KTH Computer Science
and Communication

Decentralized Algorithms for Resource Allocation in Mobile Cloud Computing Systems

SLAĐANA JOŠILO

Doctoral Thesis
Stockholm, Sweden 2018

TRITA-EECS-AVL-2018:34 KTH School of Electrical Engineering and Computer Science
ISSN 1653-5146 SE-100 44 Stockholm
ISBN 978-91-7729-751-2 Sweden

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av doktorsexamen torsdag den 31 May 2018 klockan 10.00 i sal Q2, KTH, Stockholm.

© Slađana Jošilo, May 2018

Tryck: Universitetsservice US AB

Abstract

The rapid increase in the number of mobile devices has been followed by an increase in the capabilities of mobile devices, such as the computational power, memory and battery capacity. Yet, the computational resources of individual mobile devices are still insufficient for various delay sensitive and computationally intensive applications. These emerging applications could be supported by mobile cloud computing, which allows using external computational resources. Mobile cloud computing does not only improve the users' perceived performance of mobile applications, but it also may reduce the energy consumption of mobile devices, and thus it may extend their battery life. However, the overall performance of mobile cloud computing systems is determined by the efficiency of allocating communication and computational resources. The work in this thesis proposes decentralized algorithms for allocating these two resources in mobile cloud computing systems.

In the first part of the thesis, we consider the resource allocation problem in a mobile cloud computing system that allows mobile users to use cloud computational resources and the resources of each other. We consider that each mobile device aims at minimizing its perceived response time, and we develop a game theoretical model of the problem. Based on the game theoretical model, we propose an efficient decentralized algorithm that relies on average system parameters, and we show that the proposed algorithm could be a promising solution for coordinating multiple mobile devices.

In the second part of the thesis, we consider the resource allocation problem in a mobile cloud computing system that consists of multiple wireless links and a cloud server. We model the problem as a strategic game, in which each mobile device aims at minimizing a combination of its response time and energy consumption for performing the computation. We prove the existence of equilibrium allocations of mobile cloud resources, and we use game theoretical tools for designing polynomial time decentralized algorithms with a bounded approximation ratio. We then consider the problem of allocating communication and computational resources over time slots, and we show that equilibrium allocations still exist. Furthermore, we analyze the structure of equilibrium allocations, and we show that the proposed decentralized algorithm for computing equilibria achieves good system performance.

By providing constructive equilibrium existence proofs, the results in this thesis provide low complexity decentralized algorithms for allocating mobile cloud resources for various mobile cloud computing architectures.

In loving memory of my mother, Milenija Mihajlović Jošilo.

Acknowledgments

I would like to thank my advisor György Dán for his patience, support and encouragement. His consistent guidance and valuable suggestions at every stage of the research work have helped me to complete the thesis successfully.

I would also like to thank all past and present members of the NSE department for providing a friendly environment. I am thankful to my close friends for their love, enthusiasm and help.

I am deeply thankful to my family for their love and sacrifices that they have made on my behalf.

Contents

Chapter 1

Introduction

1.1 Background

In recent years, the number of mobile devices including smartphones, tablets, sensors, and other portable devices, has been rapidly increasing. According to a recent estimate by Cisco, the number of mobile devices and connections will grow to 11.6 billion by 2021 at a compound annual growth rate (CAGR) of 8% between 2016 and 2021 [1].

Despite such a significant growth in the number of mobile devices, the computational capabilities of individual mobile devices are still limited compared to desktop computers [2]. In addition, mobile devices are mostly battery powered, and thus energy consumption is an important aspect that requires careful attention. Unfortunately, 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 [3]. Furthermore, in order to be easily carried, mobile devices must be physically light and small, which puts additional limitations on specific hardware resources, such are computational capabilities, battery capacity, and memory and disk capacity [4].

At the same time, computationally intensive applications, including augmented reality, natural language processing, face, gesture, speech and object recognition are increasingly used on mobile devices; for example, Cisco predicted that only augmented reality traffic will increase sevenfold between 2016 and 2021 [1]. Therefore, it is clear that the computational capabilities of mobile devices are increasing at a slower rate than the computational requirements of the applications.

A widely adopted approach to closing the gap between the limited computational capabilities of mobile devices and high computational requirements of the applications is mobile cloud computing [5, 6]. Mobile cloud computing augments the computational capabilities of mobile devices by allowing them to use external computational resources. By relying on external computational resources, mobile cloud computing may accelerate the execution of applications, may extend the bat-

tery lifetime of mobile devices, and may enable collaboration among mobile devices. All these potential benefits contribute to the growing interest in mobile cloud services; the mobile cloud market is expected to grow from $12.07 billion in 2016 to $74.25 billion by 2023 at a CAGR of 30.1% [7].

1.2 Challenges

The growth in the number of mobile devices over the past years has been followed by a corresponding growth in mobile data traffic. According to a recent estimate by Cisco, the overall mobile data traffic will increase sevenfold between 2016 and 2021 at a CGR of 47% [1]. Strong mobile data traffic growth puts stress on mobile cloud communication and computational infrastructures, and thus it affects users' perception of mobile cloud computing performance. Therefore, effective management of mobile cloud communication and computational resources is an important part of designing the mobile cloud computing systems.

There are two fundamental challenges facing the design of mobile cloud computing systems. The first is meeting users' requirements concerning the overall application response time. The application response time is not only affected by the limited computational resources of mobile devices, but it may also be affected by the wireless network constraints in the case of using external computational resources. The other fundamental challenge is meeting users' requirements concerning the battery lifetime of their mobile devices. The battery lifetime is mostly determined by the energy consumption rate, which may depend on several factors, such as the type of applications and network connection.

In response to these challenges, different mobile cloud architectures have been considered. The traditional architectures make use of the commercial cloud infrastructures, by allowing mobile devices to offload their computational tasks to the remote resourceful clouds, such as Amazon EC2 [8] and Windows Azure [9]. In order to meet the extremely low latency requirements of emerging delay sensitive applications, recently proposed architectures consider the execution of applications in close proximity of the end users.

Mobile edge computing (MEC) [10] proposes bringing computational resources close to the network edge, and thus it is recognized as one of the key emerging technologies for 5G networks [11]. Another architecture that enables execution of applications in close proximity to the end users is fog computing [12]. Fog computing extends MEC services by using heterogeneous devices, such as access points, edge routers and switches as the service nodes, and thus it is considered to be a potential platform for Internet of Things (IoT) applications [13, 14].

In either case, when many mobile devices compete for communication and computational resources, new challenges arise for several reasons. First, the mobile devices are heterogeneous in terms of computational capabilities and in terms of the constraints on the total energy consumption. Second, the applications running on different mobile devices may be different in terms of what computational

tasks they consist of and how often they generate the computational tasks. Third, the mobile devices may be autonomous, and hence they may be be interested in improving their own performance. These challenges additionally complicate the design of mobile cloud computing systems, especially in the case of delay sensitive and computationally intensive applications.

1.3 Thesis Structure

The structure of this thesis is as follows. In Chapter 2, we discuss characteristics of different wireless access technologies and we present both traditional centralized and emerging distributed mobile cloud computing architectures. In Chapter 3, we define the computational tasks, and we introduce performance metrics for evaluating mobile cloud computing systems. In Chapter 4, we present different formulations of the resource allocation problem for various mobile cloud computing architectures. 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

Mobile Cloud Computing Resources

Mobile cloud computing systems consist mainly of two types of resources, that is, communication and computational resources. Figure 2.1 shows an example of a mobile cloud computing system. As illustrated in the figure, mobile devices can decide whether to perform the computation using local computational resources or to offload the computation to external computational resources through communication networks.

In the case of offloading, mobile devices compete for communication and computational resources, and thus the decision of a mobile device affects both its own performance and the performance of the other mobile devices. In this chapter we present different wireless access technologies and different mobile cloud computing architectures, and we discuss the main factors affecting the performance of mobile cloud computing systems.

2.1 Communication Resources

When offloading their tasks to external computational resources mobile devices rely on wireless networks. Today's wireless networks are highly heterogeneous, and thus mobile users can usually select among different radio access technologies, such as 2.5G, 3G, 4G and Wi-Fi [15].

The most common problems facing these radio access technologies are intermittent connectivity, variable network conditions, and limited bandwidth [16, 17]. Furthermore, the communication medium is shared among users in the same area, and thus the transmission rate of a user depends on the bandwidth allocation algorithm.

The bandwidth allocation algorithm in the distributed coordination function (DCF) used in the IEEE 802.11 standard uses the CSMA/CA protocol for implementing a fair sharing of the bandwidth [18, 19]. According to the DCF algorithm, the bandwidth of an access point (AP) a is fairly shared among the set $\mathcal{N}_a$ of users connected to the AP a. Hence, given the set $\mathcal{N}_a$ the uplink rate $\omega_{i,a}$ of user $i \in \mathcal{N}_a$

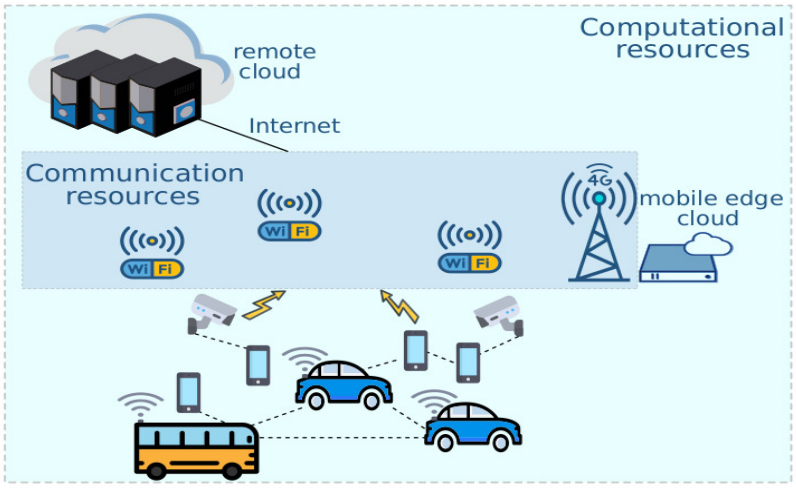

Figure 2.1: An example of a mobile cloud computing system.

can be expressed as

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

Other examples of fair bandwidth sharing mechanisms are the ones used in time-fair TDMA and OFDM based medium access protocols in which the uplink rate $\omega_{i,a}$ of user i on AP a does not depend on the specific set $\mathcal{N}_a$ of users sharing the AP, but it may depend on the total number $|\mathcal{N}_a|$ of users sharing the AP [20, 21]. Common to these protocols is that the uplink rate $\omega_{i,a}$ of user i on AP a can be user specific, and given $|\mathcal{N}_a|$ it can be expressed as

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

A model similar to the latter uplink rate model can also be used to describe the proportional-fair scheduling (PFS) in 3G networks [22].

Given the overall growth in the number of mobile devices and the latency requirements of emerging delay sensitive applications, it is clear that communication resources in mobile cloud computing systems have to be managed appropriately. There have been a few recent approaches with a strong focus on the communication related problems in mobile cloud computing systems [23, 24, 25, 26, 27]. Approaches considered in [23, 24] propose mechanisms for predicting network connectivity based on the users' movement and a database of network connectivity over geographical zones. Approaches considered in [25, 26, 27] propose collaboration among mobile devices. The latter approach is especially interesting for two

reasons. First, it can improve bandwidth utilization and can make use of device-to-device (D2D) communication, which is considered to be a promising technology for future 5G cellular networks [28, 29, 30]. Second, the concept of collaboration provides a basis for moving towards highly distributed mobile cloud computing architectures [31, 32, 33].

2.2 Mobile Cloud Computing Architectures

As illustrated in Figure 2.1, the sources of computational resources may be different, from remote commercial clouds and the clouds located at the network edge, to the user carried mobile devices and mobile devices attached to the vehicles. In the following we discuss the factors affecting the performance of mobile cloud computing systems for traditional centralized and emerging distributed mobile cloud computing architectures.

Centralized Clouds

According to a recent data from Synergy Research Group, the four leading commercial cloud providers are Amazon, Azure, IBM and Google [34]. These cloud providers usually offer three types of cloud computing services, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [35]. IaaS and PaaS offer a high level of control, flexibility, and management, and thus they are suitable for application owners that provide the end users with a final product through SaaS [36]. Therefore, from the perspective of a mobile user the most relevant cloud computing service is SaaS.

A number of recent works have investigated the performance of commercial cloud computing services [37, 38, 39, 40]. According to the reported measurements, commercial clouds may have very high end-to-end transmission delays, which mostly occur because commercial cloud infrastructures offer no guarantees on proximity of their servers to the end users. As a consequence, computation offloading to remote commercial clouds may not always be a good solution, especially in the case of delay sensitive applications.

Mobile Edge Clouds

Emerging distributed mobile edge clouds are widely accepted as a promising alternative to the centralized clouds [41]. The key idea of MEC is to install the computational and storage resources together with the existing infrastructures, such as mobile BSs [10]. This idea is especially attractive for the network operators, since it gives them an opportunity to profit from the MEC market.

By bringing computational resources close to the network edge, MEC has a potential to improve the performance perceived by the users. However, when many mobile users use the MEC resources simultaneously, they may experience a degraded performance for two reasons. First, the amount of bandwidth that a BS can

assign to a mobile user may not be sufficient for providing satisfactory transmission times, especially in the case of applications that require offloading huge volumes of data. Second, MEC provides only limited computational and storage resources compared to commercial cloud infrastructures. As a consequence, the amount of computational resources that an edge cloud can assign to a mobile device may not be sufficient for providing satisfactory execution times, especially in the case of computationally intensive applications. Therefore, using MEC resources requires optimizing not only the allocation of communication, but also the allocation of computational resources, which makes the problem inherently challenging.

Fog resources

Fog computing is about shifting away from the centralized cloud architectures towards highly distributed architectures. Compared to MEC, fog computing proposes bringing computational resources even closer to the end users [12]. The key idea of fog computing is to extend the existing centralized cloud computing architecture by allowing collaboration among distributed edge cloud resources and nearby heterogeneous devices. Todays' devices may have at disposal limited computational resources when considered alone, but by pooling their resources they can form a computationally rich distributed computing platform.

The reason for shifting towards distributed architectures is that emerging applications, such as various IoT applications have requirements that can not be always addressed by relying only on computational resources of individual devices and centralized clouds [13, 14]. On the one hand, this is due to that many individual devices are still not enough powerful to support computationally intensive applications. On the other hand, some devices may have difficulty connecting to the centralized cloud due to network bandwidth constraints.

Fog computing is not only beneficial from the perspective of computational resources, but it is also beneficial from the perspective of communication resources, because collaboration among nearby devices can make use of D2D communication, and thus it may improve bandwidth utilization [42]. However, the two biggest challenges facing fog computing are how to integrate heterogeneous devices into a common computing platform, and how to efficiently distribute the computational tasks among numerous devices [43].

Chapter 3

Computation Offloading for Mobile Systems

The applications running on the mobile devices may be partitioned into computational tasks at different granularity levels [44]. According to application partitioning requirements, there are two main classes of computation offloading frameworks. The first class considered in [45, 46, 47] does not require partitioning, and the entire application can be offloaded. The second class considered in [5, 48, 49, 50] requires application partitioning into computational tasks that can be offloaded for remote execution and computational tasks that have to be executed on the mobile device. In this chapter, we define the parameters that characterize the computational tasks and we introduce the main performance metrics that can be used to measure the efficiency of resource allocation in mobile cloud computing systems.

3.1 Computational Tasks

In what follows, we consider only the computational tasks that can be offloaded, and we characterize the mobile user i's computational task by two parameters. The first parameter is the size D_i of the input data that must be offloaded in the case of remote execution, and the second parameter is the complexity L_i, that is, 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 $L_i = D_i X_i$ [51], where X_i is the number of CPU cycles per data bit, which can be approximated by a Gamma distribution [52, 53].

The mobile devices and the cloud servers differ from the perspective how fast they can execute the same computational task $<D_i, L_i>$, which mostly depends on their CPU performance. Although every new generation of mobile devices is more and more powerful, yet the gap between the CPU performance of individual mobile devices and server grade computers remains [54].

