CoOMO: Cost-efficient Computation Outsourcing with Multi-site Offloading for Mobile-Edge Services

Tianhui Meng*, Huaming Wu†, Zhihao Shang‡, and Yubin Zhao*
*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
†Tianjin University, Tianjin, China
‡Freie Universität Berlin, Berlin, Germany
{th.meng, zhaoyb}@siat.ac.cn, whming@tju.edu.cn, zhihao.shang@fu-berlin.de

***Abstract*—Mobile phones and tablets are becoming the primary platform of choice. However, these systems still suffer from limited battery and computation resources. A popular technique in mobile edge systems is computing outsourcing that augments the capabilities of mobile systems by migrating heavy workloads to resourceful clouds located at the edges of cellular networks. In the multi-site scenario, it is possible for mobile devices to save more time and energy by offloading to several cloud service providers. One of the most important challenges is how to choose servers to offload the jobs.**

In this paper, we consider a multi-site decision problem. We present a scheme to determine the proper assignment probabilities in a two-site mobile-edge computing system. We propose an open queueing network model for an offloading system with two servers and put forward performance metrics used for evaluating the system. Then in the specific scenario of a mobile chess game, where the data transmission is small but the computation jobs are relatively heavy, we conduct offloading experiments to obtain the model parameters. Given the parameters as arrival rates and service rates, we calculate the optimal probability to assign jobs to offload or locally execute and the optimal probabilities to choose different cloud servers. The analysis results confirm that our multi-site offloading scheme is beneficial in terms of response time and energy usage. In addition, sensitivity analysis has been conducted with respect to the system arrival rate to investigate wider implications of the change of parameter values.

***Index Terms*—Computation outsourcing, Multi-site offloading, Mobile-edge computing, Queueing networks**

I. Introduction

As a key 5G enabler technology, Mobile Edge Computing (MEC) has emerged as a new computing paradigm that provides end-users with low latency in their access to applications deployed at the edge of the cloud [1], [2]. In MEC, computation offloading is a promising solution proposed to improve the mobile devices' performance by migrating heavy computation workload to resourceful servers [3]. In recent years, there has been a lot of research on mobile cloud offloading and computation outsourcing [4]–[6]. Mobile-edge computing offloading is different from the traditional client-server architecture, where a thin client always migrates computation to a server [7]. The current cloud computing infrastructure provides mobile systems with plenty and easy access to public cloud resources. Hence, there are several cloud service providers that use public clouds to address the mobile computing problems. For example, Apple's iCloud provides a service to its customer by hosting their applications and data in public clouds (i.e., Amazon EC2 and Microsoft Azure) [8]. The possibility to offload involves taking a decision regarding whether and what computation to migrate [9].

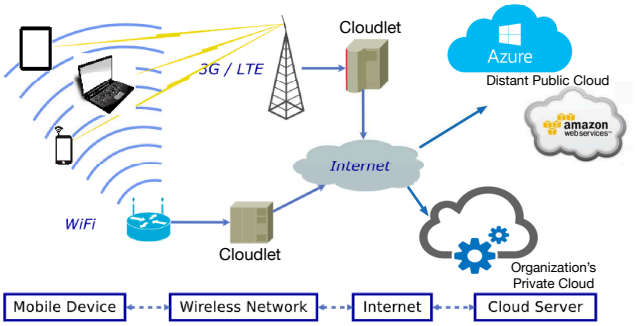


Fig. 1. Multi-site offloading scenario.

Mobile devices can access multiple cloud providers and edge servers (Fig. 1) and it is possible for mobile systems to optimize their metrics by offloading different parts of the computation to different servers [4], [10], [11]. However, the issues of poor flexibility, complex structure, and not suitable for lightweight equipment hinders the large-scale application of the existing schemes.

In this paper, we propose a cost-efficient and light-weighted scheme for multi-site offloading and outsourcing in mobile-edge services. Different from the existing schemes, the proposed scheme accelerates the decision process by leveraging open queueing network models. Given the arrival and service rates of the queues, the optimal probability to assign jobs to offload and the optimal strategy to choose offloading servers are specified for the system gain in terms of response time and energy consumption. We also propose metrics to evaluate the performance and cost of the multi-site offloading system.

In order to validate the proposed offloading scheme, experiments are conducted by including a mobile chess game in our offloading engine, where we run experiments using different

*Corresponding author: Huaming Wu.

mobile devices in different network conditions. A mobile chess game is a good candidate application to investigate the benefits of mobile cloud offloading since the data transmission is small but the computation work can be relatively heavy. The experimental results confirm that using the proposed multi-site offloading scheme more time and energy can be saved on the mobile devices. Real-time multimedia applications, especially real-time strategy games, fitness applications are some applications that benefit from this approach.

The remainder of this paper is structured as follows. Section 2 presents the system model and metrics. Section 3 shows the evaluation of the proposed offloading scheme. Finally, the paper is concluded in Section 4.

