An Edge Network Orchestrator for Mobile Augmented Reality

Qiang Liu, Siqi Huang, Johnson Opadere, Tao Han
Electrical and Computer Engineering Department,
The University of North Carolina at Charlotte, NC, United States
Email: {qliu12, shuang9, jopadere, tao.han}@uncc.edu

***Abstract*—Mobile augmented reality (MAR) involves high complexity computation which cannot be performed efficiently on resource limited mobile devices. The performance of MAR would be significantly improved by offloading the computation tasks to servers deployed with the close proximity to the users. In this paper, we design an edge network orchestrator to enable fast and accurate object analytics at the network edge for MAR. The measurement-based analytical models are built to characterize the tradeoff between the service latency and analytics accuracy in edge-based MAR systems. As a key component of the edge network orchestrator, a server assignment and frame resolution selection algorithm named FACT is proposed to mitigate the latency-accuracy tradeoff. Through network simulations, we evaluate the performance of the FACT algorithm and show the insights on optimizing the performance of edge-based MAR systems. We have implemented the edge network orchestrator and developed the corresponding communication protocol. Our experiments validate the performance of the proposed edge network orchestrator.**

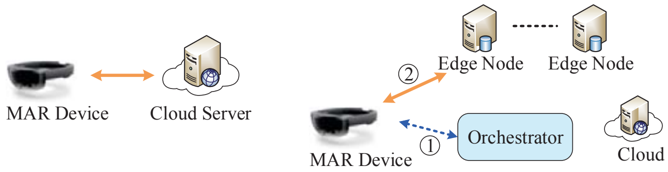
***Index Terms*—Mobile augmented reality; edge network orchestration; Mobile edge computing**

I. Introduction

Mobile augmented reality (MAR) augments a real-world environment by computer-generated sensory information such as text, sound, and graphics. With advanced MAR technologies, the information about a person's surrounding physical environment can be brought out of the digital world and overlaid with the person's perceived real world. MAR will be widely adopted in various industries such as tourism, entertainment, advertisement, education, manufacture and so on [1]. According to Digi-Capital, MAR will become the primary driver of a $108 billion virtual/augmented reality market by 2021 [2].

Since MAR performs in the semantic context of the real-world, the fast and accurate object analytics is the key component for integrating digital information with environment elements in MAR applications [3]. Although the object detection and recognition have been well studied in computer vision researches, the existing solutions are designed to run with powerful CPU/GPU and cannot be applied to MAR directly due to the limited computing resource in mobile devices [4]–[7]. Two research directions emerge for solving this problem. The first direction is tailoring the compute-intensive computer vision algorithms for executing on mobile devices [8], [9]. The state-of-the-art object analytics latency in mobile devices is about 600ms per frame [8] which is still longer than the expected latency of MAR [1].

The other research direction is offloading the compute-intensive object analysis to powerful cloud servers [10]–[13]. The cloud-based MAR solutions significantly reduce the computational latency by exploiting high-end CPU/GPU in cloud servers. However, offloading video frames to cloud may introduce excessive network latency because of the time-varying and capacity-constrained wireless channels. Therefore, reducing the network latency becomes one of the focal points in designing cloud-based MAR systems. The fundamental idea of reducing the network latency is reducing the amount of data required to be transmitted to cloud servers. The data transmission can be reduced by methods such as the frame selection, target area/object selection, and feature extraction [11]–[13].



(a) Cloud-based MAR System (b) Edge-based MAR System

Figure 1: The cloud-based vs. edge-based MAR system.

Mobile edge computing (MEC) is a new networking paradigm in which computing nodes are placed in close proximity to mobile devices for highly responsive cloud services [14], [15]. Leveraging MEC, cloud-based MAR systems can further improve their responsiveness and potentially enable large-scale MAR services. However, it is not trivial to push cloud servers to the edge of the network. Edge servers are usually less powerful than cloud servers. A single edge server may be easily overloaded by MAR services [16]. Therefore, multiple edge servers are necessary to provisioning MAR services for a large number of users. Meanwhile, the distribution of mobile users exhibits highly spatial diversity [17]. Such a user distribution will lead to imbalanced workloads among edge servers, which impairs the performance of MAR in terms of the service latency. Therefore, a well designed network orchestrator, which can dynamically dispatch MAR related computing workloads to edge servers, is needed in an edge-based MAR system as illustrated in Fig. 1.

The major challenge of designing the edge network orchestrator lies in the difficulty of mitigating the tradeoff between the analytics accuracy, network latency, and computational latency. A higher-resolution video frame usually allows a better analytics accuracy at the cost of longer network and computational latency [18]. Meanwhile, the network and computational latency may not be minimized simultaneously because of diverse workloads and heterogeneous edge servers.

978-1-5386-4128-6/18/$31.00 ©2018 IEEE

Since the network latency largely depends on the location of the server, an edge server with the closest proximity to a mobile user can provide the lowest network latency. However, the computational latency is contingent on the computing resources and workloads in the edge server. Therefore, the edge server with lowest network latency may cause higher computational latency because of the limited computing resources.

In this paper, we design and implement an edge network orchestrator which enables fast and accurate object analytics in an edge-based MAR system. In order to address the aforementioned challenge, we model the network latency, computational latency, and analytics accuracy in an edge-based MAR system according to the performance measurements obtained from our MAR testbed. Then, we formulate a multi-objective optimization problem which aims to mitigate the tradeoff between the network latency, computation latency and analytics accuracy by optimizing the edge server assignment and video frame resolution selection for MAR users. We develop a fast and accurate object analytics (FACT) algorithm which solves the multi-object optimization problem based on convex optimization theory. We evaluate the FACT algorithm through both network simulations and experiments with our edge-based MAR system implementation.

To the best of our knowledge, our work is the first to design an edge network orchestrator for edge-based MAR systems. Our contributions are summarized as follows:

- We build analytical models for investigating the tradeoff between the network latency, computational latency and analytics accuracy in an edge-based MAR system.
- We design an edge network orchestration algorithm named FACT which boosts the performance of an edge-based MAR system by optimizing the edge server assignment and video frame resolution selection for MAR users. We evaluate the performance of the FACT algorithm and provide insights on optimizing the edge-based MAR system through network simulations.
- We develop an edge-based MAR system, implement the FACT algorithm as the edge network orchestrator, and design the corresponding MAR communication protocol. We conduct experiments and validate the performance of the edge-based MAR system.