The CPU performance can be characterized by a wide range of performance metrics, among which the basic metrics are the clock cycle time and the clock frequency [55]. Yet, the frequency F at which the processor executes a computational task $<D_i, L_i>$ is often not the same as the clock frequency. This is mostly due to that the processor performs many different tasks simultaneously (e.g., the management of the underlying hardware, operating system activities, and input/output (I/O) operations).

Given that the frequency F is expressed in CPU cycles per second, where the notion of an instruction is different between different instruction set architectures (e.g. RISC, CISC and VLIW), the time T_i^{exe} needed to execute a computational task $<D_i, L_i>$ can be expressed as

$$T_i^{exe} = \frac{L_i}{F}.$$

The frequency F at which the processor executes a computational task $<D_i, L_i>$ also influences the energy consumption. According to the measurements reported in [6, 51], the energy consumption per CPU cycle is linearly proportional to the square of the frequency F. Thus, the energy E_i^{exe} needed to execute a computational task $<D_i, L_i>$ can be expressed as

$$E_i^{exe} = cF^2 L_i,$$

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

The above models for the execution time and the energy consumption capture the main characteristics of executing the computational tasks. The models are especially suitable for tasks that are characterized by a large complexity L_i, and thus that are not very sensitive to the possible interruptions that may occur during the startup, execution, and termination.

3.2 Performance Metrics

The two main performance metrics for assessing the performance of mobile cloud computing systems are the task completion time and the corresponding energy consumption of mobile devices. The factors that affect these performance metrics depend on which computational 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 and remote execution.

Task Completion Time

When a task $<D_i, L_i>$ is executed locally, the time $T_i^{c,l}$ needed to complete the task is the time needed to execute the task using local computational resources at frequency F_i

$$T_i^{c,l} = \frac{L_i}{F_i}.$$

On the contrary, when a mobile user offloads its task $< D_i, L_i >$ to external computational resources through an AP a, the task completion time consists of three parts. The first part is the time $T_{i,a}^t$ needed to transmit the amount of D_i input data through AP a at the rate $\omega_{i,a}$, and it can be expressed as

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

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

$$T_{i,e}^{exe} = \frac{L_i}{F_i^e}.$$

The third part is the time needed to transmit the result of the computation from external computational resources to the mobile device. For many applications, such as tracking, 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 [56, 57, 58]. Therefore, in the case of computation offloading through an AP a, a simple linear model can be used to model the task completion time $T_{i,a}^{c,e}$

$$T_{i,a}^{c,e} = T_{i,a}^t + T_{i,e}^{exe}.$$

Energy Consumption

When a task $<D_i, L_i>$ is executed locally, the energy consumption E_i^l of a mobile device is the energy needed to execute the task using local computational resources at frequency F_i

$$E_i^l = cF_i^2 L_i.$$

On the contrary, when a mobile user offloads its task $< D_i, L_i >$ to external computational resources through an AP 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 [59], 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 a mobile user transmits the data through AP a at rate $\omega_{i,a}$ using transmit power $P_{i,a}$, the energy consumption $E_{i,a}$ can be expressed as

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

Chapter 4

Resource Allocation in Mobile Cloud Computing Systems

When designing mobile cloud computing systems, one of the main objectives is providing a convenient solution for mobile devices to perform their applications efficiently, in terms of both the application completion time and the energy consumption. Due to increasing use of mobile devices, efficient management of mobile cloud computing resources is crucial to achieve this objective. In the following, we model the cost associated with mobile users, and we discuss different formulations of the resource allocation problem.

4.1 Cost Model

Since mobile devices are heterogeneous in terms of computational capabilities, battery states, and in terms of what type of computational tasks they have to execute, it is reasonable to introduce the notion of preferences over the performance metrics. The heterogeneity among mobile devices can be modeled using two parameters, $0 \leq \gamma_i^T \leq 1$ and $0 \leq \gamma_i^E \leq 1$, which characterize user i's preferences regarding the completion time and the energy consumption, respectively. Given these parameters, the user i's cost can be formulated as a function of the weighted completion time and the weighted energy consumption,

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

$$\textit{offloading through AP } a\textit{:} \quad C_{i,a}^e = f(\gamma_i^T T_{i,a}^{c,e}, \gamma_i^E E_{i,a}).$$

The above cost models allow a mobile user to dynamically adjust its objective to the specific application requirements, and to its current battery state by changing the values of the parameters γ_i^T and γ_i^E.

4.2 Resource Allocation Problem Formulation

In order to provide a general formulation of the resource allocation problem, in the following we consider a mobile cloud computing system that consists of a set $\mathcal{N}$ of mobile users, $|\mathcal{N}| = N$, a set $\mathcal{A}$ of communication resources, $|\mathcal{A}| = A$, and a set $\mathcal{S}$ of computational resources, $|\mathcal{S}| = S$. We use $\mathbf{X} \in \{0,1\}^{N\times A}$ and $\mathbf{Y} \in \{0,1\}^{N\times S}$ to denote communication and computational resource assignment matrices, respectively.

One potential goal could be to minimize the system cost C, which is defined as the sum over all users' costs. Since the resources are shared among the users, the system cost C is a function of $\mathbf{X}$ and $\mathbf{Y}$, that is, $C = C(\mathbf{X}, \mathbf{Y})$. Using the above notation, the problem of resource allocation in a mobile cloud computing system can be formulated as the following $0-1$ nonlinear optimization problem,

$$\min_{\mathbf{X},\mathbf{Y}} f_0(\mathbf{X}, \mathbf{Y}, C(\mathbf{X}, \mathbf{Y})) \tag{4.1}$$

$$\text{s.t.} \quad g_i(\mathbf{X}, \mathbf{Y}) \leq a_i, \forall i \in \mathcal{N}, \tag{4.2}$$

$$h(\mathbf{X}) \leq b_a, \forall a \in \mathcal{A}, \tag{4.3}$$

$$q(\mathbf{Y}) \leq c_s, \forall s \in \mathcal{S}, \tag{4.4}$$

$$\sum_{a\in\mathcal{A}} \sum_{s\in\mathcal{S}\setminus\{i\}} x_{i,a} y_{i,s} + y_{i,i} = 1, \forall i \in \mathcal{N}, \tag{4.5}$$

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

$$\mathbf{Y} \in \{0,1\}^{N\times S}. \tag{4.7}$$

The function $g_i(\mathbf{X}, \mathbf{Y})$ takes into account sharing both communication and computational resources. For example, constraint (4.2) may be used to ensure that the task completion time or the energy consumption of each user $i \in \mathcal{N}$ is lower than the threshold specified by a_i. The functions $h(\mathbf{X})$ and $q(\mathbf{Y})$ take into account sharing only one type of resources, that is, the communication and computational resources, respectively. The constraints (4.3) and (4.4) can be used to enforce a limitation on the amount of communication and computational resources that can be provided to each user, respectively. The constraint (4.5) enforces that each user $i \in \mathcal{N}$ either performs the computation locally ($x_{i,a} = 0, y_{i,i} = 1, y_{i,s} = 0, \forall a \in \mathcal{A}, \forall s \in \mathcal{S} \setminus \{i\}$) or it offloads the task to computational resource s using communication resource a ($x_{i,a} = 1, y_{i,s} = 1, x_{i,a'} = 0, y_{i,s'} = 0, \forall a' \in \mathcal{A} \setminus \{a\}, \forall s' \in \mathcal{S} \setminus \{s\}$).

Observe that the above optimization problem can be easily reduced to the completion time minimization problem by setting the completion time parameter $\gamma_i^T = 1$, and the energy consumption parameter $\gamma_i^E = 0$, for all mobile users $i \in \mathcal{N}$. Similarly, we can define the energy consumption minimization problem by setting the energy consumption parameter $\gamma_i^E = 1$, and the completion time parameter $\gamma_i^T = 0$, for all mobile users $i \in \mathcal{N}$. However, solving the problem $(4.1) - (4.7)$ may be impractical in realistic mobile cloud computing systems, because it involves searching a large solution space. With this in mind, in the following we distinguish

between the three different formulations of the resource allocation problem in mobile cloud computing systems, and we discuss the most important results from the literature.

Completion Time Minimization

Completion time minimization is usually considered in the case of delay sensitive applications such as mobile augmented reality, real-time voice and video. Most of the works that aim at minimizing the application completion time consider the joint computation partitioning and resource allocation problem, while meeting various constraints that can arise in mobile cloud computing systems. In the following we discuss two classes of works that consider independent tasks and dependent tasks, respectively.

Independent task scheduling problem: The works presented in [60, 61] studied the completion time minimization problem assuming that there is no dependency among the computational tasks. The authors in [60] considered a mobile cloud computing system that consists of a set of processors with known processing times and, a set of processors with unknown processing times. Given the set of independent tasks in the system, the authors aim at minimizing the time when the processing of the last task is completed, that is, the makespan of the given tasks. For the case when the processing time is unknown for only one of the processors, the authors proposed a constant-factor approximation algorithm for scheduling the tasks. They extended the analysis to the case of multiple processors with unknown processing times, and in this case they proposed a heuristic algorithm. The authors in [61] considered a system with stochastic task arrivals, and they defined the user's cost as the expected number of its tasks in the system, that is, the product of the user's expected task completion time and the rate at which its device generates tasks. In the considered system, mobile devices may offload their tasks either to an edge cloud or to a centralized cloud with the objective to minimize their costs under energy consumption constraints. The authors used game theoretical tools to show existence of a mixed strategy equilibrium task allocation and they developed a distributed algorithm for computig it.

Our work presented in Paper A falls into this class of completion time minimization problems. We considered a fog computing system that consists of a centralized cloud and multiple heterogeneous devices, which may process the tasks of each other. We considered stochastic task arrivals, and we modeled the task arrival process of each device as a Poisson process. We used a queuing model to capture the contention for both communication and computational resources, and we denoted by $\overline{T^c}_{i,j}(p_{i,j})$ the mean time that is needed to complete device i's task using node j's computational resources. Given the task assignment matrix $\mathbf{P} \in [0,1]^{N\times(N+1)}$, the system cost $\bar{C}(\mathbf{P})$ can be defined as the average system delay

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

where $p_{i,i}$, $p_{i,0}$ and $p_{i,j}$ indicate the probability that user 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 $\mathbf{P}$ that ensure the stability of the queuing system, the problem can be formulated as the following convex optimization problem

$$\min_{\mathbf{P} \in \mathcal{P}} \bar{C}(\mathbf{P}) \tag{4.8}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{N} \cup \{0\}} p_{i,j} = 1, \forall i \in \mathcal{N}. \tag{4.9}$$

Since devices in fog computing systems are expected to be autonomous [62], in Paper A instead of solving (4.8) − (4.9) 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, that is, according to the solution of (4.8) − (4.9), and with the performance of a greedy algorithm. We showed that the proposed algorithm achieves close to optimal system performance, and it performs good even compared with a greedy algorithm that is based on global knowledge of the system state.

Dependent task scheduling problem: There are several works that considered the computation offloading problem, while assuming dependency between the tasks [63, 64, 65, 66, 67]. The authors in [63, 64] addressed the problem of the feature-based visual analysis in a sensor network that consists of a single camera node, and multiple camera nodes, respectively. In [63] the authors proposed an approximate optimal solution for minimizing the completion time of distributed feature extraction. In [64] the authors showed that the multi-sensor completion time minimization problem is NP-hard, and they proposed a distributed solution for allocating the computing tasks that is periodically supported by the centralized coordinator, which provides the limited amount of shared information to the sensor nodes. The authors in [65, 66] considered the problem of joint computation partitioning and resource allocation for delay sensitive applications in a mobile edge cloud computing system that serves multiple mobile users. In [66] the authors extended the model from [65] to consider not only the allocation of computational, but also the allocation of communication resources. They proposed a heuristic for minimizing the average application completion time, under the constraints on the dependency of the partitioned tasks and the constraints on the amount of available resources. The authors in [67] considered the problem of computation offloading in a mobile cloud computing system where the cloud resources are shared among many users. Motivated by the observation that the completion time of the application can be reduced by maximizing the parallelism between the mobile device and the cloud

server, the authors proposed two computation offloading algorithms for two different types of the computational tasks. For sequential tasks, they proposed an algorithm that finds the optimal solution, while for concurrent tasks they propose a load-balancing heuristic.

Energy Consumption Minimization

The need for minimizing the energy consumption of mobile devices is especially prominent in the case of computationally intensive applications such as augmented reality, face and object recognition. There is a significant body of works that analyze the energy consumption minimization problem under delay constraints, and a few works that do not take the delay constraints into account. Common to these classes of works is that they both aim to extend the battery lifetime of mobile devices through energy consumption minimization. However, if the applications are delay sensitive at the same time, then the first class is more relevant, otherwise the latter class can be considered.

Energy consumption minimization subject to delay constraints: Among many works [5, 68, 69, 70, 71] that fall into this class, the work presented in [5] is considered to be one of the pioneering works in the mobile cloud computing area. The authors in [5] considered the case of a single user, and they formulated the offloading problem as a $0-1$ integer linear program. The solution of the program dictates how to partition the application so that the energy consumption of the device is minimized, while meeting the task completion time constraint.

The authors in [68] considered a mobile cloud computing architecture where each mobile device can decide whether to execute the application locally, or on its clone that runs on a virtual machine in a nearby cloud. They considered two scheduling problems, that is, the scheduling problem in the case of local execution and the scheduling problem in the case of execution in the cloud clone. They solved the corresponding convex optimization problems analytically, and showed that the energy consumption can be minimized by optimally configuring the CPU clock frequency of mobile devices in the case of local execution, and by optimally scheduling the data transmission in the case of execution in the cloud clone. The authors in [69] considered a fog computing system that consists of a set of cloud servers and a set of fog devices that are located close to the end users. They proposed an approximate approach to solve the problem of minimizing the energy consumption of the fog computing system while meeting the end users' delay constraints.

The works presented in [70, 71] considered a mobile cloud computing system that consists of multiple mobile users and one cloud server. The authors in [70] proposed a method to optimize the allocation of communication and computational resources by solving the optimization problem that minimizes the transmit power of the mobile devices, under the constraint on the maximum delay. The authors in [71] formulated the problem as a competitive game where the users aim at minimizing their energy consumption while meeting the task completion time constraints. They

showed that the game always has a pure Nash equilibrium, and that the equilibrium can be computed efficiently.

Energy consumption minimization without considering delay constraints: The works presented in [72, 73] used game theoretical tools to model and analyze the interaction among multiple devices in a mobile cloud computing system. The authors in [72] considered that each device can decide whether to perform the computation locally or to offload the computation to one of the multiple cloud servers. They modeled the problem as a congestion game with the objective to reduce the overall energy consumption of a mobile cloud computing system, including the energy consumed by mobile devices and the energy consumed by cloud servers. When formulating the game, the authors considered sharing only computational resources, and they proved the existence of a Nash equilibrium that can be computed in a polynomial time. The authors in [73] considered a mobile cloud computing architecture where multiple mobile users collaborate. By assuming that there is a centralized entity in the cloud that provides the required information to the users, the authors formulated the problem as a $0-1$ integer quadratic program and they used a heuristic to find the optimal solution. Since the average running time of the proposed heuristic increases exponentially with the number of users, they modeled the collaboration among mobile devices as a coalition game, and proposed a distributed coalition formation algorithm that does not require the assistance of the centralized entity.

Completion Time and Energy Consumption Minimization

When minimizing the completion time and the energy consumption together, the objective could be to explore the trade off of the overall application response time versus the consumed energy. The corresponding problems can be divided into two categories depending on whether they involve optimizing one objective function, or multiple objective functions simultaneously.

Single-objective computation offloading problem: There is a significant body of works that consider a single-objective offloading problem, where the system cost is defined as a linear combination of the system delay and the energy consumption [74, 75, 76, 77, 56, 78]. The authors in [74] considered the case of a single user, and based on a stochastic model of the dynamic offloading problem, they proposed a dynamic offloading policy. The authors in [75] integrated dynamic offloading with resource scheduling and they proposed two policies for minimizing the application completion time and the energy consumption of mobile devices. The first policy is based on optimally adjusting the CPU clock frequency of the mobile devices and it is relevant in the case of local execution. The second policy is based on optimally adjusting the transmission power, and it is relevant in the case of offloading. The authors in [76] considered a mobile cloud computing system that consists of multiple mobile devices, an edge cloud, and a centralized cloud. They proposed a heuristic that minimizes the weighted sum of the energy consumption and the corresponding

maximum transmission and execution times among all users, and by performing simulations they showed that their algorithm gives close to optimal performance.