II. Related Work

A number of researches exist on offloading decisions for improving system performance and saving energy [10], [12]–[15]. To achieve energy-efficient offloading under a completion time constraint, Guo et al. [13] provide a dynamic offloading decision scheme to reduce energy consumption and shorten completion time. Hu et al. [15] propose a software framework to deploy existing robotic software packages to the cloud and transform them into cloud services. Wu and Wolter investigate two types of delayed offloading policies and optimize a proposed Energy-Response time Weighted Product (ERWP) metric [16]. However, most of the prior studies in this area propose a limited form of offloading, that is limited to a single server as the offloading target [16], [17].

Besides the schemes for single-site offloading, there are a few number of works in the area of multi-site offloading decisions. Goudarzi et al. study a hybrid solution that finds the offloading solution in a timely manner with two decision algorithms [18]. Enzai et al. [19] propose a heuristic algorithm for the multi-site computation offloading problem. Since the multi-site offloading depicts a more generic mobile-edge computing model, and it contains the single-site offloading problem, we address multi-site decision problem in this paper. Compared with existing solutions, our proposed model is lighter and more flexible.

III. System Model

We propose a novel offloading policy for the multi-site offloading system by considering the following realistic scenario. Rather than directly connecting to remote cloud servers, the mobile devices use a nearby cloudlet (or an edge server) through Wi-Fi or 3G/LTE networks to obtain reliable connections and services. The term cloudlet refers to a layer connecting mobile devices and cloud servers. A cloudlet can be either a computer or a cluster of digital infrastructure which is well-connected to the Internet and provides cloud users with a rapid response and specifically customized functionalities [20]. As shown in Fig. 1, cloudlets are located close to mobile devices while the cloud servers are generally far.

The jobs of the system are generated by the mobile applications which can either execute them locally on the mobile device or offload them to cloud servers. In the latter case the mobile devices discover a nearby cloudlet and offload jobs to it before receiving the computation results sent back by the servers. The offloadable jobs can be any computation-intensive or energy-intensive jobs such as Optical Character Recognition (OCR), real-time translating and chess algorithm computation. The offloading decision mechanisms on the mobile systems are managed by the system administrator by a Mobile Device Management (MDM) system. The administrator can decide how many jobs to offload and how many to execute locally. The decision algorithm we use is a static algorithm which assigns jobs to be offloaded or executed locally randomly with a certain probability. The jobs assigned to offload are temporarily stored in the cloudlet offloading buffer before they are sent to the cloud servers.

There are several cloud servers available for executing the jobs in this scenario. The cloudlets collect jobs from nearby mobile clients and decide how many jobs to offload to each server in order to save the most execution time and energy. The cloudlet, also managed by the system administrator, uses a static strategy to offload the jobs to different servers according to certain probabilities. Finally, the jobs are offloaded to the desired servers through a network and after being processed by the servers, the results are sent back to the mobile devices.

An example of this scenario could be a financial company, where the employees use their mobile devices to do a certain computation job. There are cloudlets in each floor of the company building which the mobiles can connect and offload jobs to. After collecting the jobs, the cloudlet sends them to either the company's private cloud or a public cloud through Internet to execute the jobs. Finally, the computation results are sent back to the mobile devices.

A. Performance Analysis Model

In order to evaluate the performance of a two-site mobile offloading execution, we define an open queueing network model from our scenario shown in Fig. 2. The cloud servers and the mobile device execution are represented as queueing nodes. It is assumed the service times have an exponential distribution for each queue [21].

It is assumed that the jobs are generated by a number of mobile devices as a Poisson process with rate λ. Each time a job is generated, a decision has to be taken by the mobile

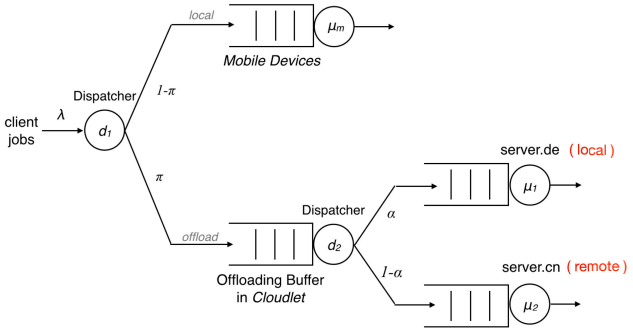


Fig. 2. A queueing model for mobile cloud offloading system

TABLE I
TABLE OF QUEUEING MODEL PARAMETERS

Parameter	Definition
λ	The arrival rate of queueing network
μ_1	The service rate of German server
μ_2	The service rate of Chinese server
μ_m	The service rate of Mobile device
d_1	The Dispatcher 1
d_2	The Dispatcher 2
π	The probability of a job assigned to the offloading queue
α	The probability of a job assigned to the server.de queue

devices whether it is offloaded to the cloud server or executed locally. There is a dispatcher d_1 in Fig. 2 which is used to allocate the offloadable jobs either to the cloud servers or to the mobile device with an offloading probability π. Thus, a ratio of $1-\pi$ of the total jobs are allocated to the mobile device and served with rate μ_m.