The rest of this paper is organized as follows. Section II briefly reviews the related work. Section III presents the system model and problem formulation. Section IV develops the FACT algorithm. Section V evaluates the performance of the FACT algorithm through network simulations. Section VI details the system implementation and experiments. Section VII concludes the paper.

II. Related Work

Our work relates to cloud-based MAR systems and edge computing load balancing algorithms. The cloud-based MAR system design has attracted many research efforts recently [1]. The main objectives in designing cloud-based MAR systems are improving the object recognition accuracy and reducing the service latency. Chen *et al.* propose a continuous real-time object recognition system which implements an active cache mechanism to improve the recognition accuracy and exploits a trigger frame selection method to reduce the latency caused by wireless networks [11]. Jain *et al.* propose two methods to improve cloud-based MAR systems [3], [12]. The first method is to improve the accuracy and latency of the object recognition by using a location-free geometric to prune down the visual search space [3]. The second method is to reduce the network latency by only transferring the distinctive features of images to the server [12].

On designing cloud-based MAR systems, the existing work does not consider the heterogeneous computing capability and diverse workloads on edge servers. Therefore, no solution is provided to properly dispatch computing workloads among edge servers for MAR services. In addition, the existing work does not allow the adaptation of video frame resolution (the amount of data transferred to servers) based on the performance of edge servers and networks.

There are many studies on the edge cloud load balancing problem. The basic idea of the existing solutions is to mitigate the tradeoff between the computational and network latency [16], [19], [20]. Jia *et al.* propose a task redirection algorithm to balance workloads among edge cloudlets and show that the load balancing scheme can significantly reduce the service response time of edge cloudlets [19]. Tong *et al.* propose a hierarchical architecture for edge clouds and design a heuristic workload dispatch algorithm to minimize the average program execution delay by adaptively placing workloads among different tires of servers [16]. Tan *et al.* propose an online job dispatching and scheduling algorithm to minimize total weighted service response time in edge clouds. [20]. These existing edge cloud load balancing solutions only focus on reducing the service latency. However, the analytics accuracy is as important as the service latency for MAR. Therefore, mitigating the tradeoff between the service latency and analytics accuracy is indispensable in designing workload dispatching algorithms for edge-based MAR systems.

III. Analytical Model of Edge-based MAR System

In this section, we describe the system model for analyzing the edge-based MAR system. The system model includes network latency, computational latency, and analytics accuracy models. The computational latency and analytics accuracy models are derived based on the performance measurements obtained from our MAR testbed.

We consider a mobile edge network with K MAR users and N heterogeneous servers including both the cloud and edge servers. Denote $\mathcal{K}$ and $\mathcal{N}$ as the set of MAR users and servers, respectively. The MAR users communicate with servers via wireless access points such as cellular base stations and WiFi hotspots. The object analytics is performed on either edge servers or cloud server. The average service latency of the kth MAR user can be defined as

$$L_k = L_k^w + L_k^t + L_k^p, \tag{1}$$

where L_k^w is the wireless latency incurred by sending a video frame from the kth user to its associated wireless access point; L_k^t is the core network latency caused by transferring the frame from the wireless access point to the server assigned to the user; and L_k^p is the computational latency of the object analytics on the server.

A. Network Latency Model

The network latency is composed of the wireless and core network latency. The wireless latency is determined by the user's video frame resolutions and wireless data rates. Since the data size of analytics results is usually small, we do not model the latency caused by transmitting the analytics results [21]. We assume that the AR video of the kth user is preprocessed into video frames with the resolution of $s_k \times s_k$ pixels. In this paper, we use s_k^2 (the number of pixels) to represent the video frame resolution of the kth MAR user. Denote σ as the number of bits required to represent the information carried by one pixel. Denote $\mathcal{S} = \{s_k^2 \,| k \in \mathcal{K}\}$ as the set of users' frame resolutions. The data size of a video frame is calculated as σs_k^2 bits. Let R_k be the average wireless data rate of the kth user. The wireless latency experienced by the kth user is modeled as

$$L_k^w = \frac{\sigma s_k^2}{R_k}. \tag{2}$$

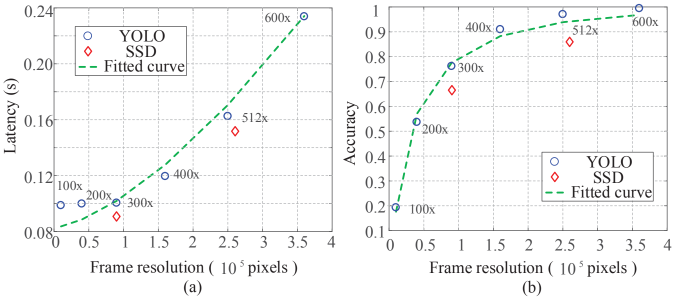
Note that we consider the simplified wireless latency model (Eq. 2) because we focus on the system level performance rather than wireless link level performance in this paper.

Since the core network usually has very high transmission capacity, its latency is mainly determined by the aggregated traffic loads in the network and the geo-distance between the wireless access points and the servers. The impact of the video frame size of a single user on the link latency of the core network is negligible. Therefore, we do not consider such an impact in our core network latency model. Denote $a_{k,n} \in \{0, 1\}$ as the server assignment indicator which indicates the kth user is served by the nth server if $a_{k,n} = 1$. Denote $\mathcal{A} = \{a_{k,n} \,| k \in \mathcal{K}, n \in \mathcal{N}\}$ as the set of users' server assignments. Here, we set $a_{k,n}$ as a binary variable to restrict that a user can be served by only one server at a time. Let $l_{k,n}$ be the core network latency between the kth user's associated wireless access point and the nth server, the core network latency of the kth user can be expressed as

$$L_k^t = \sum_{n \in \mathcal{N}} a_{k,n} l_{k,n}. \tag{3}$$

B. Computational Latency Model

The computational latency is closely related to the computational complexity of a user's task and available computational resources on servers [21]. Let c_k and f_n be the computational complexity of analyzing the kth user's video frame and the available computational resources on the nth server. We assume that the available computational resources on a server are evenly shared by the users associated with the server. Then, $f_n / \sum_{m \in \mathcal{K}} a_{m,n}$ is the computational resources allocated to one user on the nth server. Therefore, the computational latency experienced by the kth user can be modeled as



