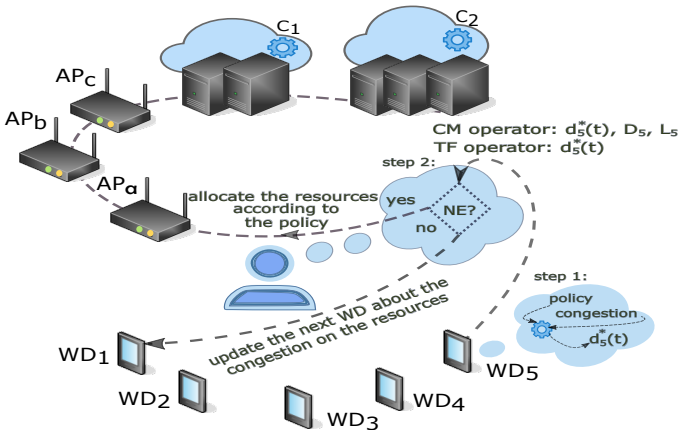

Figure 4: Example of the information exchange between the operator and WD_1 and WD_5.

Upon convergence, given the equilibrium offloading decisions of WDs, the operator allocates wireless and computing resources according to the adopted resource allocation policy. By Corollary 1 and Corollary 3 the resulting state is an SPE of the CM-COG and the TF-COG, respectively. Fig. 4 illustrates the information exchange between the operator and the WDs for the edge computing system shown in Fig. 1.

Observe that the WDs need to report only their offloading decisions in the case of the time fair operator and apart from the offloading decisions they need to reveal the characteristics of their tasks (i.e., the size D_i of the input data and the expected complexity L_i) in the case of the cost minimizing operator. Therefore, the implementation of the SPE of the TF-COG requires less information about the WDs' tasks than the implementation of the SPE of the CM-COG, and thus the time fair resource allocation policy may be a better choice in systems in which privacy and confidentiality are of major concern.

5 Price of Anarchy

We have so far analyzed the interaction between the WDs under the cost minimizing and the time fair resource allocation policies of the operator and we proposed the ILC and the JPAU algorithms for computing an equilibrium of offloading decisions of the WDs under these two policies, respectively. Furthermore, we showed that the computational complexity of the ILC algorithm (which is exponential in the worst case) can be reduced by letting WDs start to offload in non-increasing order of their task complexities, and we proved that the worst case complexity of the JPAU algorithm is polynomial in N. In this section we quantify the worst case ratio between the system performance in an SPE and the optimal performance using the

price of anarchy (PoA). We do so by providing an upper bound on the PoA of the CM-COG (denoted by $PoA_{\text{CM-COG}}$) and the TF-COG (denoted by $PoA_{\text{TF-COG}}$), respectively. Let us recall that the games $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ and $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ are strategic representations of the CM-COG and the TF-COG games, respectively. Therefore, we have $PoA_{\text{CM-COG}} = PoA(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ and $PoA_{\text{TF-COG}} = PoA(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$.

We start with the definition of the $PoA(\mathcal{P}_r, \mathcal{P}_c)$ of the strategic game played by the WDs for a policy $(\mathcal{P}_r, \mathcal{P}_c)$ of the operator for which an equilibrium allocation $\mathbf{d}^*$ of offloading decisions exists

$$PoA(\mathcal{P}_r, \mathcal{P}_c) = \frac{\max_{\mathbf{d}^* \in \mathfrak{D}^*} \sum_{i \in \mathcal{N}} C_i(\mathbf{d}^*, \mathcal{P}_r, \mathcal{P}_c)}{\min_{\mathbf{d} \in \mathfrak{D}} \sum_{i \in \mathcal{N}} C_i(\mathbf{d}, \mathcal{P}_r, \mathcal{P}_c)}, \tag{20}$$

where $\mathfrak{D}^*$ is the set of equilibria of offloading decisions under $(\mathcal{P}_r, \mathcal{P}_c)$.

5.1 Price of Anarchy of the CM-COG

In order to provide an upper bound on $PoA_{\text{CM-COG}}$, we provide an upper bound on $PoA(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ of the strategic game $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$.

Theorem 5. $PoA_{\textit{CM-COG}} = PoA(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*}) \leq \frac{3+\sqrt{5}}{2}$.

Proof. Our proof is inspired by Theorem 3.1 in [22], which provides a PoA bound for normalized weighted congestion games. Our proof extends the PoA bound to the game $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$, which is not a normalized weighted congestion game.

We start with defining the set $\mathcal{R}{=}\mathcal{N} \cup \mathcal{A} \cup \mathcal{C}$ of all resources available in the system. Furthermore, we denote by $\mathcal{R}_{d_i}$ the set of resources that WD i uses in strategy profile $\mathbf{d}$, and we use $\mathbf{d}^*$ and $\hat{\mathbf{d}}$ to denote a NE and an optimal strategy profile of $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$, respectively. Let us define the local computing weight $w_{i,i} \triangleq \sqrt{L_i / F_i^l}$ for each WD $i \in \mathcal{N}$, and the set of WDs using local computing link i $O_i(\mathbf{d}) = \{i | d_i = i\}$. Observe that either $O_i(\mathbf{d}) = \emptyset$ or $O_i(\mathbf{d}) = \{i\}$ holds since the local computing resources are not shared among WDs. We can thus express the total weight $w_i(\mathbf{d}) = \sum_{i \in O_i(\mathbf{d})} w_{i,i}$ associated with local computing link i, which is either $w_i(\mathbf{d}) = 0$ or $w_i(\mathbf{d}) = w_{i,i}$.

Using the above notation we can express the system cost $C(\mathbf{d}, \mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ for $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ in a strategy profile $\mathbf{d}$ as

$$C(\mathbf{d}, \mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*}) = \sum_{r \in \mathcal{R}} \sum_{i \in O_r(\mathbf{d})} w_r(\mathbf{d}) w_{i,r} = \sum_{r \in \mathcal{R}} w_r^2(\mathbf{d}). \tag{21}$$

Furthermore, from the definition of a NE we obtain

$$\begin{aligned} &\sum_{r \in \mathcal{R}_{d_i^*}} w_r(\mathbf{d}^*) w_{i,r} \leq \sum_{r \in \mathcal{R}_{d_i^*} \cap \mathcal{R}_{\hat{d}_i}} w_r(\mathbf{d}^*) w_{i,r} + \\ &\sum_{r \in \mathcal{R}_{d_i^*} \setminus \mathcal{R}_{\hat{d}_i}} \big(w_r(\mathbf{d}^*) + w_{i,r}\big) w_{i,r} \leq \sum_{r \in \mathcal{R}_{\hat{d}_i}} \big(w_r(\mathbf{d}^*) + w_{i,r}\big) w_{i,r}. \end{aligned} \tag{22}$$

First, by summing inequality (22) over all WDs i we obtain

$$\sum_{i\in\mathcal{N}}\sum_{r\in\mathcal{R}_{d_i^*}} w_r(\mathbf{d}^*)w_{i,r} \leq \sum_{i\in\mathcal{N}}\sum_{r\in\mathcal{R}_{\hat{d}_i}} \big(w_r(\mathbf{d}^*)+w_{i,r}\big)w_{i,r}. \tag{23}$$

Second, by reordering the summations, (23) can be rewritten as

$$\sum_{r\in\mathcal{R}}\sum_{i\in O_r(\mathbf{d}^*)} w_r(\mathbf{d}^*)w_{i,r} \leq \sum_{r\in\mathcal{R}}\sum_{i\in O_r(\hat{\mathbf{d}})} \big(w_r(\mathbf{d}^*)w_{i,r}+w_{i,r}^2\big). \tag{24}$$

Next, from the definition of the total weight $w_r(\mathbf{d}) = \sum_{i\in O_r(\mathbf{d})} w_{i,r}$ associated with resource r and from $\sum_{i\in O_r(\mathbf{d})} w_{i,r}^2 \leq w_r^2(\mathbf{d})$ we obtain

$$\sum_{r\in\mathcal{R}} w_r^2(\mathbf{d}^*) \leq \sum_{r\in\mathcal{R}} w_r(\mathbf{d}^*)w_r(\hat{\mathbf{d}}) + \sum_{r\in\mathcal{R}} w_r^2(\hat{\mathbf{d}}). \tag{25}$$

We can now use the Cauchy-Schwartz inequality ($\sum_{r\in\mathcal{R}} a_r b_r \leq \sqrt{\sum_{r\in\mathcal{R}} a_r^2 \sum_{r\in\mathcal{R}} b_r^2}$) to obtain

$$\sum_{r\in\mathcal{R}} w_r^2(\mathbf{d}^*) \leq \sqrt{\sum_{r\in\mathcal{R}} w_r^2(\mathbf{d}^*) \sum_{r\in\mathcal{R}} w_r^2(\hat{\mathbf{d}})} + \sum_{r\in\mathcal{R}} w_r^2(\hat{\mathbf{d}}). \tag{26}$$

If we divide the right and the left side of inequality (26) by $\sum_{r\in\mathcal{R}} w_r^2(\hat{\mathbf{d}}) > 0$ we can rewrite it using (21) as

$$\frac{C(\mathbf{d}^*, \mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})}{C(\hat{\mathbf{d}}, \mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})} \leq \sqrt{\frac{C(\mathbf{d}^*, \mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})}{C(\hat{\mathbf{d}}, \mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})}} + 1. \tag{27}$$

Since (27) holds for any NE of the game $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$, it holds for the worst case NE too, and thus we have

$$PoA(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*}) \leq \sqrt{PoA(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})} + 1. \tag{28}$$

By solving (28) we obtain that $PoA_{\text{CM-COG}} = PoA(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*}) \leq \frac{3+\sqrt{5}}{2}$, which proves the theorem. □

5.2 Price of Anarchy of the TF-COG

Next, using a similar approach to the one presented in the proof of Theorem 5, in what follows we provide an upper bound on $PoA_{\text{TF-COG}}$ by providing an upper bound on the $PoA(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ of the strategic game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$.

Theorem 6. $PoA_{TF\text{-}COG} = PoA(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*}) \leq N+1$.

Proof. We start with the definition of the weights in $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ for all resources $\mathcal{R}{=}\mathcal{N}\cup\mathcal{A}\cup\mathcal{C}$ available in the system

$$\widetilde{w}_{i,i} \triangleq \frac{L_i}{F_i^l}, \widetilde{w}_{i,c} \triangleq \frac{L_i}{F^c}, \widetilde{w}_{i,a} \triangleq \frac{D_i}{R_{i,a}}.$$

Using the above notation we can express the system cost $\widetilde{C}(\mathbf{d}, \mathcal{P}_r^{eq}, \mathcal{P}_c^{eq})$ for $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ in a strategy profile $\mathbf{d}$ as

$$\widetilde{C}(\mathbf{d}, \mathcal{P}_r^{eq}, \mathcal{P}_c^{eq}) = \sum_{r\in\mathcal{R}} \sum_{i\in O_r(\mathbf{d})} n_r(\mathbf{d})\widetilde{w}_{i,r} = \sum_{r\in\mathcal{R}} n_r(\mathbf{d})\widetilde{w}_r(\mathbf{d}), \tag{29}$$

where $\widetilde{w}_r(\mathbf{d}) \triangleq \sum_{i\in O_r(\mathbf{d})} \widetilde{w}_{i,r}$.

Furthermore, let us use $\mathbf{d}^*$ and $\hat{\mathbf{d}}$ to denote a NE and an optimal solution of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$, respectively.

Now, from the definition of a NE we obtain

$$\begin{aligned} \sum_{r\in\mathcal{R}_{d_i^*}} n_r(\mathbf{d}^*)\widetilde{w}_{i,r} &\leq \sum_{r\in\mathcal{R}_{d_i^*}\cap\mathcal{R}_{\hat{d}_i}} n_r(\mathbf{d}^*)\widetilde{w}_{i,r} + \\ \sum_{r\in\mathcal{R}_{d_i^*}\setminus\mathcal{R}_{\hat{d}_i}} \big(n_r(\mathbf{d}^*)+1\big)\widetilde{w}_{i,r} &\leq \sum_{r\in\mathcal{R}_{\hat{d}_i}} \big(n_r(\mathbf{d}^*)+1\big)\widetilde{w}_{i,r}. \end{aligned} \tag{30}$$

First, by summing inequality (30) over all WDs i we obtain

$$\sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_{d_i^*}} n_r(\mathbf{d}^*)\widetilde{w}_{i,r} \leq \sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_{\hat{d}_i}} \big(n_r(\mathbf{d}^*)+1\big)\widetilde{w}_{i,r}. \tag{31}$$

Second, by reordering the summations (31) can be rewritten as

$$\sum_{r\in\mathcal{R}} \sum_{i\in O_r(\mathbf{d}^*)} n_r(\mathbf{d}^*)\widetilde{w}_{i,r} \leq \sum_{r\in\mathcal{R}} \sum_{i\in O_r(\hat{\mathbf{d}})} \big(n_r(\mathbf{d}^*)\widetilde{w}_{i,r} + \widetilde{w}_{i,r}\big). \tag{32}$$

Using the definition of the total weight $\widetilde{w}_r(\mathbf{d}) \triangleq \sum_{i\in O_r(\mathbf{d})} \widetilde{w}_{i,r}$ associated with resource r we can rewrite (32) as

$$\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)\widetilde{w}_r(\mathbf{d}^*) \leq \sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)\widetilde{w}_r(\hat{\mathbf{d}}) + \sum_{r\in\mathcal{R}} \widetilde{w}_r(\hat{\mathbf{d}}). \tag{33}$$

Next, observe that $n_r(\mathbf{d}) \leq N$ must hold for any feasible strategy profile $\mathbf{d}$ and for every resource $r \in \mathcal{R}$, and that $|O_r(\mathbf{d})| \geq 1$ implies $n_r(\mathbf{d}) \geq 1$. Therefore, we have that

$\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)\widetilde{w}_r(\hat{\mathbf{d}}) \leq N\sum_{r\in\mathcal{R}} n_r(\hat{\mathbf{d}})\widetilde{w}_r(\hat{\mathbf{d}})$ and $\sum_{r\in\mathcal{R}} \widetilde{w}_r(\hat{\mathbf{d}}) \leq \sum_{r\in\mathcal{R}} n_r(\hat{\mathbf{d}})\widetilde{w}_r(\hat{\mathbf{d}})$. By using these observations in (33) we obtain the following inequality

$$\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)\widetilde{w}_r(\mathbf{d}^*) \leq (N+1)\sum_{r\in\mathcal{R}} n_r(\hat{\mathbf{d}})\widetilde{w}_r(\hat{\mathbf{d}}). \tag{34}$$

Finally, since $\sum_{r\in\mathcal{R}} n_r(\hat{\mathbf{d}})\widetilde{w}_r(\hat{\mathbf{d}}) > 0$ must hold, we can divide the right and the left side of inequality (34) by $\sum_{r\in\mathcal{R}} n_r(\hat{\mathbf{d}})\widetilde{w}_r(\hat{\mathbf{d}})$ to obtain

$$\frac{\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*)\widetilde{w}_r(\mathbf{d}^*)}{\sum_{r\in\mathcal{R}} n_r(\hat{\mathbf{d}})\widetilde{w}_r(\hat{\mathbf{d}})} \leq N+1. \tag{35}$$

Since (35) holds for any NE of the game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$, it also holds for the worst case NE, and thus using (29) we obtain

$$PoA_{\text{TF-COG}} = PoA(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*}) \leq N+1, \tag{36}$$

which proves the theorem. □

Observe that the $PoA(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ and $PoA(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ are in fact bounds on the approximation ratio of the ILC and JPAU algorithms used for computing a NE of the games $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ and $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$, respectively. Therefore, the ILC algorithm outperforms the JPAU algorithm in terms of the worst case system performance and the JPAU algorithm outperforms the ILC algorithm in terms of the worst case complexity. Consequently, the cost minimizing resource allocation policy might be a better choice than the time fair resource allocation policy in systems in which a guarantee on the worst case system performance is more important than a guarantee on the worst case computational efficiency, and vice versa.

6 Numerical Results

In the following we show results from extensive simulations to evaluate the system performance from the perspective of the operator of the WDs.

For the simulations we placed ECs and WDs uniformly at random over a square area of $1km \times 1km$, and we placed 5 APs at random on a *regular grid* with 25 points defined over the area. This uniform deployment corresponds to a dense urban area. We consider that the channel gain of WD i in the case of offloading through the same AP a depends on its distance $d_{i,a}$ from the AP and on the path loss exponent α. We use $\alpha = 4$ according to the path loss model in urban and suburban areas [23]. For simplicity we assign a bandwidth of $B_{i,a} = 5MHz$ to each communication link $(i,a) \in \mathcal{N} \times \mathcal{A}_i$. The transmit power $P_{i,a}^t$ at which WD i offloads the data through AP a is drawn from a continuous uniform distribution on $[0.05, 0.18]\,W$ according to [24]. Given the noise power P_n we calculate the transmission rate $R_{i,a}$ achievable

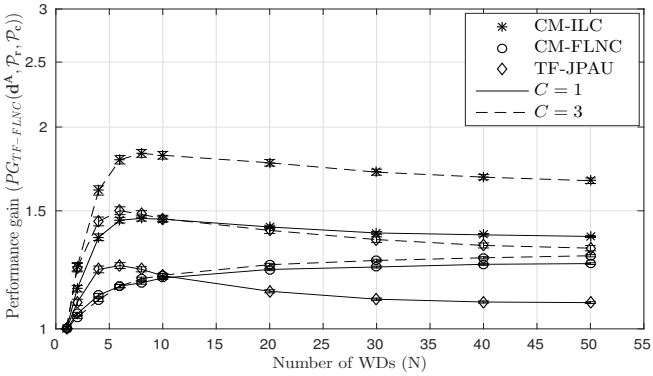

Figure 5: *Performance gain* vs. the number of WDs N for $A = 5$ APs. *Homogeneous ECs*, $F^{c,tot} = 192GHz$.

to WD i for offloading to AP a as $R_{i,a} = B_{i,a} log(1+d_{i,a}^{-\alpha} \frac{P_{i,a}^t}{P_n})$. The input data size D_i is drawn from a uniform distribution on $[0.2, 4]Mb$, and the number X of CPU cycles required per data bit is a Gamma distributed random variable with the shape $k = 0.5$ and scale $\theta = 1.6$. Given D_i and X, we calculate the complexity of a task as $L_i = D_i X$.