After an offloadable job is assigned to offloading, it is first stored in the offloading buffer in the cloudlet. The dispatcher d_2 is used to allocate the jobs to the different servers. In our scenario there are two cloud servers in the system, one server is in Germany (*server.de*) and the other one is located in China (*server.cn*). The service rates are μ_1 and μ_2 respectively. α of the jobs are allocated to *server.de* while $1-\alpha$ of the jobs are offloaded to *server.cn*. The mobile devices are assumed to be located in Germany. In the proposed two-site mobile cloud offloading system management, the two assignment probabilities π and α are the parameters that the system administrator can tune to adjust the performance. In the following sections, we explore how to determine the optimal probabilities π and α for the optimal system gain.

B. Performance Metrics

In order to evaluate the performance of the multi-site offloading system, we propose several metrics: average response time, mean energy consumption, accounting cost and Dollar per Job metric. We not only analyze the performance metrics independently, but also consider the tradeoff between different metrics in this section.

1) Average response time: The response time is the time between arrival of a job until it completes service and departs. In queueing theory, the mean response time can be computed using Little's law [22]

$$\mathbb{E}[N] = \lambda \mathbb{E}[R]. \tag{1}$$

$\mathbb{E}[\cdot]$ is the expectation of a random variable. Then the response time

$$\mathbb{E}[R] = \frac{\mathbb{E}[N]}{\lambda} = \left(\frac{\rho}{1-\rho}\right)\frac{1}{\lambda} = \frac{1}{\mu-\lambda}, \tag{2}$$

where ρ is the system utilization.

In the proposed queueing network model, the system mean response time equals the average of the response times in the offloading queue and in the local execution queue. That is the sum of products of response time and probability of assignment to each queue:

$$\hat{\mathbb{E}}[R] = (1-\pi)\cdot\mathbb{E}[R_{\text{local}}] + \pi\cdot\mathbb{E}[R_{\text{offload}}]. \tag{3}$$

Similarly, the response time of the two-site offloading queue is computed as

$$\mathbb{E}[R_{\text{offload}}] = (1-\alpha)\cdot\mathbb{E}[R_{cn}] + \alpha\cdot\mathbb{E}[R_{de}]. \tag{4}$$

Substituting Eq. 2, Eq. 4 and model parameters into Eq. 3 we get the expected response time of the multi-site offloading system as:

$$\hat{\mathbb{E}}[R] = \frac{1-\pi}{\mu_m-(1-\pi)\lambda} + \pi\left(\frac{\alpha}{\mu_1-\alpha\pi\lambda} + \frac{1-\alpha}{\mu_2-(1-\alpha)\pi\lambda}\right). \tag{5}$$

2) Energy consumption: The energy consumption is the energy spent by the mobile device in a period of time. Aaron Carroll made elaborate measurements and computations of the energy consumption of each module in mobile systems in [23]. As introduced by Carroll, the energy consumption of executing different commands are not equal. Thus we assign two energy parameters e_s and e_m to represent the energy consumption for offloading execution and local execution of jobs on the smartphone respectively. The offloading energy parameter e_s includes both the power used by data transmission and waiting for the results. Since the value of energy in mobile devices and servers are different, we assign another specific weight (θ_s and θ_m). Then the expected energy consumed per job is:

$$\mathbb{E}[\mathcal{E}] = \pi\mathbb{E}[R_{\text{offload}}]\, e_s\theta_s + (1-\pi)\mathbb{E}[R_{\text{local}}]\cdot e_m\theta_m\ . \tag{6}$$

That is the expected energy consumption equals the sum of products of the expected response time and energy parameter of each queue. Substituting the model parameters into Eq. 6, we get

$$\mathbb{E}[\mathcal{E}] = \frac{(1-\pi)e_s\theta_s}{\mu_m-(1-\pi)\lambda} + \pi e_m\theta_m\left(\frac{\alpha}{\mu_1-\alpha\pi\lambda} + \frac{1-\alpha}{\mu_2-(1-\alpha)\pi\lambda}\right). \tag{7}$$

3) Energy-response time tradeoff metric: Energy consumption and response time are two primary aspects for mobile cloud offloading systems which must be taken into consideration when making offloading decisions [21]. In order to investigate how expected response time will interact with expected energy consumption, we study the tradeoff between these two metrics, which is a nontrivial multi-objective optimization problem:

$$\textit{Tradeoff} = \mathbb{E}[R]\cdot\mathbb{E}[\mathcal{E}] = \frac{\mathbb{E}[\mathcal{E}]}{\frac{1}{\mathbb{E}[R]}}\ . \tag{8}$$

The tradeoff metric we propose is an objective function formed from the product of the expected energy consumption

(Eq. 6) and the expected response time (Eq. 3). This can be seen as energy per job metric, which is the better the lower.

4) Accounting cost: In addition to energy and performance we propose an accounting cost metric to represent the cost property of the multi-site offloading system.