The works presented in [77, 56, 78] used game theoretical tools in order to model and analyze the problem of allocating the resources in mobile cloud computing systems that serve multiple mobile users. The authors in [77] considered a centralized mobile cloud computing system where the users are served by one remote cloud. They provided a two stage game-based formulation of the problem, with the objective to minimize the users' energy consumption and the task completion time in the first stage of the game, and to maximize the cloud service providers' profit in the second stage of the game. The authors in [56, 78] formulated the problem as a strategic game in order to model the sharing of communication resources in a mobile cloud computing system that consists of multiple mobile users and a cloud server. In [56] the authors considered that devices may offload their tasks to the cloud through a single wireless link, and they provided a decentralized algorithm for computing a pure strategy Nash equilibrium of the game. In [78] they showed that the same algorithm can be used in the case of multiple identical wireless links.

Our work presented in Paper B [79] and Paper C [80] falls into this category of computation offloading problems. For each user $i \in \mathcal{N}$ we defined its cost as a linear combination of the task completion time and the energy consumption,

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

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

We considered a mobile cloud computing system that consists of a set of APs, one mobile edge cloud and multiple mobile devices that compete for both communication and computational resources. In Paper B we considered the allocation of communication and computational resources, and in Paper C we integrated the resource allocation problem into the scheduling problem by allowing mobile devices not only to choose where to perform their tasks, but also in which time slot. We used $\mathcal{A}$, $|\mathcal{A}| = A$ to denote the set of APs, and $\mathcal{T}$, $|\mathcal{T}| = T$ to denote the set of time slots. We considered that each device $i \in \mathcal{N}$ can decide in which time slot it wants to perform the computation, and in the chosen time slot it can decide whether to perform the computation locally or to offload the computation to the cloud server through one of the APs. The problem can be formulated as the following $0-1$ integer program

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

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

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

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

where the constraint (4.11) enforces that each device either performs the computation locally or it offloads the task through an AP in one of the time slots.

In Paper B and Paper C we considered that mobile devices are selfish, and we used game theoretical tools to analyze the problem in the case of a single time slot, and in the case of multiple time slots, respectively. We defined the problem as a player-specific network congestion game, for which the existence of equilibria is not known in general. We proved the existence of equilibrium allocations, and based on our constructive proofs we provided polynomial time decentralized algorithms for computing an equilibrium allocation. By providing constructive equilibrium existence proofs, and by characterizing the structure of an equilibrium allocation, our work presented in Paper B and Paper C is also important from a game theoretical perspective.

Multi-objective computation offloading problem: An interesting approach to investigate the Pareto optimal offloading decisions has been considered in [81, 82], where the authors used multi-objective optimization technique that involves two objective functions. The authors in [81] considered a sequence of tasks generated by a single mobile device that can decide which tasks to perform locally, and which tasks to offload to nearby cloudlets and remote centralized clouds. They proposed a multi-objective dynamic programming approach for making Pareto optimal offloading decisions for each of the tasks such that the energy consumption and the application completion time are minimized. The authors in [82] considered a mobile cloud computing system that consists of neighboring mobile devices that act not only as clients, but also as computational service providers. By considering neighboring mobile devices as service providers, they proposed a two-stage algorithm to select service providers such that the tasks' completion time is minimized along minimizing the energy consumption.

Chapter 5

Summary of Original Work

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

Slađana Jošilo and György Dán

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

Summary: In this paper we consider a mobile cloud computing system where multiple devices may offload their computational tasks to each other or to a cloud server 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: A Game Theoretic Analysis of Selfish Mobile Computation Offloading

Slađana Jošilo and György Dán

in Proc. of IEEE International Conference on Computer Communications (INFOCOM), 2017.

Summary: In this paper we investigate the problem of resource allocation in a mobile cloud computing system that consists of multiple mobile devices, multiple APs and an edge cloud. We consider that mobile users are selfish, and thus they aim at minimizing their 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 mobile devices, we formulate the problem as a player-specific congestion game where users compete for communication and computational resources. We prove that a pure Nash equilibrium of the game exists, and we provide a polynomial complexity 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 time of the proposed algorithm. We show that the proposed algorithm achieves the cost performance close to optimal cost performance, and the convergence time of the algorithm scales approximately linearly with the number of mobile devices.

Contribution: The author of this thesis developed the analytical model in collaboration with the second author of the paper, proved the analytical results for the case of both the 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: Decentralized Scheduling for Offloading of Periodic Tasks in Mobile Edge Computing

Slađana Jošilo and György Dán

in Proc. of IFIP Networking (NETWORKING), 2018.

Summary: In this paper we consider periodic computation offloading problem in a mobile cloud 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 a cloud server via one of multiple APs. 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 based on a game theoretical treatment of the problem we prove the existence of a pure strategy Nash equilibria. Based on the constructive equilibrium existence proof, we characterize the structure of computed equilibria, and we develop an efficient decentralized algorithm for computing it. Our numerical results show that the proposed algorithm can be used to compute an efficient resource allocation at polynomial computational complexity despite combinatorial nature of the problem. Finally, the results show that the algorithm computes equilibria with good cost performance for various scenarios of a mobile cloud computing system.

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.

Publications not included in the thesis

1. Slađana Jošilo and György Dán. "Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks". In: *to appear in IEEE Transactions on Mobile Computing (TMC)* (2018)

2. Slađana Jošilo and György Dán. "Decentralized Fog Computing Resource Management for Offloading of Periodic Tasks". In: *Poster presented at IEEE INFOCOM* (2018)

3. Slađana Jošilo, Valentino Pacifici, and György Dán. "Distributed Algorithms for Content Placement in Hierarchical Cache Networks". In: *Computer Networks* (2017)

4. Valentino Pacifici, Slađana Jošilo, and György Dán. "Distributed algorithms for content caching in mobile backhaul networks". In: *Proc. of IEEE International Teletraffic Congress*. Vol. 1. 2016, pp. 313–321

5. Slađana Jošilo et al. "Multicarrier waveforms with I/Q staggering: uniform and nonuniform FBMC formats". In: *EURASIP Journal on Advances in Signal Processing* 2014.1 (2014), p. 167

6. Slađana Jošilo et al. "Widely linear filtering based kindred co-channel interference suppression in FBMC waveforms". In: *Proc. of IEEE International Symposium on Wireless Communications Systems*. 2014, pp. 776–780

7. Slađana Jošilo, Milos Pejovic, and Slobodan Nedic. "Non-uniform FBMC-a pragmatic approach". In: *Proc. of VDE International Symposium on Wireless Communication Systems*. 2013, pp. 1–5

Chapter 6

Conclusions and Future Work

In this thesis, we considered the computation offloading problem in mobile cloud computing systems. We analyzed the problem from the perspective of mobile cloud users and with this in mind we focused on the problem of allocating resources for various mobile cloud computing architectures. By using game theoretical tools we analyzed the interactions among mobile users, and we proposed efficient and scalable algorithms for allocating mobile cloud communication and computational resources.

In the first part of this thesis, we considered a highly distributed mobile cloud computing architecture, which allows users to use cloud resources and the resources of each other. We modeled the transmission and the execution of computational tasks using queuing theory, and provided a game theoretical formulation of the problem. We used variational inequality theory to address the question whether the users can compute an efficient equilibrium task allocation in static mixed strategies in a decentralized manner, under the assumption that every user knows only the average statistics on task arrival intensities, transmission rates, and task parameters.

In the second part of the thesis, we considered mobile edge cloud computing system where users can offload their computational tasks to an edge cloud. By assuming that users are selfish, we formulated the problem as a strategic game. 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 choose not only where to perform their tasks, but also in which time slot.

There are many open questions concerning the problem of resource allocation in mobile cloud computing system. The first interesting question is whether our results from the second part of the thesis can be extended to the case of heterogeneous tasks periodicities. The second interesting question is whether efficient decentralized algorithms exist for allocating mobile cloud resources in a system where the

actual number of users is not known, which would allow for less signaling between the mobile devices and the cloud/mobile network. Finally, the resource allocation problem could be analyzed not only from the perspective of users, but also from the perspective of mobile cloud service providers. Related to the last question, one could consider a mobile cloud computing system where mobile cloud service providers cooperate in serving users computational requests in order to improve their performance benefits.

Bibliography

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

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

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

[4] 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.

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

[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*. Mar. 2012, pp. 2716–2720.

[7] Research and Markets. "Mobile Cloud Market by Application - Global Opportunity Analysis and Industry Forecast". In: *Market rep* (2017).

[8] *Amazon Elastic Computing*. `https://aws.amazon.com/ec2/`.

[9] *Microsoft Azure: Cloud Computing Platform and Services*. `https://azure.microsoft.com/en-in/`.

[10] *Multi-access Edge Computing*. `http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing`.

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

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

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

[14] Flavio Bonomi et al. "Fog computing: A platform for internet of things and analytics". In: *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014, pp. 169–186.

[15] Ehsan Aryafar et al. "RAT selection games in HetNets". In: *Proc. of IEEE INFOCOM*. 2013, pp. 998–1006.

[16] George H. Forman and John Zahorjan. "The challenges of mobile computing". In: *Computer* 27.4 (1994), pp. 38–47.

[17] Mahadev Satyanarayanan. "Fundamental challenges in mobile computing". In: *Proc. of ACM Symposium on Principles of distributed computing*. 1996, pp. 1–7.

[18] Giuseppe Bianchi. "Performance analysis of the IEEE 802.11 distributed coordination function". In: *IEEE Journal on selected areas in communications* 18.3 (2000), pp. 535–547.

[19] Martin Heusse et al. "Performance anomaly of 802.11 b". In: *Proc. of IEEE INFOCOM*. 2003, pp. 836–843.

[20] Tarun Joshi et al. "Airtime fairness for IEEE 802.11 multirate networks". In: *IEEE Transactions on Mobile Computing* 7.4 (2008), pp. 513–527.

[21] Alessandro Biagioni et al. "Adaptive subcarrier allocation schemes for wireless OFDMA systems in WiMAX networks". In: *IEEE Journal on Selected Areas in Communications* 27.2 (2009), pp. 217–225.

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

[23] Anthony J Nicholson and Brian D Noble. "Breadcrumbs: forecasting mobile connectivity". In: *Proc. of ACM*. 2008, pp. 46–57.

[24] Cong Shi et al. *IC-Cloud: Computation offloading to an intermittently-connected cloud*. Tech. rep. Georgia Institute of Technology, 2013.

[25] Eugene E Marinelli. *Hyrax: cloud computing on mobile devices using MapReduce*. Tech. rep. Carnegie-mellon univ Pittsburgh PA school of computer science, 2009.

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

[27] Insun Jang et al. "A Proxy-Based Collaboration System to Minimize Content Download Time and Energy Consumption". In: *IEEE Transactions on Mobile Computing* 16.8 (2017), pp. 2105–2117.

[28] Mohsen Nader Tehrani, Murat Uysal, and Halim Yanikomeroglu. "Device-to-device communication in 5G cellular networks: challenges, solutions, and future directions". In: *IEEE Communications Magazine* 52.5 (2014), pp. 86–92.

[29] Xuemin Shen. "Device-to-device communication in 5G cellular networks". In: *IEEE Network* 29.2 (2015), pp. 2–3.

[30] Nam Tuan Le et al. "Survey of promising technologies for 5G networks". In: *Mobile Information Systems* 2016 (2016).

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

[32] Cong Shi et al. "Serendipity: enabling remote computing among intermittently connected mobile devices". In: *Proc. of ACM Symposium on Mobile Ad Hoc Networking and Computing*. 2012, pp. 145–154.

[33] Abderrahmen Mtibaa et al. "Towards resource sharing in mobile device clouds: Power balancing across mobile devices". In: *Proc. of ACM SIGCOMM Computer Communication Review*. 2013, pp. 51–56.

[34] Synergy Research Group. *The Leading Cloud Providers Continue to Run Away with the Market*. Tech. rep. 2017.

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

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

[37] Simon Ostermann et al. "A performance analysis of EC2 cloud computing services for scientific computing". In: *International Conference on Cloud Computing*. Springer. 2009, pp. 115–131.

[38] Qiming He et al. "Case study for running HPC applications in public clouds". In: *Proc. of ACM Symposium on High Performance Distributed Computing*. 2010, pp. 395–401.

[39] Keith R Jackson et al. "Performance analysis of high performance computing applications on the amazon web services cloud". In: *Proc. of IEEE Cloud Computing Technology and Science*. 2010, pp. 159–168.

[40] Alexandru Iosup et al. "Performance analysis of cloud computing services for many-tasks scientific computing". In: *IEEE Transactions on Parallel and Distributed systems* 22.6 (2011), pp. 931–945.

[41] 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.

[42] Gábor Fodor et al. "Design aspects of network assisted device-to-device communications". In: *IEEE Communications Magazine* 50.3 (2012).

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

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

[45] Mahadev Satyanarayanan et al. "The case for vm-based cloudlets in mobile computing". In: *IEEE pervasive Computing* 8.4 (2009).

[46] Bo Zhao et al. "Mirroring smartphones for good: A feasibility study". In: *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer. 2010, pp. 26–38.

[47] Feng Xia et al. "Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing". In: *Information Systems Frontiers* 16.1 (2014), pp. 95–111.

[48] Roelof Kemp et al. "Cuckoo: a computation offloading framework for smartphones". In: *International Conference on Mobile Computing, Applications, and Services*. Springer. 2010, pp. 59–79.

[49] Byung-Gon Chun et al. "Clonecloud: elastic execution between mobile device and cloud". In: *Proc. of ACM*. 2011, pp. 301–314.

[50] Hao Qian and Daniel Andresen. "Jade: Reducing energy consumption of android app". In: *International Journal of Networked and Distributed Computing (IJNDC), Atlantis press* (2015), pp. 150–158.

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

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

[53] Wanghong Yuan and Klara Nahrstedt. "Energy-efficient CPU scheduling for multimedia applications". In: *Proc. of ACM Transactions on Computer Systems (TOCS)* 24.3 (2006), pp. 292–331.

[54] Jason Flinn. "Cyber foraging: Bridging mobile and cloud computing". In: *Synthesis Lectures on Mobile and Pervasive Computing* 7.2 (2012), pp. 1–103.

[55] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. 4th. Morgan Kaufmann Publishers Inc., 2011.

[56] Xu Chen. "Decentralized computation offloading game for mobile cloud computing". In: *IEEE Transactions on Parallel and Distributed Systems* 26.4 (2015), pp. 974–983.

[57] D. Huang, P. Wang, and D. Niyato. "A Dynamic Offloading Algorithm for Mobile Computing". In: *IEEE Transactions on Wireless Communications* 11.6 (2012), pp. 1991–1995.

[58] 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.

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

[60] 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.

[61] Valeria Cardellini et al. "A game-theoretic approach to computation offloading in mobile cloud computing". In: *Mathematical Programming* 157.2 (2016), pp. 421–449.

[62] Luis M Vaquero and Luis Rodero-Merino. "Finding your way in the fog: Towards a comprehensive definition of fog computing". In: *ACM SIGCOMM Computer Communication Review* 44.5 (2014), pp. 27–32.

[63] Emil Eriksson, György Dán, and Viktoria Fodor. "Predictive distributed visual analysis for video in wireless sensor networks". In: *IEEE Transactions on Mobile Computing* (2016), pp. 1743–1756.

[64] Emil Eriksson, György Dán, and Viktoria Fodor. "Coordinating Distributed Algorithms for Feature Extraction Offloading in Multi-Camera Visual Sensor Networks". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2017).

[65] Lei Yang et al. "Multi-user computation partitioning for latency sensitive mobile cloud applications". In: *IEEE Transactions on Computers* 64.8 (2015), pp. 2253–2266.

[66] 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.

[67] Mike Jia, Jiannong Cao, and Lei Yang. "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing". In: *Proc. of IEEE INFOCOM Workshop*. 2014, pp. 352–357.

[68] Weiwen Zhang et al. "Energy-optimal mobile cloud computing under stochastic wireless channel". In: *IEEE Transactions on Wireless Communications* 12.9 (2013), pp. 4569–4581.