(a)
(b)

Figure 2: The latency and accuracy vs. the frame resolution.

$$L_k^p = \sum_{n \in \mathcal{N}} a_{k,n} \frac{c_k}{f_n} \sum_{m \in \mathcal{K}} a_{m,n}. \tag{4}$$

In order to characterize the computational latency, we have to figure out the relationship between the computational complexity c_k and the video frame resolution $s_k \times s_k$. To do so, we implement two object recognition algorithms, YOLO [5] and SSD [6], on our workstation with Nvidia Quadro M4000 GPU. The original YOLO algorithm resizes incoming video frames to a predefined resolution, e.g., 448 x 448, before performing the object analytics. In order to measure the impact of the video frame resolution on the object analytics performance, we enable the YOLO algorithm to resize incoming video frames to different resolutions and perform the object analytics based on resized video frames. The computational latency measured in the experiments reflects the computational complexity of the object recognition under different video frame resolutions.

Fig. 2 (a) shows that the computational latency increases when the video frame resolution becomes higher. The speed of the latency increase become faster at a higher video frame resolution. Such a relationship between the computational complexity and the video frame resolution can be characterized by a convex function. For example, the measurement data can be fitted by a convex function $f(s_k^2) = 7 \times 10^{-10} {s_k}^3 + 0.083$ with the root mean square error (RMSE) of 0.01. Based on these observations, we model the computational complexity $c_k = \psi(s_k^2)$ where $\psi(s_k^2)$ is convex with respect to the video frame resolution s_k^2.

C. Analytics Accuracy Model

The analytics accuracy highly depends on the resolutions of the AR video frame. The higher video frame resolution usually results in a better mean average precision of an object recognition function [18]. Therefore, we model the analytics accuracy as a function of the video frame resolution. We define the analytics accuracy as the ratio between the number of correctly recognized objects and that of total objects in a video frame. We build the analytics accuracy model based on the aforementioned performance measurement. On calculating the accuracy, we assume that the YOLO algorithm can detect all objects in a video frame when the video frame resolution is 600×600 pixels.

Fig. 2 (b) shows the analytics accuracy of the YOLO and SSD algorithm under different video frame resolutions. There are two observations. The first one that a higher video frame resolution enables a better analytics accuracy. The second observation is that the performance gain narrows down at a high video resolution. Based on these observations, we can use a concave function to define the relationship between the analytics accuracy and the video frame resolution. For example, the concave function $f(s_k^2) = 1 - 1.578e^{-6.5\times10^{-3}s_k}$ can fit the measurement data with less than 0.03 RMSE. Therefore, we model the analytics accuracy $A_k = \xi(s_k^2)$ where $\xi(s_k^2)$ is a concave function with respect to video frame resolution s_k^2.

D. Problem Formulation

Based on the analytical model, the total service latency of the MAR users is

$$L = \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}}\left[\frac{\sigma s_k^2}{R_k} + a_{k,n}\left(l_{k,n} + \frac{\psi(s_k^2)}{f_n}\sum_{m\in\mathcal{K}} a_{m,n}\right)\right], \tag{5}$$

and the summation of the analytics accuracy of the MAR users is

$$A = \sum_{k\in\mathcal{K}} \xi(s_k^2). \tag{6}$$

On designing the edge network orchestrator, we aim to minimize the overall service latency and maximize the total analytics accuracy of the MAR users. Therefore, designing the orchestrator is a multi-objective optimization problem [22]. There is a tradeoff between the service latency and analytics accuracy. In order to characterize the tradeoff, we introduce a positive weight parameter β which reflects the preference between the service latency and analytics accuracy in optimizing the MAR system. We adopt the weighted sum method [23] to express the multi-object optimization problem as

$$\begin{aligned} \mathscr{P}1: \quad & \min_{\{\mathcal{A},\mathcal{S}\}} && F = L - \beta A \\ & s.t. && C_1: \ \xi(s_k^2) \geq \delta_k, \forall k \in \mathcal{K}, \\ &&& C_2: \ \textstyle\sum_{n\in\mathcal{N}} a_{k,n} = 1, \forall k \in \mathcal{K}, \\ &&& C_3: \ a_{k,n} \in \{0,1\}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N} \end{aligned} \tag{7}$$

where δ_k is the minimum analytics accuracy requirement of the kth user; the constraints C_2 and C_3 ensure that an individual user is assigned to one and only one server. The weight parameter β controls the latency-accuracy tradeoff. For example, a larger β indicates that the MAR system prefers a higher accuracy. As a result, the optimal solution of Problem $\mathscr{P}1$ trades the average service latency for enhancing the analytics accuracy.

IV. The FACT Algorithm

Problem $\mathscr{P}1$ is a mixed-integer nonlinear programming problem (MINLP) which is difficult to solve [24]. We develop the FACT algorithm to solve the problem based on the block coordinate descent method [25].

To solve Problem $\mathscr{P}1$, we relax binary variables $a_{k,n}$ to continuous variables $\tilde{a}_{k,n}$. Denote $\tilde{\mathcal{A}} = \{\tilde{a}_{k,n} | k \in \mathcal{K}, n \in \mathcal{N}\}$. The relaxed problem is

$$\begin{aligned} \mathscr{P}2: \quad & \min && F = L - \beta A \\ & s.t. && C_1, C_2, 0 \leq \tilde{a}_{k,n} \leq 1. \end{aligned} \tag{8}$$

Lemma 1. *The Problem $\mathscr{P}2$ is strictly convex with respect to the relaxed server assignments $\tilde{\mathcal{A}}$.*

Proof: For any feasible $\tilde{a}_{m,n}, \tilde{a}_{i,j}, \forall m, i \in \mathcal{K}, \forall n, j \in \mathcal{N}$,

$$\frac{\partial^2 F}{\partial \tilde{a}_{i,j}\partial \tilde{a}_{m,n}} = \begin{cases} \frac{2\psi(s_i^2)}{f_j}, & i = j \text{ and } m = n, \\ 0, & i \neq j \text{ or } m \neq n, \end{cases} \tag{9}$$