The accounting cost we considered is the system cost. We assume that the billing of the server in Germany (*server.de*) is more expensive than the server in China (*server.cn*). But the German server is closer to the user. Since most popular cloud service providers, like Google App Engine (GAE) and Sina App Engine (SAE), charge their users by incoming and outgoing network traffic, we define two cost weighting parameters ζ_{de} and ζ_{cn} that are multiplied by the number of jobs. Hence the cost metric is given by the sum of the probability of offloading to each server multiplied with the cost weighting parameter:

$$Cost = \pi\lambda\left[\alpha \cdot \zeta_{de} + (1-\alpha) \cdot \zeta_{cn}\right] \ . \tag{9}$$

5) Dollar per Job metric: The tradeoff between response time and cost metric is also analyzed. An objective function formed from the tradeoff of these two metrics is created to demonstrate the tradeoff situation. Since the reciprocal of the response time can be interpreted as the number of processed jobs, we call the product *Dollar per Job* metric as shown below:

$$Dollar\ per\ Job = \mathbb{E}\left[R\right] \cdot Cost = \frac{Cost}{\frac{1}{\mathbb{E}[R]}} \ . \tag{10}$$

As a system designer, one may look forward to maintaining the response time with lower cost, as for the *Dollar per Job* metric, the lower the better.

The parameters for the queueing model are measured from runtime experimental data. In Section V we evaluate the offloading system using the proposed metrics.

IV. Runtime of the Offloading Chess Game

In Table I, the parameters μ_1, μ_2 and μ_m are needed as the input of the proposed queueing network model. Thus in this section, we explore these parameters by conducting experiments. A mobile chess game is a good candidate application to investigate the benefits of mobile cloud offloading since the data transmission is small but the computation work can be relatively heavy. We develop our testbed based on the chess module CuckooChess 1.12 [24], an advanced free open source chess program under the GNU General Public License written in Java. The CuckooChess module implements many of the standard methods for computer chess programs which enable us to develop various desired chess strategies.

We show the runtime experiments of the offloading chess engine. Most chess programs are divided into two parts: an engine that computes the best move given a current board representation and a user interface. The chess engine can be run either on the mobile device or on the remote server while the interface will always remain on the mobile device.

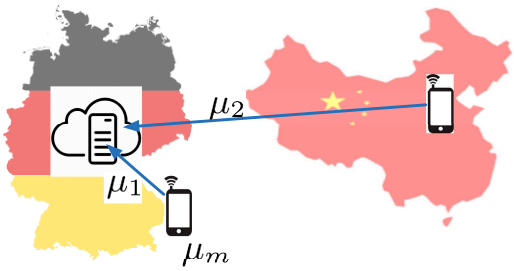


Fig. 3. The mobile offloading experiment deployment

TABLE II
Traceroute results from different networks

from-to	**Number of Hops**	**Packet Loss**	**Average RTT(ms)**
eduroam - server	7	0.10%	7.5
O2 DSL - server	15	0.20%	52.8
China ADSL - server	22	3.60%	286.0
eduroam - China server	22	3.10%	288.2

The two parts communicate with each other using a public communication protocol. The most popular protocol is the Chess Engine Communication Protocol (CECP) [25].

We vary the search depth in the tree which is a parameter commonly used to adjust the difficulty of a chess game. We investigate only the runtime of the chess moves, which is used as an indicative parameter for the performance of the device. In our evaluations, we do not consider the influence of fault occurrences in the communication networks because we only focus on the total response time of the offloading server which includes the transmission time and the server execution time.

A. Hardware and Network Specification

The experiment deployment is illustrated in Fig. 3. We use different connections to parameterize our model. The offloading server (xen-virtual-machine: 2.53 GHz 4core Xeon CPU E5649 with 8 GB RAM) is located at our institute in Berlin. For the runtime experiments we used two different mobile phones, a Samsung Galaxy S6 (2.1 GHz Exynos 7420 CPU with 3 GB RAM) and a Redmi 2 (1.2 GHz Qualcomm Snapdragon 410 CPU with 1 GB RAM). We connected the Samsung Galaxy S6 to the server through two different networks, the eduroam WLAN at the institute and the O2 DSL network from a residential area in Berlin. The Redmi 2 connected to the offloading server through ADSL network form China Unicom (a Chinese state-owned telecommunications operator) from Beijing. The mobile phones access the networks through Wi-Fi.

We could hence investigate three very different network connections to our offloading server: an excellent connection using *eduroam*, a good connection using *O2 DSL* and a less stable, and longer connection from China using *China Unicom ADSL*. In order to show the three network characteristics, we

run the MTR tool [26] for about 10 minutes in the each scenario and show the results in a table.

The first three result rows in Table II summarize the traceroute results to the offloading server host https://www.mi.fu-berlin.de/offload from the three networks. They illustrate the number of hops, the packet loss as well as the average round-trip time (RTT) for a packet to this host.

When the mobile device is connecting to the eduroam network in our institute, the average RTT from the MTR tool is 7.5 ms and it needs 7 hops to the offloading server. Meanwhile, in the O2 DSL, it takes 15 hops and the average RTT is 52.8 ms, which has more network delay than eduroam. The worst network is Unicom ADSL from China because it is far away (22 hops) from the server in Germany and the average RTT is 286 ms. The network connection from China also has the highest packet loss.

Even though the network conditions are changing from time to time, the traceroute results can still act as indications of the network characteristics in a general order of magnitude.

It is worth mentioning that in the runtime experiments we set up one offloading server which is located at our institute, while we use two mobile devices: one is in Germany and the other in China. Thus for the Chinese server queue, the measurement direction is opposite to that in the queueing network model. In the model, jobs are offloaded to the Chinese server, but the runtime measurements are taken by sending jobs from China to the German server. We assume the runtime distributions are identical for both directions so that we can use the runtime results to parameterize the service rate μ_2 for *server.cn*.