[69] Ruilong Deng et al. "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption". In: *IEEE Internet of Things Journal* 3.6 (2016), pp. 1171–1181.

[70] Stefania Sardellitti, Gesualdo Scutari, and Sergio Barbarossa. "Joint optimization of radio and computational resources for multicell mobile-edge computing". In: *IEEE Transactions on Signal and Information Processing over Networks* 1.2 (2015), pp. 89–103.

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

[72] Yang Ge et al. "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system". In: *Proc. of ACM/IEEE Symposium on Low power electronics and design*. 2012, pp. 279–284.

[73] Liyao Xiang, Baochun Li, and Bo Li. "Coalition Formation Towards Energy-Efficient Collaborative Mobile Computing". In: *Proc. of IEEE Computer Communication and Networks (ICCCN)*. 2015, pp. 1–8.

[74] Esa Hyytiä, Thrasyvoulos Spyropoulos, and Jörg Ott. "Offload (only) the right jobs: Robust offloading using the Markov decision processes". In: *Proc. of IEEE World of Wireless, Mobile and Multimedia Networks Symposium*. 2015, pp. 1–9.

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

[76] Meng-Hsi Chen, Min Dong, and Ben Liang. "Joint offloading decision and resource allocation for mobile cloud with computing access point". In: *Proc. of IEEE Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 3516–3520.

[77] Yanzhi Wang, Xue Lin, and Massoud Pedram. "A nested two stage game-based optimization framework in mobile cloud computing system". In: *Proc. of IEEE Service Oriented System Engineering Symposium*. 2013, pp. 494–502.

[78] Xu Chen et al. "Efficient multi-user computation offloading for mobile-edge cloud computing". In: *IEEE/ACM Transactions on Networking* 24.5 (2016), pp. 2795–2808.

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

[80] Slađana Jošilo and György Dán. "Decentralized Scheduling for Offloading of Periodic Tasks in Mobile Edge Computing". In: *Proc. of IFIP/TC6 Networking*. 2018.

[81] Xinchen Lyu and Hui Tian. "Adaptive receding horizon offloading strategy under dynamic environment". In: *IEEE Communications Letters* 20.5 (2016), pp. 878–881.

[82] Simin Ghasemi-Falavarjani, Mohammadali Nematbakhsh, and Behrouz Shahgholi Ghahfarokhi. "Context-aware multi-objective resource allocation in mobile cloud". In: *Computers & Electrical Engineering* 44 (2015), pp. 218–240.

[83] Slađana Jošilo and György Dán. "Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks". In: *to appear in IEEE Transactions on Mobile Computing (TMC)* (2018).

[84] Slađana Jošilo and György Dán. "Decentralized Fog Computing Resource Management for Offloading of Periodic Tasks". In: *Poster presented at IEEE INFOCOM* (2018).

[85] Slađana Jošilo, Valentino Pacifici, and György Dán. "Distributed Algorithms for Content Placement in Hierarchical Cache Networks". In: *Computer Networks* (2017).

[86] Valentino Pacifici, Slađana Jošilo, and György Dán. "Distributed algorithms for content caching in mobile backhaul networks". In: *Proc. of IEEE International Teletraffic Congress*. Vol. 1. 2016, pp. 313–321.

[87] Slađana Jošilo et al. "Multicarrier waveforms with I/Q staggering: uniform and nonuniform FBMC formats". In: *EURASIP Journal on Advances in Signal Processing* 2014.1 (2014), p. 167.

[88] Slađana Jošilo et al. "Widely linear filtering based kindred co-channel interference suppression in FBMC waveforms". In: *Proc. of IEEE International Symposium on Wireless Communications Systems*. 2014, pp. 776–780.

[89] Slađana Jošilo, Milos Pejovic, and Slobodan Nedic. "Non-uniform FBMC-a pragmatic approach". In: *Proc. of VDE International Symposium on Wireless Communication Systems*. 2013, pp. 1–5.

Paper **A**

Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems

Slađana Jošilo and György Dán

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

Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems

Slađana Jošilo and György Dán

School of Electrical Engineering,
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 a cloud server. 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 cloud server. 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.

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 cloud computational and storage resources, but also to use the resources of each other in a collaborative way [3].

Traditional centralized cloud computing allows devices to offload the computation to a cloud infrastructure with significant computational power [4–6]. 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 [7–10].

Fog computing addresses this problem by allowing collaborative computation offloading among nearby devices and distributed cloud resources close to the network edge. 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 [11–13]. 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 a cloud. We provide a game theoretical treatment of the completion time minimization problem, and we show that an equilibrium task allocation in static mixed strategies always exists. 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 a cloud server. 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

Figure 1: Fog computing system that serves 5 devices.

random variable L_i. Similar to other works [14–16], 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 a cloud server. Thus, device i chooses a member of the set $\mathcal{N} \cup \{0\}$, where 0 corresponds to the 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., $p_i(k) \in \mathcal{P}$.

We assume that all devices faithfully execute the tasks offloaded to them. This can be ensured with an adequate incentive scheme similar to those presented in [17–19], e.g., if only devices that process offloaded tasks themselves are entitled to offload tasks. Figure 1 illustrates a computation offloading system that serves five devices such that devices 1 and 2 offload their tasks through a base station to the cloud, device 3 performs computation locally, and devices 4 and 5 offload their tasks to devices 2 and 3, respectively.

2.1 Communication model

We consider that the devices communicate using a single cell uplink OFDM system [20], and there is an assignment of subcarriers to pairs of communicating nodes. We denote the transmission rate from device i to device j by $R_{i,j}$, and the transmission rate from device i to the cloud server 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 cloud server. We assume that the queues are large enough to be considered infinite, 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

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

Furthermore, the time $T^d_{i,j}(k)$ needed to deliver the input data $D_{i,k}$ from device i to $j \in$

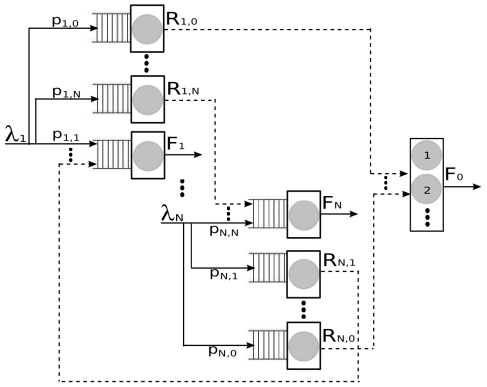

Figure 2: Fog computing system modeled as a queuing network.

$\mathcal{N} \cup \{0\}$ is the sum of the transmission time $T^t_{i,j}(k)$ and of the waiting time (if any).

Similar to other works [21–23], 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 $\lambda_i = 0$, and in which certain devices $j \in \mathcal{N}$ are not supposed to perform the computational tasks of others, by setting the transmission rates from the other devices $i \in \mathcal{N} \setminus \{j\}$ to device j to $R_{i,j} = 0$.

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 and we assume that the execution queues are large enough to be considered infinite. Unlike devices, the cloud has a large number of processors with computational capability F_0 each, and we assume that computing in the cloud begins immediately upon arrival of a task. Similar to common practice 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

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

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

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

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 [24–26].

In what follows we propose two decentralized solutions for the MCOG problem in the form of a Markov strategy and in static mixed strategies.

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. In the MBR algorithm, every device i makes a decision based on a myopic best response strategy, i.e., every device i chooses the node $j \in \mathcal{N} \cup \{0\}$ that minimizes the completion time of its task $t_{i,k}$, given the instantaneous states of the transmission and execution queues. Observe 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. The pseudo-code for computing the myopic best response strategy is shown in Figure 3.

Note that if we define the system state upon the arrival of task $t_{i,k}$ as the jobs in the transmission and execution queues and the size and complexity of the task to be offloaded, 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 equilibrium.

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

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

$p_{i,j}(k) = 0, \quad \forall j \in \mathcal{N} \cup \{0\}$
/* Estimate completion time of $t_{i,k}$ in $\forall j \in \mathcal{N} \cup \{0\}$ */
for $j = 0, \ldots, N$ **do**
  if $j = i$ **then**
    $ECompleteT(j) = T^e_{i,j}(k)$
  else
    $ECompleteT(j) = T^d_{i,j}(k) + T^e_{i,j}(k)$
  end if
end for
/* Make a greedy decision */
$i' \leftarrow \arg\min_{j \in \mathcal{N} \cup \{0\}} ECompleteT(j)$
$p_{i,i'}(k) = 1$
return $p_i(k)$

Figure 3: Pseudo code of myopic best response.

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, we can model each transmission queue with 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. This approximation allows us to model the execution queue of each device as an $M/G/1$ system, and the 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 $\rho^e_j = \sum_{i' \in \mathcal{N}} p_{i',j}\lambda_{i'} \overline{T^c}_{i',j} < 1$.

Since computing in the 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.,

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

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 $(p_i)_{i\in\mathcal{N}}$,

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

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

$$\mathcal{K}_i(p_{-i}) = \{p_i \in \mathcal{P} | \rho^t_{i,j} \leq S_t, \rho^e_{i'} \leq S_t, \forall j \in \mathcal{N} \backslash \{i\} \cup \{0\}, \forall i'\},$$

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,

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

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

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

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

$$h_{i,0} = \overline{T^t}_{i,0} + \overline{T^c}_{i,0} + \frac{\gamma_{i,0}}{2\varepsilon_{i,0}} + p_{i,0}\lambda_i\Big(\frac{^2\overline{T^t}_{i,0}}{2\varepsilon_{i,0}} + \frac{\overline{T^t}_{i,0}\gamma_{i,0}}{2\varepsilon^2_{i,0}}\Big),$$

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

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

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

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

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

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

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 [27]).

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 [28], 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 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 [29], 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*

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

then the function F is monotone.

The proof is given in Appendix .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 the 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}.

Unless otherwise noted, we placed the devices at random on a *regular grid* with 10^4 points defined over a square area of $1km \times 1km$, and the cloud is located at the center of the grid. 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, 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 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 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 [30]. We set the data transmit power P_i^t of every device i to 0.4 W according to [31] and given the noise power P_n we calculate the transmission rate $R_{i,j}$ from device i to node $j \in \mathcal{N} \setminus \{i\} \cup \{0\}$ as $R_{i,j} = B_{i,j} log(1 + P_i^t d_{i,j}^{\alpha} / P_n)$.

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 cloud is $F_0 = 64$ *GHz* [32]. 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 on $[0.7, 1]$ *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 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. The results shown are the averages of 50 simulations, together with 95% confidence intervals.

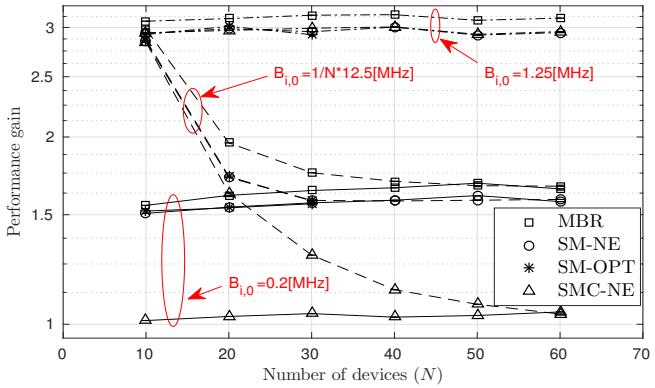

Figure 4: *Performance gain* vs. number of devices for $B_{i,0} = 0.2$ *MHz*, $B_{i,0} = 1.25$ *MHz* and $B_{i,0} = \frac{1}{N}12.5$ *MHz*.

5.1 Performance gain

We start with evaluating the performance gain as a function of the number of devices. Figure 4 shows the performance gain for the MBR, SM-NE, SM-OPT and for the SMC-NE algorithms as a function of the number of devices for the two device to cloud bandwidth assignment scenarios. 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 for the SM-OPT algorithm are shown only up to 30 devices, because the computation of the socially optimal strategy profile was computationally infeasible for larger problem instances.

Figure 4 shows the performance gain for the MBR, SM-NE, SM-OPT and for the SMC-NE algorithms as a function of the number of devices. 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 cloud, as on average it is best to do so, even if given the instantaneous system state it may be

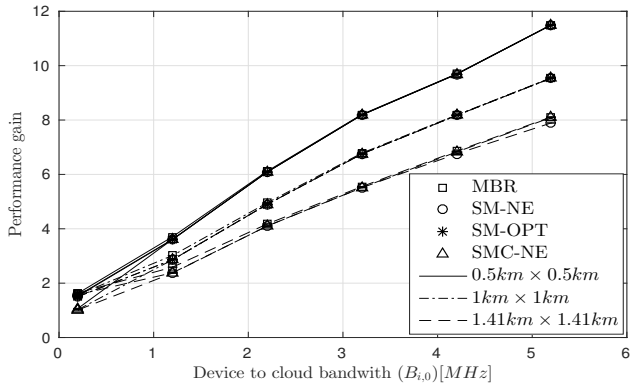

Figure 5: *Performance gain* vs. device to cloud bandwidth $B_{i,0}$ for $N = 8$ devices placed over $0.5km \times 0.5km$ square area, for $N = 30$ devices placed over $1km \times 1km$ square area, and for $N = 60$ devices placed over $1.41km \times 1.41km$ square area.

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 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 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 cloud, in the following we vary the bandwidth $B_{i,0}$ between 0.2 *MHz* and 5.2 *MHz*. Figure 5 shows the performance gain for the MBR, SM-NE, SM-OPT and for the 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.

The figure shows that the performance gain achieved by the algorithms increases with

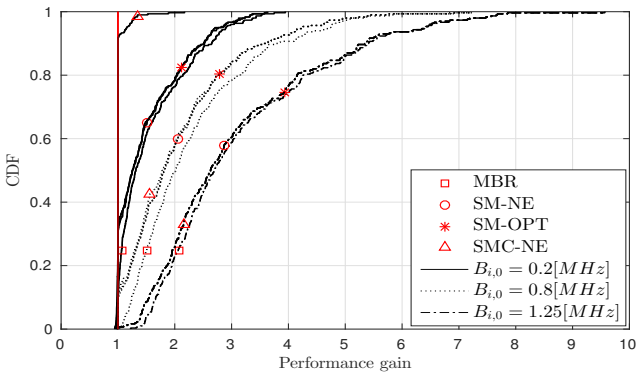

Figure 6: Distribution of the performance gain for $N = 30$ devices, $B_{i,0} = 0.2$ *MHz*, $B_{i,0} = 0.8$ *MHz* and $B_{i,0} = 1.25$ *MHz*.

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 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 on average in a smaller area.

5.3 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 4 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 6 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 the MBR algorithms achieve a performance gain below one for a small fraction

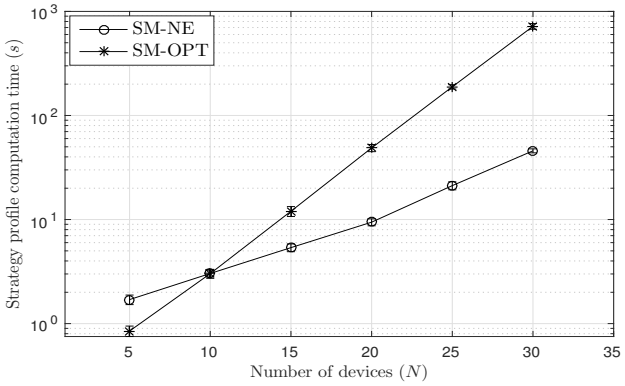

Figure 7: Time needed to compute a static mixed strategy equilibrium and a socially optimal static mixed strategy profile for $B_{i,0} = 1.25$ *MHz*.

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 gain of every device is larger than one, but 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. These results show that collaboration among the devices is ex-post individually rational only if sufficient bandwidth is provided for communication to the cloud. Thus, if ex-post individual rationality is important then the device to cloud bandwidth has to be managed appropriately.

5.4 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 7 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.