The Hessian matrix $\mathbf{H} = \left(\frac{\partial^2 F}{\partial \tilde{a}_{i,j}\partial \tilde{a}_{m,n}}\right)_{KN\times KN}$ is symmetric and positive definite. The constraints C_2 and C_3 are linear. The constraints C_1 are irrelevant to $\tilde{\mathcal{A}}$. Therefore, $\mathscr{P}2$ is strictly convex with respect to $\tilde{\mathcal{A}}$ [26]. ■

Lemma 2. *The Problem $\mathscr{P}2$ is strictly convex with respect to frame resolution $\mathcal{S}$.*

Proof: For any feasible variable s_i^2, s_j^2, $\forall i, j \in \mathcal{K}$, we have

$$\frac{\partial^2 F}{\partial s_i^2 \partial s_j^2} = \begin{cases} \frac{\partial^2 \psi}{\partial s_i^2 \partial s_j^2} \sum_{n\in\mathcal{N}} \frac{\tilde{a}_{i,n}}{f_n} \sum_{m\in\mathcal{K}} \tilde{a}_{m,n} - \beta \frac{\partial^2 \xi}{\partial s_i^2 \partial s_j^2}, & i = j, \\ 0, & i \neq j, \end{cases} \tag{10}$$

Since $\psi(s_k^2)$ is convex function, $\frac{\partial^2 \psi}{\partial s_i^2 \partial s_j^2}$ is non-negative. Since $\xi(s_k^2)$ is concave function, $\frac{\partial^2 \xi}{\partial s_i^2 \partial s_j^2}$ is non-positive. Hence, the Hessian matrix $\mathbf{H} = \left(\frac{\partial^2 F}{\partial s_i^2 \partial s_j^2}\right)_{K\times K}$ is symmetric and positive definite. Constraints C_1 are convex, and Constraints C_2 and C_3 do not apply to $\mathcal{S}$. Therefore, Problem $\mathscr{P}2$ is strictly convex with respect to $\mathcal{S}$ [26]. ■

Lemmas 1 and 2 lay the foundation for solving Problem $\mathscr{P}2$ with the block coordinate descent method [25]. Based on this method, we develop a fast and accurate object analytics (FACT) algorithm which solves Problem $\mathscr{P}2$ by sequentially fixing one variable, i.e., $\tilde{\mathcal{A}}$ or $\mathcal{S}$, and updating the other one. The pseudo code of the FACT algorithm is presented in Algorithm 1. At the beginning of the algorithm, the weight β and video frame resolution $\mathcal{S}$ are initialized. With the fixed video frame resolution, we solve Problem $\mathscr{P}2$ to obtain the optimal server assignment $\tilde{\mathcal{A}}$. Then, based on the optimized $\tilde{\mathcal{A}}$, we optimize the video frame resolution. After each iteration, the value of the objective function F is calculated. The FACT algorithm iteratively optimizes $\tilde{\mathcal{A}}$ and $\mathcal{S}$ until the objective function F is converged. In the algorithm, we introduce an arbitrary small positive number τ to evaluate the convergence of the objective function as shown in line 6 of the pseudo code. After solving Problem $\mathscr{P}2$, $\tilde{a}_{k,n}$ are converted to $a_{k,n}$ according to

$$a_{k,n} = \begin{cases} 1, & n = \arg\max_{j\in\mathcal{N}} \tilde{a}_{k,j}, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Theorem 1. *The FACT algorithm converges to the optimal*

Algorithm 1: The FACT Algorithm

```
Input: The weight $\beta$, the convergence condition $\tau$, the
       initial set of frame resolutions $\mathcal{S}_0$.
Output: The set of server assignments $\mathcal{A}$, the set of
        frame resolutions $\mathcal{S}$.
$\mathcal{S} \leftarrow \mathcal{S}_0, i \leftarrow 0$;
while True do
    $\tilde{\mathcal{A}} \leftarrow$ solve Problem $\mathscr{P}2$ with fixed $\mathcal{S}$;
    $\mathcal{S} \leftarrow$ solve Problem $\mathscr{P}2$ with fixed $\tilde{\mathcal{A}}$;
    $F_i \leftarrow L - \beta A$;
    if $|(F_i - F_{i-1})/F_i| \leq \tau$ then
        break;
    $i \leftarrow i + 1$;
for $k \in \mathcal{K}$ do
    for $n \in \mathcal{N}$ do
        if $n = \arg\max_{j \in \mathcal{N}} \tilde{a}_{k,j}$ then
            $a_{k,n} \leftarrow 1$;
        else
            $a_{k,n} \leftarrow 0$;
$\mathcal{S} \leftarrow$ solve Problem $\mathscr{P}2$ with $\mathcal{A}$;
return $\mathcal{A}, \mathcal{S}$
```

solution.

Proof: The convergence of the FACT algorithm to optimal solution can be proved by showing that Problem $\mathscr{P}2$ is strictly convex with respect to each block of variables, e.g., $\tilde{\mathcal{A}}$ and $\mathcal{S}$ [25]. The convexity of Problem $\mathscr{P}2$ is proved by lemma 1 and 2. Hence, the optimality and convergence of the FACT algorithm is guaranteed. ■

Lemma 3. *The FACT algorithm has a sub-linear convergence rate.*

Proof: The block coordinate descent method ensures a sub-linear convergence rate [27]. The FACT algorithm is designed based on the block coordinate descent method. Thus, it has a sub-linear convergence rate. ■

V. Simulation Results

In this section, we evaluate the FACT algorithm through large-scale simulations. We simulate an edge network with 50 wireless access points and 20 servers. The distribution of the wireless access points and the user traffic are derived based on network traffic traces collected from an operating mobile network consisting of 10000 base stations and 50000 mobile users. The edge servers are randomly deployed in the network. The network latency between wireless access points and edge servers are generated according to the normal distribution with the mean value of 50ms. The network latency between the wireless access points and the cloud server are also generated based on the normal distribution but with the mean value of 150ms. The wireless data rates of users are uniformly distributed between 1 to 10 Mbps. Based on the measurements from our experiments, we adopt $c_k = \psi(s_k^2) = 7 \times 10^{-10} {s_k}^3 + 0.083$ TFLOPS as the computational complexity of analyzing a $s_k \times s_k$ video frame. In the simulation, the computing capacities of the cloud and edge servers are set to 10 and 2 TFLOPS, respectively. We model as $A_k = \xi(s_k^2) = 1 - 1.578 e^{-6.5 \times 10^{-3} s_k}$ as the analytics accuracy function on both cloud and edge servers. The default value of β is 20, and the minimum video frame resolution is 40000 pixels (200 × 200).