We consider two operator policies in the evaluation. We refer to $(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ as the CM policy. Under the CM policy the WDs use the ILC algorithm for computing a NE of the game $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$, as shown in Section 3. As a baseline for comparison, we consider the TF policy $(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$, under which the WDs use the JPAU algorithm for computing a NE of the game $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$, as shown in Section 4.

As a baseline for the ILC and JPAU algorithms proposed for computing an equilibrium of offloading decisions, we use the *FastestLinkNearestCloud* (FLNC) algorithm. According to the FLNC algorithm WDs offload the computation through the AP with the highest achievable transmission rate and to the EC closest to the chosen AP. Observe that FLNC can be used with both operator policies. The results shown are the averages of 1000 simulations, together with 95% confidence intervals.

6.1 User-oriented performance

We start with considering the system performance from the point of view of the WDs. We define the *performance gain* $PG_{\text{TF}-\text{FLNC}}(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c)$ (w.r.t. the TF-FLNC) for a strategy profile $\mathbf{d}^A$ computed by algorithm $A \in \{\text{ILC}, \text{JPAU}, \text{FLNC}\}$ under a resource allocation policy $(\mathcal{P}_r, \mathcal{P}_c) \in \{(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*}), (\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})\}$ as

$$PG_{\text{TF}-\text{FLNC}}(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c) = \frac{C(\mathbf{d}^{FLNC}, \mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})}{C(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c)}.$$

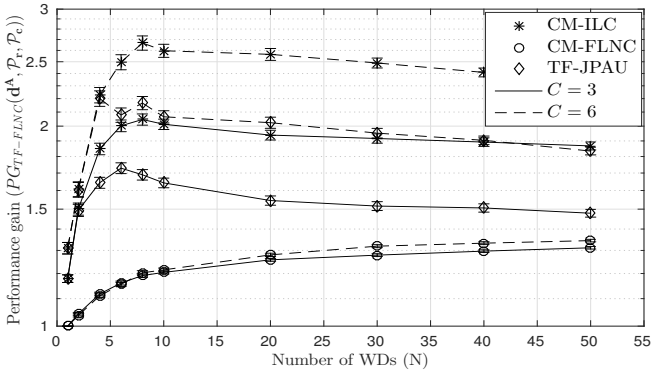

Figure 6: *Performance gain* vs. the number of WDs N for $A = 5$ APs. *Heterogeneous ECs*, $F^{c,tot} = 192GHz$.

Fig. 5 shows the *performance gain* as a function of the number N of WDs for two MEC systems, one with C=1 (F^{c_1}=192GHz) and one with C=3 (F^{c_i}=64GHz), i.e., ECs are homogeneous. The figure shows that the *performance gain* is largest when the operator uses the CM policy and WDs offload according to an equilibrium computed by the ILC algorithm. Interestingly, even CM-FLNC outperforms TF-JPAU for C=1 ECs and N>10 WDs. These results indicate that the operator's resource allocation policy has a large impact on the user-perceived performance. Overall, we can observe that the *performance gain* increases with a decreasing marginal gain in N, which suggests that the achievable *performance gain* is limited by the congestion on the APs and ECs.

Fig. 6 shows the corresponding *performance gain* for heterogeneous ECs for two MEC systems, one with C=3 ECs and one with C=6 ECs. The total cloud computing capability $F^{c,tot}$=192GHz of the system is distributed among the ECs such that F^{c_1}=32GHz and $F^{c_i} = F^{c_{i-1}} + 32GHz$, $i > 1$, for $C = 3$ ECs, and $F^{c_1} = 12GHz$ and $F^{c_i} = F^{c_{i-1}} + 8GHz$, i>1, for C=6 ECs. As in Fig. 5, the results in Fig. 6 show a decreasing marginal gain in N and confirm that the largest *performance gain* is achieved by the CM-ILC. Nonetheless, a comparison of Fig. 5 and Fig. 6 reveals that the *performance gain* is affected by the number of ECs in the system and the way the total cloud computing capability is shared among the ECs. On the one hand, the *performance gain* increases with C. On the other hand, the *performance gain* for $C = 3$ ECs is greater in the case of heterogeneous ECs than that in the case of homogeneous ECs. Thus, CM-ILC is most beneficial when edge cloud resources are heterogeneous. The improved performance is partly due to that the WDs in the baseline strategy profile (computed by the FLNC) offload their tasks through the fastest link to the EC that is closest to the chosen AP, and since WDs, APs and ECs are randomly placed over the area, the number of WDs per EC is not proportional to its computing capability, as we will see later.

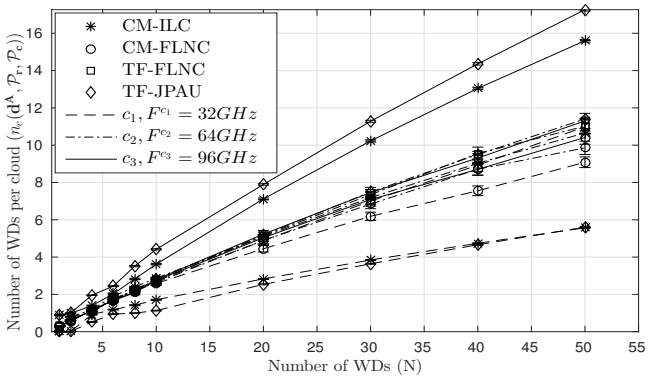

Figure 7: Congestion per EC vs. the number of WDs N for $A = 5$ APs, $C = 3$ ECs. *Heterogeneous ECs*, $F^{c,tot} = 192GHz$.

6.2 Infrastructure-oriented performance

In order to evaluate the system performance from operator's perspective, we investigate how the choice of the resource allocation policy and the algorithm for computing the offloading decisions of WDs affects the number $n_c(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c)$ of WDs per EC and the cost $C^c(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c) = \sum_{i \in O_c(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c)} C_i(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c)$ per EC. For consistency, we show results for a system with heterogeneous cloud resources, i.e., $F^{c,tot} = 192\,GHz$ divided among three ECs such that $F^{c_1} = 32\,GHz$ and $F^{c_i} = F^{c_{i-1}} + 32\,GHz$, for $i > 1$.

Fig. 7 and Fig. 8 show $n_c(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c)$ and $C^c(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c)$ for each of the ECs as a function of the number N of WDs, respectively. The results are shown for the ILC, JPAU and FLNC algorithms under both the CM and TF resource allocation policies. By looking at $n_c(\mathbf{d}^A, \mathcal{P}_r, \mathcal{P}_c)$ for all ECs for a fixed N, we observe from Fig. 7 that the ratio of the WDs that offload their tasks decreases as N increases. This happens because the number of WDs that cannot benefit from offloading due to high congestion on the shared resources increases with N. Fig. 7 also shows that the difference in the congestion experienced by the ECs is smallest when the offloading decisions of the WDs are computed by the FLNC algorithm. This is due to that in the strategy profile computed by the FLNC algorithm WDs offload their tasks to the EC that is closest to the fastest AP, and since the WDs, APs, and ECs are placed uniformly at random over the region, all ECs experience the same congestion on average. Consequently, the corresponding cost per EC, shown in Fig. 8, is inverse proportional to the computing capability of the EC.

On the contrary, in the case of equilibria computed by ILC and by JPAU (i.e. equilibria under the CM and TF policies, respectively) the congestion and the cost per EC are proportional to the computing capability of the EC as shown in Fig. 7 and Fig. 8, respectively. We also observe that the total number of WDs that offload their tasks and the total offloading cost are higher in an equilibrium computed by

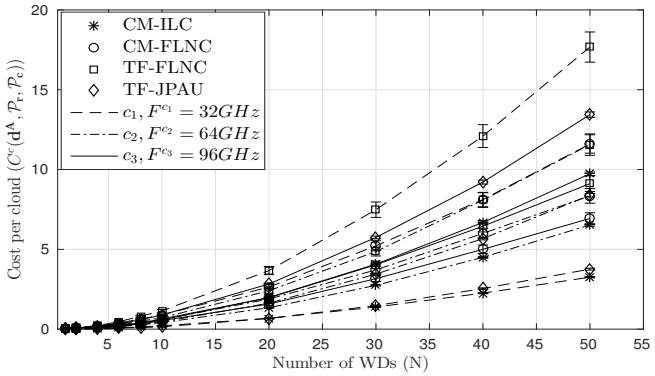

Figure 8: Cost per EC vs. the number of WDs N for $A = 5$ APs, $C = 3$ ECs. *Heterogeneous ECs*, $F^{c,tot} = 192GHz$.

the JPAU algorithm than in an equilibrium computed by the ILC algorithm. This is due to that the cloud computing resources are shared among WDs independently of their tasks' complexities in the case of the TF policy, and consequently the WDs overuse the ECs.

6.3 Computational complexity

We characterize the computational complexity of an algorithm as the number of iterations needed to compute a computation offloading strategy profile. Since $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ is a potential game, we use the AU algorithm (c.f. Fig. 2) as a baseline for comparison, as it is guaranteed to converge from an arbitrary initial strategy profile [18]. For the AU algorithm we consider three initial strategy profiles: a randomly chosen initial strategy profile (*Random*AU), an initial strategy profile in which all WDs offload their tasks such that the number of WDs offloading the computation to an EC is proportional to its computing capability (*ECProportional*AU), and an empty strategy profile where the WDs enter the game in non-increasing order of their task complexities (*JoinNon-Incr*AU). Furthermore, we consider the complexity of computing an equilibrium of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ using the JPAU algorithm.

Fig. 9 shows the number of iterations needed to compute an equilibrium of $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ and an equilibrium of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$, as a function of N for the same set of parameters as in Fig. 5. We observe that the number of iterations scales approximately linearly with N in all cases and that computing an equilibrium of $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$ using the ILC algorithm is more efficient than computing an equilibrium of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ using the JPAU algorithm; the difference is up to 50%.

We also observe that the choice of the initial strategy profile affects the complexity of computing an equilibrium of the game $\Gamma(\mathcal{P}_r^{c,*}, \mathcal{P}_c^{c,*})$, and we make three observations. First, the number of iterations required by ILC and by *JoinNon-Incr*AU is insensitive to the number of ECs, while the number of iterations required by

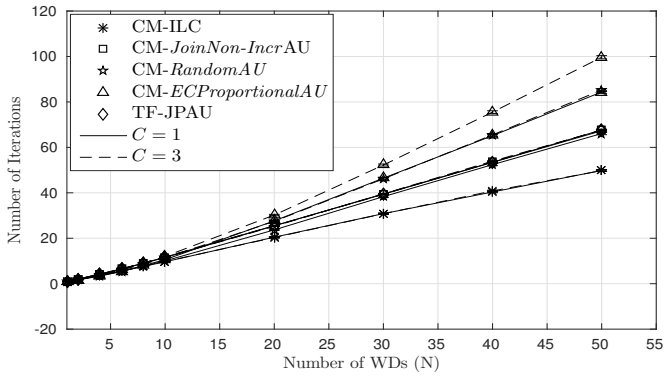

Figure 9: Number of iterations vs. the number of WDs N for $A = 5$. *Homogeneous ECs*, $F^{c,tot} = 192GHz$.

*Random*AU and by *ECProportional*AU increases with the number of ECs. This is due to that in the case of ILC and of *JoinNon-Incr*AU the WDs start using ECs in non-increasing order of their task complexities, and thus it follows from Proposition 1 that when a new WD starts offloading, WDs will not have an incentive to change between ECs. This is not true in the case of *Random*AU and of *ECProportional*AU, since they start from a strategy profile where WDs did not start to offload in the order of the complexities of their tasks, and consequently the WDs can decrease their offloading cost not only by changing between the APs, but also by changing between the ECs. Second, the *ECProportional*AU has the highest computational complexity. This is due to that *ECProportional*AU starts from an initial strategy profile that has the highest congestion on the resources and thus when a WD updates its strategy the number of WDs affected by the update step is higher than in the case of the other initial strategy profiles. Finally, the smallest computational complexity can be achieved by the proposed ILC algorithm. On the one hand, this is because the WDs do not have to choose their initial strategy as in the case of the *JoinNon-Incr*AU. On the other hand, the WDs cannot decrease their offloading cost by changing between the ECs as in the case of the *Random*AU and *ECProportional*AU.

To summarize, the proposed CM-ILC algorithm can provide a significant reduction in terms of completion times and has low computational complexity, and could be a good candidate for coordinating the offloading decisions of WDs for edge computing.

7 Related Work

There is a large body of recent works on computation offloading for mobile cloud computing [11, 15, 25–36]. Many of these works assume that the offloading decisions of devices are determined by a centralized entity with the objective to meet the energy and latency constraints of the devices [11,25–30]. [25] considered that devices offload the computation either to a computationally limited local cloud or to a

computationally rich remote cloud, and proposed a policy that schedules resources in the clouds so as to meet the delay requirements of the applications. [11, 26, 27] formulated the computation offloading problem as an optimization problem that minimizes the energy consumption of the mobile devices under latency constraints. [26] considered that devices may offload their tasks to an edge cloud through a base station, and proposed a policy for managing computing and communication resources assuming that the base station has perfect knowledge about the system. [11, 27] considered a network composed of multiple cells, each equipped with an edge cloud. [11] proposed an iterative algorithm for jointly optimizing the allocation of computing and uplink bandwidth resources, and [27] proposed an iterative algorithm for jointly optimizing the allocation of computing and both uplink and downlink bandwidth resources. [28] considered the problem of joint optimization of network selection and service placement under random mobility of users and proposed an iterative algorithm that minimizes the average system delay. [29] proposed an online algorithm for distributing the workload across multiple edge clouds, which are managed by an operator that apart from the workload distribution decides about the activation status of the edge clouds, and acts as the auctioneer that solicits bids from multiple service providers. [30] considered a mobile cloud computing system in which a centralized entity located in the cloud implements an online algorithm for scheduling the transmissions between the mobile devices and the cloud so as to minimize the energy consumption of mobile devices. Unlike these works, we propose a novel approach to address the computation offloading problem by considering the interaction between an operator that manages the allocation of wireless and computing resources and devices that make their offloading decisions in a decentralized manner.

Closer related to ours are recent works that propose decentralized algorithms based on a game theoretic treatment of the computation offloading problem [15,31–36]. Authors in [31] considered the interaction between devices that always offload their tasks and an operator that optimizes the allocation of wireless and computing resources. Compared to [32], we consider both the cost minimizing and the time fair resource allocation policies and besides the analysis of a game in the case of the cost minimizing operator, we also prove the existence of Stackelberg equilibria in the case of the time fair operator, we establish an upper bound on the price of anarchy of the resulting Stackelberg game and we propose a polynomial complexity algorithm for computing an equilibrium of the game. [33] considered that devices may offload the computation to the cloud through a single wireless link if doing so minimizes their own energy consumption, and proved the existence of equilibria when devices with the same delay budget compete only for wireless resources. [15, 34, 35] considered that devices may offload their tasks to the cloud through one of multiple wireless links so as to minimize the linear combination of the delay and the energy consumption. [15] considered the congestion only on the wireless links and proved the existence of equilibria under the assumption that a device experiences the same channel gain for all wireless links. [34] extended the equilibrium existence results of [15] to a dynamic environment, where devices may be active or inactive. [35] considered that devices may offload their tasks to the cloud through one of multiple

heterogeneous wireless links, modeled the congestion on both cloud and wireless links and provided a polynomial time algorithm for computing equilibria. [36] considered a fog computing system where multiple devices may offload their computational tasks to each other or to an edge cloud and provided an efficient algorithm for computing a mixed strategy equilibrium in a decentralized way. Our work differs significantly from these works, as we model the congestion on multiple heterogeneous wireless links and edge clouds, which are managed by an operator that can implement one of two resource allocation policies and given the resource allocation policy of the operator we consider that devices can autonomously decide whether or not to offload the computations, and if so, to which of multiple edge clouds and through which of multiple wireless links. To the best of our knowledge, ours is the first work on computation offloading for mobile cloud computing that closes the gap between the works that propose centralized solutions and the works that propose decentralized solutions.

Closest to our work in the literature on game theory is [37], which considers the effectiveness of Stackelberg strategies for atomic congestion games. Authors in [37] consider that the leader controls a subset of non-selfish players, focus on affine latency functions and on congestion games on parallel links. On the contrary, in our model the leader manages the sharing of resources, and we consider a player-specific weighted network congestion game for which the existence of equilibria is not known in general [38]. Thus, our work provides a novel game theoretic perspective on congestion games.