To support this assumption, we take another traceroute measurement from our institute to a Chinese server (baidu.com) shown in the row of Table II. From the results, one can see that the number of hops and average RTT (22 hops, 288.2 ms RTT) are nearly the same with that from China client to German server (22 hops, 286.0 ms RTT). Since our model only takes the mean response time into consideration, the transmission time distributions of both directions can be considered identical in our case.

B. Mean runtime of first 10 rounds

Fig. 4 shows the mean runtime of the first 10 chess rounds with the two different mobile devices in different network conditions. We use an average over the first ten rounds since some games end very quickly and almost all games took at least ten rounds, such that those mean values can cover most of the 60 tests. The results are shown with confidence intervals (CI). The local execution time increases rapidly up to several seconds per round on both devices with growing depth. The increase seems to be roughly exponential. However, the server has more processing power and the remote execution time increases slowly but seems to be linearly for both devices and all networks.

We assume that the processing speed of the server will be in the same order or magnitude irrespective of the network over which it is accessed. Therefore, we attribute the difference in the speed of the remote execution using the different networks to the quality of the network conditions. Using eduroam, since the server and the client are in the same LAN, the execution time is the lowest and offloading is beneficial for any search depth. While using the O2 network local execution of the chess search is faster up to a search depth of 7. For higher difficulty, from depth 8 onwards, offloading will increase the processing speed. As for the connection from China, even using a less powerful device Redmi 2, offloading takes longer and is only beneficial for search depth 9 and more.

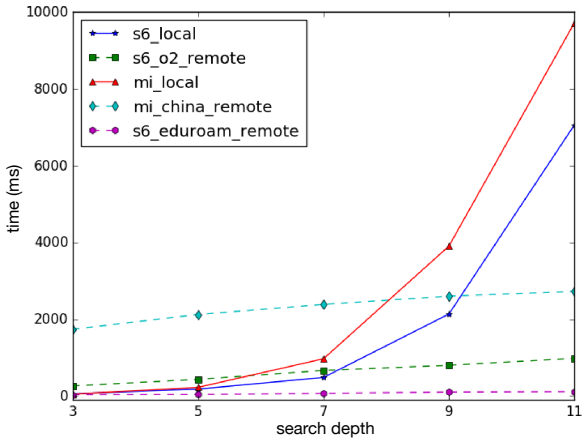


Fig. 4. Mean runtime with different mobile devices in different network conditions

C. Runtime for a full game

The times shown in Table III describe the mean time that must be invested in a full game. We have used the mean number of rounds needed per search depth and the mean time needed for the mean number of rounds. Obviously, the time per game increases most with the depth for the full computation on both mobile devices.

All remote executions increase less and offloading within the eduroam network at our institute is the fastest for a full game at all levels of difficulty. The decisions for a full game will be mostly the same as for the first ten rounds, as considered in Fig. 4.

When the network connection is stable and the user is in the same LAN as the offloading server(e.g. eduroam in our experiment), one should always offload his computation to save time and energy. While as one uses a less powerful device (Redmi 2), one should decide to offload when the search depth is larger than 6 in a good network (e.g. O2 DSL). It can be seen that offloading is beneficial even for a very powerful mobile device (Samsung S6) when the search depth is more than 8. However, if the network delay is high, it is wise to execute the computation locally on the mobile device before the search depth of 10, in which we assume the network condition does not change during one game time.

TABLE III
MEAN TIME PER GAME (MS)

device \ depth	3	5	7	9	11
S6 local	1647.02	4454.53	11752.20	44483.90	95115.70
S6 edurom remote	804.27	916.77	1203.08	1666.47	1698.15
S6 O2 remote	5253.47	8439.13	10092.90	11634.20	13967.10
Redmi local	1793.78	6099.82	23977.40	67450.70	142052.00
Redmi China remote	40650.70	40532.10	40853.60	40343.60	40972.50

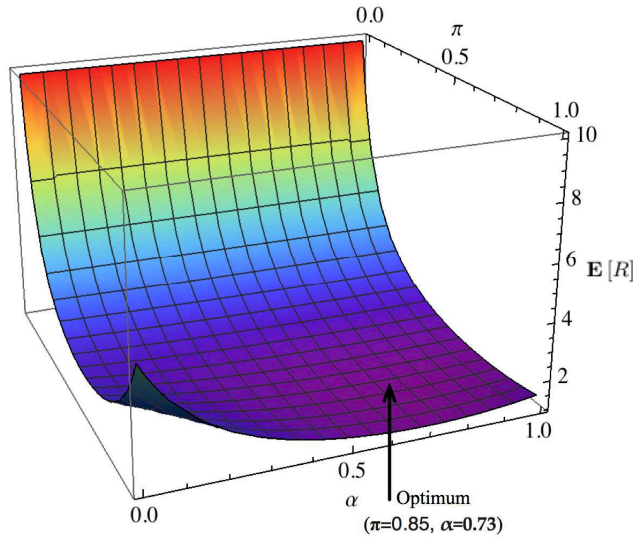


Fig. 5. System average response time over the offloading probability π and the optimal probability of choosing the German server α