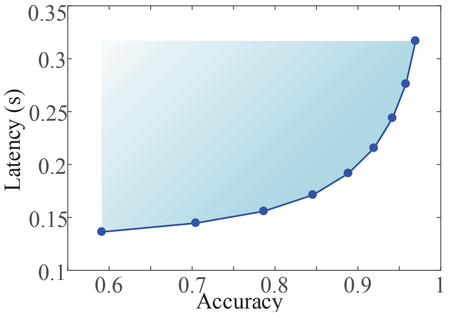


Figure 3: The Pareto boundary derived by the FACT algorithm.

In the simulation, we compare the FACT algorithm with the three categories of algorithms summarized in Table I.

- **Baseline:** the baseline algorithm has a fixed video frame resolution and randomly assigns servers to MAR users.
- **Server assignment optimized:** this category of algorithms optimize the server assignments, but the video frame resolution is predefined. We implement maximum accuracy (maxA) and minimum latency (minL) algorithms which adopt the largest and smallest frame resolutions, respectively.
- **Frame resolution optimized:** these algorithms optimize the frame resolution, but the server assignments are not optimized. We implement two methods: random server selection (RandS) and least workload server selection (LoadS).

Table I. Algorithm Comparison

	frame resolution		server selection		
	fixed	optimized	random	greedy	optimized
FACT		x			x
Baseline	x		x		
MaxA	x				x
MinL	x				x
LoadS		x		x	
RandS		x	x		

A. *The impact of β*

The value of β impacts the tradeoff between the service latency and analytics accuracy in the edge based MAR system. As shown in Fig. 3, by varying the value of β, we derive the Pareto boundary which characterizes the latency-accuracy tradeoff obtained by the FACT algorithm.

Fig. 4 shows the impact of β on the service latency and analytics accuracy of different algorithms. When β increases,

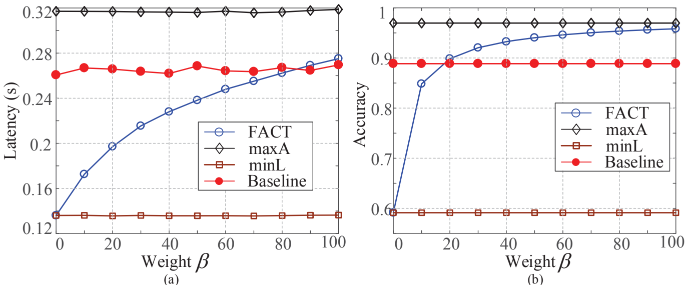


Figure 4: The system performance vs. β.

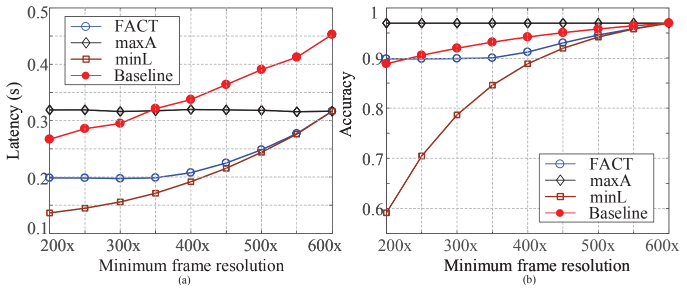


Figure 5: The system performance vs. frame resolution.

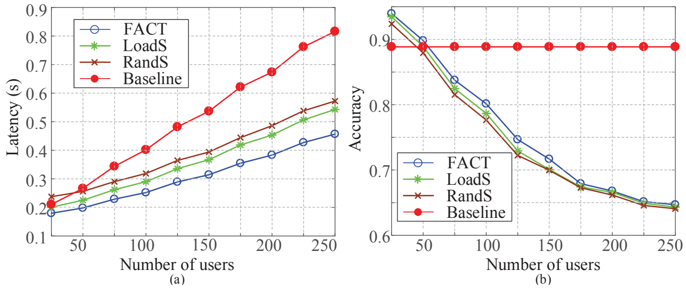


Figure 6: The system performance vs. number of users.

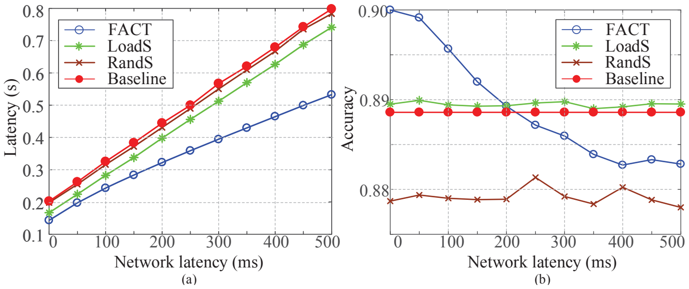


Figure 7: The system performance vs. core network latency.

the MAR system emphasizes on the performance of the analytics accuracy. As a result, the FACT algorithm trades the service latency for the analytics accuracy. Since the maxA, minL, and baseline algorithms do not optimize the video frame resolution, the variation of β does not change the latency-accuracy tradeoff. Hence, the performance of these algorithm does not show many changes versus β. The latency fluctuation of the baseline algorithm is because of the random server assignments. This simulation result also shows that, as compared to the baseline algorithm, the FACT algorithm is able to reduce about 26% service latency while maintaining the similar analytics accuracy ($\beta = 20$), and enhance about 8% analytics accuracy when ensuring a similar service latency ($\beta = 90$).