8 Conclusion

We have provided a game theoretical analysis of selfish computation offloading in a mobile edge computing system where wireless and computing resources are jointly managed by an operator, and devices make offloading decisions autonomously so as to minimize the completion times of their tasks. We consider the cost minimizing and the time fair allocation policies of the operator and we use a Stackelberg game to model the interaction between the operator and devices. We expressed the cost minimizing resource allocation policy in closed form and proved the existence of Stackelberg equilibria for both policies. Using game theoretical tools, we developed efficient decentralized approximation algorithms for computing offloading decisions of devices under both policies of the operator. Our numerical results show that the proposed algorithms are computationally efficient and that the system performance can be significantly improved through optimally allocating wireless and computing resources in a system, while allowing the devices to make their offloading decisions autonomously.

References

[1] M. Hakkarainen, C. Woodward, and M. Billinghurst, "Augmented assembly using a mobile phone," in *Proc. of IEEE/ACM ISMAR*, Sept 2008, pp. 167–168.

[2] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uwave: Accelerometer-based personalized gesture recognition and its applications," in *Proc. of IEEE PerCom*, March 2009, pp. 1–9.

[3] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 68–81.

[4] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless communications*, vol. 20, no. 3, pp. 14–22, 2013.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," Sep. 2015.

[6] S. R. Group, "The leading cloud providers continue to run away with the market," Tech. Rep., 2017.

[7] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM CCR*, vol. 44, no. 5, pp. 27–32, 2014.

[8] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM CCR*, vol. 45, no. 5, pp. 37–42, 2015.

[9] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *ACM SIGMETRICS*, 2001, pp. 50–61.

[10] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of Usenix HotCloud*, 2010.

[11] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE T-SIPN*, vol. 1, no. 2, pp. 89–103, 2015.

[12] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. of IEEE INFOCOM*, March 2012, pp. 2716–2720.

[13] T. Li, D. Baumberger, and S. Hahn, "Efficient and scalable multiprocessor fair scheduling using distributed weighted round-robin," *SIGPLAN Not.*, vol. 44, no. 4, pp. 65–74, Feb. 2009.

[14] T. Joshi, A. Mukherjee, Y. Yoo, and D. P. Agrawal, "Airtime fairness for ieee 802.11 multirate networks," *IEEE Trans. on Mob. Comp.*, pp. 513–527, 2008.

[15] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM TON*, no. 5, pp. 2795–2808, 2016.

[16] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer Mag.*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[17] S. Jošilo and G. Dán, "Decentralized scheduling for offloading of periodic tasks in mobile edge computing," in *Proc. of IFIP NETWORKING*, 2018.

[18] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[19] J. R. Marden, G. Arslan, and J. S. Shamma, "Joint strategy fictitious play with inertia for potential games," *IEEE Trans. on Automatic Control*, vol. 54, no. 2, pp. 208–220, Feb 2009.

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

[21] H. Ackermann, H. Röglin, and B. Vöcking, "On the impact of combinatorial structure on congestion games," *Journal of the ACM (JACM)*, vol. 55, no. 6, p. 25, 2008.

[22] B. Awerbuch, Y. Azar, and A. Epstein, "The price of routing unsplittable flow," in *Proc. of ACM STOC*, 2005, pp. 57–66.

[23] A. Aragon-Zavala, *Antennas and propagation for wireless communication systems*. John Wiley & Sons, 2008.

[24] E. Casilari, J. M. Cano-García, and G. Campos-Garrido, "Modeling of current consumption in 802.15. 4/zigbee sensor motes," *Sensors*, vol. 10, no. 6, pp. 5443–5468, 2010.

[25] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *IEEE GC Wkshps*, 2015, pp. 1–6.

[26] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. on Wireless Communications*, pp. 1397–1411, 2017.

[27] A. Al-Shuwaili, O. Simeone, A. Bagheri, and G. Scutari, "Joint uplink/downlink optimization for backhaul-limited mobile cloud computing with user scheduling," *IEEE T-SIPN*, pp. 787–802, 2017.

[28] G. Bin, F. L. Zhi, Zhou and, and X. Fei, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. of IEEE INFOCOM*, 2019.

[29] C. Shutong, J. Lei, W. Lin, and L. Fangming, "An online market mechanism for edge emergency demand response via cloudlet control," in *Proc. of IEEE INFOCOM*, 2019.

[30] F. Liu, P. Shu, and J. C. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[31] S. Jošilo and G. Dán, "Joint allocation of computing and wireless resources to autonomous devices in mobile edge computing," in *Proc. of ACM SIGCOMM Mecomm'18 Workshop*, 2018.

[32] ——, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. of IEEE INFOCOM*, 2019.

[33] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy aware offloading for competing users on a shared communication channel," *IEEE Trans. on Mob. Comp*, vol. 16, no. 1, pp. 87–96, 2017.

[34] J. Zheng, Y. Cai, Y. Wu, and X. S. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. on Mob. Comp.*, 2018.

[35] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Trans. on Mob. Comp*, vol. 18, no. 1, pp. 207–220, 2019.

[36] ——, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Transactions on Networking*, 2018.

[37] D. Fotakis, "Stackelberg strategies for atomic congestion games," *Theory of Computing Systems*, vol. 47, no. 1, pp. 218–249, 2010.

[38] I. Milchtaich, "The equilibrium existence problem in finite network congestion games," in *Proc. of WINE*, 2006, pp. 87–98.

[39] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of IEEE INFOCOM*, 2017.

A Appendix

A.1 Proof of Theorem 1

By inspecting the leading minors of the Hessian matrix of (3) it is easy to show that (9) is neither convex nor concave in $\mathbf{u}$ and $\mathbf{p}$ already for the case when there are only two WDs sharing a resource. Furthermore, it is easy to see from expressions (1) and (2) that the optimal solution of (9) cannot be unique, since any non-zero scalar multiple of feasible policies $(\mathcal{P}_r, \mathcal{P}_c)$ yields the same objective value, and hence if there is an optimal solution then there is a continuum of optimal solutions.

To make the solution unique with respect to scalar multiplication, let us introduce normalization constraints on the sums of the provisioning coefficients, and obtain

$$\min_{(\mathbf{u},\mathbf{p})\in\mathfrak{A}_c} C(\mathbf{d},\mathbf{u},\mathbf{p}) \tag{37}$$

$$\text{s.t.} \quad \textstyle\sum_{j\in O_a(\mathbf{d})} u_{j,a} = 1, \quad \forall a \in \mathcal{A} \tag{38}$$

$$\textstyle\sum_{j\in O_c(\mathbf{d})} p_{j,c} = 1. \quad \forall c \in \mathcal{C} \tag{39}$$

Observe that due to the normalization constraint the cost function $C(\mathbf{d},\mathbf{u},\mathbf{p})$ can be rewritten as

$$C'(\mathbf{d},\mathbf{u},\mathbf{p}) = \sum_{a\in\mathcal{A}} \sum_{i\in O_a(\mathbf{d})} \frac{D_i}{R_{i,a}u_{i,a}} + \sum_{c\in\mathcal{C}} \sum_{i\in O_c(\mathbf{d})} \frac{L_i}{F^c p_{i,c}} + \sum_{i\in\mathcal{N}\setminus O(\mathbf{d})} C_i^l$$

Unlike problem (9), problem (37)-(39) is a convex minimization problem, and thus its optimal solution must satisfy the Karush–Kuhn–Tucker (KKT) conditions. To define the Lagrangian dual of (37)-(39), we denote by $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ the dual variables associated with constraints (38) and (39) and by $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ the non-negative dual variables associated with constraints $\mathbf{u}\succeq 0$ and $\mathbf{p}\succeq 0$. Using this notation, we express the Lagrangian associated with (37)-(39) as

$$\mathcal{L}(\mathbf{d},\mathbf{u},\mathbf{p},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta}) = C'(\mathbf{d},\mathbf{u},\mathbf{p}) + \sum_{a\in\mathcal{A}} \alpha_a \Big(\sum_{j\in O_a(\mathbf{d})} u_{j,a} - 1 \Big)$$
$$- \sum_{a\in\mathcal{A}} \sum_{j\in O_a(\mathbf{d})} \gamma_{j,a} u_{j,a} + \sum_{c\in\mathcal{C}} \beta_c \Big(\sum_{j\in O_c(\mathbf{d})} p_{j,c} - 1 \Big) - \sum_{c\in\mathcal{C}} \sum_{j\in O_c(\mathbf{d})} \delta_{j,c} p_{j,c}.$$

Finally, we define the Lagrangian dual problem as $\max_{\boldsymbol{\alpha}\in\mathbb{R}^A,\boldsymbol{\beta}\in\mathbb{R}^C,\boldsymbol{\gamma},\boldsymbol{\delta}\succeq 0} \min_{\mathbf{u},\mathbf{p}\succeq 0} \mathcal{L}(\mathbf{d},\, \mathbf{u},\mathbf{p},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta})$, and we formulate the following KKT conditions.

Observe that $u_{i,a} = 0$ and $p_{i,c} = 0$ would lead to an infinite completion time for WD i's task, and thus $u_{i,a} > 0$ and $p_{i,c} > 0$ must hold. Therefore, $\gamma_{i,a} = 0$ and $\delta_{i,c} = 0$ must hold in order to have the complementary slackness conditions satisfied. Finally, from the stationarity conditions we can express $u_{i,a}$ and $p_{i,c}$ as

$$u_{i,a} = \sqrt{D_i/\alpha_a R_{i,a}}, \forall a \in \mathcal{A}, \forall i \in O_a(\mathbf{d}), \tag{40}$$

Stationarity:	$\frac{\partial \mathcal{L}(\mathbf{d},\mathbf{u},\mathbf{p},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta})}{\partial u_{i,a}} = 0, \forall a \in \mathcal{A}, \forall i \in O_a(\mathbf{d})$
	$\frac{\partial \mathcal{L}(\mathbf{d},\mathbf{u},\mathbf{p},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta})}{\partial p_{i,c}} = 0, \forall c \in \mathcal{C}, \forall i \in O_c(\mathbf{d})$
Primal feasibility:	$\sum_{j \in O_a(\mathbf{d})} u_{j,a} = 1, \forall a \in \mathcal{A}$
	$\sum_{j \in O_c(\mathbf{d})} p_{j,c} = 1, \forall c \in \mathcal{C}$
Dual feasibility:	$\gamma_{i,a}, \delta_{i,c} \geq 0, \forall i \in \mathcal{N}, \forall a \in \mathcal{A}, \forall c \in \mathcal{C}$
Complementary slackness:	$-\gamma_{i,a} u_{i,a} = 0, \forall a \in \mathcal{A}, \forall i \in O_a(\mathbf{d})$
	$-\delta_{i,c} p_{i,c} = 0, \forall c \in \mathcal{C}, \forall i \in O_c(\mathbf{d})$

and

$$p_{i,c} = \sqrt{L_i / \beta_c F^c}, \forall c \in \mathcal{C}, \forall i \in O_c(\mathbf{d}). \tag{41}$$

By substituting (40) and (41) in the primal feasibility equations we can obtain the expressions for α_a and β_c, and we can rewrite equations (40) and (41) as $u_{i,a} = \frac{\sqrt{D_i / R_{i,a}}}{\sum_{j \in O_a(\mathbf{d})} \sqrt{D_j / R_{j,a}}}$ and $p_{i,c} = \frac{\sqrt{L_i / F^c}}{\sum_{j \in O_c(\mathbf{d})} \sqrt{L_j / F^c}}$, which proves the theorem.

A.2 Proof of Theorem 4

The JPAU algorithm starts from an empty system, adds WDs into the game one at a time in the induction phase and lets WDs to update their best replies one at a time in the update phase. We denote by $\mathcal{N}_n$ the set of WDs that participate in the game upon an induction step $1 \leq n \leq N$ and we use $\mathbf{d}(n) = (d_i(n), d_{-i}(n))$ to denote a strategy profile played by WD i and the other WDs $j \in \mathcal{N}_n \setminus \{i\}$. Observe that for $N = 1$, there is only one WD i in the game playing its best reply $d_i^*(1)$, and thus $\mathbf{d}^*(1) = d_i^*(1)$ is a NE of the game.

For $N > 1$ let us assume that in induction step $n-1$ WDs play a NE $\mathbf{d}^*(n-1)$, and let us consider a WD $i \in \mathcal{N} \setminus \mathcal{N}_n$ added into the game by the JPAU algorithm in induction step n. If a best reply $d_i^*(n)$ of WD i is to perform the computation locally, then $\mathbf{d}^*(n) = (d_i^*(n), \mathbf{d}^*(n-1))$ is a NE of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ since the congestion on the APs and the ECs remained unchanged. Otherwise, let us assume that a best reply of WD i is offloading through AP a to EC c, i.e., $d_i^*(n) = (a, c)$. Observe that $\mathbf{d}(n) = (d_i^*(n), \mathbf{d}^*(n-1))$ may or may not be a NE of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$. If there is no WD that wants to deviate from its strategy played in $\mathbf{d}^*(n-1)$ then $\mathbf{d}(n)$ is a NE of the game. Otherwise, one or all of the following cases can happen: (i) there is a WD $j \in O_{(a,c)}(\mathbf{d}(n))$ that wants to update its best reply by changing its strategy from (a, c) to local computing, (ii) there is a WD $j \in O_{(a,c')}(\mathbf{d}(n))$ that wants to update its best reply by changing its strategy from (a, c'), $c' \neq c$ to local computing, (iii) there is a WD $j \in O_a(\mathbf{d}(n))$ that wants to update its best reply

by changing its offloading strategy from $(a,\cdot)$ to $(a',\cdot)$, $a' \neq a$, (iv) there is a WD $j \in O_{(a',c)}(\mathbf{d}(n))$ that wants to update its best reply by changing its strategy from (a',c), $a' \neq a$ to local computing. Observe that WDs $j \in O_c(\mathbf{d}(n))$ cannot decrease their offloading cost by changing between the ECs, since EC c was WD i's best reply (i.e., $(n_c(\mathbf{d}^*(n-1))+1)/F^c \leq (n_{c'}(\mathbf{d}^*(n-1))+1)/F^{c'}$), and thus it is also a best reply for all WDs $j \in O_c(\mathbf{d}(n))$.

The JPAU algorithm lets WDs to update their best replies in the following order. If case (i) happens, the JPAU algorithm allows one of the WDs $j \in O_{(a,c)}(\mathbf{d}(n))$ to stop offloading. Now, in the updated strategy profile $\mathbf{d}'(n) = (j, d_{-j}(n))$ we have that $n_a(\mathbf{d}'(n)) = n_a(\mathbf{d}^*(n-1))$ and $n_c(\mathbf{d}'(n)) = n_c(\mathbf{d}^*(n-1))$ hold for every AP $a \in \mathcal{A}$ and every EC $c \in \mathcal{C}$, and thus $\mathbf{d}'(n)$ is a NE of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$. Otherwise, if case (ii) happens, the JPAU algorithm allows one of WDs $j \in O_{(a,c')}(\mathbf{d}(n))$ to stop offloading. In the updated strategy profile $\mathbf{d}'(n) = (j, d_{-j}(n))$ we have that $n_a(\mathbf{d}'(n)) = n_a(\mathbf{d}^*(n-1))$ for every AP a, $n_c(\mathbf{d}'(n)) = n_c(\mathbf{d}^*(n-1))+1$, $n_{c'}(\mathbf{d}'(n)) = n_{c'}(\mathbf{d}^*(n-1))-1$ and $n_{c''}(\mathbf{d}'(n)) = n_{c''}(\mathbf{d}^*(n-1))$ for every $c'' \in \mathcal{C} \setminus \{c, c'\}$. Since WD j was offloading its task to EC c' in a NE $\mathbf{d}^*(n-1)$ (i.e. before WD i enter the game) we have that $(n_c(\mathbf{d}^*(n-1))+1)/F^c > n_{c'}(\mathbf{d}^*(n-1))/F^{c'}$. Therefore WDs $k \in O_c(\mathbf{d}'(n))$ can decrease their offloading cost by changing their strategy from offloading to EC c to offloading to EC c'. Let us now consider the updated strategy profile $\mathbf{d}'(n) = ((\cdot, c'), d'_{-k}(n))$ after a WD $k \in O_c(\mathbf{d}'(n))$ performed its best reply $d_k^*(n) = (\cdot, c')$. We have that $n_a(\mathbf{d}'(n)) = n_a(\mathbf{d}^*(n-1))$ and $n_c(\mathbf{d}'(n)) = n_c(\mathbf{d}^*(n-1))$ hold for every AP $a \in \mathcal{A}$ and every EC $c \in \mathcal{C}$, and thus $\mathbf{d}'(n)$ is a NE of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$.

Now, let us assume that neither (i) nor (ii) happened. In that case, the JPAU algorithm allows a sequence of update steps in which WDs $j \in O(\mathbf{d}(n))$ are allowed to decrease their offloading costs. Let us recall that WDs $j \in O(\mathbf{d}(n))$ cannot decrease their offloading cost by changing between ECs, and thus in the resulting sequence of update steps WDs only change between APs. Furthermore, observe that the sequence starts with an update step performed by WD $j \in O_a(\mathbf{d}(n))$, which corresponds to case (iii). It follows from [39] that this sequence of update steps is finite. Observe that after the sequence terminates in the updated strategy profile $\mathbf{d}'(n)$, we have that $n_{a''}(\mathbf{d}'(n)) = n_{a''}(\mathbf{d}^*(n-1))+1$ holds for AP a'' through which a WD started offloading in the last update step, and thus some of the WDs $j \in O_{a''}(\mathbf{d}'(n))$ may want to stop offloading. Observe that the case when there is a WD $j \in O_{(a'',c)}(\mathbf{d}'(n))$ that wants to stop offloading corresponds to case (i) and the case when there is a WD $j \in O_{(a'',c')}(\mathbf{d}'(n))$, $c' \neq c$ that wants to stop offloading corresponds to case (ii). In the discussion above we showed that the JPAU algorithm terminates in a NE of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ in both cases. Let us now consider that none of the previous two cases happened. Then, if there are no WDs $j \in O_{a'}(\mathbf{d}'(n))$, $a' \neq a''$ that want to stop offloading then the JPAU algorithm terminates in a NE because there are no WDs that want to start offloading either because $n_r(\mathbf{d}'(n)) \geq n_r(\mathbf{d}^*(n-1))$, $\forall r \in \mathcal{A} \cup \mathcal{C}$.