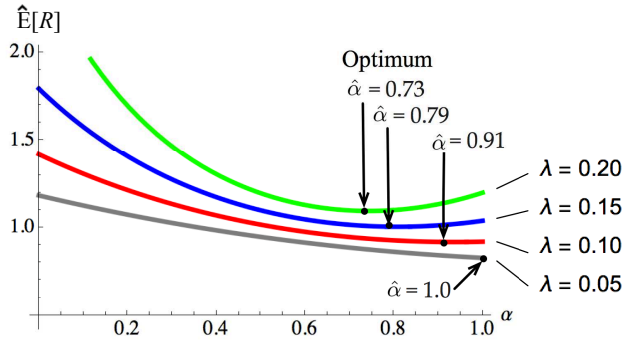


Fig. 6. Mean system response time changing with the probability of choosing the German server α

V. EVALUATION OF THE MULTI-SITE OFFLOADING SCHEME

In this Section, we evaluate the performance metrics proposed in Section III-B using the model analysis results. We parameterize the queues in the proposed model for multi-site offloading scenario using the experiment results from our offloading chess engine.

A. Response Time Analysis

In Fig. 5 we show the average system response time $\mathbb{E}[R]$ versus the offloading probability π and the probability of choosing the German server α. The light color area at the bottom of the figure is the area where the system witnesses the shortest average response time. Searching from the figure, we get the optimum point ($\pi = 0.85, \alpha = 0.73$) for the average response time metric. Besides, one can also see the edge in the top ($\pi = 0$, no jobs are offloaded) shows the highest response time when all jobs are executed on the mobile device. For the sake of brevity, in the following analysis we set the offloading probability $\pi = 0.85$ so that we mainly study how the system performance interacts with the probability of choosing the German server α.

We can further formulate the optimization of the mean response time metric for the offloading assignment as:

$$\arg\min_{\alpha} \ \hat{\mathbb{E}}[R] \ , \tag{11}$$

and we find the probability of choosing the German server α to *server.de* and *server.cn* $(1-\alpha)$ queues such that $\mathbb{E}[R]$ is minimized when all queues are in operation. After setting the offloading probability π as a constant, the optimal probability α can be computed by taking the derivative of the expected response time $\mathbb{E}[R]$:

$$\frac{\mathrm{d}\,\hat{\mathbb{E}}[R]}{\mathrm{d}\alpha} = 0 \ , \tag{12}$$

Then the optimal probability $\hat{\alpha}$ to assign a job to the *server.de* queue is

$$\hat{\alpha} = \frac{\lambda\pi\mu_1 - 2\mu_1\mu_2 + (\mu_1 + \mu_2 - \lambda\pi)\sqrt{\mu_1\mu_2}}{\lambda\pi\,(\mu_1 - \mu_2)} \ . \tag{13}$$

Substituting the numerical instances of model parameters into Eq. 13, we get exactly the same optimal value $\hat{\alpha} = 0.73$ as shown in Fig. 5.

We show the two-dimensional result of the mean system response time changing with the probability of choosing the German α in Fig. 6. The results are presented with four different arrival rates $\lambda = 0.05, 0.10, 0.15$ and 0.20 respectively. We vary the probability of choosing the German server α from 0 to 1. When $\alpha = 1$ all offloading jobs go to *server.de*, whereas for $\alpha = 0$ all jobs are executed in *server.cn*.

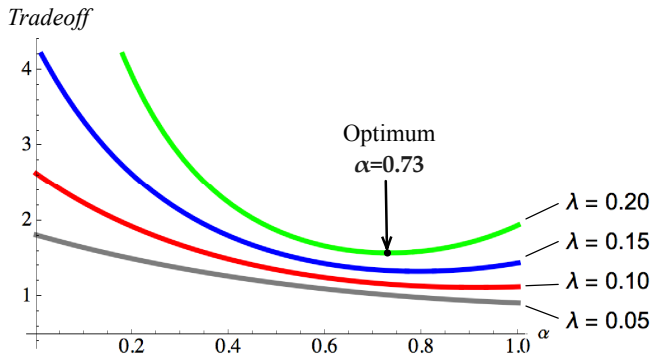


Fig. 7. Response time and energy tradeoff over the optimal probability of choosing the German server α

From the figure, we found the optimal probability of choosing the German server $\hat{\alpha}$ to minimize the system average response time for different arrival rates. When the workload is low ($\lambda = 0.05$), $\mathbb{E}[R]$ is minimized when $\alpha = 1$, that is all jobs are offloaded to *server.de*. As the system workload increases, the optimal probability $\hat{\alpha}$ decreases and the jobs are offloaded to both the German and Chinese servers. When $\lambda = 0.2$, the optimal probability α is 0.73, the same as the value we derived from Eq. 13.

B. Tradeoff Analysis

We present the tradeoff of energy consumption and response time in Fig. 7. As the average response time metric, we formulate the optimization of the *Tradeoff* metric as:

$$\arg\min_{\alpha} \ \textit{Tradeoff} \ . \tag{14}$$

We found that the optimal probability of choosing the German server α for the tradeoff metric is the same as the one for the average response time metric (Fig. 6). The reason is that in the multi-site offloading scenario, the response time dominates the tradeoff metric. When the response time shorter, the mobile device can save more energy by using the results sent from the servers.