B. The impact of AR video frame resolution

The AR video frame resolution impacts not only the transmission and computational latency but also the analytics accuracy. In order to evaluate such impacts, we vary the minimum video frame resolutions in the simulation. The maximum video frame resolution is fixed at 600×600 pixels. The optimized video frame resolution is between the minimum and maximum video frame resolutions. The video frame resolution of the baseline algorithm is defined as the mean value of the minimum and maximum video frame resolutions. As shown in Fig. 5 (a), the service latency of the minL and baseline algorithm increases versus the minimum video frame resolution. When the minimum video frame resolution is less than 350×350 pixels, the FACT algorithm maintains the system performance (both the service latency and analytics accuracy) because of the frame resolution optimization. When the minimum video frame resolution is larger than 500×500 pixels, the FACT algorithm has the same service latency as the minL algorithm because of the optimal frame resolution equals to the minimum frame resolution. Fig. 5 (b) shows that the FACT algorithm obtains a lower latency at the cost of analytics accuracy. However, the latency-accuracy tradeoff is mitigated. For example, as compared with the baseline algorithm, the FACT algorithm reduces about 40% latency at the cost of around 3% accuracy when the minimum video frame resolution is 400×400 pixels.

C. The impact of the number of users

Fig. 6 evaluates the impact of the number of users on the performance of the edge-based MAR system. When the number of users increases, the edge servers experience more workloads. Thus, the computational latency increases. As compared with the other algorithms, the FACT algorithm achieves the smallest latency. Since the FACT, LoadS and RandS algorithms optimize the video frame resolution, the performance differences between the FACT algorithm and the other two algorithms shows the improvement gained from the optimal server assignment. Both RandS and baseline algorithms adopt the random server selection. Hence, the performance difference between these algorithms reflect the advantages of the frame resolution optimization.

As compared with the baseline algorithm, the FACT algorithm gains up to 38% service latency reduction with less than 10% loss of analytics accuracy when the number of users is 100. When the number of users increases, the servers are overloaded. In order to maintain a low service latency, the FACT algorithm aggressively reduces the video frame resolution, which is the reason for the 10% loss in the analytics accuracy. As compared with LoadS and RandS algorithms, the FACT algorithm improves both the latency and accuracy performance.

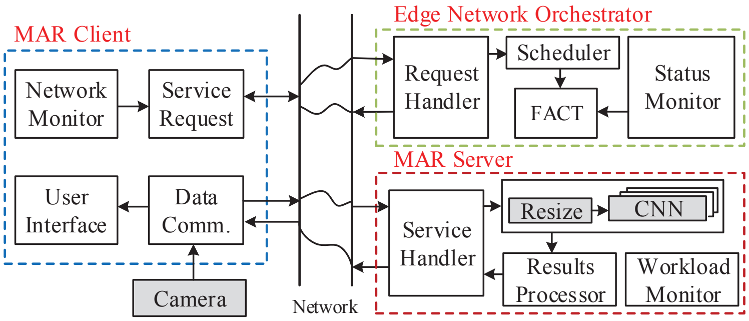


Figure 8: The overview of edge-based MAR system.

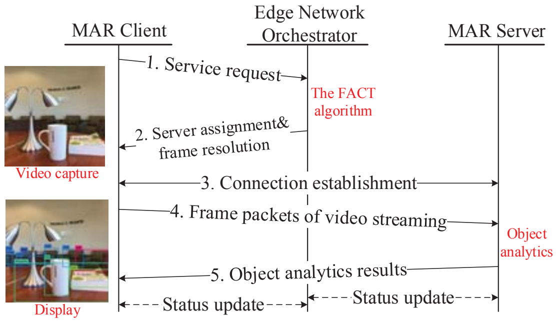

MAR Client
Edge Network Orchestrator
MAR Server
1. Service request
The FACT algorithm
2. Server assignment& frame resolution
Video capture
3. Connection establishment
4. Frame packets of video streaming
Object analytics
5. Object analytics results
Display
Status update
Status update

Figure 9: The MAR communication protocol.

D. The impact of the average network latency

Fig. 7 shows the impact of the average network latency on the performance of the edge-based MAR system. Here, the average network latency reflects the average transmission delay between the wireless access points and servers. As shown in the figure, the service latency of the MAR system increases versus the average network latency. However, the FACT algorithm is able to minimize the service latency at the cost of a slight decrease of the analytics accuracy. The latency gap between the FACT, LoadS and RandS algorithms reflects the gain obtained through optimizing the server assignment. The latency gap between the baseline and RandS algorithms is due to the video frame resolution optimization.

VI. The System Implementation and Experiments

In this section, we implement the edge-based MAR system including the edge network orchestrator, MAR clients and MAR servers, and develop the corresponding MAR communication protocol. We conduct experiments based on the implementation to validate the performance of the edge network orchestrator and MAR communication protocol.

A. The edge-base MAR system implementation

Fig. 8 overviews the edge-based MAR system which consists of three major components: the edge network orchestrator, MAR client and MAR server.

The edge network orchestrator: the orchestrator is responsible for optimizing the server assignment and video frame resolution for MAR users. The core of the orchestrator is the FACT algorithm which performs the optimization. In order to realize the FACT algorithm, three auxiliary modules are implemented. The first one is the request handler which puts the incoming service requests into a FIFO queue and dispatches the server assignments and video frame resolutions derived from the FACT algorithm to the corresponding MAR users. The second module is the scheduler which manages the service queue and invokes the FACT algorithm based on the queue length and predefined optimization interval. In the implementation, the FACT algorithm is invoked if the queue length equals to 10 or a 100ms timer is expired after the last optimization. The optimization trigger, e.g., the queue length and timer, is adjustable in the implementation. The third module is the system status monitor which monitors and collects the system information including users' wireless data rates, network latency between users and servers, and workloads on servers. This information is inputted to the FACT algorithm for optimizing the server assignment and video frame resolution. The orchestrator is deployed at network edge and can be accessed by MAR users with very short latency. We also implement the baseline, maxA, minL, LoadS, and RandS algorithms in the network orchestrator for the performance comparison.

The MAR Client: The MAR client captures the real-world video, sends frames to a server and overlays the received information with its corresponding objects. There are four functional modules implemented in the MAR client. The first one is the network status monitoring module which measures wireless data rates and the network latency between the user and servers. This model also triggers a service request to the orchestrator for the performance optimization when the user's service quality is lower than a threshold. The second one is the user interface that captures the real-world environments and display the analytics results received from the server. The third one is the service request module which sends service request to the orchestrator. The service request message contains the information of the user's wireless data rate and network latency toward edge and cloud servers. It also parses the control information about the frame resolution and server assignment received from the orchestrator. The fourth module is the data communication module which streams the AR video frames to an assigned server, and receives the analytics results from the server. This module is also responsible for resizing the video frames according to the frame resolution selected by the orchestrator. Since the current implementation of our MAR server only supports six video frame resolutions, we select one of the supported resolutions to approximate the video frame resolution determined by the orchestrator.