Otherwise, let us assume that a WD $j \in O_{(a',c)}(\mathbf{d}'(n))$, $a' \neq a''$ (i.e., $n_{a'}(\mathbf{d}'(n)) = n_{a'}(\mathbf{d}^*(n-1))$) wants to stop offloading because the congestion in EC c increased, i.e., because $n_c(\mathbf{d}'(n)) = n_c(\mathbf{d}^*(n-1))+1$ holds. Observe that if cases (i)-(iii)

did not happen, we have that $a'' = a$, which corresponds to case (iv). After a WD $j \in O_{(a',c)}(\mathbf{d}'(n))$ stops to offload, in the updated strategy profile $\mathbf{d}'(n) = (j, d'_{-j}(n))$ we have that $n_{a''}(\mathbf{d}'(n)) = n_{a''}(\mathbf{d}^*(n-1)) + 1$, $n_{a'}(\mathbf{d}'(n)) = n_{a'}(\mathbf{d}^*(n-1)) - 1$, $n_b(\mathbf{d}'(n)) = n_b(\mathbf{d}^*(n-1))$ for APs $b \in \mathcal{A} \setminus \{a'', a'\}$ and $n_c(\mathbf{d}'(n)) = n_c(\mathbf{d}^*(n-1))$ for every EC $c \in \mathcal{C}$. The JPAU algorithm first allows one of the WDs $k \in O_{a''}(\mathbf{d}'(n))$ to decrease its offloading cost by changing its strategy from $(a'', \cdot)$ to $(a', \cdot)$. If such a WD k exists, after it performs the best reply we have that $n_a(\mathbf{d}'(n)) = n_a(\mathbf{d}^*(n-1))$ and $n_c(\mathbf{d}'(n)) = n_c(\mathbf{d}^*(n-1))$ hold for every AP $a \in \mathcal{A}$ and every EC $c \in \mathcal{C}$ in the updated strategy profile $\mathbf{d}'(n) = ((a', \cdot), d'_{-k}(n))$. Thus, there is no WD that can further decrease its cost and the JPAU algorithm terminates in a NE of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$. Otherwise, if there is no WD $k \in O_{a''}(\mathbf{d}'(n))$ that can decrease its offloading cost by changing the strategy from $(a'', \cdot)$ to $(a', \cdot)$, then the JPAU algorithm allows a sequence of the update steps in which WDs $k \in O(\mathbf{d}'(n))$ are allowed to decrease their offloading costs by changing between the APs. It follows from [39] that this sequence of the update steps is finite. Let us now consider a strategy profile $\mathbf{d}'(n)$ after the last update step in the sequence was performed. We can either have $n_a(\mathbf{d}'(n)) = n_a(\mathbf{d}^*(n-1))$ for every AP $a \in \mathcal{A}$ or $n_{a''}(\mathbf{d}'(n)) = n_{a''}(\mathbf{d}^*(n-1)) + 1$, $n_{a'}(\mathbf{d}'(n)) = n_{a'}(\mathbf{d}^*(n-1)) - 1$ for some $a' \in \mathcal{A} \setminus \{a''\}$ and $n_b(\mathbf{d}'(n)) = n_b(\mathbf{d}^*(n-1))$ for $b \in \mathcal{A} \setminus \{a'', a'\}$. In the first case the JPAU algorithm terminates in a NE of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$ since $n_c(\mathbf{d}'(n)) = n_c(\mathbf{d}^*(n-1))$ still holds for every EC $c \in \mathcal{C}$ (the WDs were not changing between ECs). In the second case we have that some of the WDs $k \in \mathcal{N}_n \setminus O(\mathbf{d}'(n))$ that are performing computation locally may want to start to offload through AP a'. If such a WD does not exist, $\mathbf{d}'(n)$ is a NE of $\Gamma(\mathcal{P}_r^{t,*}, \mathcal{P}_c^{t,*})$, since there is no WD that can further decrease its cost. Otherwise, if such a WD k exists its best reply is (a', c) since $n_c(\mathbf{d}'(n)) = n_c(\mathbf{d}^*(n-1))$ and $(n_c(\mathbf{d}^*(n-1)) + 1)/F^c \leq (n_{c'}(\mathbf{d}^*(n-1)) + 1)/F^{c'}$. Observe that in the updated strategy profile $\mathbf{d}'(n) = ((a', c), d'_{-k}(n))$ we have that $n_{a'}(\mathbf{d}'(n)) = n_{a'}(\mathbf{d}^*(n-1))$ and $n_b(\mathbf{d}'(n)) \geq n_b(\mathbf{d}^*(n-1))$ for $b \in \mathcal{A} \setminus \{a'\}$, and thus WDs $j \in O_{a'}(\mathbf{d}'(n))$ cannot decrease their offloading cost. Furthermore, WDs $j \in O(\mathbf{d}'(n)) \setminus O_{a'}(\mathbf{d}'(n))$ cannot decrease their offloading cost either since they could not do so before WD k started offloading. Finally, we observe that WDs $O_{(a'',c)}(\mathbf{d}'(n))$ do not want to stop offloading because they did not want to do so after the sequence of update steps of type (iii) terminated, and consequently only an update step of type (iv) may happen.

In the following we show that an update step of type (iv) can happen a finite number of times. First, observe that $n_{a'}(\mathbf{d}'(n)) = n_{a'}(\mathbf{d}^*(n-1))$, $a' \neq a''$ holds. Thus, a WD $j \in O_{(a',c)}(\mathbf{d}'(n))$, may want to stop offloading only because the congestion in EC c increased. Second, observe that a WD $j \in O_c(\mathbf{d}'(n))$ for which a best reply in one of the previous steps was to share EC c with $n_c(\mathbf{d}^*(n-1))$ WDs, will not have an incentive to perform an improvement step of type (iv) after a WD $k \in \mathcal{N}_n \setminus O(\mathbf{d}'(n))$ starts to offload to EC c, i.e., after the congestion in EC c increases to $n_c(\mathbf{d}^*(n-1)) + 1$ again. The same holds for all WDs that decide to change their strategy from local computing to offloading to EC c. Consequently, the length of the sequence of update steps of type (iv) is at most $n_c(\mathbf{d}^*(n-1))$, which proves the theorem.

A.3 Proof of Proposition 3

First, observe that in a sequence of update steps of type (iii) each WD $j \in O(\mathbf{d}^*(n-1))$ can deviate at most once. This is due to that when a WD $j \in O(\mathbf{d}^*(n-1))$ moves to an AP a it brings the system to a state where $n_a(\mathbf{d}'(n)) = n_a(\mathbf{d}^*(n-1)) + 1$ holds and since $n_a(\mathbf{d}'(n))$ can only decrease in the following improvement steps, WD j will not have an incentive to deviate again.

In the worst case scenario $A \geq 3$, $C = 1$, all WDs offload their tasks in a NE $\mathbf{d}^*(n-1)$ i.e., $O(\mathbf{d}^*(n-1)) = \mathcal{N}_{n-1}$ and case (iii) happens such that every WD $j \in O(\mathbf{d}^*(n-1))$ changes between APs exactly once. Furthermore, in the worst case scenario, after an $|\mathcal{N}_{n-1}|$ long sequence of update steps of type (iii), one of the WDs stops to offload due to increased congestion only in the EC. Observe that in the resulting strategy profile there is one AP a'' on which the congestion increased, one AP a' on which the congestion decreased and the congestion on the other APs has not changed compared with the congestions in $\mathbf{d}^*(n-1)$. Now, observe that in the worst case scenario each of the WDs that offload through an AP $a \in \mathcal{A} \setminus \{a''\}$ changes between the APs, and consequently in the worst case scenario $n_{a''}(\mathbf{d}^*(n-1)) = 1$. Since the remaining $|\mathcal{N}_{n-1}| - 1$ WDs do not have an incentive to move to the AP a'' because of the increased congestion, it follows from the definition of a best reply that in the resulting sequence each of $|\mathcal{N}_{n-1}| - 1$ WDs updates its strategy at most $A-2$ times. Furthermore, it follows from the proof of Theorem 4 that an update of type (iv) can happen at most $n_c(\mathbf{d}^*(n-1))$ times, where $n_c(\mathbf{d}^*(n-1)) = |\mathcal{N}_{n-1}|$ for $C = 1$ and $O(\mathbf{d}^*(n-1)) = \mathcal{N}_{n-1}$. Finally, we obtain that a NE is reached after at most $|\mathcal{N}_{n-1}| + (1 + (A-2)(|\mathcal{N}_{n-1}| - 1) + 1)|\mathcal{N}_{n-1}|$, which proves the result.

Paper **E**

Joint Wireless and Edge Computing Resource Management with Dynamic Network Slice Selection

Slađana Jošilo and György Dán

submitted to IEEE/ACM Transactions on Networking (ToN).

Joint Wireless and Edge Computing Resource Management with Dynamic Network Slice Selection

Slađana Jošilo and György Dán
School of Electrical Engineering and Computer Science
KTH, Royal Institute of Technology, Stockholm, Sweden
E-mail: {josilo, gyuri}@kth.se

Abstract

Network slicing is a promising approach for enabling low latency computation offloading in edge computing systems. In this paper, we consider an edge computing system under network slicing in which the wireless devices generate latency sensitive computational tasks. We address the problem of joint dynamic assignment of computational tasks to slices, management of radio resources across slices and management of radio and computing resources within slices. We formulate the *Joint Slice Selection and Edge Resource Management* (JSS-ERM) problem as a mixed-integer problem with the objective to minimize the completion time of computational tasks. We show that the JSS-ERM problem is NP-hard and develop an approximation algorithm with bounded approximation ratio based on a game theoretic treatment of the problem. We provide extensive simulation results to show that network slicing can improve the system performance compared to no slicing and that the proposed solution can achieve significant gains compared to the equal slicing policy. Our results also show that the computational complexity of the proposed algorithm is approximately linear in the number of devices.

1 Introduction

Network slicing is emerging as an enabler for providing logical networks that are customized to meet the needs of different kinds of applications, mostly in 5G mobile networks. Horizontal network slices are designed for specific classes of applications, e.g., streaming visual analytics, real-time control, or media delivery, while vertical network slices are designed for specific industries. Slicing is expected to allow flexible and efficient end-to-end provisioning of bandwidth, composition of in-network processing, e.g., in the form of service chains composed of virtual network functions (VNF), and the allocation of dedicated computing resources. At the same time it provides performance isolation. Slicing is particularly appealing in combination with

edge computing, as network slicing could allow low latency access to customized computing services located in edge clouds [1, 2].

Flexibility in network slicing is achieved through service orchestration. Orchestration focuses on the deployment and service-aware adaptation of VNFs and edge cloud services based on predicted workloads. Recent works in the area addressed the joint placement and routing of service function chains, formulated as a virtual network embedding problem [3], and the problem of joint resource dimensioning and routing [4, 5]. Typical objectives are maximization of the service capacity or profit under physical (bandwidth and computational power) resource constraints, or the minimization of the energy consumption subject to satisfying service demand.

Common to the works on service orchestration is that they assume that each application is mapped to a specific slice deterministically, and assume a static resource pool per slice so as to ensure performance isolation [3–5]. A deterministic mapping is, however, not mandatory in practice. While there may be a designated (default) slice for every application, most proposed architectures for network slicing define a set of allowed slices, and the assignment of an application to a slice can be decided dynamically based on the current workload and SLA requirements [6]. The dynamic assignment of applications to slices thus results in a mixture of workloads in the slices, and consequently calls for flexibility in allocating resources to slices.

The importance of resource management across slices has been widely accepted in the case of the radio access network (RAN) [6]. Such inter-slice resource allocation should happen at short time scales, taking into account slice-level service level agreements (SLAs) and technological constraints (e.g., available RAN technology, such as 5GNR or WiFi-Lic). Recent work in the area has focused on system aspects of virtualizing RANs [7], and on the allocation of virtual resource block groups to slices so as to maximize efficiency [8], but has not considered of the potential impact of inter-slice resource management on service orchestration and on the dynamic assignment of applications to slices. It is thus so far unclear how to perform joint resource management within and across slices, considering the orchestration of communication and computing resources simultaneously.

In this paper we address the problem of joint dynamic slice selection, inter-slice radio resource management and intra-slice radio and computing resource management for latency sensitive workloads, and make three important contributions. First, we formulate the joint slice selection and edge resource management (JSS-ERM) problem, and show that it is NP-hard. Second, we analyze the optimal solution structure, and we develop an efficient approximation algorithm with bounded approximation ratio inspired by a game theoretic treatment of the problem. Third, we provide extensive numerical results to show that the resulting system performance significantly outperforms baseline resource allocation policies.

The rest of the paper is organized as follows. Section 2 introduces the system model and Section 3 the problem formulation. Sections 4 and 5 provide the analytical results, and Section 6 shows numerical results. Section 7 discusses related work and Section 8 concludes the paper.

2 System Model

We consider a slicing enabled mobile backhaul including mobile edge computing (MEC) resources that serves a set $\mathcal{N}=\{1,2,\ldots,N\}$ of wireless devices (WDs) that generate computationally intensive tasks. WDs can offload their tasks through a set $\mathcal{A}=\{1,2,\ldots,A\}$ of access points (APs) to a set $\mathcal{C}=\{1,2,\ldots,C\}$ of edge clouds (ECs). APs and ECs form the set $\mathcal{E} \triangleq \mathcal{A} \cup \mathcal{C}$ of edge resources. We denote by $\mathcal{S}=\{1,2,\ldots,S\}$ the set of slices in the network, which include certain combinations of computing resources (e.g., CPUs, GPUs, NPUs and/or FPGAs), optimized for executing some types of tasks. For ease of reference, the key notations used in the paper are summarized in Table 1.

We characterize a task generated by WD i by the size D_i of the input data and by its complexity, which we define as the *expected* number of instructions required to perform the computation. Since the WDs and the slices may have different instruction set architectures, the number of instructions required to execute the same task may also differ. Hence, for a task generated by WD i we denote by L_i and $L_{i,s}$ the expected number of instructions required to perform the computation locally and in slice s, respectively. Similar to other works [9–11], we consider that D_i, L_i and $L_{i,s}$ can be estimated from measurements by applying the methods described in [12–14].

We consider that each WD i generates a computational task at a time; each task is atomic and can be either offloaded for computation or performed locally on the WD it was generated at. In the case of offloading, the WD will be assigned to exactly one slice $s \in \mathcal{S}$ and within the slice to exactly one AP $a \in \mathcal{A}_i \subseteq \mathcal{A}$ through which it can offload the computation to exactly one EC $c \in \mathcal{C}$. Therefore, we define the set of feasible decisions for WD i as $\mathfrak{D}_i \triangleq \{i\} \cup \{(a,c,s) | a \in \mathcal{A}_i, c \in \mathcal{C}, s \in \mathcal{S}\}$ and we use variable $d_i \in \mathfrak{D}_i$ to indicate the decision for WD i's task (i.e., $d_i = i$ indicates that WD i performs the task locally and $d_i = (a,c,s)$ indicates that WD i should offload its task through AP a to EC c in slice s). Furthermore, we define a decision vector $\mathbf{d} \triangleq (d_i)_{i \in \mathcal{N}}$ as the collection of the decisions of all WDs and we define the set $\mathfrak{D} \triangleq \times_{i \in \mathcal{N}} \mathfrak{D}_i$, i.e., the set of all possible decision vectors.

For a decision vector $\mathbf{d} \in \mathfrak{D}$ we define the set $O_{(a,s)}(\mathbf{d}) \triangleq \{i \in \mathcal{N} | d_i = (a,\cdot,s), a \in \mathcal{A}_i, s \in \mathcal{S}\}$ of all WDs that use AP a in slice s and the set $O_a(\mathbf{d}) = \cup_{s \in \mathcal{S}} O_{(a,s)}(\mathbf{d})$ of all WDs that use AP a. Similarly, we define the set $O_{(c,s)}(\mathbf{d}) \triangleq \{i \in \mathcal{N} | d_i = (\cdot,c,s), c \in \mathcal{C}, s \in \mathcal{S}\}$ of all WDs that use EC c in slice s and the set $O_c(\mathbf{d}) = \cup_{s \in \mathcal{S}} O_{(c,s)}(\mathbf{d})$ of all WDs that use EC c. Finally, we define the local computing singleton set $O_i(\mathbf{d}) \subset \{i, \emptyset\}$ for WD i (i.e., $O_i(\mathbf{d}) = \{i\}$ when WD i performs the computation locally and $O_i(\mathbf{d}) = \emptyset$ otherwise) and the set $O_l(\mathbf{d}) = \cup_{i \in \mathcal{N}} O_i(\mathbf{d})$ of all WDs that perform the computation locally.

Fig. 1 shows an example of a slicing enabled MEC system that consists of $N = 7$ WDs, $C = 2$ ECs and $A = 3$ APs and $S = 4$ slices. In this example we have that 2 out of 7 WDs perform the computation locally and 5 out of 7 WDs offload their tasks. In what follows we discuss our models of communication and computing resources.