C. Dollar per Job Analysis

The accounting cost metric changes with the probability of choosing the German server α as depicted in Fig. 8. It is assumed that the Germany server charges more than the Chinese server as the cost parameters are $\zeta_{de} = 2.5$ and $\zeta_{cn} = 1$. We also present the results with four different arrival rates $\lambda = 0.05, 0.10, 0.15$ and 0.20 respectively.

Obviously, the accounting cost metric increases monotonically with growing probability of choosing the German server α, since the larger α the more jobs are offloaded to the more expensive server *server.de*. Similarly, the larger the system arrival rate λ the more the user has to pay. However, from Fig. 9 when the system workload is heavy ($\lambda = 0.20$) we can find the optimal probability $\alpha = 0.49$ for the *Dollar per Job* metric. As the system arrival rate decreases, it is cheaper to

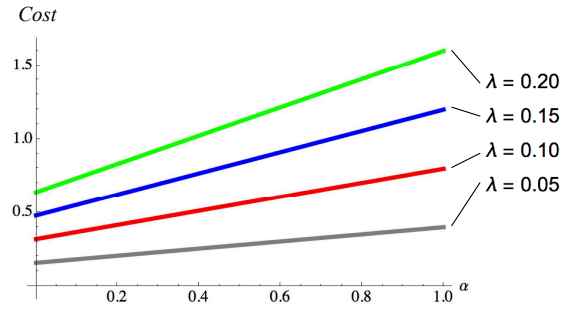


Fig. 8. Accounting cost changing with the probability of choosing the German server α

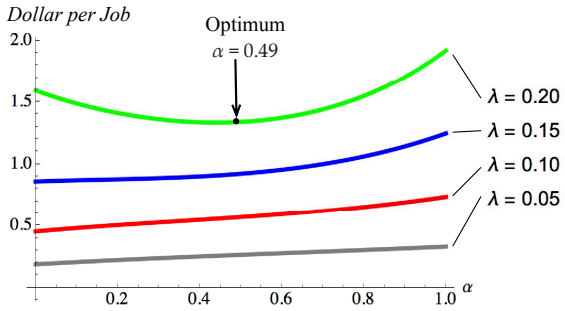


Fig. 9. Dollar per Jobs metric over the probability of choosing the German server α

offload all the jobs to the Chinese server. That is we get the lowest *Dollar per Job* metric when $\alpha = 0$.

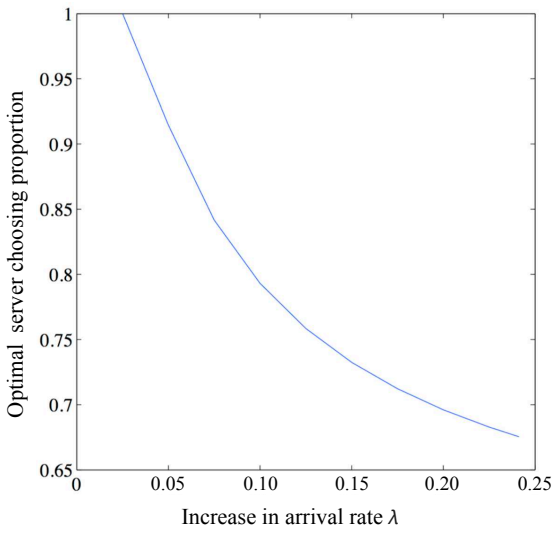
Given the system parameters as arrival rates and service rates, we can calculate the optimal offloading probability π and probability of choosing the German server α for different performance metrics and give advice to system administrators as to how to configure the multi-site offloading system.

D. Sensitivity Analysis

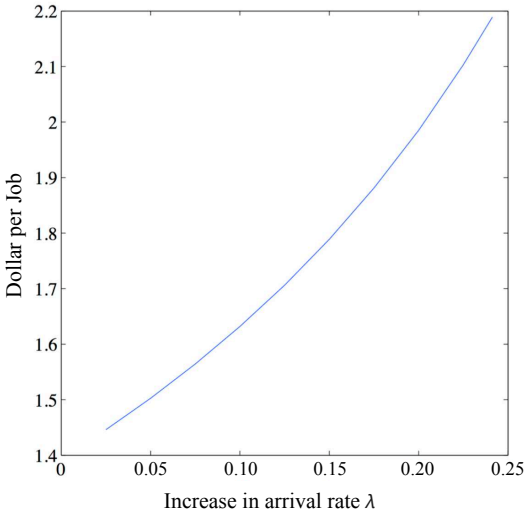
Sensitivity analyses were implemented to explore the effects of increasing the system arrival rate λ on the probability of choosing the German server $\hat{\alpha}$ (Fig. 10(a)) and the *Dollar per Job* metric (Fig. 10(b)). As expected, increasing the system arrival rate necessarily made the offloading less cost-effective and the *Dollar per Job* metric increased accordingly. However, for the optimal probability of choosing the German server an increase in the arrival rate will cause its decrease because as the system workload is heavier some jobs should be offloaded to *server.cn* to avoid the long waiting time in the queue of *server.de*.