6 Related Work

There is a large body of work on augmenting the execution of computationally intensive applications using cloud resources [6, 21, 33–35]. [33] studied the problem of maximizing the throughput mobile data stream applications through partitioning, and proposed a genetic algorithm as a solution. [34] considered multiple QoS factors in a 2-tiered cloud infrastructure, and proposed a heuristic for minimizing the users' cost. [35] proposed an iterative algorithm that minimizes the users' overall energy consumption, while meeting latency constraints. [21] 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. [6] 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. [15] 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. [36] 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. [37] 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. [38] 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. [14] 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. [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 a 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. [40] 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. [41] formulated the power consumption-delay tradeoff prob-

lem 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. [42] 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.

.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

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

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

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

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

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

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

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

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

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

where

$$h^t_{i,j} = \frac{\lambda^2}{\varepsilon_{i,j}} \big(^2\overline{T^t}_{i,j} + \frac{\gamma_{i,j}\overline{T^t}_{i,j}}{\varepsilon_{i,j}}\big)\big(1 + p_{i,j}\frac{\lambda\overline{T^t}_{i,j}}{\varepsilon_{i,j}}\big),$$

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

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 $M^s_i = \frac{1}{2}(M^T_i + M_i)$ is positive semidefinite.

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

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

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

where

$$h^t_{j,i} = \frac{\lambda^2}{\varepsilon_{j,i}} \left({}^2\overline{T^t}_{j,i} + \frac{\gamma_{j,i} \overline{T^t}_{j,i}}{\varepsilon_{j,i}}\right)\left(1 + p_{j,i} \frac{\lambda \overline{T^t}_{j,i}}{\varepsilon_{j,i}}\right),$$

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

$${}^o h^s_{j,i} = \frac{\lambda \overline{T^c}_{i,i} \lambda \overline{T^c}_{j,i}}{\zeta_i^2} \left(1 + \frac{\lambda}{\zeta_i}(p_{i,i} \overline{T^c}_{i,i} + p_{j,i} \overline{T^c}_{j,i})\right)$$

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

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

Furthermore, let us define matrix T^p_i as

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

Now, matrix M_i can be rewritten as

$$M_i = \frac{\lambda^2}{\zeta_i^2}\left(\overline{T^c}_i\,\overline{T^c}_i^T \circ \left(I + E + \frac{2\lambda}{\zeta_i}T_i^p\right)\right) + \overline{T^t}_i,$$

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 $\overline{T^c}_i\,\overline{T^c}_i^T$ follows from the definition. Observe that matrix $\overline{T^t}_i$ 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 [43], the proof reduces to showing that matrix $I + E + \frac{2\lambda}{\zeta_i}T_i^p$ is positive semidefinite. To do so, we will show that the minimum eigenvalue of the matrix $\frac{2\lambda}{\zeta_i}T_i^p$ is greater than or equal to -1. To do so, let us denote by e the all-ones vector and define the vector $t_i^p = (p_{1,i}\overline{T^c}_{1,i}\ p_{2,i}\overline{T^c}_{2,i}\ldots p_{N,i}\overline{T^c}_{N,i})$. Now, we can express matrix T_i^p as

$$T_i^p = \frac{1}{2}\left(t_i^p e^T + e(t_i^p)^T\right).$$

The characteristic polynomial of the symmetric matrix T_i^p is given by [44]

$$\frac{k^{N-2}}{2}\left(k^2 - 2(e^T t_i^p)k + (e^T t_i^p)^2 - N\|t_i^p\|^2\right).$$

We observe that T_i^p has $N-2$ zero eigenvalues, and one non-negative and one non-positive eigenvalue given by $k_+ = \left(e^T t_i^p + \sqrt{N}\|t_i^p\|\right)/2$ and $k_- = \left(e^T t_i^p - \sqrt{N}\|t_i^p\|\right)/2$, respectively. Therefore, the minimum eigenvalue of the matrix $\frac{2\lambda}{\zeta_i}T_i^p$ is greater than -1 if

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

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

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

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.

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 cloud computing technologies for internet of things: A primer," *Digital Communications and Networks*, 2017.

[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] D. Kovachev, Y. Cao, and R. Klamma, "Mobile cloud computing: a comparison of application models," *arXiv preprint arXiv:1107.4940*, 2011.

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

[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 Journal on selected areas in communications*, pp. 1065–1082, 2014.

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

[9] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[10] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proceedings of the tenth international conference on Information and knowledge management*. ACM, 2001, pp. 310–317.

[18] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 640–651.

[19] 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.

[20] E. Yaacoub and Z. Dawy, *Resource Allocation in Uplink OFDMA Wireless Systems: Optimal Solutions and Practical Implementations*. John Wiley & Sons, 2012, vol. 24.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

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

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

[31] 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 on Internet measurement conference*. ACM, 2009, pp. 280–293.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

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

[39] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: enabling remote computing among intermittently connected mobile devices," in *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2012, pp. 145–154.

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

[41] 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.

[42] 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.

[43] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

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

Paper B

A Game Theoretic Analysis of Selfish Mobile Computation Offloading

Slađana Jošilo and György Dán

in Proc. of IEEE International Conference on Computer Communications (INFOCOM), 2017.

A Game Theoretic Analysis of Selfish Mobile Computation Offloading

Slađana Jošilo and György Dán

School of Electrical Engineering,
KTH Royal Institute of Technology, Stockholm, Sweden
E-mail: {josilo,gyuri}@kth.se.

Abstract

Offloading computation to a mobile cloud is a promising approach for enabling the use of computationally intensive applications by mobile devices. In this paper we consider autonomous devices that maximize their own performance by choosing one of many wireless access points for computation offloading. We develop a game theoretic model of the problem, prove the existence of pure strategy Nash equilibria, and provide a polynomial time 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. We provide a bound on the price of anarchy of the game, thus our algorithm serves as an approximation algorithm for the global computation offloading cost minimization problem. We use extensive simulations to provide insight into the performance and the convergence time of the algorithms in various 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.

1 Introduction

Computationally intensive applications, including augmented reality, natural language processing, face, gesture and object recognition, and various forms of user profiling for recommendations [1, 2] are increasingly used on mobile devices. Many of these applications consume a significant amount of energy, which can be detrimental to battery life, and together with potentially slow response times due to the limited computational power of the mobile handsets, it may limit user acceptance.

A promising approach to extend the battery lifetime of mobile handsets and to serve the computational needs of computationally intenstive applications is mobile cloud computing [3, 4]. Mobile cloud computing allows to offload the computations through a wireless access point to a cloud infrastructure with significant computational power. The computations can be performed in the cloud and the results sent back to the mobile handset. Commercial cloud

infrastructures, such as Amazon EC2, may have plentyful computational resources, but they may not be able to provide sufficiently low response times for many applications. It may thus be better to offload computations to less resourceful mobile edge computing (MEC) infrastructures, which are considered an enabler of 5G mobile networks, as they are located close to the network edge [5].

The proximity of MEC resources to the network edge ensures low propagation times, but when many mobile devices attempt to offload computations simultaneously, the response times could be affected by the contention between the mobile devices for MEC computing resources and for wireless communication resources [6,7]. The problem is inherently difficult for various reasons. First, the computational tasks of the mobile devices may have different complexities and may need the transmission of different amounts of data. Second, each device could aim at minimizing a combination of its response time and energy consumption for performing the computation. Third, the number of offloading choices for each mobile device increases with the number of access points. Thus, developing scalable algorithms that coordinate the offloading decisions of the mobile devices to ensure the efficient use of MEC resources and to provide predictable performance to the mobile devices is a challenging problem.

In this paper we address this problem by considering the allocation of cloud and wireless resources among mobile devices that can choose either to offload their computation to a cloud through one of many access points or to perform the computation locally. We make three important contributions to solve the problem. First, based on a game theoretical treatment of the problem, we propose an efficient distributed 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 distributed algorithm can be used for computing equilibria when the cloud computing resources scale directly proportional with the number of mobile users. Finally, by establishing a bound on the price of anarchy of the strategic game, we show that the proposed algorithms have a bounded approximation ratio. We use extensive simulations to illustrate the computational efficiency of the algorithms and to evaluate their approximation ratio 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. We provide a bound on the approximation ratio 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 (MU). To facilitate the analysis, we make the assumption that the set of MUs changes slowly, e.g., in the order of seconds or minutes, similar to other works [4, 8–10]. Each MU has a computationally intensive task to perform, which is characterized by the size D_i of

the input data (e.g., in bytes), and by the number L_i of CPU cycles required to perform the computation. 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 $\mathcal{A}=\{1,2,...,A\}$.

2.1 Communication model

If the MU 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 server through one of the APs. Thus, together with local computing MU i can choose one element of the set $\mathfrak{D}_i=\{0,1,2,...,A\}$, 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 $n_a(\mathbf{d})$ the number of MUs that use AP a for computation offloading, and by $n(\mathbf{d})=\sum_{a\in\mathcal{A}} n_a(\mathbf{d})$ the number of MUs that offload. Similarily, for an AP $a \in \mathcal{A}$ we denote by $O_a(\mathbf{d}) = \{i|d_i = a\}$ the set of MUs that offload using AP a, and we define the set of offloaders as $O(\mathbf{d}) = \cup_{a\in\mathcal{A}} O_a(\mathbf{d})$.

We denote by $R_{i,a}$ the PHY rate of MU i on AP a, which depends on the physical layer signal characteristics and the corresponding channel gain. We denote by $\omega_{i,a}(\mathbf{d})$ the uplink rate that MU i receives when she offloads via AP a. $\omega_{i,a}(\mathbf{d})$ depends on the PHY rate $R_{i,a}$ and on the number $n_a(\mathbf{d})$ of MUs that offload via AP a

$$\omega_{i,a}(\mathbf{d}) = \frac{R_{i,a}}{n_a(\mathbf{d})}. \tag{1}$$

This model of the uplink rate can be used to model the bandwidth sharing in TDMA and OFDMA based MAC protocols [11].

The uplink rate $\omega_{i,a}(\mathbf{d})$ together with the input data size D_i determines the transmission time $T_{i,a}^{c,off}(\mathbf{d})$ of MU i for offloading via AP a,

$$T_{i,a}^{c,off}(\mathbf{d}) = \frac{D_i}{\omega_{i,a}(\mathbf{d})}. \tag{2}$$

To model the energy consumption of the MUs, we assume that MU i uses a constant transmit power of P_i for sending the data, thus the energy consumption of MU i for offloading the input data of size D_i via AP a is

$$E_{i,a}^{c}(\mathbf{d}) = \frac{D_i P_i}{\omega_{i,a}(\mathbf{d})}. \tag{3}$$

2.2 Computation model

In what follows we introduce our model of the time and energy consumption in the case of local computing and in the case of computation offloading.

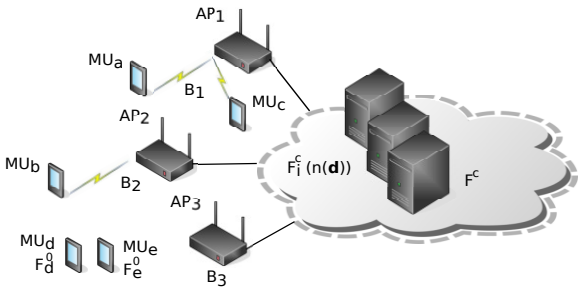

Figure 1: An example of a mobile cloud computing system.

2.2.1 Local computing

In the case of local computing the task has to be processed using local computing resources. We denote by F_i^0 the computational capability of MU i, and hence the local computing time needed to perform its computation task that requires L_i CPU cycles can be expressed as

$$T_i^0 = \frac{L_i}{F_i^0}. \tag{4}$$

We consider that the energy consumption of local computing is proportional to the computation time, thus denoting by v_i the consumed energy per CPU cycle, we obtain

$$E_i^0 = v_i L_i. \tag{5}$$

2.2.2 Cloud computing

In the case of cloud computing, after the data are transmitted via an AP, processing is done at the cloud server. We denote by F^c the computation capability of the cloud, and we consider that the computation capability $F_i^c(n(\mathbf{d}))$, assigned to MU i by the cloud is a non-increasing function of the number $n(\mathbf{d})$ of MUs that offload. Given $F_i^c(n(\mathbf{d}))$ we use a linear model to compute the execution time of a task $<D_i, L_i>$ that is offladed by MU i,

$$T_i^{c,exe} = \frac{L_i}{F_i^c(n(\mathbf{d}))}. \tag{6}$$

Figure 1 shows an example of a mobile cloud computing system in which three of five MUs offload their tasks using one of three APs.

2.3 Cost Model

We model the cost of a MU as a linear combination of the time it takes to finish the computation and the corresponding energy consumption. We use the weights γ_i^T attributed

to the time it takes to finish the computation and γ_i^E attributed to energy consumption, in order to model the delay sensitivity of the application and the consumed energy, respectively, $0 \leq \gamma_i^E, \gamma_i^T \leq 1$.

Using these notation, for the case of local computing the cost of MU i is determined by the local computing time and the consumed energy per CPU cycle

$$C_i^0 = \gamma_i^T T_i^0 + \gamma_i^E E_i^0 = (\frac{\gamma_i^T}{F_i^0} + \gamma_i^E v_i) L_i. \tag{7}$$

For the case of offloading via AP a, the cost of MU i is determined by the transmission time, the corresponding transmit energy, and the computing time in the cloud

$$C_{i,a}^c(\mathbf{d}) = \gamma_i^T (T_i^{c,exe} + T_{i,a}^{c,off}(\mathbf{d})) + \gamma_i^E E_{i,a}^c(\mathbf{d}) = (\gamma_i^T + \gamma_i^E P_i) \frac{D_i}{\omega_{i,a}(\mathbf{d})} + \gamma_i^T \frac{L_i}{F_i^c(n(\mathbf{d}))}. \tag{8}$$

Similar to previous works [7, 12, 13], 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.

To define the cost of MU i in strategy profile $\mathbf{d}$, let us introduce the indicator function $I(d_i, a)$ for MU i as

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

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

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

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

2.4 Computation Offloading Game

We consider that each MU aims at minimizing its 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 $C(\mathbf{d})$.

We thus consider that the MUs play a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$, in which the players are the MUs. 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 cost by changing her strategy, i.e., a pure Nash equilibrium of the game Γ.

$\mathbf{d} = ImprovementPath(\mathbf{d})$

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: $\quad d_i = d_i'$
3: **end while**
4: **return d**

Figure 2: Pseudo code of the *ImprovementPath* algorithm.

Definition 1. A Nash equilibrium (NE) of the strategic game $<\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$ is a strategy profile 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, according to the *ImprovementPath* Algorithm shown in Figure 2, 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 improvement path in which all improvement steps are best reply a *best improvement path*. Observe that in a NE all MUs play their best replies to each others' strategies.

3 Equilibria and the JPBR Algorithm

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

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

$$D_{O\to O}(\mathbf{d}) = \{a \in \mathcal{A} | \exists i \in O_a(\mathbf{d}), \exists b \in \mathcal{A} \setminus \{a\}, (n_b(\mathbf{d})+1)/R_{i,b} < n_a(\mathbf{d})/R_{i,a}\}.$$

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

$$D_{O\to L}(\mathbf{d}) = \{a \in \mathcal{A} | \exists i \in O_a(\mathbf{d}), C_{i,a}^c(\mathbf{d}) > C_i^0\}$$

Definition 3. The reluctance to offload via AP a of MU i in a strategy profile $\mathbf{d}$ is $\rho_i(\mathbf{d}) = \frac{C_{i,a}^c(\mathbf{d})}{C_i^0}$.

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

3.1 The ImproveAP Algorithm

Using these definitions, let us start with investigating whether the simple *Improvement-Path* algorithm 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.

Example 1. *Consider a MCOG with* $\mathcal{N} = \{a,b,c,d,e\}$ *and* $\mathcal{A} = \{1,2,3\}$ *as illustrated in Figure 1. Figure 3 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.*

d_i	d_a	d_b	d_c	d_d	d_e
d(0)	1	2	1	0	0
d(1)	1	2	2	0	0
d(2)	1	0	2	0	0
d(3)	1	0	2	2	0
d(4)	1	0	2	2	2
d(5)	1	0	1	2	2
d(6)	1	3	1	2	2
d(7)	1	3	1	2	0
d(8)	1	3	1	0	0
d(9)	1	2	1	0	0

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

Figure 3: A cyclic improvement path in a computation offloading game with 3 APs and 5 MUs. 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.