Table 1: Summary of key notations

Notation	Description
$\mathcal{N}$	Set of N WDs
$\mathcal{A}$	Set of A APs
$\mathcal{A}_i$	Set of APs available for offloading to WD i
$\mathcal{C}$	Set of C ECs
$\mathcal{E}$	Set of APs and ECs, $\mathcal{E} = \mathcal{A} \cup \mathcal{C}$
$\mathcal{S}$	Set of S slices
$\mathcal{P}_b$	Inter-slice radio resource allocation policy
$\mathcal{P}^s_{w_a}$	Intra-slice radio resource allocation policy
$\mathcal{P}^s_{w_c}$	Intra-slice computing power allocation policy
b^s_a	Inter-slice radio resource provisioning coefficient
$w^s_{i,a}$	Intra-slice radio resource provisioning coefficient
$w^s_{i,c}$	Intra-slice computing power provisioning coefficient
D_i	Mean size of the input data for WD i
L_i	Mean WD i's task complexity when executing locally
$L_{i,s}$	Mean WD i's task complexity when offloading in slice s
F^l_i	Computational capability of WD i
T^{ex}_i	Local execution time of WD i
$R_{i,a}$	Uplink PHY rate of WD i towards AP a
F^s_c	Computing capability of EC c in slice s
$\mathfrak{D}_i$	Set of feasible offloading decisions for WD i
d_i	Offloading decision for WD i, $d_i \in \mathfrak{D}_i$
$\mathbf{d}$	Offloading decision vector, $\mathbf{d} = (d_i)_{i \in \mathcal{N}}$
$O_{(a,s)}(\mathbf{d})$	Set of WDs offloading in slice s through AP a in $\mathbf{d}$
$O_a(\mathbf{d})$	Set of WDs offloading through AP a in $\mathbf{d}$
$O_{(c,s)}(\mathbf{d})$	Set of WDs offloading in slice s to EC c in $\mathbf{d}$
$O_c(\mathbf{d})$	Set of WDs offloading to EC c in $\mathbf{d}$
$O_l(\mathbf{d})$	Set of WDs performing the computation locally in $\mathbf{d}$
$W^s_{i,a}(\mathbf{d}, \mathcal{P}_b, \mathcal{P}^s_{w_a})$	Uplink rate of WD $i \in O_{(a,s)}(\mathbf{d})$
$T^{tx,s}_{i,a}(\mathbf{d}, \mathcal{P}_b, \mathcal{P}^s_{w_a})$	Transmission time of WD $i \in O_{(a,s)}(\mathbf{d})$
$F^s_{i,c}(\mathbf{d}, \mathcal{P}^s_{w_c})$	Computing capability of WD $i \in O_{(c,s)}(\mathbf{d})$
$T^{ex,s}_{i,c}(\mathbf{d}, \mathcal{P}^s_{w_c})$	Task execution time of WD $i \in O_{(c,s)}(\mathbf{d})$
$C_i(\mathbf{d}, \mathcal{P}_b, \mathcal{P}^s_{w_a}, \mathcal{P}^s_{w_c})$	Cost of WD i
$C^s(\mathbf{d}, \mathcal{P}_b, \mathcal{P}^s_{w_a}, \mathcal{P}^s_{w_c})$	Cost in slice s
$C(\mathbf{d}, \mathcal{P}_b, \mathcal{P}_{w_a}, \mathcal{P}_{w_c})$	System cost

2.1 Communication Resources

Communication resources in the system are managed at two levels: at the network level and at the slice level.

At the network level, the radio resources of each AP $a \in \mathcal{A}$ are shared across the

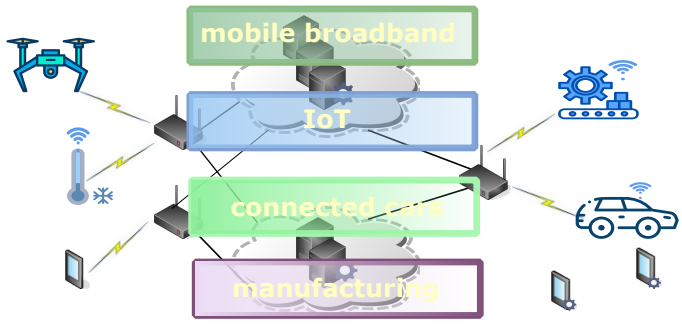

Figure 1: An example of a slicing enabled MEC system that consists of $N = 7$ WDs, $C = 2$ ECs and $A = 3$ APs and $S = 4$ slices.

slices according to the *inter-slice radio resource allocation policy* $\mathcal{P}_b : \mathfrak{D} \to \mathbb{R}_{[0,1]}^{|\mathcal{A}|\times|\mathcal{S}|}$, which determines the inter-slice radio resource provisioning coefficients $b_a^s \in [0,1]$, $\forall (a,s) \in \mathcal{A} \times \mathcal{S}$ such that $\sum_{s\in\mathcal{S}} b_a^s \leq 1$, $\forall a \in \mathcal{A}$.

At the slice level, the radio resources assigned to each slice $s \in \mathcal{S}$ are shared among the WDs according to an *intra-slice radio resource allocation policy* $\mathcal{P}_{w_a}^s : \mathfrak{D} \to \mathbb{R}_{[0,1]}^{|\mathcal{A}|\times|\mathcal{N}|}$, which determines the intra-slice radio resource provisioning coefficients $w_{i,a}^s \in [0,1]$, $\forall a \in \mathcal{A}$ and $\forall i \in O_{(a,s)}(\mathbf{d})$ such that $\sum_{i\in O_{(a,s)}(\mathbf{d})} w_{i,a}^s \leq 1$, $\forall (a,s) \in \mathcal{A} \times \mathcal{S}$.

We denote by $R_{i,a}$ the achievable PHY rate of WD i at AP a. $R_{i,a}$ depends on physical signal characteristics, such as path loss and fading, and on the modulation-coding scheme. Given $R_{i,a}$ we can express the actual uplink rate of WD i at AP a in slice s as

$$W_{i,a}^s(\mathbf{d}, \mathcal{P}_b, \mathcal{P}_{w_a}^s) = b_a^s w_{i,a}^s R_{i,a}. \tag{1}$$

The uplink rate (1) together with the input data size D_i determines the transmission time of WD $i \in O_{(a,s)}(\mathbf{d})$,

$$T_{i,a}^{tx,s}(\mathbf{d}, \mathcal{P}_b, \mathcal{P}_{w_a}^s) = \frac{D_i}{W_{i,a}^s(\mathbf{d}, \mathcal{P}_b, \mathcal{P}_{w_a}^s)}. \tag{2}$$

Similar to previous works [10, 15–17] we make the assumption that the time needed to transmit the results of the computation from the EC to the WD can be neglected because for many applications (e.g., face recognition and tracking) the size of the output data is significantly smaller than the size D_i of the input data.

2.2 Computing Resources

Our system model distinguishes between edge cloud resources and local computing resources.

2.2.1 Edge Cloud Resources

We consider that each slice $s \in \mathcal{S}$ is equipped with a certain combination of computing resources optimized for executing specific types of tasks (e.g, CPUs, GPUs, NPUs, FPGAs), and we denote by F_c^s the computing capability of EC c in slice s. The computing resources within a slice are shared among the WDs according to the *intra-slice computing power allocation policy* $\mathcal{P}_{w_c}^s : \mathfrak{D} \to \mathbb{R}_{[0,1]}^{|\mathcal{C}|\times|\mathcal{N}|}$, which determines the intra-slice computing power provisioning coefficients $w_{i,c}^s \in [0,1]$, $\forall c \in \mathcal{C}$ and $\forall i \in O_{(c,s)}(\mathbf{d})$ such that $\sum_{i \in O_{(c,s)}(\mathbf{d})} w_{i,c}^s = 1$, $\forall (c,s) \in \mathcal{C} \times \mathcal{S}$.

Given the computing capability F_c^s we can express the computing capability allocated to WD i in EC c in slice s as

$$F_{i,c}^s(\mathbf{d}, \mathcal{P}_{w_c}^s) = w_{i,c}^s F_c^s. \tag{3}$$

In order to account for the diversity of computing resources provided by different slices we use the coefficient $h_{i,s} \in \mathbb{R}_{\geq 0}$ to capture how well a slice s is tailored for executing a task generated by WD i and we express the expected number of instructions $L_{i,s}$ required to execute a task generated by WD i in slice s as $L_{i,s} = L_i / h_{i,s}$ (i.e., a high $h_{i,s}$ indicates that a task generated by WD i is a good match for the computing resources in slice s). Thus, in our model the computing capability (3) together with the expected task complexity $L_{i,s}$ determines the task execution time of WD $i \in O_{(c,s)}(\mathbf{d})$ as

$$T_{i,c}^{ex,s}(\mathbf{d}, \mathcal{P}_{w_c}^s) = \frac{L_{i,s}}{F_{i,c}^s(\mathbf{d}, \mathcal{P}_{w_c}^s)}. \tag{4}$$

2.2.2 Local Computing Resources

We denote by F_i^l the computing capability of WD i and we express the local execution time T_i^{ex} of WD i as

$$T_i^{ex} = \frac{L_i}{F_i^l}. \tag{5}$$

2.3 Cost Model

We define the system cost as the aggregate completion time of all WDs. Before providing a formal definition, we introduce the shorthand notation

$$E_{i,e}^s = \begin{cases} \frac{D_i}{R_{i,e}} & \text{if } i \in \mathcal{N}, e \in \mathcal{E} \cap \mathcal{A}, s \in \mathcal{S} \\ \frac{L_{i,s}}{F_e^s} & \text{if } i \in \mathcal{N}, e \in \mathcal{E} \cap \mathcal{C}, s \in \mathcal{S}, \end{cases} \tag{6}$$

$$b_e^s = \begin{cases} b_e^s & \text{if } e \in \mathcal{E} \cap \mathcal{A}, s \in \mathcal{S} \\ 1 & \text{if } e \in \mathcal{E} \cap \mathcal{C}, s \in \mathcal{S}. \end{cases} \tag{7}$$

Cost of WD i: When offloading, the task completion time consists of two parts: the time needed to transmit the data pertaining to a task through an AP and the time needed to execute a task in an EC. In the case of local computing, the task completion time depends only on the local execution time. Therefore, the cost of WD i can be expressed as

$$C_i(\mathbf{d},\mathcal{P}_b,\mathcal{P}^s_{w_a},\mathcal{P}^s_{w_c})=\begin{cases}\frac{E^s_{i,a}}{b^s_a w^s_{i,a}}+\frac{E^s_{i,c}}{w^s_{i,c}}, & I_{\{d_i=(a,c,s)\}}=1,\\ T^{ex}_i, & I_{\{d_i=i\}}=1.\end{cases} \tag{8}$$

where $I_{\{d_i=d\}}=1$ if $d_i=d$ and $I_{\{d_i=d\}}=0$ otherwise.

Cost per slice: We express the cost in slice s as

$$C^s(\mathbf{d},\mathcal{P}_b,\mathcal{P}^s_{w_a},\mathcal{P}^s_{w_c})=\sum_{e\in\mathcal{E}}\sum_{i\in O_{(e,s)}(\mathbf{d})}\frac{E^s_{i,e}}{b^s_e w^s_{i,e}}. \tag{9}$$

System cost: Finally, we express the system cost as

$$C(\mathbf{d},\mathcal{P}_b,\mathcal{P}_{w_a},\mathcal{P}_{w_c})=\sum_{s\in\mathcal{S}}C^s(\mathbf{d},\mathcal{P}_b,\mathcal{P}^s_{w_a},\mathcal{P}^s_{w_c})+\sum_{i\in O_l(\mathbf{d})}C^l_i, \tag{10}$$

where $(\mathcal{P}_{w_a},\mathcal{P}_{w_c})=((\mathcal{P}^s_{w_a},\mathcal{P}^s_{w_c}))_{s\in\mathcal{S}}$ denotes the collection of slices' policies.

3 Problem Formulation

We consider that the network operator aims at minimizing the system cost $C(\mathbf{d},\mathcal{P}_b,\mathcal{P}_{w_a},\mathcal{P}_{w_c})$ by finding an optimal vector $\hat{\mathbf{d}}$ of offloading decisions, and an optimal collection $(\hat{\mathcal{P}}_b,\hat{\mathcal{P}}_{w_a},\hat{\mathcal{P}}_{w_c})$ of policies for sharing the edge resources across slices and within slices. We refer to the problem as the *Joint Slice Selection and Edge Resource Management* (JSS-ERM) problem. Since the WDs generate atomic tasks that cannot be further split, the JSS-ERM is a mixed-integer optimization problem, and can be formulated as

$$\min_{\mathbf{d},\mathcal{P}_b,\mathcal{P}_{w_a},\mathcal{P}_{w_c}} C(\mathbf{d},\mathcal{P}_b,\mathcal{P}_{w_a},\mathcal{P}_{w_c}) \tag{11}$$

$$\text{s.t.}\sum_{d\in\mathfrak{D}_i} I_{\{d_i=d\}}=1,\forall i\in\mathcal{N}, \tag{12}$$

$$C_i(\mathbf{d},\mathcal{P}_b,\mathcal{P}^s_{w_a},\mathcal{P}^s_{w_c})\le T^{ex}_i,\forall i\in\mathcal{N}, \tag{13}$$

$$\sum_{s\in\mathcal{S}} b^s_a\le 1,\forall a\in\mathcal{A}, \tag{14}$$

$$\sum_{j\in O_{(e,s)}(\mathbf{d})} w^s_{j,e}\le 1,\forall e\in\mathcal{E},\forall s\in\mathcal{S}, \tag{15}$$

$$b^s_a\ge 0,\forall a\in\mathcal{A},\forall s\in\mathcal{S}, \tag{16}$$

$$w^s_{i,e}\ge 0,\forall i\in\mathcal{N},\forall e\in\mathcal{E},\forall s\in\mathcal{S}. \tag{17}$$

Constraint (12) enforces that each WD either performs the computation locally or offloads its task to exactly one logical resource $(a, c, s) \in \mathcal{A} \times \mathcal{C} \times \mathcal{S}$; constraint (13) ensures that the task completion time in the case of offloading is not greater than the task completion time in the case of local computing; constraint (14) enforces a limitation on the amount of communication resources of an AP that can be provided to each slice; constraint (15) enforces a limitation on the amount of communication resources of an AP and the amount of computing resources of an EC that can be provided to each WD in each slice.

Theorem 1. *The JSS-ERM defined by (11)-(17) is NP-hard.*

Proof. We provide the proof in Section 4.2. □

In what follows we develop an approximation scheme for the JSS-ERM problem based on decomposition of the problem, and by adopting a game theoretic interpretation of one of the subproblems.

4 Network Slice Orchestration and Edge Resource Allocation

In what follows we show that the JSS-ERM problem can be solved through solving a series of smaller optimization problems. To do so, we start with considering the problem of finding the collection $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of optimal resource allocation policies for a given vector $\mathbf{d}$ of offloading decisions.

Lemma 1. *Consider an offloading decision vector $\boldsymbol{d}$ for which the constraint (13) can be satisfied. Furthermore, define the problem of finding a collection $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of optimal resource allocation policies as*

$$\min_{\mathcal{P}_b, \mathcal{P}_{w_a}, \mathcal{P}_{w_c}} C(\boldsymbol{d}, \mathcal{P}_b, \mathcal{P}_{w_a}, \mathcal{P}_{w_c}) \tag{18}$$

$$s.t. (13) - (17). \tag{19}$$

Then, the collection $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of optimal resource allocation policies sets the provisioning coefficients according to

$$\hat{w}^s_{i,e} = \frac{\sqrt{E^s_{i,e}}}{\sum\limits_{j \in O_{(e,s)}(\boldsymbol{d})} \sqrt{E^s_{j,e}}}, \forall e \in \mathcal{E}, \forall s \in \mathcal{S}, \forall i \in O_{(e,s)}(\boldsymbol{d}), \tag{20}$$

$$\hat{b}^s_a = \frac{\sum\limits_{j \in O_{(a,s)}(\boldsymbol{d})} \sqrt{E^s_{j,a}}}{\sum\limits_{s' \in \mathcal{S}} \sum\limits_{j \in O(a,s')(\boldsymbol{d})} \sqrt{E^{s'}_{j,a}}}, \forall a \in \mathcal{A}, \forall s \in \mathcal{S}. \tag{21}$$