The MAR Server: The MAR server is developed to process the video frames and send the analytics results back to MAR users. The server is designed to serve multiple users simultaneously through multi-threading. There are four major modules implemented on the server. The first one is the service handler module which performs the authentication and establishes a socket connection with MAR users. This module is also responsible for dispatching the analytics results to corresponding MAR users.

The second one is the object analytics module, which decompresses the frames and performs the object analytics for MAR users. The object analytics module is designed based on the YOLO framework with the GPU acceleration [5]. In order to allow the object analytics module to analyze video frames with different resolutions, we loaded six models with different input resolutions. Therefore, the implemented object analytics module supports six video frame resolutions. If the resolution of incoming video frames does not match any of these video frame resolutions, the object analytics module resizes the input video frames into one of the supported resolutions and then analyzes the resized video frames.

The third one is the results processing module which prepares the information related to the recognized objects. The fourth module is the workload monitor which monitors workloads on the server and updates the workload information to the orchestrator periodically.

B. The edge-based MAR communication protocol

As illustrated in Fig. 9, the proposed communication protocol enables the MAR service in five steps.

1) The MAR user sends a service request to the orchestrator. The service request message also includes the user's wireless data rates and network latency measurements. After the service is initialized, the service request message is used to periodically update the user's wireless data rates and network latency measurements to the orchestrator.
2) Upon receiving the service request, the orchestrator decides the server assignment and frame resolution, and sends them back to the user.
3) The user establishes a connection to the assigned server, and informs the server of its configuration information.
4) After the connection is established, the user sends its AR video frames to the server for the object analytics.
5) The MAR server detects and recognizes objects in the video frames, and sends the results back to the user.

C. Performance evaluation

An edge-based MAR testbed as shown in Fig. 10 is developed to evaluate the performance of the proposed edge network orchestrator. The MAR clients are implemented in Raspberry Pis and LXC containers connected to the emulated network via wireless routers. The edge network is emulated by Mininet [28]. Using Mininet, we can configure the core network latency and evaluate its impact on the system performance. Three edge MAR servers are implemented using three NVIDIA Jetson TX development kits, and one cloud MAR server is implemented on a Dell workstation with Nvidia Quadro M4000 GPU.

Fig. 11 (a) shows the latency and accuracy versus the number of users. Although the latency increase with the growth of number of users, the FACT algorithm achieves significant latency reduction as compared to the other algorithms. For example, when the number of users is 8, the FACT algorithm reduces 27% latency with only 1% accuracy loss as compared to the baseline algorithm.

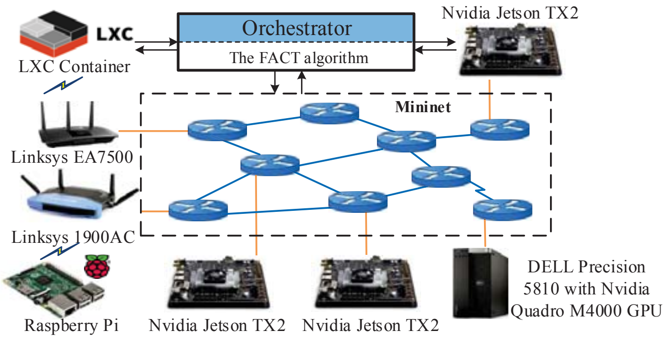


Figure 10: The edge-base MAR testbed.

Fig. 11 (b) shows the service latency and analytics accuracy versus the core network latency. The service latency increases with the growth of the core network latency. However, the FACT algorithm outperforms the other algorithms. For instance, it achieves about 25% latency reduction at the cost of less than 1% accuracy loss when the average core network latency is 100 ms.

Fig. 11 (c) evaluates the impact of wireless channel conditions on the service latency and analytics accuracy. We use the Keysight J7211A attenuation control unit to vary the channel fading between the wireless access point and a MAR user. In this way, we evaluate the system performance under different wireless channel conditions. A larger channel fading increases the signal attenuation and leads to a lower wireless data rate. As a result, the service latency increases. Owing to the video frame resolution optimization, the FACT algorithm is able to achieve a low service latency while maintaining a high analytics accuracy.

Fig 12 shows the tradeoff between the latency and accuracy with different β. A larger β indicates the system prefers a higher analytics accuracy. When β is small, e.g., $\beta = 20$, the FACT algorithm trades the analytics accuracy for the service latency. Therefore, it achieves more than 40% service latency reduction at the cost of about 10% analytics accuracy loss as compared with the maxA algorithm. When β is large, e.g., $\beta = 60$, the FACT algorithm prioritizes the analytics accuracy improvement. As a result, it achieves an almost perfect analytics accuracy. At the same time, the FACT algorithm achieves a lower latency than the maxA, LoadS, and RandS algorithms.

VII. Conclusion

In this paper, we design an edge network orchestrator to improve the responsiveness and analytics accuracy of the edge-based MAR system. We build analytical models for studying the latency-accuracy tradeoff in edge-based MAR systems, and develop the FACT algorithm to improve the system performance by optimizing the server assignment and frame resolution selection. The performance of the FACT algorithm is evaluated through network simulations. In addition, we implement the edge-based MAR system with the proposed network orchestrator and the corresponding communication protocol. The performance of the edge network orchestrator

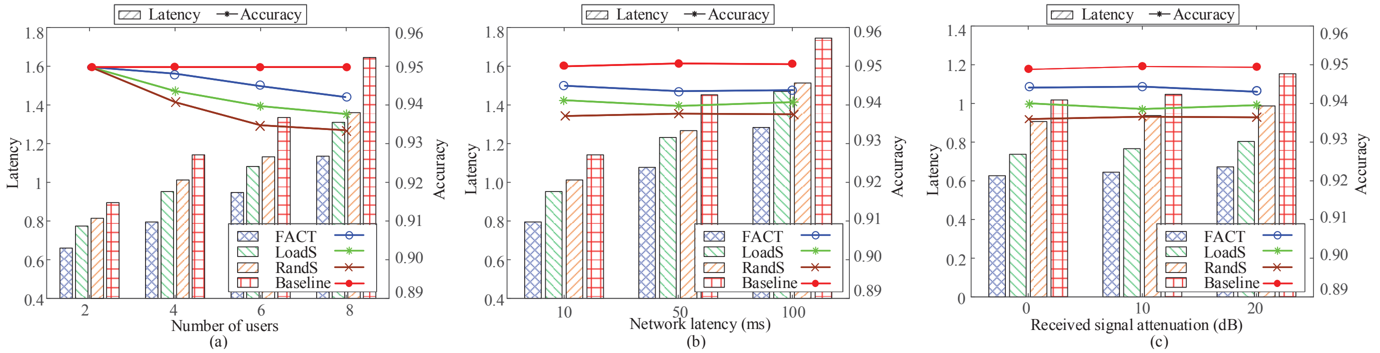


Figure 11: The system performance measurements with different configurations.

and the edge-based MAR system is validated in our experiments.