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, $\frac{1}{R_{b,3}}(\gamma_b^T + \gamma_b^E P_b)D_b + 5\gamma_b^T \frac{L_b}{F^c} > \frac{1}{R_{b,3}}(\gamma_b^T + \gamma_b^E P_b)D_b + 3\gamma_b^T \frac{L_b}{F^c}$ 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 $\gamma_d^T \frac{L_d}{F^c} > \frac{1}{R_{d,2}}(\gamma_d^T + \gamma_d^E P_d)D_d$. Similarly, it follows from inequalities (4) and (7) that $\gamma_e^T \frac{L_e}{F^c} > \frac{1}{R_{e,2}}(\gamma_e^T + \gamma_e^E P_e)D_e$. 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 $i, i' \in \mathcal{N}$, $i \neq i'$.

An important consequence of the cycle in the improvement path is that the MCOG does not allow a generalized ordinal potential function, and the *ImprovementPath* algorithm

$\mathbf{d} = ImproveAP(\mathbf{d})$

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

Figure 4: Pseudo code of the *ImproveAP* algorithm.

cannot be used for computing NE. Although improvement paths may cycle, as we next show, improvement paths are finite if we only allow the MUs to change between APs but not to start or to stop offloading. We refer to this algorithm as the *ImproveAP* algorithm, and show its pseudo code in Figure 4. Our first result shows that all improvement paths generated by the *ImproveAP* algorithm are finite.

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

Proof. We prove the lemma by showing that the function

$$\Phi(\mathbf{d}) = \sum_{a'=1}^{A} \sum_{n=1}^{n_{a'}(\mathbf{d})} log(n) - \sum_{a'=1}^{A} \sum_{i'=1}^{N} log(R_{i',a'}) I(d_{i'}, a').$$

decreases strictly at every improvement step generated by the *ImproveAP* algorithm.

Let us consider an improvement step made by MU i in which she changes from offloading via AP b to offloading via AP a. First, 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 that $n_a(a,d_{-i})/n_b(b,d_{-i}) < R_{i,a}/R_{i,b}$. Since $n_a(a,d_{-i}), n_b(b,d_{-i}) > 0$, and $R_{i,a}, R_{i,b} > 0$ this is equivalent to

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

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

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

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

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

It follows from (12) that $\Phi(a,d_{-i}) - \Phi(b,d_{-i}) < 0$. Since the number of strategy profiles is finite, $\Phi(\mathbf{d})$ can not decrease infinitely and the *ImproveAP* algorithm terminates after a finite number of improvement steps. □

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

3.2 The JPBR Algorithm

In what follows we use the *ImproveAP* algorithm as a building block for proving that a NE always exists in the MCOG even if it cannot be computed using the *ImprovementPath* algorithm.

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.

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. Otherwise, if MU i decides to offload, she would play her best reply which is given by $d_i^*(1) = \arg\max_{a\in\mathcal{A}} R_{i,a}$. 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 5. 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 $d^*_{N^{(t)}}(t) = a \in \mathcal{A}$, 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 $i \in O_a(\mathbf{d}(t))$ offloading using AP $b \in \mathcal{A}\setminus\{a\}$ becomes a best reply, (ii) for some MUs $i \in O(\mathbf{d}(t))$ local computing becomes a best reply.

Let us first consider case (i) and let MUs execute the *ImproveAP* 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. In the resulting strategy profile the number of MUs that offload will be $n(\mathbf{d}^*(t-1))+1$. Furthermore, there will be one AP (denoted by a') for which the number of offloaders is $n_{a'}(\mathbf{d}^*(t-1))+1$, while for the other APs $a \neq a'$ 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.

Update phase of JPBR algorithm

/* Corresponds to case (i) */
Let $\mathbf{d}'(t) = \text{ImproveAP}(\mathbf{d}(t))$
/* 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**
    $b \leftarrow \arg\max_{a \in D_{O\to L}} \rho_{(a,1)}(\mathbf{d}'(t))$
                /* AP 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 $\mathbf{d}'(t) = \text{ImproveAP}(\mathbf{d}'(t))$
    end if
  end while
end if

Figure 5: Pseudo code of the update phase of the JPBR algorithm.

Note that if case (i) did not happen, i.e. $|D_{O\to O}\mathbf{d}(t)| = 0$, then AP a' is the same AP a that was chosen by MU $N^{(t)}$ when it 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 AP a' to local computation, $n_a(\mathbf{d}'(t)) = n_a(\mathbf{d}^*(t-1))$ holds for every AP $a \in \mathcal{A}$ 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 MU i that wants to change her strategy from offloading through AP $b \in \mathcal{A} \setminus \{a'\}$ 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 $(b, 1)$ for some $b \neq a'$). We denote the resulting strategy profile by $\mathbf{d}'(t)$. Due to this improvement step $n_b(\mathbf{d}'(t)) = n_b(\mathbf{d}^*(t-1)) - 1$, and thus 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, then there

is no more MU that would like to stop offloading either because $n(\mathbf{d}'(t)) = n(\mathbf{d}^*(t-1))$. 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 a MU that did not offload before, then we reached a NE. Otherwise some MUs that offload via AP $a \in \mathcal{A} \setminus \{b\}$ may want to connect to AP b, and we let them execute the *ImproveAP* 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 1. *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 (ii) 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 by choosing another AP. 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. Therefore, a NE is reached after at most $(N^{(t)} - 2) + (N^{(t)} - 2) + (N^{(t)} - 3) + (N^{(t)} - 2)(A-1)$ update steps. □

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 2. *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, we can consider MU i as a 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 a 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 2 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 MCOG with polynomial complexity. In what follows we show that a much simpler algorithm can be used for computing an equilibrium if the cloud computation capability assigned to MU i is independent of the other players' strategies, $F_i^c(n(\mathbf{d})) = F^c$, and thus of the number of MUs that offload. This model can be relevant for large cloud computing infrastructures, in which the cloud computing resources scale with the number of MUs, and we refer to this model as the *elastic* cloud model. In the case of the *elastic* cloud model the cost function in the case of offloading can be expressed as

$$C_{i,a}^c(\mathbf{d}) = (\gamma_i^T + \gamma_i^E P_i) D_i \frac{n_a(\mathbf{d})}{R_{i,a}} + \gamma_i^T \frac{L_i}{F^c}. \tag{13}$$

Before we formulate the theorem, let us recall the definition of a generalized ordinal potential from [14].

Definition 4. A function $\Phi : \times \mathfrak{D}_i \to \mathbb{R}$ is a generalized ordinal potential function for the strategic game $< \mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i >$ 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 3. *The MCOG with elastic cloud admits the generalized ordinal potential function*

$$\Phi(\boldsymbol{d}) = \sum_{a'=1}^{A} \sum_{n=1}^{n_{a'}(\boldsymbol{d})} log(n) - \sum_{a'=1}^{A} \sum_{i'=1}^{N} log(M_{i'} R_{i',a'}) I(d_{i'}, a'), \tag{14}$$

and hence it possesses a pure strategy Nash equilibrium.

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 (13), the condition $C_i(a, d_{-i}) < C_i(0, d_{-i})$ implies that

$$(\gamma_i^T + \gamma_i^E P_i) D_i \frac{n_a(a, d_{-i})}{R_{i,a}} + \gamma_i^T \frac{L_i}{F^c} < (\frac{\gamma_i^T}{F_i^0} + \gamma_i^E v_i) L_i.$$

After algebraic manipulations we obtain

$$\frac{n_a(a, d_{-i})}{R_{i,a}} < M_i \triangleq \frac{\gamma_i^E v_i + \gamma_i^T (1/F_i^0 - 1/F^c)}{\gamma_i^T + \gamma_i^E P_i} \cdot \frac{L_i}{D_i}. \tag{15}$$

Since $n_a(a, d_{-i}) > 0$ and $M_i R_{i,a} > 0$, (15) implies that

$$\log(n_a(a, d_{-i})) < \log(M_i R_{i,a}). \tag{16}$$

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

$$\Phi(a, d_{-i}) = \sum_{n=1}^{n_a(a,d_{-i})} \log(n) + \sum_{a' \neq a} \sum_{n=1}^{n_{a'}(a,d_{-i})} \log(n)$$

$$- \log(M_i R_{i,a}) - \sum_{a'=1}^{A} \sum_{i' \neq i} \log(M_{i'} R_{i',a'}) I(d_{i'}, a'),$$

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

$$\Phi(0, d_{-i}) = \sum_{n=1}^{n_a(0,d_{-i})} \log(n) + \sum_{a' \neq a} \sum_{n=1}^{n_{a'}(0,d_{-i})} \log(n)$$

$$- \sum_{a'=1}^{A} \sum_{i' \neq i} \log(M_{i'} R_{i',a'}) I(d_{i'}, a').$$

Since $n_a(a, d_{-i}) = n_a(0, d_{-i}) + 1$, we obtain $\Phi(a, d_{-i}) - \Phi(0, d_{-i}) = \log(n_a(a, d_{-i})) - \log(M_i R_{i,a})$. It follows from (16) 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 (13), the condition $C_i(a, d_i) < C_i(b, d_i)$ implies that

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

which is equivalent to

$$\frac{n_a(a,d_{-i})}{n_b(b,d_{-i})} < \frac{R_{i,a}}{R_{i,b}}. \tag{17}$$

Since $n_a(a,d_{-i}), n_b(b,d_{-i}) > 0$, and $R_{i,a}, R_{i,b} > 0$ (17) implies that

$$\log(n_a(a,d_{-i})) - \log(n_b(b,d_{-i})) < \log(R_{i,a}) - \log(R_{i,b}). \tag{18}$$

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

$$\Phi(a,d_{-i}) = \sum_{n=1}^{n_a(a,d_{-i})} \log(n) + \sum_{n=1}^{n_b(a,d_{-i})} \log(n) + \sum_{a' \neq a,b} \sum_{n=1}^{n_{a'}(a,d_{-i})} \log(n)$$
$$- \log(M_i R_{i,a}) - \sum_{a'=1}^{A} \sum_{i' \neq i} \log(M_{i'} R_{i',a'}) I(d_{i'}, a').$$

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 $\Phi(a,d_{-i}) - \Phi(b,d_{-i}) = \log(n_a(a,d_{-i})) - \log(n_b(b,d_{-i})) - (\log(R_{i,a}) - \log(R_{i,b}))$. It follows from (18) that $\Phi(a,d_{-i}) - \Phi(b,d_{-i}) < 0$, 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 [14]. Therefore, the existence of a generalized ordinal potential function allows us to use the *ImprovementPath* Algorithm (c.f., Figure 2) 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.*

5 Price of Anarchy

We have so far shown that NE exist and provided low complexity algorithms for computing a NE. 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

$$PoA = \frac{\max_{\mathbf{d}^*} \sum_{i \in \mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d} \in \mathfrak{D}} \sum_{i \in \mathcal{N}} C_i(\mathbf{d})}. \tag{19}$$

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

Theorem 4. *The price of anarchy for the computation offloading game is upper bounded by*

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

Proof. First we show that if there is a NE in which all MUs 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 $\sum_{i\in\mathcal{N}} C_i(d_i^*, d_{-i}^*) \leq \sum_{i\in\mathcal{N}} C_i^0$, and if all MUs 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 $d_i = a$ for some $a \in \mathcal{A}$, 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, $\omega_{i,a}(d_i, d_{-i}) \leq R_{i,a}$ and $F_i^c(n(d_i, d_{-i})) \leq F^c$, which implies that

$$C_{i,a}^c(d_i, d_{-i}) = (\gamma_i^T + \gamma_i^E P_i)\frac{D_i}{\omega_{i,a}(d_i, d_{-i})} + \gamma_i^T \frac{L_i}{F_i^c(n(d_i, d_{-i}))} \geq (\gamma_i^T + \gamma_i^E P_i)\frac{D_i}{R_{i,a}} + \gamma_i^T \frac{L_i}{F^c} = C_{i,a}^{\bar{c}}.$$

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

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

which proves the theorem. □

6 Numerical Results

We use extensive simulations to evaluate the cost performance and the computational time of the *JPBR* algorithm. We consider that the APs and MUs are placed over a $1km \times 1km$ region. The APs are located at grid points in the region, while the MUs are placed uniformly at random. We consider that the channel gain of MU i to AP a is 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 [15]. The channel bandwidth B_a of every AP a was set to $\mu = 5$ *MHz*, while the data transmit power P_i of every MU i was set to to $0.4W$ according to [16]. The computational capability F_i^0 of MU i was drawn from a continuous uniform distribution with parameters $[0.5, 1]$ *Gcycles*, while the computation capability of the cloud F^c was set to 100 *Gcycles* [17]. Unless otherwise noted, 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 was set to $10^{-11}(F_i^0)^2$ according to measurements reported in [4, 18]. The weights attributed to energy consumption γ_i^E and the response time γ_i^T were drawn from a continuous uniform distribution on $[0, 1]$.

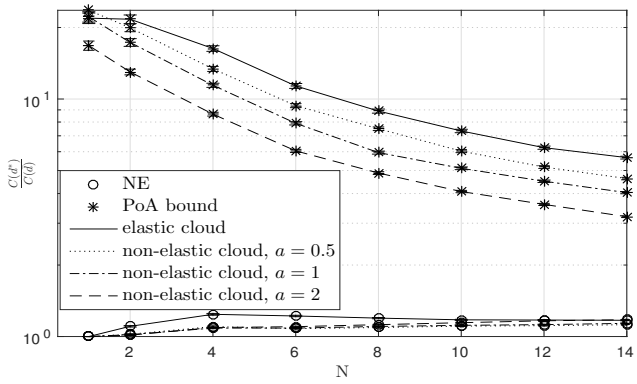

Figure 6: The *cost ratio* and the upper bound on the PoA for the *elastic* and *non-elastic* cloud ($a = 0.5, 1, 2$), $A = 3$ APs. The results shown are the averages of 600 simulations, together with 95% confidence intervals.

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 use the system cost that can be achieved if all MUs execute their computation tasks locally, which coincides with the bound on the PoA.

6.1 Optimal vs. Equilibrium Cost

Figure 6 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 cloud computational capability assigned to a MU that offloads is a reciprocal function of the number of MUs that offload, i.e. $F_i^c(\mathbf{d}) = \frac{F^c}{an(\mathbf{d})}$. We refer to this latter case as a *non-elastic* cloud and to the coefficient a as the *cloud provisioning coefficient*. A coefficient of $a = 1$ corresponds to a cloud with fixed amount of resources, $a < 1$ to resources that scale slower than the demand, while $a > 1$ 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* ($a = 0.5, 1$ and 2).

The results in Figure 6 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

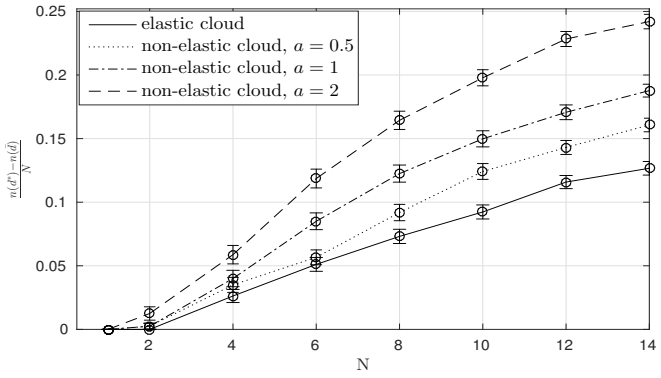

Figure 7: *Offloading difference ratio* vs. number of MUs N for the *elastic* and *non-elastic* cloud ($a = 0.5, 1, 2$), $A = 3$ APs. The results shown are the averages of 600 simulations, together with 95% confidence intervals.

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. We can also observe that the upper bound on the PoA decreases as a increases, and thus the problem becomes computationally easier for larger values of a.

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 7 for the same set of parameters as in Figure 6. The results show that the *offloading difference ratio* increases with the number of MUs, which explains the increased *cost ratio* observed in Figure 6, 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 [19]. 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 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=0.45 *Gcycles*.