Proof. First, observe that constraint (13) can be omitted since we assumed that the decision vector $\mathbf{d}$ is such that constraint (13) can be satisfied. Furthermore, by inspecting the leading minors of the Hessian matrix of the objective function (18) it is easy to show that the matrix is positive semidefinite on the domain defined by (19), and thus problem (18)-(19) is convex. Therefore, the optimal solution of the problem must satisfy the Karush–Kuhn–Tucker (KKT) conditions and thus we can formulate the corresponding Lagrangian dual problem. To do so, let us define $\mathbf{b} \triangleq (b_a^s)_{s\in\mathcal{S},a\in\mathcal{A}}$ and $\mathbf{w}^s \triangleq (w_{i,e}^s)_{i\in\mathcal{N},e\in\mathcal{E}}$, and let us introduce non-negative Lagrange multiplier vectors $\boldsymbol{\alpha} = (\alpha_a)_{a\in\mathcal{A}}$, $\boldsymbol{\beta} = (\beta_e^s)_{e\in\mathcal{E},s\in\mathcal{S}}$, $\boldsymbol{\gamma} = (\gamma_a^s)_{a\in\mathcal{A},s\in\mathcal{S}}$ and $\boldsymbol{\delta} = (\delta_{i,e}^s)_{i\in O_{(e,s)}(\mathbf{d}),e\in\mathcal{E},s\in\mathcal{S}}$ for constraints in (19), respectively. Next, let us define the Lagrangian dual problem corresponding to problem (18)-(19) as $\max\limits_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta}\succeq 0} \min\limits_{\mathbf{b},\mathbf{w}\succeq 0} \mathcal{L}(\mathbf{b},\mathbf{w},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta})$, where the Lagrangian is given by

$$\begin{aligned}\mathcal{L}(\mathbf{b},\mathbf{w},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta}) = &\sum_{s'\in\mathcal{S}}\sum_{e'\in\mathcal{E}} \frac{1}{b_{e'}^{s'}}\Big(\sum_{j\in O_{(e',s')}(\mathbf{d})} \frac{E_{j,e'}^{s'}}{w_{j,e'}^{s'}}\Big) + \\ &\sum_{a'\in\mathcal{A}} \alpha_{a'}\Big(\sum_{s'\in\mathcal{S}} b_{a'}^{s'} - 1\Big) + \sum_{e'\in\mathcal{E}}\sum_{s'\in\mathcal{S}} \beta_{e'}^{s'}\Big(\sum_{j\in O_{(e',s')}(\mathbf{d})} w_{j,e'}^{s'} - 1\Big)\Big) \\ &- \sum_{a'\in\mathcal{A}}\sum_{s'\in\mathcal{S}} \gamma_{a'}^{s'} b_{a'}^{s'} - \sum_{e'\in\mathcal{E}}\sum_{s'\in\mathcal{S}}\sum_{j\in O_{(e',s')}(\mathbf{d})} \delta_{j,e'}^{s'} w_{j,e'}^{s'} + \sum_{j\in O_l(\mathbf{d})} C_j^l.\end{aligned}$$

Now, we can express the KKT conditions as follows

$$\textbf{stationarity:} \quad \sum_{j\in O_{(a,s)}(\mathbf{d})} \frac{E_{j,a}^s}{w_{j,a}^s}\cdot\frac{1}{(b_a^s)^2} = \alpha_a - \gamma_a^s, a\in\mathcal{A}, s\in\mathcal{S}, \tag{22}$$

$$\frac{E_{i,e}^s}{b_e^s (w_{i,e}^s)^2} = \beta_e^s - \delta_{i,e}^s, e\in\mathcal{E}, s\in\mathcal{S}, i\in O_{(e,s)}(\mathbf{d}), \tag{23}$$

$$\textbf{pr. feasibility:} \quad (19), \tag{24}$$

$$\textbf{du. feasibility:} \quad \boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta} \succeq 0, \tag{25}$$

$$\textbf{co. slackness:} \quad \alpha_a\Big(\sum_{s'\in\mathcal{S}} b_a^{s'} - 1\Big), a\in\mathcal{A}, \tag{26}$$

$$\beta_e^s\Big(\sum_{j\in O_{(e,s)}(\mathbf{d})} w_{j,e}^s - 1\Big) = 0, e\in\mathcal{E}, s\in\mathcal{S}, \tag{27}$$

$$-\gamma_a^s b_a^s = 0, a\in\mathcal{A}, s\in\mathcal{S} \tag{28}$$

$$-\delta_{i,e}^s w_{i,e}^s = 0, e\in\mathcal{E}, s\in\mathcal{S}, i\in O_{(e,s)}(\mathbf{d}). \tag{29}$$

We proceed with finding $\hat{w}_{i,e}^s$. First, from the KKT dual feasibility condition $\boldsymbol{\delta}\succeq 0$ and complementary slackness condition (29) we obtain that $\delta_{i,e}^s = 0$ must hold for every $e\in\mathcal{E}$, $s\in\mathcal{S}$ and $i\in O_{(e,s)}(\mathbf{d})$ as otherwise $w_{i,e}^s = 0$ would lead to infinite value of the objective function. Then, from the KKT stationarity condition (23) and complementary slackness condition (27) we obtain the expression (20) for coefficients $\hat{w}_{i,e}^s$. Finally, by substituting expression (20) into the KKT stationarity condition (22)

and by following the same approach as for finding $\hat{w}^s_{i,e}$ we obtain the expression (21) for coefficients $\hat{b}^s_a$, which proves the result. □

As a first step in the decomposition, let us consider the problem of finding the optimal collection $(\mathcal{P}^*_{w_a}, \mathcal{P}^*_{w_c}) = ((\mathcal{P}^{s,*}_{w_a}, \mathcal{P}^{s,*}_{w_c}))_{s\in\mathcal{S}}$ of resource allocation policies of slices for a given vector $\mathbf{d}$ of offloading decisions and a given policy $\mathcal{P}_b$.

Proposition 1. *Consider an offloading decision vector $\boldsymbol{d}$ for which constraint (13) can be satisfied and a policy $\mathcal{P}_b$ for setting the inter-slice radio resource provisioning coefficients $b^s_a, \forall a \in \mathcal{A}, \forall s \in \mathcal{S}$. Then the solution to the problem*

$$\min_{\mathcal{P}^s_{w_a}, \mathcal{P}^s_{w_c}} C^s(\boldsymbol{d}, \mathcal{P}_b, \mathcal{P}^s_{w_a}, \mathcal{P}^s_{w_c}) \tag{30}$$

$$s.t. (13), (15), (17). \tag{31}$$

is given by (20), i.e., $(\mathcal{P}^{s,*}_{w_a}, \mathcal{P}^{s,*}_{w_c}) = (\hat{\mathcal{P}}^s_{w_a}, \hat{\mathcal{P}}^s_{w_c}), \forall s \in \mathcal{S}$.

Proof. The result can be proved by following the approach presented in the proof of Lemma 1. □

As a second step, let us consider the problem of finding an optimal policy $\mathcal{P}^*_b$ for a given vector $\mathbf{d}$ of offloading decisions $\mathbf{d}$ and the optimal collection $(\mathcal{P}^*_{w_a}, \mathcal{P}^*_{w_c}) = (\hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of the slices' policies.

Proposition 2. *Consider an offloading decision vector $\boldsymbol{d}$ for which the constraint (13) can be satisfied. Furthermore, let us substitute (20) into (11)-(17) and define the problem of finding an optimal inter-slice radio resource allocation policy $\mathcal{P}^*_b$, i.e., a solution to*

$$\min_{\mathcal{P}_b} \sum_{s'\in\mathcal{S}} \sum_{a'\in\mathcal{A}} \frac{1}{b^{s'}_{a'}} \Big(\sum_{j\in O_{(a',s')}(\boldsymbol{d})} \sqrt{E^{s'}_{j,a'}} \Big)^2 \tag{32}$$

$$s.t. (13), (14) \text{ and } (16). \tag{33}$$

Then, the optimal inter-slice radio resource allocation policy $\mathcal{P}^*_b$ sets the inter-slice provisioning coefficients according to (21), i.e., $\mathcal{P}^*_b = \hat{\mathcal{P}}_b$.

Proof. The result can be proved by following the approach presented in the proof of Lemma 1. □

By combining the above two results, we are now ready to show that the JSS-ERM problem can be decomposed into a sequence of optimization problems.

Theorem 2. *The solution to problem (18)-(19) can be obtained by finding the optimal policies $(\hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ first, and finding the optimal policy $\hat{\mathcal{P}}_b$ second, i.e.,*

$$\min_{\mathcal{P}_b, \mathcal{P}_{w_a}, \mathcal{P}_{w_c}} C(\boldsymbol{d}, \mathcal{P}_b, \mathcal{P}_{w_a}, \mathcal{P}_{w_c}) = \min_{\mathcal{P}_b} \min_{\mathcal{P}_{w_a}, \mathcal{P}_{w_c}} C(\boldsymbol{d}, \mathcal{P}_b, \mathcal{P}_{w_a}, \mathcal{P}_{w_c}). \tag{34}$$

Proof. The result follows from the proofs of Lemma 1, Proposition 1 and Proposition 2. □

Furthermore, as the next theorem shows, we can use this decomposition structure also for computing the optimal offloading decision vector.

Theorem 3. *The solution to problem (11)-(17) can be obtained by finding the optimal collection* $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ *of resource allocation policies first, and finding an optimal offloading decision vector* $\hat{\boldsymbol{d}}$ *second, i.e.,*

$$\min_{\boldsymbol{d},\mathcal{P}_b,\mathcal{P}_{w_a},\mathcal{P}_{w_c}} C(\boldsymbol{d},\mathcal{P}_b,\mathcal{P}_{w_a},\mathcal{P}_{w_c}) = \min_{\boldsymbol{d}} \min_{\mathcal{P}_b} \min_{\mathcal{P}_{w_a},\mathcal{P}_{w_c}} C(\boldsymbol{d},\mathcal{P}_b,\mathcal{P}_{w_a},\mathcal{P}_{w_c}). \tag{35}$$

Proof. It is easy to see that the exact values of the provisioning coefficients are functions of $\hat{\mathbf{d}}$. However, the optimal policies according to which the resources are shared are the same for every offloading decision vector $\mathbf{d} \in \mathfrak{D}$, as defined by (20) and (21). Therefore, one can solve the problem (18)-(19) first, assuming an arbitrary offloading decision vector $\mathbf{d}$, and then given the solution $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of (18)-(19) find the optimal offloading decision vector $\hat{\mathbf{d}}$ that will determine the exact values of the provisioning coefficients. This proves the result. □

4.1 Discussion and Practical Implications

So far we have shown that the JSS-ERM problem can be decomposed into a $S+2$ coupled resource allocation problems that can be solved sequentially. It is of interest to discuss the relationship between the decomposition and the potential implementation of a resource allocation and orchestration framework.

The proposed decomposition results in an optimization problem to be solved at the network level (eqns. (32)-(33)) and one in each slice ((eqns. (30)-(31))), followed by the problem of finding an optimal offloading decision vector. This structure is aligned with the slice-based network architecture proposed in [6], where inter-slice radio resource allocation and service orchestration are performed by a centralized entity, the *slice resource orchestrator* (SRO), while intra-slice radio and computing resource management is performed by the slices themselves, i.e., each slice manages its own radio and computing resources.

Fig. 2 illustrates the interaction between the SRO and slices in the potential implementation of a resource allocation and orchestration framework.

4.2 Problem Complexity

In what follows we provide a result concerning the complexity of the JSS-ERM problem. For notational convenience let us first define the set of resources $\tilde{\mathcal{R}} \triangleq$

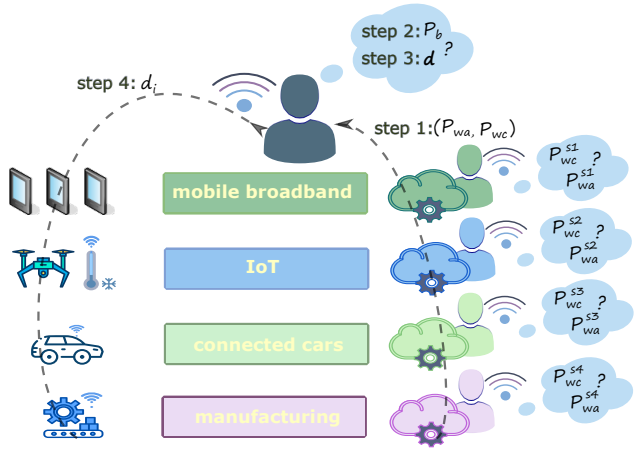

Figure 2: An example of the potential implementation of a resource allocation and orchestration framework.

$\{\{\mathcal{A}\times\mathcal{S}\}\cup\{\mathcal{C}\times\mathcal{S}\}\cup\mathcal{N}\}$ and let us introduce the following shorthand notation

$$q_{i,(a,s)} \triangleq \sqrt{\frac{D_i}{R_{i,a}}},\ q_{i,(c,s)} \triangleq \sqrt{\frac{L_i}{h_{i,s}}},\ q_{i,i} \triangleq \sqrt{L_i},\ q_r(\mathbf{d}) \triangleq \sum_{j\in O_r(\mathbf{d})} q_{j,r}, \forall r \in \tilde{\mathcal{R}}. \quad (36)$$

First, by substituting (20) into (8) and by using the notation introduced in (36), we can express the cost of WD i under a policy $\mathcal{P}_b$ and the collection $(\hat{\mathcal{P}}_{w_a},\hat{\mathcal{P}}_{w_c})$ of optimal allocation policies of slices as

$$\tilde{C}_i(\mathbf{d}) = \sum_{r\in\tilde{\mathcal{R}}_{d_i}} m_r q_{i,r} q_r(\mathbf{d}), \quad (37)$$

where $\tilde{\mathcal{R}}_{d_i}$ is the set of resources that WD i uses for performing its task in $\mathbf{d}$ (i.e., $\tilde{\mathcal{R}}_{d_i}\subset\tilde{\mathcal{R}}$) and $m_{(a,s)} = 1/b_a^s$, $m_{(c,s)} = 1/F_c^s$ and $m_i = 1/F_i^l$.

Second, by summing the expressions (37) over all WDs $i\in\mathcal{N}$ and by reordering the summations we can express the system cost (10) under a policy $\mathcal{P}_b$ and the collection $(\hat{\mathcal{P}}_{w_a},\hat{\mathcal{P}}_{w_c})$ of optimal allocation policies of slices as

$$\tilde{C}(\mathbf{d}) = \sum_{r\in\tilde{\mathcal{R}}} m_r q_r^2(\mathbf{d}). \quad (38)$$

Next, let us define the set of resources $\bar{\mathcal{R}} \triangleq \{\mathcal{A}\cup\{\mathcal{C}\times\mathcal{S}\}\cup\mathcal{N}\}$ and a coefficient $q_{i,a} \triangleq q_{i,(a,s)} = \sqrt{D_i/R_{i,a}}$. By substituting (21) into (37), we can express the cost of WD i under the collection $(\hat{\mathcal{P}}_b,\hat{\mathcal{P}}_{w_a},\hat{\mathcal{P}}_{w_c})$ of optimal allocation policies as

$$\bar{C}_i(\mathbf{d}) = \sum_{r\in\bar{\mathcal{R}}_{d_i}} m_r q_{i,r} q_r(\mathbf{d}), \quad (39)$$

where $\bar{\mathcal{R}}_{d_i}$ is the set of resources that WD i uses for performing its task in $\mathbf{d}$ (i.e., $\bar{\mathcal{R}}_{d_i} \subset \bar{\mathcal{R}}$) and $m_a = 1$.

Finally, by summing the expressions (39) over all WDs $i \in \mathcal{N}$ and by reordering the summations we can express the system cost (38) under the collection $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of optimal allocation policies as

$$\bar{C}(\mathbf{d}) = \sum_{r \in \bar{\mathcal{R}}} m_r q_r^2(\mathbf{d}). \tag{40}$$

Theorem 4. *Consider the problem of finding the optimal vector $\hat{\boldsymbol{d}}$ of offloading decisions of WDs under the collection $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of optimal allocation policies that set provisioning coefficients according to (20) and (21)*

$$\min_{\boldsymbol{d}} \bar{C}(\boldsymbol{d}) \tag{41}$$

$$s.t. (12). \tag{42}$$

Problem (41)-(42) is NP-hard.

Proof. We prove the NP-hardness of the problem by reduction from the *Minimum Sum of Squares* problem (SP19 problem in [18]): given a finite set $\mathcal{B}$, a size $s(b) \in \mathbb{Z}^+, \forall b \in \mathcal{B}$ and positive integers $K \leq |\mathcal{B}|$ and J, the question is whether $\mathcal{B}$ can be partitioned into K disjoint subsets $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K$ such that $\sum_{k=1}^{K} \Big(\sum_{b \in \mathcal{B}_k} s(b) \Big)^2 \leq J$.

For the reduction we set $S = 1$, $C = 0$ and $F_i^l = 0$, $\forall i \in \mathcal{N}$, i.e., in this simplified version of the problem $\tilde{\mathcal{R}} = \mathcal{A}$. Next, we let $\mathcal{N} = \mathcal{B}$, $|\mathcal{A}| = K$, $R_{i,a} = R_i$, $\forall i \in \mathcal{N}$, $\forall a \in \mathcal{A}$ and $\sqrt{D_i / R_i} = s(b)$. Then, it follows from (38) that the optimal solution of (41)-(42) provides the solution to the SP19 problem. As SP19 is NP-hard, problem (41)-(42) is also NP-hard, which proves the theorem. □

Proof of Theorem 1. The result follows from Theorem 3 and Theorem 4. □

5 Approximation Scheme for the JSS-ERM Problem

In what follows we propose the *choose offloading slice* (COS) algorithm for computing an approximation to the optimal solution of the JSS-ERM problem. In particular, the algorithm serves as an approximation scheme to the problem of finding an optimal offloading decision vector. The algorithm starts from an offloading decision vector $\mathbf{d}^0$ in which all WDs perform computation locally and it lets WDs update their offloading decisions one at a time, based on their local cost function $\tilde{C}_i(\mathbf{d})$. We show the pseudo code of the algorithm in Fig. 3.

$\mathbf{d}^* = \text{COS}(\mathbf{d}^0, \mathcal{P}_b, \mathcal{P}_w^*, \mathcal{P}_w^*)$

$\mathbf{d} \leftarrow \mathbf{d}^0$
while $\exists\, WD\, j \in \mathcal{N}$ *s.t.* $d_j \neq \arg\min_{d'_j \in \mathfrak{D}_j} \tilde{C}_j(d'_j, d_{-j})$
  $d_j^* = \arg\min_{d'_j \in \mathfrak{D}_j} \tilde{C}_j(d'_j, d_{-j})$, $\mathbf{d} = (d_j^*, d_{-j})$
end
$\mathbf{d}^* = \mathbf{d}$

Figure 3: Pseudo code of the COS algorithm.