VIII. Acknowledgment

This work is partially supported by the US National Science Foundation under Grant No. 1731675 and the UNC-Charlotte Faculty Research Grant.

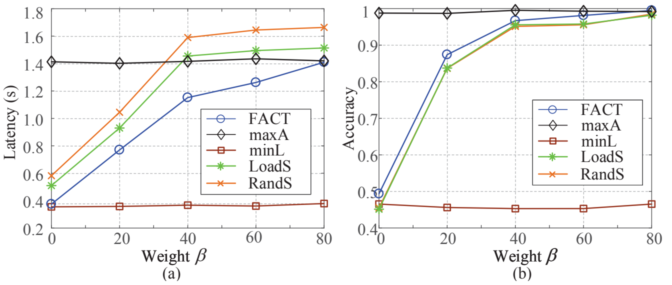


Figure 12: The performance measurements vs. β.

References

[1] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, “Mobile Augmented Reality Survey: From Where We Are to Where We Go,” *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[2] Digital Capital, “Augmented/Virtual Reality Report Q3 2017,” 2017. [Online]. Available: http://www.digi-capital.com/news/2017/01/after-mixed-year-mobile-ar-to-drive-108-billion-vrar-market-by-2021/#.WWfASlLMwdU

[3] P. Jain, J. Manweiler, and R. Roy Choudhury, “OverLay: Practical Mobile Augmented Reality,” in *ACM, MobiSys 2015*, Florence, Italy, May 2015, pp. 331–344.

[4] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in *NIPS 2015*, Montreal, Canada, Dec. 2015, pp. 91–99.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *IEEE CVPR 2016*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector,” in *ECCV*, Amsterdam, Netherlands, Oct. 2016.

[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional Architecture for Fast Feature Embedding,” in *ACM, MM 2014*, Orlando, Florida, USA, Nov. 2014, pp. 675–678.

[8] L. N. Huynh, Y. Lee, and R. K. Balan, “DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications,” in *ACM MobiSys 2017*, Niagara Falls, New York, USA, Jun. 2017, pp. 82–95.

[9] X. Zeng, K. Cao, and M. Zhang, “MobileDeepPill: A Small-Footprint Mobile Deep Learning System for Recognizing Unconstrained Pill Images,” in *ACM MobiSys 2017*, Niagara Falls, New York, USA, Jun. 2017, pp. 82–95.

[10] Z. Huang, W. Li, P. Hui, and C. Peylo, “CloudRidAR: A Cloud-Based Architecture for Mobile Augmented Reality,” in *ACM MobiSys workshop 2014*, Bretton Woods, NH, USA, Jun. 2014, pp. 29–34.

[11] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, “Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices,” in *ACM Sensys 2015*, Seoul, South Korea, Nov. 2015, pp. 155–168.

[12] P. Jain, J. Manweiler, and R. Roy Choudhury, “Low Bandwidth Offload for Mobile AR,” in *ACM CoNEXT 2016*, Irvine, California, USA, Dec. 2016, pp. 237–251.

[13] R. Shea, A. Sun, S. Fu, and J. Liu, “Towards Fully Offloaded Cloud-based AR: Design, Implementation and Experience,” in *ACM MMSys 2017*, Taipei, Taiwan, Jun. 2017, pp. 321–330.

[14] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision And Challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[15] M. Satyanarayanan, “The Emergence of Edge Computing,” *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

[16] L. Tong, Y. Li, and W. Gao, “A Hierarchical Edge Cloud Architecture for Mobile Computing,” in *IEEE INFOCOM 2016*, San Francisco, CA, USA, April 2016, pp. 1–9.

[17] B. Cici, M. Gjoka, A. Markopoulou, and C. T. Butts, “On the Decomposition of Cell Phone Activity Patterns and Their Connection with Urban Ecology,” in *ACM MobiHoc 2015*, Hangzhou, China, Jun. 2015, pp. 317–326.

[18] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, “Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors,” *arXiv preprint arXiv:1611.10012*, 2016.

[19] M. Jia, W. Liang, Z. Xu, and M. Huang, “Cloudlet Load Balancing in Wireless Metropolitan Area Networks,” in *IEEE INFOCOM 2016*, San Francisco, CA, USA, April 2016, pp. 1–9.

[20] H. Tan, Z. Han, X.-Y. Li, and F. C. Lau, “Online Job Dispatching and Scheduling in Edge-Clouds,” in *IEEE INFOCOM 2017*, Atlanta, GA, USA, May 2017, pp. 1–9.

[21] 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, 2015.

[22] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001, vol. 16.

[23] R. T. Marler and J. S. Arora, “The Weighted Sum Method for Multi-Objective Optimization: New Insights,” *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[24] M. R. Bussieck and A. Pruessner, “Mixed-Integer Nonlinear Programming,” *SIAG/OPT Newsletter: Views & News*, vol. 14, no. 1, pp. 19–22, 2003.

[25] L. Grippo and M. Sciandrone, “On The Convergence Of The Block Nonlinear Gauss–Seidel Method Under Convex Constraints,” *Operations research letters*, vol. 26, no. 3, pp. 127–136, 2000.

[26] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[27] A. Beck and L. Tetruashvili, “On the convergence of block coordinate descent type methods,” *SIAM journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[28] Team Mininet, “Mininet-An Instant Virtual Network on your Laptop (or other PC),” 2014.