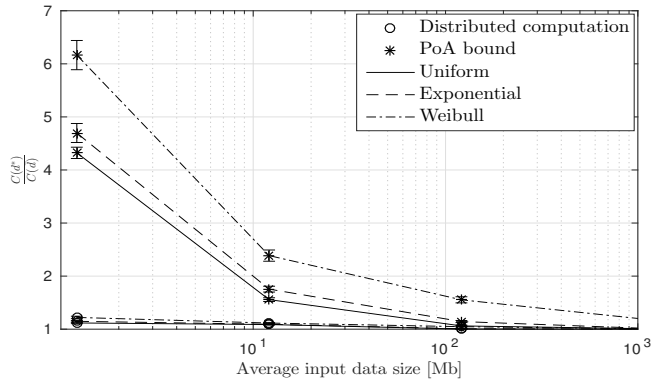

Figure 8: The *cost ratio* and the upper bound on the PoA for the *elastic* and *non-elastic* cloud ($a = 1$), uniform, exponential and Weibull distributions of the input data sizes, $A = 3$ APs, $N = 12$ MUs. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

Figure 8 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 (a=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 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.

6.3 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 ($a = 1$), $A = 10$ and $A = 100$ APs. Figure 9 shows the number of iterations as a function of the number of MUs for two orderings of adding MUs: in the first case the MUs are added in random order, while in the second case the MUs are added in increasing order of their ratio $\frac{D_i}{C_i^0 L_i}$. We refer to the latter as the *least reluctance first* (*LRF*) order. Intuitively, one would expect that the *LRF* order results in a smaller number of iterations, since the MUs with lower $\frac{D_i}{C_i^0 L_i}$ ratio have lower computational capability to execute computationally more demanding tasks with smaller offloading data size than the MUs with higher $\frac{D_i}{C_i^0 L_i}$. However, the simulation results show that the number of iterations is

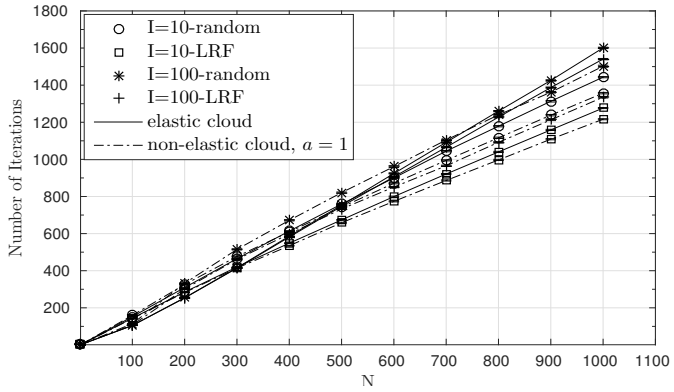

Figure 9: Number of iterations vs. number of MUs N for the *elastic* and *non-elastic* cloud ($a = 1$), A =10 and 100 APs. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

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 an efficient decentralized algorithm for coordinating computation offloading among autonomous MUs.

7 Related Work

Most previous works considered the problem of energy efficient computation offloading for a single mobile user [3,4,6,20,21], and thus they do not consider the allocation of resources between mobile users.

Some recent works considered the problem of energy efficient computation offloading for multiple mobile users [8,9,22]. [8] studied the partitioning problem for mobile data stream applications, and proposed a genetic algorithm as a heuristic for solving the optimization problem that maximizes throughput. [22] considered a two-tiered cloud infrastructure model under user mobility in a location-time workflow framework, and proposed a heuristic for minimizing the users' cost. [9] provided an iterative algorithm for solving the optimization problem that minimizes the mobile users' energy consumption by joint allocation of wireless and cloud resources.

A few recent works provided a game theoretic treatment of computation offloading problem [7, 23–27]. [23] considers a two-stage problem, where first each mobile user decides which parts of a task to offload so as to minimize its energy consumption and to

meet its service response deadline, and then the cloud allocates computational resources to the offloaded tasks. [24] considered a three-tier cloud architecture and stochastic task arrivals, and provided a distributed algorithm for the computation of an equilibrium. [26] considered tasks that arrive at the same time, a single wireless link, and elastic cloud, and showed the existence of equilibria when all mobile users have the same delay budget. Our work differs from [23] in that we consider that the allocation of cloud resources is known to the mobile users, from [24] in that we consider contention in the wireless access, and from [26] in that we consider multiple wireless links and a non-elastic cloud.

Most related to our work are [7,25,27]. [7] considered a single wireless link and an elastic cloud, assumed upload rates to be determined by the Shannon capacity of an interference channel, and showed that the game is a potential game. [25] 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. Unlike these works, we consider time-fair bandwidth sharing and the case of a non-elastic cloud. [27] considered multiple wireless links, fair bandwidth sharing and a non-elastic cloud, and claims the game to have an exact potential.

The importance of our contribution from a game theoretical perspective is that the computation offloading game with non-elastic cloud is a player-specific congestion game for which the existence of equilibria is not known in general [28], thus the JPBR algorithm and our proof of equilibrium existence advance the state of the art in the study of equilibria in general congestion games.

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," Sep. 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*, April 2013, pp. 1285–1293.

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, 2015.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124 – 143, 1996.

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

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

[17] 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.

[18] 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.

[19] G. Hardin, "The tragedy of the commons," *Science*, vol. 162, no. 3859, pp. 1243–1248, 1968.

[20] 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.

[21] 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.

[22] 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.

[23] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *SOSE, 2013 IEEE 7th Int. Symp. on*, Mar. 2013, pp. 494–502.

[24] 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.

[25] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, to appear.

[26] 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.

[27] 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.

[28] I. Milchtaich, "Congestion games with player-specific payoff functions," *Games and Economic Behavior*, vol. 13, no. 1, pp. 111 – 124, 1996.

Paper C

Decentralized Scheduling for Offloading of Periodic Tasks in Mobile Edge Computing

Slađana Jošilo and György Dán

in Proc. of IFIP Networking (NETWORKING), 2018.

Decentralized Scheduling for Offloading of Periodic Tasks in Mobile Edge Computing

Slađana Jošilo and György Dán

School of Electrical Engineering,
KTH Royal Institute of Technology, Stockholm, Sweden
E-mail: {josilo,gyuri}@kth.se.

Abstract

Motivated by various surveillance applications, we consider wireless devices that periodically generate computationally intensive tasks. The devices aim at maximizing their performance by choosing when to perform the computations and whether or not to offload their computations to a cloud resource via one of multiple wireless access points. We propose a game theoretic model of the problem, give insight into the structure of equilibrium allocations and provide an efficient algorithm for computing pure strategy Nash equilibria. Extensive simulation results show that the performance in equilibrium is significantly better than in a system without coordination of the timing of the tasks' execution, and the proposed algorithm has an average computational complexity that is linear in the number of devices.

1 Introduction

Mobile edge computing (MEC) is considered to become an enabler of a variety of Internet of Things (IoT) applications that are based on a pervasive deployment of wireless sensors. Examples range from water pipeline surveillance [1], through pursuit problems and discrete manufacturing [2] to body area networks [3]. Many of these applications involve the periodic collection of sensory data, which need to be processed timely to enable control decisions. Processing often requires some form of data analytics, e.g., visual analysis, which is computationally demanding.

The key advantage of MEC compared to centralized cloud infrastructures is that computational resources are located close to the network edge [4]. Thus, even though MEC infrastructures may be less resource-rich than centralized clouds, such as Microsoft Azure or AWS, due to their proximity to the sensors they may be able to provide response times that make them suitable for computation offloading for real-time applications.

The proximity of MEC resources makes low response times for individual sensors possible, but when multiple wireless sensors attempt to offload to the MEC simultaneously, the

response times might increase due to contention for the communication and the computational resources [5–7]. Coordination is thus essential for maintaining low response times in the case of MEC computation offloading.

Coordination for offloading periodic tasks involves deciding whether or not to offload the computations, deciding which of the available wireless communication channels to use for offloading, and in the case of periodic tasks, it involves deciding when to collect sensory data and when to offload the computation. In addition coordination should respect that sensors may be managed by different entities, with individual interests. The resulting coordination problem not only has a huge solution space with a combinatorial structure, but it also requires consideration of the potentially diverse requirements of the sensors in terms of response time and energy consumption for performing the computation. Efficient coordination of computation offloading for wireless sensors with periodic tasks is thus a complex problem.

In this paper we address this problem by considering the allocation of cloud and wireless resources among wireless devices that generate tasks periodically. The devices can choose the time slot in which to perform their periodic task, and can decide whether to offload their computation to a cloud through one of many access points or to perform the computation locally. We provide a game theoretical treatment of the problem, and prove the existence of pure strategy Nash equilibria. Our proof provides a characterization of the structure of the equilibria, and serves as an efficient decentralized algorithm for coordinating the offloading decisions of the wireless devices. We use extensive simulations to assess the benefits of coordinated computation offloading compared to uncoordinated computation offloading where devices choose a time slot at random, and in the chosen time slot play an equilibrium allocation. Our results show that the proposed algorithm computes equilibria with good system performance in a variety of scenarios in terms of task periodicity, the number of devices and the number of access points.

The rest of the paper is organized as follows. In Section 2 we present the system model and the problem formulation. In Section 3 we present algorithmic and analytical results. In Section 4 we show numerical results and in Section 5 we discuss related work. Section 6 concludes the paper.

2 System Model and Problem Formulation

We consider a computation offloading system that consists of N devices, A acces points (APs) and a cloud service. We denote by $\mathcal{N}=\{1,2,...,N\}$ and $\mathcal{A}=\{1,2,...,A\}$ the set of devices and the set of APs, respectively. Each device generates a computationally intensive task periodically every T time units. Device i's task is characterized 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. We make the reasonable assumption that the number X of CPU cycles required per bit can be modeled by a random variable following a Gamma distribution [8,9], and assume $E[X]$ to be known from previous measurements. Thus, assuming independence the mean number of CPU cycles can be expressed as $L_i = D_i E[X]$.

We consider that time is partitioned into T time slots, and we denote by $\mathcal{T}=\{1,2,...,T\}$ the set of time slots. Each device can choose one time slot in which it wants to perform the computation and in the chosen time slot it can decide whether to perform the computation locally or to offload the computation to the cloud server through one of the APs. Therefore, each device $i \in \mathcal{N}$ can choose one element of the set $\mathfrak{D}_i = \{\mathcal{A} \cup \{0\}\} \times \mathcal{T}$, where 0 corresponds to local computing. We denote by $d_i \in \mathfrak{D}_i$ the decision of MU 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. The considered model of homogeneous task periodicities is reasonable for surveillance of homogeneous physical phenomena, we leave the case of heterogeneous periodicities to be subject of future work.

For a strategy profile $\mathbf{d}$ 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 $O_t(\mathbf{d}) = \cup_{a\in\mathcal{A}} O_{(t,a)}(\mathbf{d})$, and the total number of devices that offload in time slot t as $n_t(\mathbf{d})=\sum_{a\in\mathcal{A}} n_{(t,a)}(\mathbf{d})$. Finally, we denote by $O(\mathbf{d}) = \cup_{t\in\mathcal{T}} O_t(\mathbf{d})$ the set of all devices that offload in strategy profile $\mathbf{d}$.

2.1 Local computing

In the case of local computing each device has to use its own computing resources in order to perform the computation. We consider that different devices may have different computational capabilities and we denote by F_i^0 the computational capability of device i. Furthermore, we consider that the computational capability F_i^0 of device i is independent of the chosen time slot, and hence the time that is needed for device i to perform its computation task that requires L_i CPU cycles can be expressed as

$$T_i^0 = \frac{L_i}{F_i^0}. \tag{1}$$

In order to express the energy consumption in the case of local computing we denote by v_i the energy consumption per CPU cycle [10], and we express the energy that device i would spend on performing a computation task that requires L_i CPU cycles as

$$E_i^0 = v_i L_i. \tag{2}$$

2.2 Computation offloading

In the case of computation offloading the computation is performed in the cloud, but the input data for the computation task need to be transmitted through one of the APs. In what follows we introduce our communication and computation models that describe how the wireless medium and the cloud computing resources are shared among devices that offload their tasks, respectively.

2.2.1 Communication model

We consider that the uplink rate $\omega_{i,(t,a)}(\mathbf{d})$ that device i can achieve if it offloads through AP a in time slot t 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. Furthermore, we consider that each device is characterized by PHY rate $R_{i,a}$, which depends on device specific parameters such as physical layer signal characteristics and the channel conditions. Therefore, the uplink rate of device i on AP a can be different from the uplink rates of the other devices on the same AP and can be expressed as

$$\omega_{i,(t,a)}(\mathbf{d}) = R_{i,a} \times f_a(n_{(t,a)}(\mathbf{d})). \tag{3}$$

This communication model can be used to model throughput sharing mechanisms in TDMA and OFDMA based MAC protocols [11].

Given the uplink rate $\omega_{i,(t,a)}(\mathbf{d})$, the time needed for device i to transmit the input data of size D_i through AP a in time slot t can be expressed as

$$T^{tx}_{i,(t,a)}(\mathbf{d}) = \frac{D_i}{\omega_{i,(t,a)}(\mathbf{d})}. \tag{4}$$

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 [12, 13]. 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

$$E^{tx}_{i,(t,a)}(\mathbf{d}) = P_{i,a} T^{tx}_{i,(t,a)}(\mathbf{d}). \tag{5}$$

2.2.2 Computation model

We denote by F^c the computational capability of the cloud service, and we consider that the computational capability $F^c_{i,t}(\mathbf{d})$ that device i would receive from the cloud in time slot t is a non-increasing function $f_i(n_t(\mathbf{d}))$ of the total number $n_t(\mathbf{d})$ of devices that offload in time slot t

$$F^c_{i,t}(\mathbf{d}) = F^c \times f_i(n_t(\mathbf{d})). \tag{6}$$

Therefore, the time needed for performing device i's task in the cloud may be different in different time slots, and given the number L_i of CPU cycles needed for the computation task it can be expressed as

$$T^{exe}_{i,t}(\mathbf{d}) = \frac{L_i}{F^c_{i,t}(\mathbf{d})}. \tag{7}$$

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. Figure 1 shows an example of a mobile cloud computing system where devices can choose one slot out of four time slots to perform the computation. In the case of computation offloading, each device in the chosen time slot

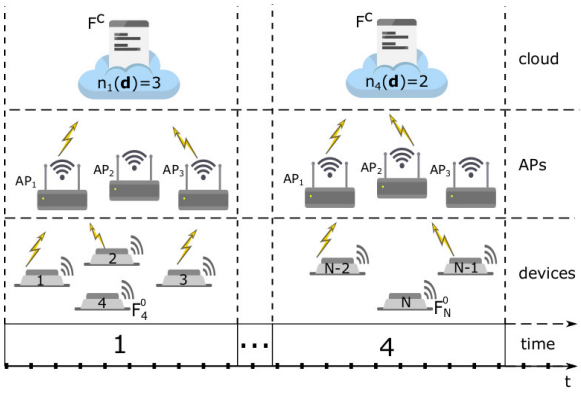

Figure 1: An example of a mobile cloud computing system than consists of N devices, $T = 4$ time slots, and $A = 3$ APs.

can offload its task to the cloud through one of three APs, e.g., in time slot 1 devices 1 and 2 offload their tasks through AP 1, device 3 offloads its task through AP 3, and device 4 performs the computation locally.

2.3 Cost Model

We consider that devices are interested in minimizing a linear combination of their computing time and their energy consumption, and denote by $0 \leq \gamma_i^T, \gamma_i^E \leq 1$ the corresponding weights, respectively. We can then express the cost of device i in the case of local computation as

$$C_i^0 = \gamma_i^T T_i^0 + \gamma_i^E E_i^0. \tag{8}$$

Similarly, we can express the cost of device i in the case of offloading through AP a in time slot t as

$$C_{i,(t,a)}^c(\mathbf{d}) = \gamma_i^T (T_{i,t}^{exe}(\mathbf{d}) + T_{i,(t,a)}^{tx}(\mathbf{d})) + \gamma_i^E E_{i,(t,a)}^{tx}(\mathbf{d}). \tag{9}$$

In (9) we made the common assumption that the time needed to transmit the result of the computation from the cloud service to the device can be neglected [5, 7, 14, 15], 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. We can thus express the cost of device i in strategy profile $\mathbf{d}$ as

$$C_i(\mathbf{d}) = \sum_{d_i \in \mathcal{T} \times \{0\}} \mathbf{1}_{(t,0)}(d_i) \cdot C_i^0 + \sum_{d_i \in \mathcal{T} \times \mathcal{A}} \mathbf{1}_{(t,a)}(d_i) \cdot C_{i,(t,a)}^c(\mathbf{d}), \tag{10}$$

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.