Theorem 5. *Consider an allocation policy $\mathcal{P}_b$ and the collection $(\hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of optimal allocation policies of slices. The COS algorithm terminates after a finite number of the iterations.*

Proof. The proof is based on a game theoretic treatment of the problem

$$\min_{\mathbf{d}} \tilde{C}(\mathbf{d}) \tag{43}$$

$$\text{s.t.}(12), \tag{44}$$

in which the inter-slice radio resource provisioning coefficients are set according to an arbitrary policy $\mathcal{P}_b$ and the intra-slice radio and computing power provisioning coefficients are set according to the optimal policies $\hat{\mathcal{P}}_{w_a}$ and $\hat{\mathcal{P}}_{w_c}$, respectively.

In what follows we show that the problem (43)-(44) can be interpreted as a congestion game $\Gamma(\mathcal{P}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c}) =< \mathcal{N}, (\mathfrak{D}_i)_{i\in\mathcal{N}}, (\tilde{C}_i)_{i\in\mathcal{N}} >$ with resource-dependent weights $q_{i,r}$, $i \in \mathcal{N}$, $r \in \tilde{\mathcal{R}}$, and the cost of WD i in the resulting game is given by (37). First, observe that $q_{i,r}$ can be interpreted as the weight that WD i contributes to the congestion when using resource $r \in \tilde{\mathcal{R}}$ and thus $q_r(\mathbf{d})$ can be interpreted as the total congestion on resource r in strategy profile $\mathbf{d}$. This in fact implies that the cost (37) of WD i in strategy profile $\mathbf{d}$ depends on its own resource-dependent weights $q_{i,r}$ and on the total congestion $q_r(\mathbf{d})$ on the resources it uses. Therefore, it follows from [19] that the problem (43)-(44) can be interpreted as a congestion game $\Gamma(\mathcal{P}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ with resource dependent weights. Consequently, the COS algorithm terminates after a finite number of iterations iff the game $\Gamma(\mathcal{P}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ has a pure strategy Nash equilibrium. [1]

Since the cost $c_r(\mathbf{d}) \triangleq m_r q_r(\mathbf{d})$ of sharing every resource $r \in \tilde{\mathcal{R}}$ is an affine function of the congestion $q_r(\mathbf{d})$ on resource r, it follows from Theorem 4.2 in [19]

[1] A pure strategy Nash equilibrium of a strategic game is a collection $\mathbf{d}^*$ of decisions (called a strategy profile) for which $\tilde{C}_i(d_i^*, d_{-i}^*) \leq \tilde{C}_i(d_i, d_{-i}^*)$, $\forall d_i$, where d_i^* and d_{-i}^* are standard game theoretical notations for an improvement step of player i and for the collection of decisions (strategies) of all players other than i, respectively.

that the game $\Gamma(\mathcal{P}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ has the exact potential function[2] given by

$$\Psi(\mathbf{d}) = \sum_{i \in \mathcal{N}} \sum_{r \in \bar{\mathcal{R}}_{d_i}} q_{j,r} c_r^{\leq i}(\mathbf{d}), \tag{46}$$

where $c_r^{\leq i}(\mathbf{d}) = m_r q_r^{\leq i}(\mathbf{d})$ and $q_r^{\leq i}(\mathbf{d}) = \sum_{\{j \in O_r(\mathbf{d}) | j \leq i\}} q_{i,r}$.

It is well known that in a finite strategic game that admits an exact potential all improvement paths[3] are finite [20] and thus the existence of the exact potential function (46) allows us to use the COS algorithm for computing a pure strategy Nash equilibrium $\mathbf{d}^*$ of the game $\Gamma(\mathcal{P}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$, which proves the result. □

Theorem 6. *The COS algorithm terminates after a finite number of the iterations for the collection $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of optimal allocation policies.*

Proof. By following the same approach as in the proof of Theorem 5, it is easy to show that given the collection $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of optimal allocation policies, the problem (41)-(42) can be interpreted as a congestion game $\Gamma(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c}) =< \mathcal{N}, (\mathfrak{D}_i)_{i \in \mathcal{N}}, (\bar{C}_i)_{i \in \mathcal{N}} >$ with resource-dependent weights $q_{i,r}$, $i \in \mathcal{N}$, $r \in \bar{\mathcal{R}}$, and the cost of WD i in the resulting game is given by (39).

Since $m_a = 1, \forall a \in \mathcal{A}$, the cost $c_r(\mathbf{d}) \triangleq m_r q_r(\mathbf{d})$ of sharing every resource $r \in \bar{\mathcal{R}}$ is an affine function of the congestion on resource r. Therefore, the game $\Gamma(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ is also an exact potential game, and thus the COS algorithm computes a pure strategy Nash equilibrium $\mathbf{d}^*$ of the game $\Gamma(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$, which proves the result. □

In general, the number of improvement steps can be exponential in a potential game, but as we show next the COS algorithm can compute an equilibrium $\mathbf{d}^*$ of offloading decisions efficiently.

Theorem 7. *The COS algorithm terminates in $\mathcal{O}(n \frac{C^{min}}{C^{max}} \log \frac{\sum_{i \in \mathcal{N}} T_i^{ex}}{\Psi^{min}})$ iterations, where $n \geq 1$, C^{min} and C^{max} are system parameter dependent constants and Ψ^{min} is the minimum value of the potential function.*

Proof. First, let us define the minimum cost that WD i can achieve as $C_i^{min} \triangleq \min\{C_i^l, \min_{(a,c,s) \in \mathcal{A} \times \mathcal{C} \times \mathcal{S}} (D_i / R_{i,a} + L_{i,s} / F_c^s)\}$ and let $C^{min} \triangleq \min_{i \in \mathcal{N}} C_i^{min}$. Furthermore, let us define the maximum cost that WD i can achieve if it was the only

[2] A function $\Psi : \times_i(\mathfrak{D}_i) \to \mathbb{R}$ is an exact potential for a finite strategic game if for an arbitrary strategy profile (d_i, d_{-i}) and for any improvement step d_i^* the following holds:

$$\Psi(d_i, d_{-i}) - \Psi(d_i^*, d_{-i}) = \tilde{C}_i(d_i, d_{-i}) - \tilde{C}_i(d_i^*, d_{-i}). \tag{45}$$

[3] An improvement path is a sequence of strategy profiles in which one player at a time changes its strategy through performing an improvement step.

WD in the system as $C_i^{max} \triangleq \max\{C_i^l, \min_{(a,c,s)\in\mathcal{A}\times\mathcal{C}\times\mathcal{S}}(D_i/R_{i,a} + L_{i,s}/F_c^s)\}$, and let $C^{max} = \max_{i\in\mathcal{N}} C_i^{max}$.

Consider now an iteration of the COS algorithm where the offloading decision of WD i is updated from d_i to d_i^*. We can then write

$$\begin{aligned}\Psi(d_i, d_{-i}) - \Psi(d_i^*, d_{-i}) &= \tilde{C}_i(d_i, d_{-i}) - \tilde{C}_i(d_i^*, d_{-i}) \\ &\geq -C^{max} \geq -\frac{C^{max}}{C^{min}}\Psi(d_i, d_{-i}),\end{aligned} \tag{47}$$

where the equality follows from the definition of the exact potential function (45), the first inequality follows from the fact that $\tilde{C}_i(d_i, d_{-i}) - \tilde{C}_i(d_i^*, d_{-i}) > 0$ since d_i^* is an improvement step of WD i and the last inequality follows from the fact that $\Psi(d_i, d_{-i}) \geq C^{min}$ for any vector $\mathbf{d}$ of offloading decisions.

Therefore, from (47) we obtain $\Psi(d_i^*, d_{-i}) \leq (1 + \frac{C^{max}}{C^{min}})$, i.e., the COS algorithm decreases the potential function by at least a factor of $(1 + \frac{C^{max}}{C^{min}})$. Next, observe that from the definition of the constants C^{max} and C^{min} we have $\frac{C^{max}}{C^{min}} \geq 1$. Hence, since $(1+x)^{\frac{n}{x}} \leq e^n$ holds for $x, n \geq 1$, we obtain that after every $n\frac{C^{min}}{C^{max}}$ iterations of the COS algorithm $(1 + \frac{C^{max}}{C^{min}})^{n\frac{C^{min}}{C^{max}}} \leq e^n$, and thus every $n\frac{C^{min}}{C^{max}}$ iteration decreases the potential function by a constant factor (n can be chosen as a smallest positive constant for which $n\frac{C^{min}}{C^{max}} \geq 1$). Furthermore, since the COS algorithm starts from an offloading decision vector $\mathbf{d}^0$ in which all WDs perform computation locally, the potential function begins at the value $\Psi(\mathbf{d}^0) = \sum_{i\in\mathcal{N}} T_i^{ex}$ and cannot drop lower than Ψ^{min}. Therefore, the COS algorithm converges in $\mathcal{O}(n\frac{C^{min}}{C^{max}} \log \frac{\sum_{i\in\mathcal{N}} T_i^{ex}}{\Psi^{min}})$ iterations, which proves the result. □

In what follows we address the efficiency of the COS algorithm in terms of the cost approximation ratio.

Theorem 8. *The COS algorithm is a 2.62-approximation algorithm for the optimization problem (43)-(44) in terms of the system cost, i.e.,* $\frac{\tilde{C}(\mathbf{d}^*)}{\tilde{C}(\hat{\mathbf{d}})} \leq 2.62$.

Proof. Let us denote by $\mathfrak{D}^* \subseteq \mathfrak{D}$ the set of all vectors of offloading decisions that can be computed using the COS algorithm given any policy $\mathcal{P}_b$ and the collection $(\hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ of the optimal resource allocation policies of slices. Furthermore, let us consider a vector $\mathbf{d}^* \in \mathfrak{D}^*$ and an arbitrary vector $\hat{\mathbf{d}} \in \mathfrak{D}$ of offloading decisions. Since there is no WD i for which the cost $\tilde{C}_i(\mathbf{d}^*)$ can be decreased by unilaterally changing its offloading decision we have the following

$$\begin{aligned}\tilde{C}_i(\mathbf{d}^*) \leq \sum_{r\in\tilde{\mathcal{R}}_{d_i^*}\cap\tilde{\mathcal{R}}_{\hat{d}_i}} m_r q_{i,r} q_r(\mathbf{d}^*) + \\ \sum_{r\in\tilde{\mathcal{R}}_{d_i^*}\setminus\tilde{\mathcal{R}}_{\hat{d}_i}} m_r\big(q_r(\mathbf{d}^*) + q_{i,r}\big)q_{i,r} \leq \sum_{r\in\tilde{\mathcal{R}}_{\hat{d}_i}} m_r\big(q_r(\mathbf{d}^*) + q_{i,r}\big)q_{i,r},\end{aligned} \tag{48}$$

where $\tilde{\mathcal{R}}_{d_i^*} \subset \tilde{\mathcal{R}}$ and $\tilde{\mathcal{R}}_{\hat{d}_i} \subset \tilde{\mathcal{R}}$ denote the the set of resources that WD i uses in $\mathbf{d}^*$ and $\hat{\mathbf{d}}$, respectively. By summing (48) over all WDs $i \in \mathcal{N}$ and by reordering the summations we obtain

$$\tilde{C}(\mathbf{d}^*) \leq \sum_{r \in \mathcal{R}} \sum_{i \in O_r(\hat{\mathbf{d}})} m_r \big(q_r(\mathbf{d}^*) q_{i,r} + q_{i,r}^2\big). \tag{49}$$

From the definition (36) of the total weight $q_r(\mathbf{d})$ on resource $r \in \tilde{\mathcal{R}}$ and from $\sum\limits_{i \in O_r(\mathbf{d})} q_{i,r}^2 \leq q_r^2(\mathbf{d})$ we obtain

$$\tilde{C}(\mathbf{d}^*) \leq \sum_{r \in \mathcal{R}} m_r q_r(\mathbf{d}^*) q_r(\hat{\mathbf{d}}) + \sum_{r \in \mathcal{R}} m_r q_r^2(\hat{\mathbf{d}}).$$

Next, let us recall the Cauchy-Schwartz inequality $\sum\limits_{r \in \mathcal{R}} a_r b_r \leq \sqrt{\sum\limits_{r \in \mathcal{R}} a_r^2 \sum\limits_{r \in \mathcal{R}} b_r^2}$. By defining $a_r \triangleq \sqrt{m_r} q_r(\mathbf{d}^*)$ and $b_r \triangleq \sqrt{m_r} q_r(\hat{\mathbf{d}})$ we obtain the following

$$\begin{aligned} \tilde{C}(\mathbf{d}^*) \leq & \sqrt{\sum_{r \in \mathcal{R}} m_r q_r^2(\mathbf{d}^*) \sum_{r \in \mathcal{R}} m_r q_r^2(\hat{\mathbf{d}})} \\ & + \sum_{r \in \mathcal{R}} m_r q_r^2(\hat{\mathbf{d}}). \end{aligned} \tag{50}$$

By dividing the right and the left side of (50) by $\sum\limits_{r \in \mathcal{R}} q_r^2(\hat{\mathbf{d}}) > 0$ and by using (38) we obtain

$$\frac{\tilde{C}(\mathbf{d}^*)}{\tilde{C}(\hat{\mathbf{d}})} \leq \sqrt{\frac{\tilde{C}(\mathbf{d}^*)}{\tilde{C}(\hat{\mathbf{d}})}} + 1. \tag{51}$$

Since (51) holds for any vector $\mathbf{d}^* \in \mathfrak{D}^*$ of offloading decisions computed by the COS algorithm and for any vector $\hat{\mathbf{d}} \in \mathfrak{D}$ of offloading decisions of the WDs, it holds for the worst vector $\mathbf{d}^* = \arg\max_{\mathbf{d} \in \mathfrak{D}^*} \tilde{C}(\mathbf{d})$ of offloading decisions that can be computed using the COS algorithm and for the optimal $\hat{\mathbf{d}} = \arg\min_{\mathbf{d} \in \mathfrak{D}} \tilde{C}(\mathbf{d})$ solution too. Therefore, by solving (51) we obtain that the cost approximation ratio $\frac{\tilde{C}(\mathbf{d}^*)}{\tilde{C}(\hat{\mathbf{d}})}$ of the COS algorithm is upper bounded by $(3+\sqrt{5})/2 \cong 2.62$, which proves the theorem. □

Theorem 9. *The COS algorithm is a* 2.62*-approximation algorithm for the optimization problem (41)-(42) in terms of the system cost, i.e.,* $\frac{\tilde{C}(\mathbf{d}^*)}{\tilde{C}(\hat{\mathbf{d}})} \leq 2.62$.

Proof. The result can be easily obtained by following the approach used to prove Theorem 8. □

Finally, from Theorem 3 and Theorem 9 we obtain the approximation ratio bound for the proposed decomposition-based algorithm.

Theorem 10. *Given the collection* $(\hat{\mathcal{P}}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})$ *of optimal allocation policies, the proposed decomposition-based algorithm computes a* 2.62*-approximation solution to the JSS-ERM problem.*

6 Numerical Results

We used extensive simulations to evaluate the performance of the proposed resource allocation algorithm. To capture the potentially uneven spatial distribution of ECs, WDs and APs in a dense urban area, we consider a square area of $1km \times 1km$ in which WDs and 3 ECs are placed uniformly at random and 5 APs are placed at random on a *regular grid* with 25 points. The channel gain of WD i to AP a depends on their Euclidean distance $d_{i,a}$ and on the path loss exponent α, which we set to 4 according to the path loss model in urban and suburban areas [21]. We set the bandwidth B_a of 2 APs to $18MHz$ and the bandwidth of 3 APs to $27MHz$, corresponding to 25 and 75 resource blocks that are $12 \times 60KHz$ and $12 \times 30KHz$ subcarriers wide [22, 23], respectively. We consider that the transmit power $P_{i,a}$ of every WD i is uniformly distributed on $[10^{-6}, 0.1]\,W$ according to [24]. We calculate the total thermal noise in a $B_a MHz$ channel as $N_0(\text{dBm}) = -174 + 10log(B_a)$ according to [25] and the transmission rate $R_{i,a}$ achievable to WD i at AP a as $R_{i,a} = B_a log(1 + d_{i,a}^{-\alpha} \frac{P_{i,a}}{N_0})$.

To set the values for the computational capabilities of the WDs, we consider a line of Samsung Galaxy phones, from the oldest version with 1 core operating at $1GHz$ to the one of the newest versions with 8 cores operating at $2.84GHz$. We consider that EC c_1 is equipped with 36 vCPUs operating at $2.3GHz$ and 96 vCPUs operating at $3.6GHz$. We consider that EC c_2 and EC c_3 are equipped with 1 GPU each (with 2048 parallel processing cores operating at $557MHz$ and 2496 parallel processing cores operating at $560MHz$, respectively). Given the measurements reported in [26–28] we assume that a WD, a CPU and a GPU can execute on average 2, 3 and 1 instructions per cycle (IPC), respectively. Based on this, we consider that the computational capability F_i^l of every WD i is uniformly distributed on $[2, 45.4]\,GIPS$, where the lower and the upper bound correspond to the oldest and the newest version of the phone, respectively. Similarly, we calculate the computational capabilities of ECs, and set them to $F_{c1} = 1285.2GIPS$, $F_{c_2} = 1140.7GIPS$ and $F_{c_3} = 1397.8GIPS$.