VI. Conclusions

In this paper we presented CoOMO, a cost-efficient multi-site offloading scheme for mobile computation outsourcing. CoOMO aims to provide a light-weighted, yet powerful and efficient offloading assignment decision for mobile cloud services. First, an open queueing network model is proposed for



(a) Effect of increase in arrival rate λ on the optimal probability $\hat{\alpha}$



(b) Effect of increase in arrival rate λ on *Dollar per Job* metric

Fig. 10. Sensitivity analysis results

the multi-site offloading system and performance metrics used for evaluating the system are presented. The experimental results confirm that our multi-site offloading scheme is beneficial in terms of response time and energy when the recommended assignment probabilities are applied. We found that the optimal probability for choosing the offloading server for the energy-response time tradeoff metric is the same as the one for the average response time metric. When the system workload is heavy, we found the optimal probability $\alpha = 0.49$ for the *Dollar per Job* metric. However, as the system arrival rate is low, it is always cheaper to offload all the jobs to the remote Chinese server.

ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China (No. 2018YFB1004804), Science and Technology Planning Project of Guangdong Province (No. 2019B010137002) and the Basic Research Program of Shenzhen (JCYJ20180302145731531).

REFERENCES

[1] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–23, 2019.

[2] S. Lin and J. Wan, "Context-awareness enhances 5g mec resource allocation," in *International Symposium on Pervasive Systems, Algorithms and Networks*. Springer, 2019, pp. 347–358.

[3] A. Anand, G. De Veciana, and S. Shakkottai, "Joint scheduling of urllc and embb traffic in 5g wireless networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 477–490, 2020.

[4] Y. Zhang, J. Zhou, Y. Xiang, L. Y. Zhang, F. Chen, S. Pang, and X. Liao, "Computation outsourcing meets lossy channel: Secure sparse robustness decoding service in multi-clouds," *IEEE Transactions on Big Data*, 2017.

[5] Z. Shan, K. Ren, M. Blanton, and C. Wang, "Practical secure computation outsourcing: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–40, 2018.

[6] S. Debnath and B. Bhuyan, "An efficient computation and storage outsourcing process with verification for light weight devices," in *International Conference on Innovation in Modern Science and Technology*. Springer, 2019, pp. 94–103.

[7] B. Zhou and R. Buyya, "Augmentation techniques for mobile cloud computing: A taxonomy, survey, and future directions," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–38, 2018.

[8] M. B. Terefe, H. Lee, N. Heo, G. C. Fox, and S. Oh, "Energy-efficient multisite offloading policy using markov decision process for mobile cloud computing," *Pervasive and Mobile Computing*, 2016.

[9] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4192–4203, 2019.

[10] R. Kumari, S. Kaushal, and N. Chilamkurti, "Energy conscious multi-site computation offloading for mobile cloud computing," *Soft Computing*, vol. 22, no. 20, pp. 6751–6764, 2018.

[11] D. Sulaiman and A. Barker, "Mamoc-android: Multisite adaptive computation offloading for android applications," in *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, 2019, pp. 68–75.

[12] A. Khalili, S. Zarandi, and M. Rasti, "Joint resource allocation and offloading decision in mobile edge computing," *IEEE Communications Letters*, vol. 23, no. 4, pp. 684–687, 2019.

[13] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2019.

[14] F. Altaf, S. Maity *et al.*, "Linearly homomorphic signature based secure computation outsourcing in vehicular adhoc networks," in *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)*, vol. 1. IEEE, 2019, pp. 1–7.

[15] B. Hu, H. Wang, P. Zhang, B. Ding, and H. Che, "Cloudroid: A cloud framework for transparent and qos-aware robotic computation outsourcing," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. IEEE, 2017, pp. 114–121.

[16] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, 2017.

[17] Q. Wang and K. Wolter, "Reducing task completion time in mobile offloading systems through online adaptive local restart," in *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '15. New York, NY, USA: ACM, 2015, pp. 3–13.

[18] M. Goudarzi, M. Zamani, and A. T. Haghighat, "A fast hybrid multi-site computation offloading for mobile cloud computing," *Journal of Network and Computer Applications*, vol. 80, pp. 219–231, 2017.

[19] N. I. M. Enzai and M. Tang, "A heuristic algorithm for multi-site computation offloading in mobile cloud computing," *Procedia Computer Science*, vol. 80, pp. 1232–1241, 2016.

[20] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[21] H. Wu, W. Knottenbelt, and K. Wolter, "Analysis of the energy-response time tradeoff for mobile cloud offloading using combined metrics," in *Teletraffic Congress (ITC 27), 2015 27th International*. IEEE, 2015, pp. 134–142.

[22] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1990.

[23] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone." in *USENIX annual technical conference*, vol. 14. Boston, MA, 2010.

[24] P. Österlund, "Cuckoochess 1.12," http://hem.bredband.net/petero2b/javachess/index.html.

[25] T. Mann and H.G.Muller, "Chess engine communication protocol," https://home.hccnet.nl/h.g.muller/engine-intf.html.

[26] "My traceroute," https://github.com/traviscross/mtr.