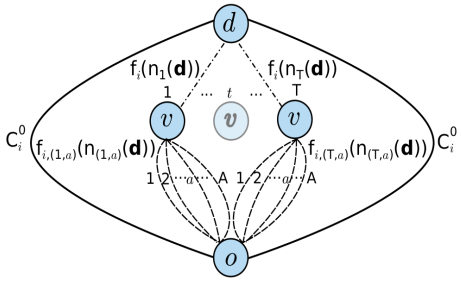

Figure 2: Network model of the *multi-slot computation offloading game* (MSCOG).

2.4 Multi-slot computation offloading game

We consider that the objective of each device is to minimize its own total cost (10), i.e., to find a strategy

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

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. Since devices may be autonomous entities with individual interests, 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 MSCOG. The MSCOG is a player specific network congestion game, as illustrated in Figure 2.

Our objective is to answer the fundamental question whether there is a strategy profile from which no device would want to deviate, i.e., a pure strategy Nash equilibrium.

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,

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

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 no device can perform a best improvement step in a NE.

3 Computing Equilibria

3.1 Single time slot ($T = 1$)

We start with considering the case $T = 1$, i.e., a single time slot.

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 [16]. Observe that if for $T = 1$ we contract the edge (v,d) in the network shown in Figure 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. Let us define the local computation cost of player i in $\tilde{\Gamma}$ as $\tilde{C^0}_i(N-n_1(\mathbf{d})) = C_i^0 - f_i(1+n_1(\mathbf{d}))+c$, and the cost of offloading through AP a as $\tilde{f}_{i,a}(n_{(1,a)}(\mathbf{d})) = f_{i,a}(n_{(1,a)}(\mathbf{d}))+c$, 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 $\tilde{C^c}_{i,a} = C^c_{i,a} - f_i(n_1(\mathbf{d}))$, 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 $\boldsymbol{d}^*$ 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 [17]. □

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.

3.2 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) $\min_{t' \in \mathcal{T}} n_{t'}(\boldsymbol{d}^*) \leq n_t(\boldsymbol{d}^*) \leq \min_{t' \in \mathcal{T}} n_{t'}(\boldsymbol{d}^*) + 1$ *for* $\forall t,t' \in \mathcal{T}$,
(ii) if $n_t(\boldsymbol{d}^*) = n_{t'}(\boldsymbol{d}^*) + 1$ *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* $n_{t'}(\boldsymbol{d}^*) \leq n_t(\boldsymbol{d}^*) \leq n_{t'}(\boldsymbol{d}^*) + 1$.

Proof. Clearly, all statements hold for $T = 1$. Assume that $T > 1$ and $\exists t,t' \in \mathcal{T}$ such that $n_t(\mathbf{d}^*) > n_{t'}(\mathbf{d}^*) + 1$. 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 $n_t(\mathbf{d}^*) = n_{t'}(\mathbf{d}^*) + 1$, 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 $n_t(\mathbf{d}^*) < n_{t'}(\mathbf{d}^*)$. 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 $n_t(\mathbf{d}^*) \geq n_{t'}(\mathbf{d}^*)$. Second, assume that $n_t(\mathbf{d}^*) > n_{t'}(\mathbf{d}^*) + 1$ 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 $n_t(\mathbf{d}^*) \leq n_{t'}(\mathbf{d}^*) + 1$ must hold. □

In what follows we prove our main result concerning the existence of an equilibria in 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.

3.3 The *MyopicBest* (MB) Algorithm

We prove Theorem 3 using the MB algorithm, shown in Figure 3. The MB algorithm adds players one at a time, and lets them play their best replies given the other players' strategies. Our proof is thus 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 for $N \leq T$.*

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, and the MB algorithm computes such a strategy profile. □

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 $d_i^* = (t,0)$, 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 $j \in O_t(\mathbf{d}) \setminus O_{(t,a)}(\mathbf{d})$ may have an incentive to make an improvement step because their cloud computing cost has increased. Among these players, the MB algorithm allows players $j \in O_{(t,a)}(\mathbf{d})$ to perform best improvement steps, using the *DoublePokeDeviator* (DPD) algorithm shown in Figure 4. 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

$\mathbf{d}^* = MB(\mathcal{N}, \mathcal{TA}, F^c, F_i^0)$

1: Let $N \leftarrow 1$
2: **for** $N = 1 \ldots |\mathcal{N}|$ **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** $\exists j \in O_t(\mathbf{d}'), \exists d_j \in \mathfrak{D}_j$ s.t. $C_j(d_j, d'_{-j}) < C_j(d'_j, d'_{-j})$ **then**
11: /*Players $j \in O_t(\mathbf{d}')$ 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 3: Pseudo code of the *MB* algorithm.

in $\mathbf{d}$ is not smaller than the number of offloaders in the NE $\mathbf{d}^*(N-1)$. The DPD algorithm allows either one player of the first type, or one player of the second type to perform a best improvement step, and as we show next it terminates in a finite number of steps.

Proposition 1. *Let $\mathbf{d}$ be a strategy profile in which there is at least one player $j \in O_{(t,a)}(\mathbf{d})$*

$(\mathbf{d}, t, A') = DPD(\mathbf{d}, \mathbf{d}^*(N-1), (t,a), A')$

/*Players that want to stop to offload*/
$D'_1 = \{j | d_j = (t,a), (t,0) = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d_{-j})\}$
/*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 $i \in D'_1$
    $d_i = (t,0)$
  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 4: Pseudo code of the *DPD* algorithm.

that can be chosen by the DPD algorithm. Then the length of a best improvement path generated by the DPD algorithm is at most $N-1$.

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 DPD algorithm terminates since it allows only players that play the same strategy as the last deviator to perform best improvement steps. Furthermore, if $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$, then $n_{(t,a)}(\mathbf{d}') = n_{(t,a)}(\mathbf{d}^*(N-1))$ for every $(t,a) \in \mathcal{T} \times \mathcal{A}$, and thus $\mathbf{d}'$ is a NE.

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 DPD algorithm terminates, because there is no player that plays the same strategy as the last deviator and that can decrease its cost using the DPD algorithm. 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 best improvement path generated by the DPD algorithm, and thus the algorithm terminates in at most $N-1$ best improvement steps, which proves the proposition. □

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 proven that the DPD algorithm generates a finite number of finite best improvement paths. In the following we use this result for proving the convergence of the MB algorithm.

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 1 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 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 $j \in O_{t'}(\mathbf{d}')$ 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 $j \in O_{t'}(\mathbf{d}')$ 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 $n_t(\mathbf{d}) = n_t(\mathbf{d}') + 1$ 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 $j \in O_{t'}(\mathbf{d}')$ that offload in the same time slot as the last deviator may want to change their offloading strategy to (t,a). Furthermore, players $j \in O_{t'}(\mathbf{d}') \setminus O_{(t',b)}(\mathbf{d}')$ 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 $j \in O_{t'}(\mathbf{d}')$ 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 $n_t(\mathbf{d}) = n_t(\mathbf{d}') + 1$ 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.

If case (ii) or case (iii) happens and there is no player $j \in O_{t'}(\mathbf{d}')$ that wants to deviate, the MB algorithm allows players from the other time slots $t \in \mathcal{T} \setminus \{t'\}$ to perform best

$(\mathbf{d},(t,a),A') = SID(\mathbf{d},A')$

/*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 5: Pseudo code of the *SID* algorithm.

improvement steps using *SelfImposedDeviator* (SID) algorithm shown in Figure 5. 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 $j \in O(\mathbf{d}') \setminus O_{t'}(\mathbf{d}')$ 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 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 $j \in O(\mathbf{d}') \setminus O_{t'}(\mathbf{d}') \cup L(\mathbf{d}')$ 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 $n_t(\mathbf{d}) = n_t(\mathbf{d}') + 1$ 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. Finally, let us consider case (iii) such that there is a player $j \in O_{t'}(\mathbf{d}') \setminus O_{(t',b)}(\mathbf{d}')$ 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 addding 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.*

Implementation considerations: 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 MEC resources, and the number of users sharing the APs and the cloud. In practice these information can be provided by the MEC. As discussed in [5,7,18], two main advantages of such a decentralized implementation compared to a centralized one are that the MEC can be relieved from complex centralized management, and devices do not need to reveal their parameters, but only their most recent decisions.

4 Numerical Results

In the following we show simulation results to evaluate the cost performance and the computational efficiency of the MB algorithm. 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 [19]. For simplicity we assign a bandwidth of 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 [20]. 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. The computational capability F_i^0 of device i is drawn from a continuous uniform distribution on $[0.5, 1]$ GHz, while the computation capability of the cloud is $F^c = 100$ GHz [21]. We consider 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 [9, 10]. 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 three algorithms as a basis for comparison for the proposed *MB* algorithm. In the first algorithm players 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 $T = 1$. 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. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

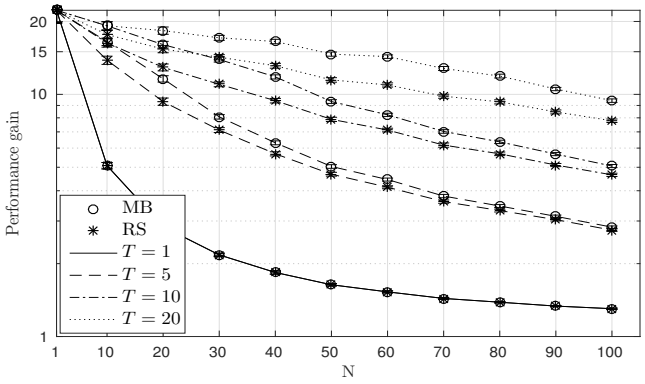

Figure 6: *Performance gain* vs. number of devices (N).

4.1 Performance gain vs number of devices

Figure 6 shows the *performance gain* as a function of the number N of devices for $A = 4$ APs. The results show that the *performance gain* decreases with the number of devices for the MB algorithm for all values of T, for the RS algorithm and for the deterministic worst case $T = 1$. 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 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.

Figure 7 shows the ratio of players that offload for the same set of parameters as in Figure 6. 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 Figure 6. 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 Figure 6.

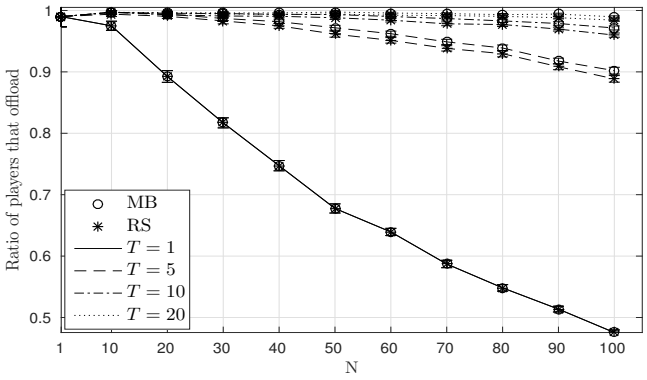

Figure 7: Ratio of offloaders vs. number of devices (N).

4.2 Performance gain vs number of APs

Figure 8 shows the *performance gain* as a function of the number A of APs for $N = 50$ devices. 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 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.

4.3 Computational Complexity

In order to assess the computational efficiency of the MB algorithm 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 NE. Figure 9 shows the number of iterations as a function of the number N of devices for $A = 4$ APs. 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 Figure 9 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. The reason is that in the case of the MB algorithm

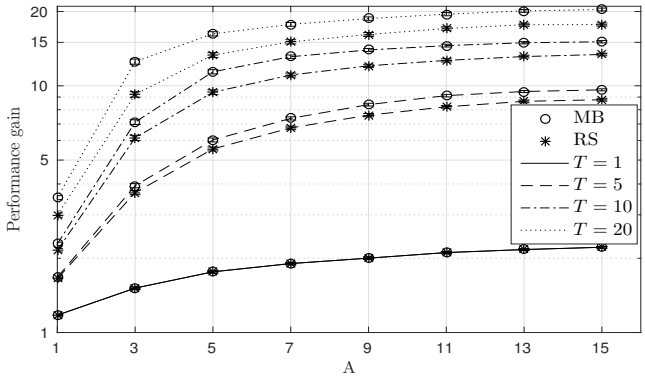

Figure 8: *Performance gain* vs. number of APs (A).

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 Figure 9 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 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.

Overall, our results show that the proposed MB algorithm can compute efficient allocations for periodic task offloading at low computational complexity.

5 Related Work

The scheduling of periodic tasks received significant attention for real-time systems [22, 23], but without considering communications. Similarily, the scheduling of communication resources has been considered without considering computation [24]. Most works that considered both communication and computation considered a single device [6, 10, 25–27], and thus they do not consider the allocation of resources between devices.

Related to our work are recent works on energy efficient computation offloading for multiple mobile users [28–30]. [28] proposed a genetic algorithm for maximizing the throughput in a partitioning problem for mobile data stream applications, while [29] proposed a heuristic for minimizing the users' cost in a two-tiered cloud infrastructure with user

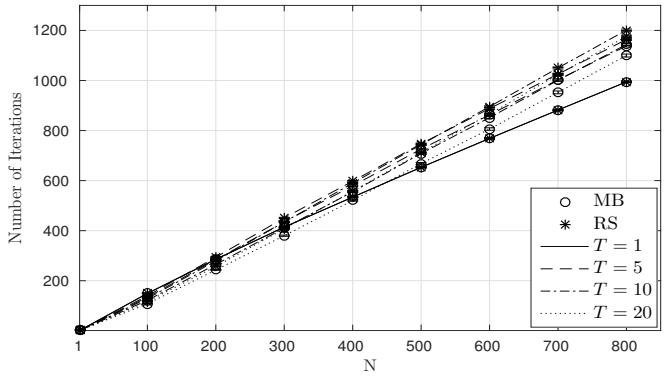

Figure 9: Number of iterations vs. number of devices (N).

mobility in a location-time workflow framework. [30] 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 offloading problem for a single time slot [5, 7, 18, 31–34]. [31] considers a two-stage game, where first each mobile user chooses the parts of its task to offload, and then the cloud allocates computational resources to the offloaded parts. [32] considered a three-tier cloud architecture, and provided a distributed algorithm for the computing a mixed strategy equilibrium. [33] 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. [5] 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. [18] 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. [7] considered multiple wireless links, equal bandwidth sharing and a non-elastic cloud, and provided a polynomical time algorithm for computing equilibria. Compared to these works, our model of periodic tasks considers the scheduling of tasks over time slots and wireless resources, and is thus a first step towards bridging the gap between early works on scheduling [23] and recent works on computation offloading [5, 7].

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 [16], thus the proposed algorithm and our proof of existence advance the state of the art in the study of equilibria in network congestion games.

6 Conclusion

We provided a game theoretic treatment of computation offloading for periodic tasks. We proved the existence of equilibrium allocations, characterized their structure and provided a polynomial time decentralized algorithm for computing equilibria. Simulations show that the proposed algorithm achieves good system performance for a wide range of system sizes and task periodicities. Our results show that periodic computation offloading can be efficiently coordinated using low complexity algorithms despite the vast solution space and the combinatorial nature of the problem. 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] S. Oh, P. Chen, M. Manzo, and S. Sastry, "Instrumenting wireless sensor networks for real-time surveillance," in *Proc. IEEE ICRA*, May 2006, pp. 3128–3133.

[3] X. Zhu, S. Han, P. C. Huang, A. K. Mok, and D. Chen, "Mbstar: A real-time communication protocol for wireless body area networks," in *Proc. of ERCTS*, Jul. 2011, pp. 57–66.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," Sep. 2015.

[5] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *Proc. of IEEE PDS*.

[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*, April 2013, pp. 1285–1293.

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

[8] 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.

[9] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of Usenix HotCloud*, 2010.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] I. Milchtaich, "The equilibrium existence problem in finite network congestion games," in *Proc. of WINE*, 2006, pp. 87–98.

[17] ——, "Congestion games with player-specific payoff functions," *Games and Economic Behavior*, vol. 13, no. 1, pp. 111 – 124, 1996.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] V. Cardellini et al., "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, 2015.

[33] 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.

[34] 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.