The input data size D_i is drawn from a uniform distribution on $[1.7, 10]Mb$ according to measurements in [29]. The number X of instructions per data bit follows a Gamma distribution [30] with shape parameter $k = 75$ and scale $\theta = 50$. Given D_i and X, we calculate the complexity of a task as $L_i = D_i X$.

Motivated by Amazon EC2 instances [31] designed to support different kinds of applications (e.g., G3 and P2 instances for graphics-intensive and general-purpose GPU applications, and C5 and I3 instances for compute-intensive and non-virtualized workloads), we evaluate the system performance for the following four cases.

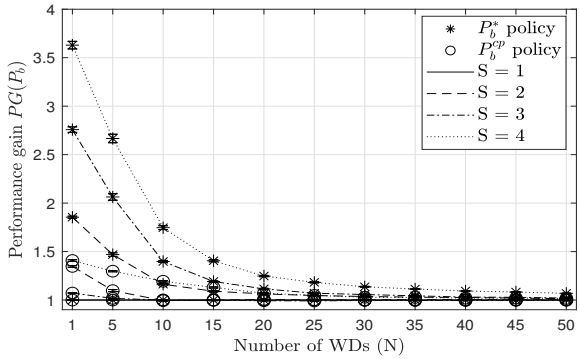

Figure 4: *Performance gain* vs. number of WDs N.

S= 1: The slice s_1 contains all ECs, and thus is able to support all of the above applications.
S= 2: The ECs are sliced such that slice s_1 supports the G3.4 instance and slice s_2 supports instances C5 and I3.
S= 3: The ECs are sliced such that slices s_1 and s_2 support P2 and G3s instances, respectively and slice s_3 supports instances C5 and I3.
S= 4: The ECs are sliced such that slices s_1, s_2, s_3 and s_4 support P2, G3s, C5 and I3 instances, respectively.

The coefficients $\frac{1}{h_{i,s}}$ were drawn from a continuous uniform distribution on $[0,1]$ and unless otherwise noted, the results are shown for all of the above scenarios.

We use two bandwidth allocation policies $\mathcal{P}_b$ of the slice orchestrator as a basis for comparison. The first policy $\mathcal{P}_b^{cp}$ shares the bandwidth of each AP a among slices proportionally to the ECs' resources that slices have. The second policy $\mathcal{P}_b^{eq}$ gives an equal share of the bandwidth of each AP a to each slice s. Observe that the COS algorithm computes an approximation vector $\mathbf{d}^*$ of offloading decisions for both policies (c.f. Theorem 5 and Theorem 8). The results shown are the averages of 300 simulations, together with 95% confidence intervals.

6.1 System Performance

We start with considering the system performance from the point of view of the slice orchestrator. To do so we define the system performance gain $PG(\mathcal{P}_b)$ for an inter-slice radio allocation policy $\mathcal{P}_b$ w.r.t. the policy $\mathcal{P}_b^{eq}$ as

$$PG(\mathcal{P}_b) = \frac{C(\mathbf{d}^*, \mathcal{P}_b^{eq}, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})}{C(\mathbf{d}^*, \mathcal{P}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})}.$$

Fig. 4 shows $PG(\mathcal{P}_b)$ as a function of the number N of WDs for the optimal $\mathcal{P}_b^*$ and for the cloud proportional $\mathcal{P}_b^{cp}$ allocation policy of the operator. We observe

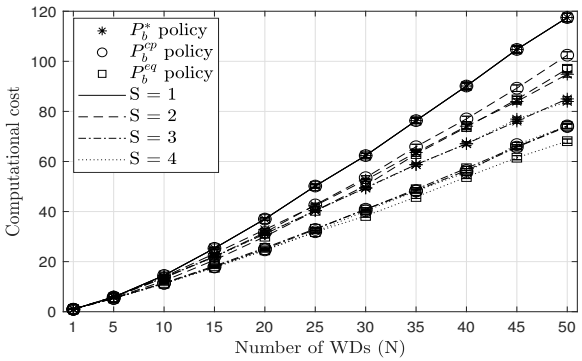

Figure 5: *Computational complexity* vs. number of WDs N.

that $PG(\mathcal{P}_b^*) = PG(\mathcal{P}_b^{cp}) = 1$ when $S = 1$ because the three solutions are equivalent when there is no slicing. On the contrary, for $S > 1$ we observe that $PG(\mathcal{P}_b^*) > 1$ and $PG(\mathcal{P}_b^{cp}) > 1$, which is due to that the policy $\mathcal{P}_b^{eq}$ does not take into account that the slices might have different amounts of ECs' resources. We also observe that the policy $\mathcal{P}_b^*$ achieves better performance gain (up to 2.5 times greater) than the policy $\mathcal{P}_b^{cp}$ because $\mathcal{P}_b^*$ assigns the WDs to slices not only based on the amount of ECs' resources the slices have, but also based on how well the slices are tailored for executing their tasks. This effect is especially evident when there are few WDs, because in this case WDs tend to offload their tasks, and thus the system cost is mostly determined by the offloading cost. On the contrary, as the number N of WDs increases, the gap between considered inter-slice radio allocation policies vanishes because the system cost becomes mostly determined by the WDs that perform the computation locally.

6.2 Computational Cost

Fig. 5 shows the number of iterations in which the COS algorithm computes a decision vector $\mathbf{d}^*$ as a function of the number N of WDs under the optimal $\mathcal{P}_b^*$, the cloud proportional $\mathcal{P}_b^{cp}$ and the equal $\mathcal{P}_b^{eq}$ inter-slice radio allocation policy of the slice orchestrator.

Interestingly, the number of updates decreases with the number S of slices. This is due to that the congestion on the logical resources decreases as S increases, and thus the COS algorithm updates the offloading decisions less frequently. We also observe that the number of updates scales approximately linearly with N under all considered policies of the slice orchestrator, and thus we can conclude that the COS algorithm is computationally efficient, which makes it a good candidate for computing an approximation $\mathbf{d}^*$ to the optimal vector $\hat{\mathbf{d}}$ of offloading decisions of WDs.

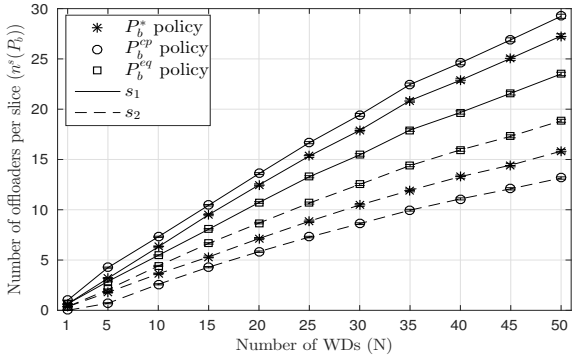

Figure 6: *Number of offloaders per slice* vs. number of WDs N.

6.3 Performance Within the Slices

We continue with considering the performance from the point of view of the slices. For an inter-slice radio allocation policy $\mathcal{P}_b$, we denote by $n^s(\mathcal{P}_b)$ the number of offloaders per slice in the vector $\mathbf{d}^*$ of offloading decisions computed by the COS algorithm and we define the cost ratio $CR^s(\mathcal{P}_b)$ per slice w.r.t. the system cost as

$$CR^s(\mathcal{P}_b) = \frac{C^s(\mathbf{d}^*, \mathcal{P}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})}{C(\mathbf{d}^*, \mathcal{P}_b, \hat{\mathcal{P}}_{w_a}, \hat{\mathcal{P}}_{w_c})}.$$

Fig. 6 and Fig. 7 show $n^s(\mathcal{P}_b)$ and $CR^s(\mathcal{P}_b)$, respectively for the optimal $\mathcal{P}_b^*$, the cloud proportional $\mathcal{P}_b^{cp}$ and the equal $\mathcal{P}_b^{eq}$ inter-slice radio allocation policy of the slice orchestrator. The results are shown for $S = 2$ and the red lines in Fig. 7 show the share of the ECs' resources among the slices s_1 and s_2 (i.e, slices s_1 and s_2 have approximately 72% and 28% of the resources, respectively). We observe from Fig. 6 and Fig. 7, respectively that the gap between $n^{s_1}(\mathcal{P}_b)$ and $n^{s_2}(\mathcal{P}_b)$ and the gap between $CR^{s_1}(\mathcal{P}_b)$ and $CR^{s_2}(\mathcal{P}_b)$ are highest in the case of the policy $\mathcal{P}_b^{cp}$ and lowest in the case of the policy $\mathcal{P}_b^{eq}$. Therefore, WDs whose tasks are a better match with the EC resources in slice s_2 than those in slice s_1 cannot fully exploit the ECs' resources in slice s_2 under the policy $\mathcal{P}_b^{cp}$, which allocates bandwidth resources proportionally to the ECs' resources. Similarly, WDs whose tasks are a better match with the EC resources in slice s_1 than in slice s_2 cannot fully exploit the ECs' resources in slice s_1 under the policy $\mathcal{P}_b^{eq}$, which allocates bandwidth resources equally. On the contrary, the results show that the optimal policy $\mathcal{P}_b^*$ finds a good match between the EC resources in the slices and the WDs' preferences for different types of computing resources, which makes it a good candidate for dynamic resource management for network slicing coupled with edge computing.

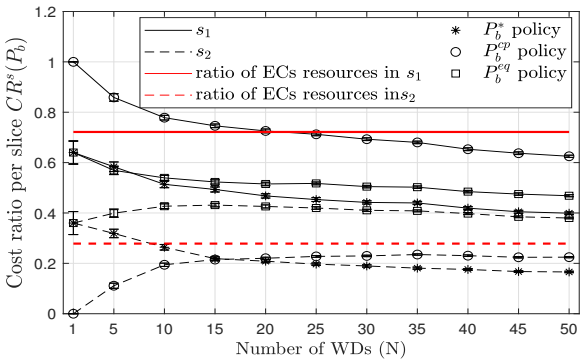

Figure 7: *Cost ratio per slice* vs. number of WDs N.

7 Related Work

Closest to our work a recent game theoretic treatments of the computation offloading problem [32–36]. In [32] the authors considered devices that compete for cloud resources so as to minimize their energy consumption, and proved that an equilibrium of offloading decision can be computed in polynomial time. In [33] the authors considered devices that maximize their performance and a profit maximizing service provider, and used backward induction for deriving near optimal strategies for the devices and the operator. In [34] the authors considered that devices can offload their tasks to a cloud through multiple identical wireless links, modeled the congestion on wireless links, and used a potential function argument for proposing a decentralized algorithm for computing an equilibrium. In [35] the authors considered that devices can offload their tasks to a cloud through multiple heterogeneous wireless links, modeled the congestion on wireless and cloud resources, showed that the game played by devices is not a potential game and proposed a decentralized algorithm for computing an equilibrium. In [36] the authors modeled the interaction between devices and a single network operator as a Stackelberg game, and provided an algorithm for computing a subgame perfect equilibrium. Unlike these works, we consider the computation offloading problem together with network slicing and we analyze the interaction between the network operator and the slices.

Another line of works considers the network slicing resource allocation problem [37–41]. In [37] the authors considered an auction-based model for allocating edge cloud resources to slices and proposed an algorithm for allocating resources to slices so as to maximize the total network revenue. In [38] the authors considered the radio resources slicing problem and proposed an approximation algorithm for maximizing the sum of the users' utilities. In [39] the authors modeled the interaction between slices that compete for bandwidth resources with the objective to maximize the sum of their users' utilities, and proposed an admission control algorithm under which

the slices can reach an equilibrium. In [40] the authors proposed a deep learning architecture for sharing the resources among network slices in order to meet the users' demand within the slices. In [41] the authors considered a radio access network slicing problem and proposed two approximation algorithms for maximizing the total network throughput. Unlike these works, we consider a slicing enabled edge system in which the slice resource orchestrator assigns WDs to slices and shares radio resources across slices, while the slices manage their own radio and computing resources with the objective to maximize overall system performance.

To the best of our knowledge ours is the first work to consider slicing and computation offloading to edge clouds jointly, capturing the interaction between the slice resource orchestrator and the slices.

8 Conclusion

We have considered the computation offloading problem in an edge computing system under network slicing in which slices jointly manage their own communication and computing resources and the slice resource orchestrator manages communication resources among slices and assigns the WDs to slices. We formulated the problem of minimizing the sum over all WDs' task completion times as a mixed-integer problem, proved that the problem is NP-hard and proposed a decomposition of the problem into a sequence of optimization problems. We proved that the proposed decomposition does not change the optimal solution of the original problem, proposed an efficient approximation algorithm for solving the decomposed problem and proved that the algorithm has bounded approximation ratio. Our numerical results show that the proposed algorithm is computationally efficient. They also show that dynamic allocation of slice resources is essential for maximizing the benefits of edge computing, and slicing could be beneficial for improving overall system performance.

References

[1] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, 2017.

[2] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "Mec in 5g networks," *Sophia Antipolis, France, ETSI, White Paper*, 2018.

[3] M. Rost and S. Schmid, "Virtual network embedding approximations: Leveraging randomized rounding," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 2071–2084, Oct. 2019.

[4] B. Farkiani, B. Bakhshi, and S. A. MirHassani, "A fast near-optimal approach for energy-aware sfc deployment," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1360–1373, Dec 2019.

[5] I. Jang, D. Suh, S. Pack, and G. Dán, "Joint optimization of service function placement and flow distribution for service function chaining," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2532–2541, Nov 2017.

[6] A. Zafeiropoulos et al., *5G PPP Architecture Working Group: View on 5G Architecture*, S. Redana and Ö. Bulakci, Eds. European Commission, Jun. 2019, vol. Version 3.0.

[7] X. Foukas, M. K. Marina, and K. Kontovasilis, "Orion: Ran slicing for a flexible and cost-effective multi-service mobile network architecture," in *Proc. of ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2017, pp. 127–140.

[8] C. Chang, N. Nikaein, and T. Spyropoulos, "Radio access network resource slicing for flexible service execution," in *Proc. of IEEE INFOCOM 2018 Workshops)*, April 2018, pp. 668–673.

[9] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 771–786, 2018.

[10] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2014.

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

[12] J. L. D. Neto, S.-Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, "Uloof: a user level online offloading framework for mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 17, no. 11, pp. 2660–2674, 2018.

[13] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[14] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[15] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE TMC*, vol. 18, no. 1, pp. 207–220, 2018.

[16] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

[17] S. Jošilo and G. Dán, "Joint management of wireless and computing resources for computation offloading in mobile edge clouds," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[18] M. R. Garey and D. S. Johnson, *Computers and intractability.* wh freeman New York, 2002, vol. 29.

[19] T. Harks, M. Klimm, and R. H. Möhring, "Characterizing the existence of potential functions in weighted congestion games," *Theory of Computing Systems*, pp. 46–70, 2011.

[20] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[21] S. R. Saunders and A. Aragón-Zavala, *Antennas and propagation for wireless communication systems.* John Wiley & Sons, 2007.

[22] E. TSGR, "Lte: Evolved universal terrestrial radio access (e-utra)," *Physical channels and modulation (3GPP TS 36.211 version 10.0.0. 0 Release 10) ETSI TS*, 2011.

[23] A. Zaidi, F. Athley, J. Medbo, U. Gustavsson, G. Durisi, and X. Chen, *5G Physical Layer: Principles, Models and Technology Components.* Academic Press, 2018.

[24] M. Lauridsen, L. Noël, T. B. Sørensen, and P. Mogensen, "An empirical lte smartphone power model with a view to energy efficiency evolution." *Intel Technology Journal*, vol. 18, no. 1, 2014.

[25] N. Da Dalt and A. Sheikholeslami, *Understanding Jitter and Phase Noise: A Circuits and Systems Perspective.* Cambridge University Press, 2018.

[26] L. Codrescu, W. Anderson, S. Venkumanhanti, M. Zeng, E. Plondke, C. Koob, A. Ingle, C. Tabony, and R. Maule, "Hexagon dsp: An architecture optimized for mobile multimedia and communications," *IEEE Micro*, vol. 34, no. 2, pp. 34–43, 2014.

[27] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An energy efficiency feature survey of the intel haswell processor," in *2015 IEEE*

international parallel and distributed processing symposium workshop, 2015, pp. 896–904.

[28] Y. Takefuji, *GPU Parallel Computing for Machine Learning in Python: How to Build a Parallel Computer*. Independently published, 2017.

[29] L. Fletcher, L. Petersson, and A. Zelinsky, "Road scene monotony detection in a fatigue management driver assistance system," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, 2005, pp. 484–489.

[30] J. R. Lorch and A. J. Smith, "Pace: A new approach to dynamic voltage scaling," *IEEE Transactions on Computers*, vol. 53, no. 7, pp. 856–869, 2004.

[31] https://aws.amazon.com/ec2/instance types/, "Amazon ec2 instance types."

[32] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *ACM/IEEE Symposium on low power electronics and design*, 2012, pp. 279–284.

[33] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *IEEE Service Oriented System Engineering Symposium*, 2013, pp. 494–502.

[34] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[35] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *IEEE INFOCOM*, 2017, pp. 1–9.

[36] ——, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *IEEE INFOCOM*, 2019, pp. 2467–2475.

[37] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5g: An auction-based model," in *IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[38] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Pérez, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3044–3058, 2017.

[39] P. Caballero, A. Banchs, G. De Veciana, X. Costa-Pérez, and A. Azcorra, "Network slicing for guaranteed rate services: Admission control and resource allocation games," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6419–6432, 2018.

[40] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Deepcog: Cognitive network management in sliced 5g networks with deep learning," in *IEEE INFOCOM*, 2019, pp. 280–288.

[41] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia, "The slice is served: Enforcing radio access network slicing in virtualized 5g systems," in *IEEE INFOCOM*, 2019, pp. 442–450.